

Handbook of Natural Computing

Main Editor
Grzegorz Rozenberg

Editors
Thomas Bäck
Joost N. Kok

Handbook of Natural Computing

With 734 Figures and 75 Tables



Editors

Grzegorz Rozenberg
LIACS
Leiden University
Leiden, The Netherlands
and
Computer Science Department
University of Colorado
Boulder, USA

Joost N. Kok
LIACS
Leiden University
Leiden, The Netherlands

Thomas Bäck
LIACS
Leiden University
Leiden, The Netherlands

ISBN 978-3-540-92909-3 ISBN 978-3-540-92910-9 (eBook)
ISBN 978-3-540-92911-6 (print and electronic bundle)
DOI 10.1007/978-3-540-92910-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010933716

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Natural Computing is the field of research that investigates human-designed computing inspired by nature as well as computing taking place in nature, that is, it investigates models and computational techniques inspired by nature, and also it investigates, in terms of information processing, phenomena taking place in nature.

Examples of the first strand of research include neural computation inspired by the functioning of the brain; evolutionary computation inspired by Darwinian evolution of species; cellular automata inspired by intercellular communication; swarm intelligence inspired by the behavior of groups of organisms; artificial immune systems inspired by the natural immune system; artificial life systems inspired by the properties of natural life in general; membrane computing inspired by the compartmentalized ways in which cells process information; and amorphous computing inspired by morphogenesis. Other examples of natural-computing paradigms are quantum computing and molecular computing, where the goal is to replace traditional electronic hardware, by, for example, bioware in molecular computing. In quantum computing, one uses systems small enough to exploit quantum-mechanical phenomena to perform computations and to perform secure communications more efficiently than classical physics, and, hence, traditional hardware allows. In molecular computing, data are encoded as biomolecules and then tools of molecular biology are used to transform the data, thus performing computations.

The second strand of research, computation taking place in nature, is represented by investigations into, among others, the computational nature of self-assembly, which lies at the core of the nanosciences; the computational nature of developmental processes; the computational nature of biochemical reactions; the computational nature of bacterial communication; the computational nature of brain processes; and the systems biology approach to bionetworks where cellular processes are treated in terms of communication and interaction, and, hence, in terms of computation.

Research in natural computing is genuinely interdisciplinary and forms a bridge between the natural sciences and computer science. This bridge connects the two, both at the level of information technology and at the level of fundamental research. Because of its interdisciplinary character, research in natural computing covers a whole spectrum of research methodologies ranging from pure theoretical research, algorithms, and software applications to experimental laboratory research in biology, chemistry, and physics.

Computer Science and Natural Computing

A preponderance of research in natural computing is centered in computer science. The spectacular progress in Information and Communication Technology (ICT) is highly supported by the evolution of computer science, which designs and develops the instruments needed for this progress: computers, computer networks, software methodologies, etc. As ICT has such a tremendous impact on our everyday lives, so does computer science.

However, there is much more to computer science than ICT: it is the science of information processing and, as such, a fundamental science for other disciplines. On one hand, the only common denominator for research done in such diverse areas of computer science is investigating various aspects of information processing. On the other hand, the adoption of Information and Information Processing as central notions and thinking habit has been an important development in many disciplines, biology and physics being prime examples. For these scientific disciplines, computer science provides not only instruments but also a way of thinking.

We are now witnessing exciting interactions between computer science and the natural sciences. While the natural sciences are rapidly absorbing notions, techniques, and methodologies intrinsic to information processing, computer science is adapting and extending its traditional notion of computation, and computational techniques, to account for computation taking place in nature around us. Natural Computing is an important catalyst for this two-way interaction, and this handbook constitutes a significant record of this development.

The Structure of the Handbook

Natural Computing is both a well-established research field with a number of classical areas, and a very dynamic field with many more recent, novel research areas. The field is vast, and so it is quite usual that a researcher in a specific area does not have sufficient insight into other areas of Natural Computing. Also, because of its dynamic development and popularity, the field constantly attracts more and more scientists who either join the active research or actively follow research developments.

Therefore, the goal of this handbook is two-fold:

- (i) to provide an authoritative reference for a significant and representative part of the research in Natural Computing, and
- (ii) to provide a convenient gateway to Natural Computing for motivated newcomers to this field.

The implementation of this goal was a challenge because this field and its literature are vast — almost all of its research areas have an extensive scientific literature, including specialized journals, book series, and even handbooks. This implies that the coverage of the whole field in reasonable detail and within a reasonable number of pages/volumes is practically impossible.

Thus, we decided to divide the presented material into six areas. These areas are by no means disjoint, but this division is convenient for the purpose of providing a representative picture of the field — representative with respect to the covered research topics and with respect to a good balance between classical and emerging research trends.

Each area consists of individual chapters, each of which covers a specific research theme. They provide necessary technical details of the described research, however they are self-contained and of an expository character, which makes them accessible for a broader audience. They also provide a general perspective, which, together with given references, makes the chapters valuable entries into given research themes.

This handbook is a result of the joint effort of the handbook editors, area editors, chapter authors, and the Advisory Board. The choice of the six areas by the handbook editors in consultation with the Advisory Board, the expertise of the area editors in their respective

areas, the choice by the area editors of well-known researchers as chapter writers, and the peer-review for individual chapters were all important factors in producing a representative and reliable picture of the field. Moreover, the facts that the Advisory Board consists of 68 eminent scientists from 20 countries and that there are 105 contributing authors from 21 countries provide genuine assurance for the reader that this handbook is an authoritative and up-to-date reference, with a high level of significance and accuracy.

Handbook Areas

The material presented in the handbook is organized into six areas: Cellular Automata, Neural Computation, Evolutionary Computation, Molecular Computation, Quantum Computation, and Broader Perspective.

Cellular Automata

Cellular automata are among the oldest models of computation, dating back over half a century. The first cellular automata studies by John von Neumann in the late 1940s were biologically motivated, related to self-replication in universal systems. Since then, cellular automata gained popularity in physics as discrete models of physical systems, in computer science as models of massively parallel computation, and in mathematics as discrete-time dynamical systems. Cellular automata are a natural choice to model real-world phenomena since they possess several fundamental properties of the physical world: they are massively parallel, homogeneous, and all interactions are local. Other important physical constraints such as reversibility and conservation laws can be added as needed, by properly choosing the local update rule. Computational universality is common in cellular automata, and even starkly simple automata are capable of performing arbitrary computation tasks. Because cellular automata have the advantage of parallelism while obeying natural constraints such as locality and uniformity, they provide a framework for investigating realistic computation in massively parallel systems. Computational power and the limitations of such systems are most naturally investigated by time- and space-constrained computations in cellular automata. In mathematics — in terms of symbolic dynamics — cellular automata are viewed as endomorphisms of the full shift, that is, transformations that are translation invariant and continuous in the product topology. Interesting questions on chaotic dynamics have been studied in this context.

Neural Computation

Artificial neural networks are computer programs, loosely modeled after the functioning of the human nervous system. There are neural networks that aim to gain understanding of biological neural systems, and those that solve problems in artificial intelligence without necessarily creating a model of a real biological system. The more biologically oriented neural networks model the real nervous system in increasing detail at all relevant levels of information processing: from synapses to neurons to interactions between modules of interconnected neurons. One of the major challenges is to build artificial brains. By reverse-engineering the mammalian brain in silicon, the aim is to better understand the functioning of the (human)

brain through detailed simulations. Neural networks that are more application-oriented tend to drift further apart from real biological systems. They come in many different flavors, solving problems in regression analysis and time-series forecasting, classification, and pattern recognition, as well as clustering and compression. Good old multilayered perceptrons and self-organizing maps are still pertinent, but attention in research is shifting toward more recent developments, such as kernel methods (including support vector machines) and Bayesian techniques. Both approaches aim to incorporate domain knowledge in the learning process in order to improve prediction performance, e.g., through the construction of a proper kernel function or distance measure or the choice of an appropriate prior distribution over the parameters of the neural network. Considerable effort is devoted to making neural networks efficient so that large models can be learned from huge databases in a reasonable amount of time. Application areas include, among many others, system dynamics and control, finance, bioinformatics, and image analysis.

Evolutionary Computation

The field of evolutionary computation deals with algorithms gleaned from models of organic evolution. The general aim of evolutionary computation is to use the principles of nature's processes of natural selection and genotypic variation to derive computer algorithms for solving hard search and optimization tasks. A wide variety of instances of evolutionary algorithms have been derived during the past fifty years based on the initial algorithms, and we are now witnessing astounding successes in the application of these algorithms: their fundamental understanding in terms of theoretical results; understanding algorithmic principles of their construction; combination with other techniques; and understanding their working principles in terms of organic evolution. The key algorithmic variations (such as genetic algorithms, evolution strategies, evolutionary programming, and genetic programming) have undergone significant developments over recent decades, and have also resulted in very powerful variations and recombinations of these algorithms. Today, there is a sound understanding of how all of these algorithms are instances of the generic concept of an evolutionary search approach. Hence the generic term "evolutionary algorithm" is nowadays being used to describe the generic algorithm, and the term "evolutionary computation" is used for the field as a whole. Thus, we have observed over the past fifty years how the field has integrated the various independently developed initial algorithms into one common principle. Moreover, modern evolutionary algorithms benefit from their ability to adapt and self-adapt their strategy parameters (such as mutation rates, step sizes, and search distributions) to the needs of the task at hand. In this way, they are robust and flexible metaheuristics for problem-solving even without requiring too much special expertise from their users. The feature of self-adaptation illustrates the ability of evolutionary principles to work on different levels at the same time, and therefore provides a nice demonstration of the universality of the evolutionary principle for search and optimization tasks. The widespread use of evolutionary computation reflects these capabilities.

Molecular Computation

Molecular computing is an emergent interdisciplinary field concerned with programming molecules so that they perform a desired computation, or fabricate a desired object, or

control the functioning of a specific molecular system. The central idea behind molecular computing is that data can be encoded as (bio)molecules, e.g., DNA strands, and tools of molecular science can be used to transform these data. In a nutshell, a molecular program is just a collection of molecules which, when placed in a suitable substrate, will perform a specific function (execute the program that this collection represents). The birth of molecular computing is often associated with the 1994 breakthrough experiment by Leonard Adleman, who solved a small instance of a hard computational problem solely by manipulating DNA strands in test tubes. Although initially the main effort of the area was focused on trying to obtain a breakthrough in the complexity of solving hard computational problems, this field has evolved enormously since then. Among the most significant achievements of molecular computing have been contributions to understanding some of the fundamental issues of the nanosciences. One notable example among them is the contribution to the understanding of self-assembly, a central concept of the nanosciences. The techniques of molecular programming were successfully applied in experimentally constructing all kinds of molecular-scale objects or devices with prescribed functionalities. Well-known examples here are self-assembly of Sierpinski triangles, cubes, octahedra, DNA-based logic circuits, DNA “walkers” that move along a track, and autonomous molecular motors. A complementary approach to understanding bioinformation and computation is through studying the information-processing capabilities of cellular organisms. Indeed, cells and nature “compute” by “reading” and “rewriting” DNA through processes that modify DNA (or RNA) sequences. Research into the computational abilities of cellular organisms has the potential to uncover the laws governing biological information, and to enable us to harness the computational power of cells.

Quantum Computation

Quantum computing has been discussed for almost thirty years. The theory of quantum computing and quantum information processing is simply the theory of information processing with a classical notion of information replaced by its quantum counterpart. Research in quantum computing is concerned with understanding the fundamentals of information processing on the level of physical systems that realize/implement the information. In fact, quantum computing can be seen as a quest to understand the fundamental limits of information processing set by nature itself. The mathematical description of quantum information is more complicated than that of classical information — it involves the structure of Hilbert spaces. When describing the structure behind known quantum algorithms, this reduces to linear algebra over complex numbers. The history of quantum algorithms spans the last fifteen years, and some of these algorithms are extremely interesting, and even groundbreaking — the most remarkable are Shor’s factorization in polynomial time and Grover’s search algorithm. The nature of quantum information has also led to the invention of novel cryptosystems, whose security is not based on the complexity of computing functions, but rather on the physical properties of quantum information. Quantum computing is now a well-established discipline, however implementation of a large-scale quantum computer continues to be extremely challenging, even though quantum information processing primitives, including those allowing secure cryptography, have been demonstrated to be practically realizable.

Broader Perspective

In contrast to the first five areas focusing on more-established themes of natural computing, this area encompasses a perspective that is broader in several ways. First, the reader will find here treatments of certain well-established and specific techniques inspired by nature (e.g., simulated annealing) not covered in the other five areas. Second, the reader will also find application-centered chapters (such as natural computing in finance), each covering, in one chapter, a collection of natural computing methods, thus capturing the impact of natural computing as a whole in various fields of science or industry. Third, some chapters are full treatments of several established research fields (such as artificial life, computational systems biology, evolvable hardware, and artificial immune systems), presenting alternative perspectives and cutting across some of the other areas of the handbook, while introducing much new material. Other elements of this area are fresh, emerging, and novel techniques or perspectives (such as collision-based computing, nonclassical computation), representing the leading edge of theories and technologies that are shaping possible futures for both natural computing and computing in general. The contents of this area naturally cluster into two kinds (sections), determined by the essential nature of the techniques involved. These are “Nature-Inspired Algorithms” and “Alternative Models of Computation”. In the first section, “Nature-Inspired Algorithms”, the focus is on algorithms inspired by natural processes realized either through software or hardware or both, as additions to the armory of existing tools we have for dealing with well-known practical problems. In this section, we therefore find application-centered chapters, as well as chapters focusing on particular techniques, not otherwise dealt with in other areas of the handbook, which have clear and proven applicability. In the second section, “Alternative Models of Computation”, the emphasis changes, moving away from specific applications or application areas, toward more far-reaching ideas. These range from developing computational approaches and “computational thinking” as fundamental tools for the new science of systems biology to ideas that take inspiration from nature as a platform for suggesting entirely novel possibilities of computing.

Handbook Chapters

In the remainder of this preface we will briefly describe the contents of the individual chapters. These chapter descriptions are grouped according to the handbook areas where they belong and given in the order that they appear in the handbook. This section provides the reader with a better insight into the contents, allowing one to design a personal roadmap for using this handbook.

Cellular Automata

This area is covered by nine chapters.

The first chapter, “Basic Concepts of Cellular Automata”, by Jarkko J. Kari, reviews some classical results from the theory of cellular automata, relations between various concepts of injectivity and surjectivity, and some basic dynamical system concepts related to chaos in cellular automata. The classical results discussed include the celebrated Garden-of-Eden and Curtis–Hedlund–Lyndon theorems, as well as the balance property of surjective cellular

automata. All these theorems date back to the 1960s. The results are provided together with examples that illustrate proof ideas. Different variants of sensitivity to initial conditions and mixing properties are introduced and related to each other. Also undecidability results concerning cellular automata are briefly discussed.

A popular mathematical approach is to view cellular automata as dynamical systems in the context of symbolic dynamics. Several interesting results in this area were reported as early as 1969 in the seminal paper by G.A. Hedlund, and still today this research direction is among the most fruitful sources of theoretical problems and new results. The chapter “Cellular Automata Dynamical Systems”, by Alberto Dennunzio, Enrico Formenti, and Petr Kůrka, reviews some recent developments in this field. Recent research directions considered here include subshifts attractors and signal subshifts, particle weight functions, and the slicing construction. The first two concern one-dimensional cellular automata and give precise descriptions of the limit behavior of large classes of automata. The third one allows one to view two-dimensional cellular automata as one-dimensional systems. In this way combinatorial complexity is decreased and new results can be proved.

Programming cellular automata for particular tasks requires special techniques. The chapter “Algorithmic Tools on Cellular Automata”, by Marianne Delorme and Jacques Mazoyer, covers classical algorithmic tools based on signals. Linear signals as well as signals of nonlinear slope are discussed, and basic transformations of signals are addressed. The chapter provides results on using signals to construct functions in cellular automata and to implement arithmetic operations on segments. The methods of folding the space-time, freezing, and clipping are also introduced.

The time-complexity advantage gained from parallelism under the locality and uniformity constraints of cellular automata can be precisely analyzed in terms of language recognition. The chapter “Language Recognition by Cellular Automata”, by Véronique Terrier, presents results and questions about cellular automata complexity classes and their relationships to other models of computations. Attention is mainly directed to real-time and linear-time complexity classes, because significant benefits over sequential computation may be obtained at these low time complexities. Both parallel and sequential input modes are considered. Separate complexity classes are given also for cellular automata with one-way communications and two-way communications.

The chapter “Computations on Cellular Automata”, by Jacques Mazoyer and Jean-Baptiste Yunès, continues with the topic of algorithmic techniques in cellular automata. This chapter uses the basic tools, such as signals and grids, to build natural implementations of common algorithms in cellular automata. Examples of implementations include real-time multiplication of integers and the prime number sieve. Both parallel and sequential input and output modes are discussed, as well as composition of functions and recursion.

The chapter “Universality in Cellular Automata”, by Nicolas Ollinger, is concerned with computational universalities. Concepts of universality include Turing universality (the ability to compute any recursive function) and intrinsic universality (the ability to simulate any other cellular automaton). Simulations of Boolean circuits in the two-dimensional case are explained in detail in order to achieve both kinds of universality. The more difficult one-dimensional case is also discussed, and seminal universal cellular automata and encoding techniques are presented in both dimensions. A detailed chronology of important papers on universalities in cellular automata is also provided.

A cellular automaton is reversible if every configuration has only one previous configuration, and hence its evolution process can be traced backward uniquely. This naturally

corresponds to the fundamental time-reversibility of the microscopic laws of physics. The chapter “Reversible Cellular Automata”, by Kenichi Morita, discusses how reversible cellular automata are defined, as well as their properties, how they are designed, and their computing abilities. After providing the definitions, the chapter surveys basic properties of injectivity and surjectivity. Three design methods of reversible cellular automata are provided: block rules, partitioned, and second-order cellular automata. Then the computational power of reversible cellular automata is discussed. In particular, simulation methods of irreversible cellular automata, reversible Turing machines, and some other universal systems are given to clarify universality of reversible cellular automata. In spite of the strong constraint of reversibility, it is shown that reversible cellular automata possess rich information processing capabilities, and even very simple ones are computationally universal.

A conservation law in a cellular automaton is a statement of the invariance of a local and additive energy-like quantity. The chapter “Conservation Laws in Cellular Automata”, by Siamak Taati, reviews the basic theory of conservation laws. A general mathematical framework for formulating conservation laws in cellular automata is presented and several characterizations are summarized. Computational problems regarding conservation laws (verification and existence problems) are discussed. Microscopic explanations of the dynamics of the conserved quantities in terms of flows and particle flows are explored. The related concept of dissipating energy-like quantities is also discussed.

The chapter “Cellular Automata and Lattice Boltzmann Modeling of Physical Systems”, by Bastien Chopard, considers the use of cellular automata and related lattice Boltzmann methods as a natural modeling framework to describe and study many physical systems composed of interacting components. The theoretical basis of the approach is introduced and its potential is illustrated for several applications in physics, biophysics, environmental science, traffic models, and multiscale modeling. The success of the technique can be explained by the close relationship between these methods and a mesoscopic abstraction of many natural phenomena.

Neural Computation

This area is covered by ten chapters.

Spiking neural networks are inspired by recent advances in neuroscience. In contrast to classical neural network models, they take into account not just the neuron’s firing rate, but also the time moment of spike firing. The chapter “Computing with Spiking Neuron Networks”, by Hélène Paugam-Moisy and Sander Bohte, gives an overview of existing approaches to modeling spiking neural neurons and synaptic plasticity, and discusses their computational power and the challenge of deriving efficient learning procedures.

Image quality assessment aims to provide computational models to predict the perceptual quality of images. The chapter “Image Quality Assessment — A Multiscale Geometric Analysis-Based Framework and Examples”, by Xinbo Gao, Wen Lu, Dacheng Tao, and Xuelong Li, introduces the fundamentals and describes the state of the art in image quality assessment. It further proposes a new model, which mimics the human visual system by incorporating concepts such as multiscale analysis, contrast sensitivity, and just-noticeable differences. Empirical results clearly demonstrate that this model resembles subjective perception values and reflects the visual quality of images.

Neurofuzzy networks have the important advantage that they are easy to interpret. When applied to control problems, insight about the process characteristics at different operating regions can be easily obtained. Furthermore, nonlinear model predictive controllers can be developed as a nonlinear combination of several local linear model predictive controllers that have analytical solutions. Through several applications, the chapter “Nonlinear Process Modelling and Control Using Neurofuzzy Networks”, by Jie Zhang, demonstrates that neurofuzzy networks are very effective in the modeling and control of nonlinear processes.

Similar to principal component and factor analysis, independent component analysis is a computational method for separating a multivariate signal into additive subcomponents. Independent component analysis is more powerful: the latent variables corresponding to the subcomponents need not be Gaussian and the basis vectors are typically nonorthogonal. The chapter “Independent Component Analysis”, by Seungjin Choi, explains the theoretical foundations and describes various algorithms based on those principles.

Neural networks has become an important method for modeling and forecasting time series. The chapter “Neural Networks for Time-Series Forecasting”, by G. Peter Zhang, reviews some recent developments (including seasonal time-series modeling, multiperiod forecasting, and ensemble methods), explains when and why they are to be preferred over traditional forecasting models, and also discusses several practical data and modeling issues.

Support vector machines have been extensively studied and applied in many domains within the last decade. Through the so-called kernel trick, support vector machines can efficiently learn nonlinear functions. By maximizing the margin, they implement the principle of structural risk minimization, which typically leads to high generalization performance. The chapter “SVM Tutorial — Classification, Regression and Ranking”, by Hwanjo Yu and Sungchul Kim, describes these underlying principles and discusses support vector machines for different learning tasks: classification, regression, and ranking.

It is well known that single-hidden-layer feedforward networks can approximate any continuous target function. This still holds when the hidden nodes are automatically and randomly generated, independent of the training data. This observation opened up many possibilities for easy construction of a broad class of single-hidden-layer neural networks. The chapter “Fast Construction of Single-Hidden-Layer Feedforward Networks”, by Kang Li, Guang-Bin Huang, and Shuzhi Sam Ge, discusses new ideas that yield a more compact network architecture and reduce the overall computational complexity.

Many recent experimental studies demonstrate the remarkable efficiency of biological neural systems to encode, process, and learn from information. To better understand the experimentally observed phenomena, theoreticians are developing new mathematical approaches and tools to model biological neural networks. The chapter “Modeling Biological Neural Networks”, by Joaquin J. Torres and Pablo Varona, reviews some of the most popular models of neurons and neural networks. These not only help to understand how living systems perform information processing, but may also lead to novel bioinspired paradigms of artificial intelligence and robotics.

The size and complexity of biological data, such as DNA/RNA sequences and protein sequences and structures, makes them suitable for advanced computational tools, such as neural networks. Computational analysis of such databases aims at exposing hidden information that provides insights that help in understanding the underlying biological principles. The chapter “Neural Networks in Bioinformatics”, by Ke Chen and Lukasz A. Kurgan, focuses on proteins. In particular it discusses prediction of protein secondary structure, solvent accessibility, and binding residues.

Self-organizing maps is a prime example of an artificial neural network model that both relates to the actual (topological) organization within the mammalian brain and at the same time has many practical applications. Self-organizing maps go back to the seminal work of Teuvo Kohonen. The chapter “Self-organizing Maps”, by Marc M. Van Hulle, describes the state of the art with a special emphasis on learning algorithms that aim to optimize a predefined criterion.

Evolutionary Computation

This area is covered by thirteen chapters.

The first chapter, “Generalized Evolutionary Algorithms”, by Kenneth De Jong, describes the general concept of evolutionary algorithms. As a generic introduction to the field, this chapter facilitates an understanding of specific instances of evolutionary algorithms as instantiations of a generic evolutionary algorithm. For the instantiations, certain choices need to be made, such as representation, variation operators, and the selection operator, which then yield particular instances of evolutionary algorithms, such as genetic algorithms and evolution strategies, to name just a few.

The chapter “Genetic Algorithms — A Survey of Models and Methods”, by Darrell Whitley and Andrew M. Sutton, introduces and discusses (including criticism) the standard genetic algorithm based on the classical binary representation of solution candidates and a theoretical interpretation based on the so-called schema theorem. Variations of genetic algorithms with respect to solution representations, mutation operators, recombination operators, and selection mechanisms are also explained and discussed, as well as theoretical models of genetic algorithms based on infinite and finite population size assumptions and Markov chain theory concepts. The authors also critically investigate genetic algorithms from the perspective of identifying their limitations and the differences between theory and practice when working with genetic algorithms. To illustrate this further, the authors also give a practical example of the application of genetic algorithms to resource scheduling problems.

The chapter “Evolutionary Strategies”, by Günter Rudolph, describes a class of evolutionary algorithms which have often been associated with numerical function optimization and continuous variables, but can also be applied to binary and integer domains. Variations of evolutionary strategies, such as the $(\mu+\lambda)$ -strategy and the (μ,λ) -strategy, are introduced and discussed within a common algorithmic framework. The fundamental idea of self-adaptation of strategy parameters (variances and covariances of the multivariate normal distribution used for mutation) is introduced and explained in detail, since this is a key differentiating property of evolutionary strategies.

The chapter “Evolutionary Programming”, by Gary B. Fogel, discusses a historical branch of evolutionary computation. It gives a historical perspective on evolutionary programming by describing some of the original experiments using evolutionary programming to evolve finite state machines to serve as sequence predictors. Starting from this canonical evolutionary programming approach, the chapter also presents extensions of evolutionary programming into continuous domains, where an attempt towards self-adaptation of mutation step sizes has been introduced which is similar to the one considered in evolutionary strategies. Finally, an overview of some recent applications of evolutionary programming is given.

The chapter “Genetic Programming — Introduction, Applications, Theory and Open Issues”, by Leonardo Vanneschi and Riccardo Poli, describes a branch of evolutionary

algorithms derived by extending genetic algorithms to allow exploration of the space of computer programs. To make evolutionary search in the domain of computer programs possible, genetic programming is based on LISP S-expression represented by syntax trees, so that genetic programming extends evolutionary algorithms to tree-based representations. The chapter gives an overview of the corresponding representation, search operators, and technical details of genetic programming, as well as existing applications to real-world problems. In addition, it discusses theoretical approaches toward analyzing genetic programming, some of the open issues, as well as research trends in the field.

The subsequent three chapters are related to the theoretical analysis of evolutionary algorithms, giving a broad overview of the state of the art in our theoretical understanding. These chapters demonstrate that there is a sound theoretical understanding of capabilities and limitations of evolutionary algorithms. The approaches can be roughly split into convergence velocity or progress analysis, computational complexity investigations, and global convergence results.

The convergence velocity viewpoint is represented in the chapter “The Dynamical Systems Approach — Progress Measures and Convergence Properties”, by Silja Meyer-Nieberg and Hans-Georg Beyer. It demonstrates how the dynamical systems approach can be used to analyze the behavior of evolutionary algorithms quantitatively with respect to their progress rate. It also provides a complete overview of results in the continuous domain, i.e., for all types of evolution strategies on certain objective functions (such as sphere, ridge, etc.). The chapter presents results for undisturbed as well as for noisy variants of these objective functions, and extends the approach to dynamical objective functions where the goal turns into optimum tracking. All results are presented by means of comparative tables, so the reader gets a complete overview of the key findings at a glance.

The chapter “Computational Complexity of Evolutionary Algorithms”, by Thomas Jansen, deals with the question of optimization time (i.e., the first point in time during the run of an evolutionary algorithm when the global optimum is sampled) and an investigation of upper bounds, lower bounds, and the average time needed to hit the optimum. This chapter presents specific results for certain classes of objective functions, most of them defined over binary search spaces, as well as fundamental limitations of evolutionary search and related results on the “no free lunch” theorem and black box complexity. The chapter also discusses the corresponding techniques for analyses, such as drift analysis and the expected multiplicative distance decrease.

Concluding the set of theoretical chapters, the chapter “Stochastic Convergence”, by Günter Rudolph, addresses theoretical results about the properties of evolutionary algorithms concerned with finding a globally optimal solution in the asymptotic limit. Such results exist for certain variants of evolutionary algorithms and under certain assumptions, and this chapter summarizes the existing results and integrates them into a common framework. This type of analysis is essential in qualifying evolutionary algorithms as global search algorithms and for understanding the algorithmic conditions for global convergence.

The remaining chapters in the area of evolutionary computation report some of the major current trends.

To start with, the chapter “Evolutionary Multiobjective Optimization”, by Eckart Zitzler, focuses on the application of evolutionary algorithms to tasks that are characterized by multiple, conflicting objective functions. In this case, decision-making becomes a task of identifying good compromises between the conflicting criteria. This chapter introduces the concept and a variety of state-of-the-art algorithmic concepts to use evolutionary algorithms

for approximating the so-called Pareto front of solutions which cannot be improved in one objective without compromising another. This contribution presents all of the required formal concepts, examples, and the algorithmic variations introduced into evolutionary computation to handle such types of problems and to generate good approximations of the Pareto front.

The term “memetic algorithms” is used to characterize hybridizations between evolutionary algorithms and more classical, local search methods (and agent-based systems). This is a general concept of broad scope, and in order to illustrate and characterize all possible instantiations, the chapter “Memetic Algorithms”, by Natalio Krasnogor, presents an algorithmic engineering approach which allows one to describe these algorithms as instances of generic patterns. In addition to explaining some of the application areas, the chapter presents some theoretical remarks, various different ways to define memetic algorithms, and also an outlook into the future.

The chapter “Genetics-Based Machine Learning”, by Tim Kovacs, extends the idea of evolutionary optimization to algorithmic concepts in machine learning and data mining, involving applications such as learning classifier systems, evolving neural networks, and genetic fuzzy systems, to mention just a few. Here, the application task is typically a data classification, data prediction, or nonlinear regression task — and the quality of solution candidates is evaluated by means of some model quality measure. The chapter covers a wide range of techniques for applying evolutionary computation to machine learning tasks, by interpreting them as optimization problems.

The chapter “Coevolutionary Principles”, by Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. de Jong, deals with a concept modeled after biological evolution in which an explicit fitness function is not available, but solutions are evaluated by running them against each other. A solution is evaluated in the context of the other solutions, in the actual population or in another. Therefore, these algorithms develop their own dynamics, because the point of comparison is not stable, but coevolving with the actual population. The chapter provides a fundamental understanding of coevolutionary principles and highlights theoretical concepts, algorithms, and applications.

Finally, the chapter “Niching in Evolutionary Algorithms”, by Ofer M. Shir, describes the biological principle of niching in nature as a concept for using a single population to find, occupy, and keep multiple local minima in a population. The motivation for this approach is to find alternative solutions within a single population and run of evolutionary algorithms, and this chapter discusses approaches for niching, and the application in the context of genetic algorithms as well as evolutionary strategies.

Molecular Computation

This area is covered by eight chapters.

The chapter “DNA Computing — Foundations and Implications”, by Lila Kari, Shinno-suke Seki, and Petr Sosík, has a dual purpose. The first part outlines basic molecular biology notions necessary for understanding DNA computing, recounts the first experimental demonstration of DNA computing by Leonard Adleman in 1994, and recaps the 2001 milestone wet laboratory experiment that solved a 20-variable instance of 3-SAT and thus first demonstrated the potential of DNA computing to outperform the computational ability of an unaided human. The second part describes how the properties of DNA-based information, and in particular the Watson–Crick complementarity of DNA single strands, have influenced

areas of theoretical computer science such as formal language theory, coding theory, automata theory, and combinatorics on words. More precisely, it explores several notions and results in formal language theory and coding theory that arose from the problem of the design of optimal encodings for DNA computing experiments (hairpin-free languages, bond-free languages), and more generally from the way information is encoded on DNA strands (sticker systems, Watson–Crick automata). Lastly, it describes the influence that properties of DNA-based information have had on research in combinatorics on words, by presenting several natural generalizations of classical concepts (pseudopalindromes, pseudoperiodicity, Watson–Crick conjugate and commutative words, involutively bordered words, pseudoknot bordered words), and outlining natural extensions in this context of two of the most fundamental results in combinatorics of words, namely the Fine and Wilf theorem and the Lyndon–Schützenberger result.

The chapter “Molecular Computing Machineries — Computing Models and Wet Implementations”, by Masami Hagiya, Satoshi Kobayashi, Ken Komiya, Fumiaki Tanaka, and Takashi Yokomori, explores novel computing devices inspired by the biochemical properties of biomolecules. The theoretical results section describes a variety of molecular computing models for finite automata, as well as molecular computing models for Turing machines based on formal grammars, equality sets, Post systems, and logical formulae. It then presents molecular computing models that use structured molecules such as hairpins and tree structures. The section on wet implementations of molecular computing models, related issues, and applications includes: an enzyme-based DNA automaton and its applications to drug delivery, logic gates and circuits using DNAzymes and DNA tiles, reaction graphs for representing various dynamics of DNA assembly pathways, DNA whiplash machines implementing finite automata, and a hairpin-based implementation of a SAT engine for solving the 3-SAT problem.

The chapter “DNA Computing by Splicing and by Insertion–Deletion”, by Gheorghe Păun, is devoted to two of the most developed computing models inspired by DNA biochemistry: computing by splicing, and computing by insertion and deletion. DNA computing by splicing was defined by Tom Head already in 1987 and is based on the so-called splicing operation. The splicing operation models the recombination of DNA molecules that results from cutting them with restriction enzymes and then pasting DNA molecules with compatible ends by ligase enzymes. This chapter explores the computational power of the splicing operation showing that, for example, extended splicing systems starting from a finite language and using finitely many splicing rules can generate only the family of regular languages, while extended splicing systems starting from a finite language and using a regular set of rules can generate all recursively enumerable languages. Ways in which to avoid the impractical notion of a regular infinite set of rules, while maintaining the maximum computational power, are presented. They include using multisets and adding restrictions on the use of rules such as permitting contexts, forbidding contexts, programmed splicing systems, target languages, and double splicing. The second model presented, the insertion–deletion system, is based on a finite set of axioms and a finite set of contextual insertion rules and contextual deletion rules. Computational power results described here include the fact that insertion–deletion systems with context-free insertion rules of words of length at most one and context-free deletion rules of words of unbounded length can generate only regular languages. In contrast, for example, the family of insertion–deletion systems where the insertion contexts, deletion contexts, and the words to be inserted/deleted are all of length at most one, equals the family of recursively enumerable languages.

The chapter “Bacterial Computing and Molecular Communication”, by Yasubumi Sakakibara and Satoshi Hiyama, investigates attempts to create autonomous cell-based Turing machines, as well as novel communication paradigms that use molecules as communication media. The first part reports experimental research on constructing *in vivo* logic circuits as well as efforts towards building *in vitro* and *in vivo* automata in the framework of DNA computing. Also, a novel framework is presented to develop a programmable and autonomous *in vivo* computer in a bacterium. The first experiment in this direction uses DNA circular strands (plasmids) together with the cell’s protein-synthesis mechanism to execute a finite state automaton in *E. coli*. Molecular communication is a new communication paradigm that proposes the use of molecules as the information medium, instead of the traditional electromagnetic waves. Other distinctive features of molecular communication include its stochastic nature, its low energy consumption, the use of an aqueous transmission medium, and its high compatibility with biological systems. A molecular communication system starts with a sender (e.g., a genetically modified or an artificial cell) that generates molecules, encodes information onto the molecules (called information molecules), and emits the information molecules into a propagation environment (e.g., aqueous solution within and between cells). A molecular propagation system (e.g., lipid bilayer vesicles encapsulating the information molecules) actively transports the information molecules to an appropriate receiver. A receiver (e.g., a genetically modified or an artificial cell) selectively receives the transported information molecules, and biochemically reacts to the received information molecules, thus “decoding” the information. The chapter describes detailed examples of molecular communication system designs, experimental results, and research trends.

The chapter “Computational Nature of Gene Assembly in Ciliates”, by Robert Brijder, Mark Daley, Tero Harju, Nataša Jonoska, Ion Petre, and Grzegorz Rozenberg, reviews several approaches and results in the computational study of gene assembly in ciliates. Ciliated protozoa contain two functionally different types of nuclei, the macronucleus and the micronucleus. The macronucleus contains the functional genes, while the genes of the micronucleus are not functional due to the presence of many interspersing noncoding DNA segments. In addition, in some ciliates, the coding segments of the genes are present in a permuted order compared to their order in the functional macronuclear genes. During the sexual process of conjugation, when two ciliates exchange genetic micronuclear information and form two new micronuclei, each of the ciliates has to “decrypt” the information contained in its new micronucleus to form its new functional macronucleus. This process is called gene assembly and involves deleting the noncoding DNA segments, as well as rearranging the coding segments in the correct order. The chapter describes two models of gene assembly, the intermolecular model based on the operations of circular insertion and deletion, and the intramolecular model based on the three operations of “loop, direct-repeat excision”, “hairpin, inverted-repeat excision”, and “double-loop, alternating repeat excision”. A discussion follows of the mathematical properties of these models, such as the Turing machine computational power of contextual circular insertions and deletions, and properties of the gene assembly process called invariants, which hold independently of the molecular model and assembling strategy. Finally, the template-based recombination model is described, offering a plausible hypothesis (supported already by some experimental data) about the “bioware” that implements the gene assembly.

The chapter “DNA Memory”, by Masanori Arita, Masami Hagiya, Masahiro Takinoue, and Fumiaki Tanaka, summarizes the efforts that have been made towards realizing Eric Baum’s dream of building a DNA memory with a storage capacity vastly larger than

the brain. The chapter first describes the research into strategies for DNA sequence design, i.e., for finding DNA sequences that satisfy DNA computing constraints such as uniform melting temperature, avoidance of undesirable Watson–Crick bonding between sequences, preventing secondary structures, avoidance of base repeats, and absence of forbidden sequences. Various implementations of memory operations, such as access, read, and write, are described. For example, the “access” to a memory word in Baum’s associative memory model, where a memory word consists of a single-stranded portion representing the address and a double-stranded portion representing the data, can be implemented by using the Watson–Crick complement of the address fixed to a solid support. In the Nested Primer Molecular Memory, where the double-stranded data is flanked on both sides by address sequences, the data can be retrieved by Polymerase Chain Reaction (PCR) using the addresses as primer pairs. In the multiple hairpins DNA memory, the address is a catenation of hairpins and the data can be accessed only if the hairpins are opened in the correct order by a process called DNA branch migration. After describing implementations of writable and erasable hairpin memories either in solution or immobilized on surfaces, the topic of *in vivo* DNA memory is explored. As an example, the chapter describes how representing the digit 0 by regular codons, and the digit 1 by wobbled codons, was used to encode a word into an essential gene of *Bacillus subtilis*.

The chapter “Engineering Natural Computation by Autonomous DNA-Based Biomolecular Devices”, by John H. Reif and Thomas H. LaBean, overviews DNA-based biomolecular devices that are autonomous (execute steps with no external control after starting) and programmable (the tasks executed can be modified without entirely redesigning the DNA nanostructures). Special attention is given to DNA tiles, roughly square-shaped DNA nanostructures that have four “sticky-ends” (DNA single strands) that can specifically bind them to other tiles via Watson–Crick complementarity, and thus lead to the self-assembly of larger and more complex structures. Such tiles have been used to execute sequential Boolean computation via linear DNA self-assembly or to obtain patterned 2D DNA lattices and Sierpinski triangles. Issues such as error correction and self-repair of DNA tiling are also addressed. Other described methods include the implementation of a DNA-based finite automaton via disassembly of a double-stranded DNA nanostructure effected by an enzyme, and the technique of whiplash PCR. Whiplash PCR is a method that can achieve state transitions by encoding both transitions and the current state of the computation on the same DNA single strand: The free end of the strand (encoding the current state) sticks to the appropriate transition rule on the strand forming a hairpin, is then extended by PCR to a new state, and finally is detached from the strand, this time with the new state encoded at its end. The technique of DNA origami is also described, whereby a scaffold strand (a long single DNA strand, such as from the sequence of a virus) together with many specially designed staple strands (short single DNA strands) self-assemble by folding the scaffold strand — with the aid of the staples — in a raster pattern that can create given arbitrary planar DNA nanostructures. DNA-based molecular machines are then described such as autonomous DNA walkers and programmable DNA nanobots (programmable autonomous DNA walker devices). A restriction-enzyme-based DNA walker consists of a DNA helix with two sticky-ends (“feet”) that moves stepwise along a “road” (a DNA nanostructure with protruding “steps”, i.e., single DNA strands).

The chapter “Membrane Computing”, by Gheorghe Păun, describes theoretical results and applications of membrane computing, a branch of natural computing inspired by the architecture and functioning of living cells, as well as from the organization of cells in tissues, organs, or other higher-order structures. The cell is a hierarchical structure of compartments, defined by membranes, that selectively communicate with each other. The computing model

that abstracts this structure is a membrane system (or P system, from the name of its inventor, Gheorghe Păun) whose main components are: the membrane structure, the multisets of objects placed in the compartments enveloped by the membranes, and the rules for processing the objects and the membranes. The rules are used to modify the objects in the compartments, to transport objects from one compartment to another, to dissolve membranes, and to create new membranes. The rules in each region of a P system are used in a maximally parallel manner, nondeterministically choosing the applicable rules and the objects to which they apply. A computation consists in repeatedly applying rules to an initial configuration of the P system, until no rule can be applied anymore, in which case the objects in a priori specified regions are considered the output of the computation. Several variants of P systems are described, including P systems with symport/antiport rules, P systems with active membranes, splicing P systems, P systems with objects on membranes, tissue-like P systems, and spiking neural P systems. Many classes of P systems are able to simulate Turing machines, hence they are computationally universal. For example, P systems with symport/antiport rules using only three objects and three membranes are computationally universal. In addition, several types of P systems have been used to solve NP-complete problems in polynomial time, by a space–time trade-off. Applications of P systems include, among others, modeling in biology, computer graphics, linguistics, economics, and cryptography.

Quantum Computation

This area is covered by six chapters.

The chapter “Mathematics for Quantum Information Processing”, by Mika Hirvensalo, contains the standard Hilbert space formulation of finite-level quantum systems. This is the language and notational system allowing us to speak, describe, and make predictions about the objects of quantum physics. The chapter introduces the notion of quantum states as unit-trace, self-adjoint, positive mappings, and the vector state formalism is presented as a special case. The physical observables are introduced as complete collections of mutually orthogonal projections, and then it is discussed how this leads to the traditional representation of observables as self-adjoint mappings. The minimal interpretation, which is the postulate connecting the mathematical objects to the physical world, is presented. The treatment of compound quantum systems is based mostly on operative grounds. To provide enough tools for considering the dynamics needed in quantum computing, the formalism of treating state transformations as completely positive mappings is also presented. The chapter concludes by explaining how quantum versions of finite automata, Turing machines, and Boolean circuits fit into the Hilbert space formalism.

The chapter “Bell’s Inequalities — Foundations and Quantum Communication”, by Časlav Brukner and Marek Žukowski, is concerned with the nature of quantum mechanics. It presents the evidence that excludes two types of hypothetical deterministic theories: neither a nonlocal nor a noncontextual theory can explain quantum mechanics. This helps to build a true picture of quantum mechanics, and is therefore essential from the philosophical point of view. The Bell inequalities show that nonlocal deterministic theories cannot explain the quantum mechanism, and the Kochen–Specker theorem shows that noncontextual theories are not possible as underlying theories either. The traditional Bell theorem and its variants, GHZ and CHSH among them, are presented, and the Kochen–Specker theorem is discussed. In this chapter, the communication complexity is also treated by showing how the violations

of classical locality and noncontextuality can be used as a resource for communication protocols. Stronger-than quantum violations of the CHSH inequality are also discussed. They are interesting, since it has been shown that if the violation of CHSH inequality is strong enough, then the communication complexity collapses into one bit (hence the communication complexity of the true physical world seems to settle somewhere between classical and stronger-than quantum).

The chapter “Algorithms for Quantum Computers”, by Jamie Smith and Michele Mosca, introduces the most remarkable known methods that utilize the special features of quantum physics in order to gain advantage over classical computing. The importance of these methods is that they form the core of designing discrete quantum algorithms. The methods presented and discussed here are the quantum Fourier transform, amplitude amplification, and quantum walks. Then, as specific examples, Shor’s factoring algorithm (quantum Fourier transform), Grover search (amplitude amplification), and element distinctness algorithms (quantum random walks) are presented. The chapter not only involves traditional methods, but it also contains discussion of continuous-time quantum random walks and, more importantly, an extensive presentation of an important recent development in quantum algorithms, viz., tensor network evaluation algorithms. Then, as an example, the approximate evaluation of Tutte polynomials is presented.

The chapter “Physical Implementation of Large-Scale Quantum Computation”, by Kalle-Antti Suominen, discusses the potential ways of physically implementing quantum computers. First, the DiVincenzo criteria (requirements for building a successful quantum computer) are presented, and then quantum error correction is discussed. The history, physical properties, potentials, and obstacles of various possible physical implementations of quantum computers are covered. They involve: cavity QED, trapped ions, neutral atoms and single electrons, liquid-form molecular spin, nuclear and electron spins in silicon, nitrogen vacancies in diamond, solid-state qubits with quantum dots, superconducting charge, flux and phase quantum bits, and optical quantum computing.

The chapter “Quantum Cryptography”, by Takeshi Koshiya, is concerned with quantum cryptography, which will most likely play an important role in future when quantum computers make the current public-key cryptosystems unreliable. It gives an overview of classical cryptosystems, discusses classical cryptographic protocols, and then introduces the quantum key distribution protocols BB84, B92, and BBM92. Also protocol OTU00, not known to be vulnerable under Shor’s algorithm, is presented. In future, when quantum computers are available, cryptography will most probably be based on quantum protocols. The chapter presents candidates for such quantum protocols: KKNY05 and GC01 (for digital signatures). It concludes with a discussion of quantum commitment, oblivious transfer, and quantum zero-knowledge proofs.

The complexity class BQP is the quantum counterpart of the classical class BPP. Intuitively, BQP can be described as the class of problems solvable in “reasonable” time, and, hence, from the application-oriented point of view, it will likely become the most important complexity class in future, when quantum computers are available. The chapter “BQP-Complete Problems”, by Shengyu Zhang, introduces the computational problems that capture the full hardness of BQP. In the very fundamental sense, no BQP-complete problems are known, but the promise problems (the probability distribution of outputs is restricted by promise) bring us as close as possible to the “hardest” problems in BQP, known as BQP-complete promise problems. The chapter discusses known BQP-complete promise problems. In particular, it is shown how to establish the BQP-completeness of the Local Hamiltonian Eigenvalue

Sampling problem and the Local Unitary Phase Sampling problem. The chapter concludes with an extensive study showing that the Jones Polynomial Approximation problem is a BQP-complete promise problem.

Broader Perspective

This area consists of two sections, “Nature-Inspired Algorithms” and “Alternative Models of Computation”.

Nature-Inspired Algorithms

This section is covered by six chapters.

The chapter “An Introduction to Artificial Immune Systems”, by Mark Read, Paul S. Andrews, and Jon Timmis, provides a general introduction to the field. It discusses the major research issues relating to the field of Artificial Immune Systems (AIS), exploring the underlying immunology that has led to the development of immune-inspired algorithms, and focuses on the four main algorithms that have been developed in recent years: clonal selection, immune network, negative selection, and dendritic cell algorithms; their use in terms of applications is highlighted. The chapter also covers evaluation of current AIS technology, and details some new frameworks and methodologies that are being developed towards more principled AIS research. As a counterpoint to the focus on applications, the chapter also gives a brief outline of how AIS research is being employed to help further the understanding of immunology.

The chapter on “Swarm Intelligence”, by David W. Corne, Alan Reynolds, and Eric Bonabeau, attempts to demystify the term Swarm Intelligence (SI), outlining the particular collections of natural phenomena that SI most often refers to and the specific classes of computational algorithms that come under its definition. The early parts of the chapter focus on the natural inspiration side, with discussion of social insects and stigmergy, foraging behavior, and natural flocking behavior. Then the chapter moves on to outline the most successful of the computational algorithms that have emerged from these natural inspirations, namely ant colony optimization methods and particle swarm optimization, with also some discussion of different and emerging such algorithms. The chapter concludes with a brief account of current research trends in the field.

The chapter “Simulated Annealing”, by Kathryn A. Dowsland and Jonathan M. Thompson, provides an overview of Simulated Annealing (SA), emphasizing its practical use. The chapter explains its inspiration from the field of statistical thermodynamics, and then overviews the theory, with an emphasis again on those aspects that are important for practical applications. The chapter then covers some of the main ways in which the basic SA algorithm has been modified by various researchers, leading to improved performance for a variety of problems. The chapter briefly surveys application areas, and ends with several useful pointers to associated resources, including freely available code.

The chapter “Evolvable Hardware”, by Lukáš Sekanina, surveys this growing field. Starting with a brief overview of the reconfigurable devices used in this field, the elementary principles and open problems are introduced, and then the chapter considers, in turn, three main areas: extrinsic evolution (evolving hardware using simulators), intrinsic evolution (where the evolution is conducted within FPGAs, FPTAs, and so forth), and adaptive hardware

(in which real-world adaptive hardware systems are presented). The chapter finishes with an overview of major achievements in the field.

The first of two application-centered chapters, “Natural Computing in Finance — A Review”, by Anthony Brabazon, Jing Dang, Ian Dempsey, Michael O’Neill, and David Edelman, provides a rather comprehensive account of natural computing applications in what is, at the time of writing (and undoubtedly beyond), one of the hottest topics of the day. This chapter introduces us to the wide range of different financial problems to which natural computing methods have been applied, including forecasting, trading, arbitrage, portfolio management, asset allocation, credit risk assessment, and more. The natural computing areas that feature in this chapter are largely evolutionary computing, neural computing, and also agent-based modeling, swarm intelligence, and immune-inspired methods. The chapter ends with a discussion of promising future directions.

Finally, the chapter “Selected Aspects of Natural Computing”, by David W. Corne, Kalyanmoy Deb, Joshua Knowles, and Xin Yao, provides detailed accounts of a collection of example natural computing applications, each of which is remarkable or particularly interesting in some way. The thrust of this chapter is to provide, via such examples, an idea of both the significant impact that natural computing has already had, as well as its continuing significant promise for future applications in all areas of science and industry. While presenting this eclectic collection of marvels, the chapter also aims at clarity and demystification, providing much detail that helps see how the natural computing methods in question were applied to achieve the stated results. Applications covered include Blondie24 (the evolutionary neural network application that achieves master-level skill at the game of checkers), the design of novel antennas using evolutionary computation in conjunction with developmental computing, and the classic application of learning classifier systems that led to novel fighter-plane maneuvers for the USAF.

Alternative Models of Computation

This section is covered by seven chapters.

The chapter “Artificial Life”, by Wolfgang Banzhaf and Barry McMullin, traces the roots, raises key questions, discusses the major methodological tools, and reviews the main applications of this exciting and maturing area of computing. The chapter starts with a historical overview, and presents the fundamental questions and issues that Artificial Life is concerned with. Thus the chapter surveys discussions and viewpoints about the very nature of the differences between living and nonliving systems, and goes on to consider issues such as hierarchical design, self-construction, and self-maintenance, and the emergence of complexity. This part of the chapter ends with a discussion of “Coreworld” experiments, in which a number of systems have been studied that allow spontaneous evolution of computer programs. The chapter moves on to survey the main theory and formalisms used in Artificial Life, including cellular automata and rewriting systems. The chapter concludes with a review and restatement of the main objectives of Artificial Life research, categorizing them respectively into questions about the origin and nature of life, the potential and limitations of living systems, and the relationships between life and intelligence, culture, and other human constructs.

The chapter “Algorithmic Systems Biology — Computer Science Propels Systems Biology”, by Corrado Priami, takes the standpoint of computing as providing a philosophical foundation for systems biology, with at least the same importance as mathematics, chemistry,

or physics. The chapter highlights the value of algorithmic approaches in modeling, simulation, and analysis of biological systems. It starts with a high-level view of how models and experiments can be tightly integrated within an algorithmic systems biology vision, and then deals in turn with modeling languages, simulations of models, and finally the postprocessing of results from biological models and how these lead to new hypotheses that can then re-enter the modeling/simulation cycle.

The chapter “Process Calculi, Systems Biology and Artificial Chemistry”, by Pierpaolo Degano and Andrea Bracciali, concentrates on the use of process calculi and related techniques for systems-level modeling of biological phenomena. This chapter echoes the broad viewpoint of the previous chapter, but its focus takes us towards a much deeper understanding of the potential mappings between formal systems in computer science and systems interpretation of biological processes. It starts by surveying the basics of process calculi, setting out their obvious credentials for modeling concurrent, distributed systems of interacting parts, and mapping these onto a “cells as computers” view. After a process calculi treatment of systems biology, the chapter goes on to examine process calculi as a route towards artificial chemistry. After considering the formal properties of the models discussed, the chapter ends with notes on some case studies showing the value of process calculi in modeling biological phenomena; these include investigating the concept of a “minimal gene set” prokaryote, modeling the nitric oxide-cGMP pathway (central to many signal transduction mechanisms), and modeling the calyx of Held (a large synapse structure in the mammalian auditory central nervous system).

The chapter on “Reaction–Diffusion Computing”, by Andrew Adamatzky and Benjamin De Lacy Costello, introduces the reader to the concept of a reaction–diffusion computer. This is a spatially extended chemical system, which processes information via transforming an input profile of ingredients (in terms of different concentrations of constituent ingredients) into an output profile of ingredients. The chapter takes us through the elements of this field via case studies, and it shows how selected tasks in computational geometry, robotics, and logic can be addressed by chemical implementations of reaction–diffusion computers. After introducing the field and providing a treatment of its origins and main achievements, a classical view of reaction–diffusion computers is then described. The chapter moves on to discuss varieties of reaction–diffusion processors and their chemical constituents, covering applications to the aforementioned tasks. The chapter ends with the authors’ thoughts on future developments in this field.

The chapter “Rough–Fuzzy Computing”, by Andrzej Skowron, shifts our context towards addressing a persistent area of immense difficulty for classical computing, which is the fact that real-world reasoning is usually done in the face of inaccurate, incomplete, and often inconsistent evidence. In essence, concepts in the real world are vague, and computation needs ways to address this. We are hence treated, in this chapter, to an overarching view of rough set theory, fuzzy set theory, their hybridization, and applications. Rough and fuzzy computing are broadly complementary approaches to handling vagueness, focusing respectively on capturing the level of distinction between separate objects and the level of membership of an object in a set. After presenting the basic concepts of rough computing and fuzzy computing in turn, in each case going into some detail on the main theoretical results and practical considerations, the chapter goes on to discuss how they can be, and have been, fruitfully combined. The chapter ends with an overview of the emerging field of “Wisdom Technology” (Wistech) as a paradigm for developing modern intelligent systems.

The chapter “Collision-Based Computing”, by Andrew Adamatzky and Jérôme Durand-Lose, presents and discusses the computations performed as a result of spatial localizations in

systems that exhibit dynamic spatial patterns over time. For example, a collision may be between two gliders in a cellular automaton, or two separate wave fragments within an excitable chemical system. This chapter introduces us to the basics of collision-based computing and overviews collision-based computing schemes in 1D and 2D cellular automata as well as continuous excitable media. Then, after some theoretical foundations relating to 1D cellular automata, the chapter presents a collision-based implementation for a 1D Turing machine and for cyclic tag systems. The chapter ends with discussion and presentation of “Abstract Geometrical Computation”, which can be seen as collision-based computation in a medium that is the continuous counterpart of cellular automata.

The chapter “Nonclassical Computation — A Dynamical Systems Perspective”, by Susan Stepney, takes a uniform view of computation, in which inspiration from a dynamical systems perspective provides a convenient way to consider, in one framework, both classical discrete systems and systems performing nonclassical computation. In particular, this viewpoint presents a way towards computational interpretation of physical embodied systems that exploit their natural dynamics. The chapter starts by discussing “closed” dynamical systems, those whose dynamics involve no inputs from an external environment, examining their computational abilities from a dynamical systems perspective. Then it discusses continuous dynamical systems and shows how these too can be interpreted computationally, indicating how material embodiment can give computation “for free”, without the need to explicitly implement the dynamics. The outlook then broadens to consider open systems, where the dynamics are affected by external inputs. The chapter ends by looking at constructive, or developmental, dynamical systems, whose state spaces change during computation. These latter discussions approach the arena of biological and other natural systems, casting them as computational, open, developmental, dynamical systems.

Acknowledgements

This handbook resulted from a highly collaborative effort. The handbook and area editors are grateful to the chapter writers for their efforts in writing chapters and delivering them on time, and for their participation in the refereeing process.

We are indebted to the members of the Advisory Board for their valuable advice and fruitful interactions. Additionally, we want to acknowledge David Fogel, Pekka Lahti, Robert LaRue, Jason Lohn, Michael Main, David Prescott, Arto Salomaa, Kai Salomaa, Shinnosuke Seki, and Rob Smith, for their help and advice in various stages of production of this handbook. Last, but not least, we are thankful to Springer, especially to Ronan Nugent, for intense and constructive cooperation in bringing this project from its inception to its successful conclusion.

Leiden; Edinburgh; Nijmegen;
Turku; London, Ontario
October 2010

Grzegorz Rozenberg (Main Handbook Editor)
Thomas Bäck (Handbook Editor and Area Editor)
Joost N. Kok (Handbook Editor and Area Editor)
David W. Corne (Area Editor)
Tom Heskes (Area Editor)
Mika Hirvensalo (Area Editor)
Jarkko J. Kari (Area Editor)
Lila Kari (Area Editor)

Editor Biographies



Prof. Dr. Grzegorz Rozenberg

Prof. Rozenberg was awarded his Ph.D. in mathematics from the Polish Academy of Sciences, Warsaw, and he has since held full-time positions at Utrecht University, the State University of New York at Buffalo, and the University of Antwerp. Since 1979 he has been a professor at the Department of Computer Science of Leiden University, and an adjunct professor at the Department of Computer Science of the University of Colorado at Boulder, USA. He is the founding director of the Leiden Center for Natural Computing.

Among key editorial responsibilities over the last 30 years, he is the founding Editor of the book series *Texts in Theoretical Computer Science* (Springer) and *Monographs in Theoretical Computer Science* (Springer), founding Editor of the book series *Natural Computing* (Springer), founding Editor-in-Chief of the journal *Natural Computing* (Springer), and founding Editor of Part C (Theory of Natural Computing) of the journal *Theoretical Computer Science* (Elsevier). Altogether he's on the Editorial Board of around 20 scientific journals.

He has authored more than 500 papers and 6 books, and coedited more than 90 books. He coedited the “Handbook of Formal Languages” (Springer), he was Managing Editor of the “Handbook of Graph Grammars and Computing by Graph Transformation” (World Scientific), he coedited “Current Trends in Theoretical Computer Science” (World Scientific), and he coedited “The Oxford Handbook of Membrane Computing” (Oxford University Press).

He is Past President of the European Association for Theoretical Computer Science (EATCS), and he received the Distinguished Achievements Award of the EATCS “in recognition of his outstanding scientific contributions to theoretical computer science”. Also he is a cofounder and Past President of the International Society for Nanoscale Science, Computation, and Engineering (ISNSCE).

He has served as a program committee member for most major conferences on theoretical computer science in Europe, and among the events he has founded or helped to establish are the International Conference on Developments in Language Theory (DLT), the International

Conference on Graph Transformation (ICGT), the International Conference on Unconventional Computation (UC), the International Conference on Application and Theory of Petri Nets and Concurrency (ICATPN), and the DNA Computing and Molecular Programming Conference.

In recent years his research has focused on natural computing, including molecular computing, computation in living cells, self-assembly, and the theory of biochemical reactions. His other research areas include the theory of concurrent systems, the theory of graph transformations, formal languages and automata theory, and mathematical structures in computer science.

Prof. Rozenberg is a Foreign Member of the Finnish Academy of Sciences and Letters, a member of the Academia Europaea, and an honorary member of the World Innovation Foundation. He has been awarded honorary doctorates by the University of Turku, the Technical University of Berlin, and the University of Bologna. He is an ISI Highly Cited Researcher.

He is a performing magician, and a devoted student of and expert on the paintings of Hieronymus Bosch.



Prof. Dr. Thomas Bäck

Prof. Bäck was awarded his Ph.D. in Computer Science from Dortmund University in 1994, for which he received the Best Dissertation Award from the Gesellschaft für Informatik (GI). He has been at Leiden University since 1996, where he is currently full Professor for Natural Computing and the head of the Natural Computing Research Group at the Leiden Institute of Advanced Computer Science (LIACS).

He has authored more than 150 publications on natural computing technologies. He wrote a book on evolutionary algorithms, “Evolutionary Algorithms in Theory and Practice” (Oxford University Press), and he coedited the “Handbook of Evolutionary Computation” (IOP/Oxford University Press).

Prof. Bäck is an Editor of the book series Natural Computing (Springer), an Associate Editor of the journal Natural Computing (Springer), an Editor of the journal Theoretical Computer Science (Sect. C, Theory of Natural Computing; Elsevier), and an Advisory Board member of the journal Evolutionary Computation (MIT Press). He has served as program chair for all major conferences in evolutionary computation, and is an Elected Fellow of the International Society for Genetic and Evolutionary Computation for his contributions to the field.

His main research interests are the theory of evolutionary algorithms, cellular automata and data-driven modelling, and applications of these methods in medicinal chemistry, pharmacology and engineering.



Prof. Dr. Joost N. Kok

Prof. Kok was awarded his Ph.D. in Computer Science from the Free University in Amsterdam in 1989, and he has worked at the Centre for Mathematics and Computer Science in Amsterdam, at Utrecht University, and at the Åbo Akademi University in Finland. Since 1995 he has been a professor in computer science, and since 2005 also a professor in medicine at Leiden University. He is the Scientific Director of the Leiden Institute of Advanced Computer Science, and leads the research clusters Algorithms and Foundations of Software Technology.

He serves as a chair, member of the management team, member of the board, or member of the scientific committee of the Faculty of Mathematics and Natural Sciences of Leiden University, the ICT and Education Committee of Leiden University, the Dutch Theoretical Computer Science Association, the Netherlands Bioinformatics Centre, the Centre for Mathematics and Computer Science Amsterdam, the Netherlands Organisation for Scientific Research, the Research Foundation Flanders (Belgium), the European Educational Forum, and the International Federation for Information Processing (IFIP) Technical Committee 12 (Artificial Intelligence).

Prof. Kok is on the steering, scientific or advisory committees of the following events: the Mining and Learning with Graphs Conference, the Intelligent Data Analysis Conference, the Institute for Programming and Algorithms Research School, the Biotechnological Sciences Delft–Leiden Research School, and the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. And he has been a program committee member for more than 100 international conferences, workshops or summer schools on data mining, data analysis, and knowledge discovery; neural networks; artificial intelligence; machine learning; computational life science; evolutionary computing, natural computing and genetic algorithms; Web intelligence and intelligent agents; and software engineering.

He is an Editor of the book series Natural Computing (Springer), an Associate Editor of the journal Natural Computing (Springer), an Editor of the journal Theoretical Computer Science (Sect. C, Theory of Natural Computing; Elsevier), an Editor of the Journal of Universal

Computer Science, an Associate Editor of the journal Computational Intelligence (Wiley), and a Series Editor of the book series Frontiers in Artificial Intelligence and Applications (IOS Press).

His academic research is concentrated around the themes of scientific data management, data mining, bioinformatics, and algorithms, and he has collaborated with more than 20 industrial partners.

Advisory Board

Shun-ichi Amari

RIKEN Brain Science Institute
Saitama
Japan

Wolfgang Banzhaf

Memorial University of Newfoundland
St. John's
Canada

David Barber

University College London
London
UK

Thomas Bartz-Beielstein

Fachhochschule Köln
Cologne
Germany

Nino Boccaro

University of Illinois at Chicago
Chicago, Illinois
USA

Jürgen Branke

Warwick Business School
Coventry
UK

Gilles Brassard

Université de Montréal
Montreal, Quebec
Canada

Larry Bull

University of the West of England
Bristol
UK

Cristian Calude

University of Auckland
Auckland
New Zealand

Luca Cardelli

Microsoft Research
Cambridge
UK

Gianpiero Cattaneo

Università degli Studi di Milano–Bicocca
Milan
Italy

Bastien Chopard

Université de Genève
Geneva
Switzerland

David Cliff

University of Bristol
Bristol
UK

Anne Condon

University of British Columbia
Vancouver
Canada

A. C. C. (Ton) Coolen

King's College London
London
UK

David W. Corne

Heriot-Watt University
Edinburgh
UK

David Davis
VGO Oil & Gas
Richmond, Texas
USA

Kenneth De Jong
George Mason University
Fairfax, Virginia
USA

Marco Dorigo
Université libre de Bruxelles
Brussels
Belgium

Włodzisław Duch
Nicolaus Copernicus University
Toruń
Poland

Enrico Formenti
Université de Nice–Sophia Antipolis
Nice
France

Eric Goles
Universidad Adolfo Ibáñez
Santiago
Chile

Erik Goodman
Michigan State University
East Lansing, Michigan
USA

Masami Hagiya
University of Tokyo
Tokyo
Japan

David Harel
Weizmann Institute
Rehovot
Israel

Tom Head
Binghamton University
Binghamton, New York
USA

Tom Heskes
Radboud Universiteit Nijmegen
Nijmegen
The Netherlands

Mika Hirvensalo
University of Turku
Turku
Finland

Alexander S. Holevo
Steklov Mathematical Institute, Russian
Academy of Sciences
Moscow
Russia

Owen Holland
University of Essex
Colchester, Essex
UK

Nataša Jonoska
University of South Florida
Tampa, Florida
USA

H. J. (Bert) Kappen
Radboud Universiteit Nijmegen
Nijmegen
The Netherlands

Jarkko J. Kari
University of Turku
Turku
Finland

Lila Kari
University of Western Ontario
London, Ontario
Canada

Joshua Knowles
University of Manchester
Manchester
UK

Petr Kůrka
Institute of Advanced Studies, Charles Univ.
Czech Republic

Wolfgang Maass
Technische Universität Graz
Graz
Austria

Thomas Martinetz
Universität zu Lübeck
Lübeck
Germany

Giancarlo Mauri
Università degli Studi di Milano–Bicocca
Milan
Italy

Jacques Mazoyer
Aix-Marseille Université and CNRS
Marseille
France

Tal Mor
Technion
Haifa
Israel

Kenichi Morita
Hiroshima University
Higashi-Hiroshima City
Japan

Michael C. Mozer
University of Colorado
Boulder
USA

Akira Namatame
National Defense Academy
Yokosuka, Kanagawa
Japan

Erkki Oja
Helsinki University of Technology
Helsinki
Finland

Nicolas Ollinger
Université de Provence
Aix–Marseille
France

Günther Palm
Universität Ulm
Germany

Gheorghe Păun
Romanian Academy, Bucharest, Romania
and University of Seville
Seville
Spain

Marcus Pivato
Trent University
Peterborough, Ontario
Canada

Przemysław Prusinkiewicz
University of Calgary
Calgary, Alberta
Canada

John H. Reif
Duke University
Durham, North Carolina
USA

Peter Ross
Napier University
Edinburgh
UK

Paul W. K. Rothmund
Caltech, Pasadena
California
USA

Günter Rudolph
Technische Universität Dortmund
Dortmund
Germany

Arto Salomaa
University of Turku
Turku
Finland

Marc Schoenauer
INRIA
Paris
France

Hans-Paul Schwefel
Technische Universität Dortmund
Dortmund
Germany

Nadrian C. (Ned) Seeman
New York University
New York
USA

Akira Suyama
University of Tokyo
Tokyo
Japan

Siamak Taati
University of Turku
Turku
Finland

Jon Timmis
University of York
York
UK

Carme Torras
Universitat Politècnica de Catalunya
Barcelona
Spain

Michel Verleysen
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Darrell Whitley
Colorado State University
Fort Collins, Colorado
USA

Stewart W. Wilson
Prediction Dynamics
Concord
USA

David Wolpert
NASA Ames Research Center
Moffett Field, California
USA

Xin Yao
University of Birmingham
Birmingham
UK

Anton Zeilinger
Universität Wien
Vienna
Austria

Area Editors

Cellular Automata

Jarkko J. Kari
University of Turku
Turku
Finland
jkari@utu.fi

Neural Computation

Tom Heskes
Radboud Universiteit Nijmegen
Nijmegen
The Netherlands
t.heskes@science.ru.nl

Joost N. Kok
LIACS
Leiden University
Leiden
The Netherlands
joost@liacs.nl

Evolutionary Computation

Thomas Bäck
LIACS
Leiden University
Leiden
The Netherlands
baeck@liacs.nl

Molecular Computation

Lila Kari
University of Western Ontario
London
Canada
lila@csd.uwo.ca

Quantum Computation

Mika Hirvensalo
University of Turku
Turku
Finland
mikhirve@utu.fi

Broader Perspective

David W. Corne
Heriot-Watt University
Edinburgh
UK
dwcorne@macs.hw.ac.uk

Table of Contents

Preface	v
Editor Biographies	xvii
Advisory Board	xxxi
Area Editors	xxxv
List of Contributors	xliii

Volume 1

Section I: Cellular Automata	1
<i>Jarkko J. Kari</i>	
1 Basic Concepts of Cellular Automata	3
<i>Jarkko J. Kari</i>	
2 Cellular Automata Dynamical Systems	25
<i>Alberto Dennunzio · Enrico Formenti · Petr Kůrka</i>	
3 Algorithmic Tools on Cellular Automata	77
<i>Marianne Delorme · Jacques Mazoyer</i>	
4 Language Recognition by Cellular Automata	123
<i>Véronique Terrier</i>	
5 Computations on Cellular Automata	159
<i>Jacques Mazoyer · Jean-Baptiste Yunès</i>	
6 Universalities in Cellular Automata	189
<i>Nicolas Ollinger</i>	
7 Reversible Cellular Automata	231
<i>Kenichi Morita</i>	
8 Conservation Laws in Cellular Automata	259
<i>Siamak Taati</i>	
9 Cellular Automata and Lattice Boltzmann Modeling of Physical Systems	287
<i>Bastien Chopard</i>	

Section II: Neural Computation	333
<i>Tom Heskes and Joost N. Kok</i>	
10 Computing with Spiking Neuron Networks	335
<i>Hélène Paugam-Moisy · Sander Bohte</i>	
11 Image Quality Assessment — A Multiscale Geometric Analysis-Based Framework and Examples	377
<i>Xinbo Gao · Wen Lu · Dacheng Tao · Xuelong Li</i>	
12 Nonlinear Process Modelling and Control Using Neurofuzzy Networks	401
<i>Jie Zhang</i>	
13 Independent Component Analysis	435
<i>Seungjin Choi</i>	
14 Neural Networks for Time-Series Forecasting	461
<i>G. Peter Zhang</i>	
15 SVM Tutorial — Classification, Regression and Ranking	479
<i>Hwanjo Yu · Sungchul Kim</i>	
16 Fast Construction of Single-Hidden-Layer Feedforward Networks	507
<i>Kang Li · Guang-Bin Huang · Shuzhi Sam Ge</i>	
17 Modeling Biological Neural Networks	533
<i>Joaquin J. Torres · Pablo Varona</i>	
18 Neural Networks in Bioinformatics	565
<i>Ke Chen · Lukasz A. Kurgan</i>	
19 Self-organizing Maps	585
<i>Marc M. Van Hulle</i>	

Volume 2

Section III: Evolutionary Computation	623
<i>Thomas Bäck</i>	
20 Generalized Evolutionary Algorithms	625
<i>Kenneth De Jong</i>	
21 Genetic Algorithms — A Survey of Models and Methods	637
<i>Darrell Whitley · Andrew M. Sutton</i>	
22 Evolutionary Strategies	673
<i>Günter Rudolph</i>	

23 Evolutionary Programming	699
<i>Gary B. Fogel</i>	
24 Genetic Programming — Introduction, Applications, Theory and Open Issues	709
<i>Leonardo Vanneschi · Riccardo Poli</i>	
25 The Dynamical Systems Approach — Progress Measures and Convergence Properties	741
<i>Silja Meyer-Nieberg · Hans-Georg Beyer</i>	
26 Computational Complexity of Evolutionary Algorithms	815
<i>Thomas Jansen</i>	
27 Stochastic Convergence	847
<i>Günter Rudolph</i>	
28 Evolutionary Multiobjective Optimization	871
<i>Eckart Zitzler</i>	
29 Memetic Algorithms	905
<i>Natalio Krasnogor</i>	
30 Genetics-Based Machine Learning	937
<i>Tim Kovacs</i>	
31 Coevolutionary Principles	987
<i>Elena Popovici · Anthony Bucci · R. Paul Wiegand · Edwin D. de Jong</i>	
32 Niching in Evolutionary Algorithms	1035
<i>Ofer M. Shir</i>	

Volume 3

Section IV: Molecular Computation	1071
<i>Lila Kari</i>	
33 DNA Computing — Foundations and Implications	1073
<i>Lila Kari · Shinnosuke Seki · Petr Sosík</i>	
34 Molecular Computing Machineries — Computing Models and Wet Implementations	1129
<i>Masami Hagiya · Satoshi Kobayashi · Ken Komiya · Fumiaki Tanaka · Takashi Yokomori</i>	
35 DNA Computing by Splicing and by Insertion–Deletion	1185
<i>Gheorghe Păun</i>	

36 Bacterial Computing and Molecular Communication	1203
<i>Yasubumi Sakakibara · Satoshi Hiyama</i>	
37 Computational Nature of Gene Assembly in Ciliates	1233
<i>Robert Brijder · Mark Daley · Tero Harju · Nataša Jonoska · Ion Petre · Grzegorz Rozenberg</i>	
38 DNA Memory	1281
<i>Masanori Arita · Masami Hagiya · Masahiro Takinoue · Fumiaki Tanaka</i>	
39 Engineering Natural Computation by Autonomous DNA-Based Biomolecular Devices	1319
<i>John H. Reif · Thomas H. LaBean</i>	
40 Membrane Computing	1355
<i>Gheorghe Păun</i>	
Section V: Quantum Computation	1379
<i>Mika Hirvensalo</i>	
41 Mathematics for Quantum Information Processing	1381
<i>Mika Hirvensalo</i>	
42 Bell's Inequalities — Foundations and Quantum Communication	1413
<i>Časlav Brukner · Marek Żukowski</i>	
43 Algorithms for Quantum Computers	1451
<i>Jamie Smith · Michele Mosca</i>	
44 Physical Implementation of Large-Scale Quantum Computation	1493
<i>Kalle-Antti Suominen</i>	
45 Quantum Cryptography	1521
<i>Takeshi Koshiha</i>	
46 BQP-Complete Problems	1545
<i>Shengyu Zhang</i>	

Volume 4

Section VI: Broader Perspective – Nature-Inspired Algorithms	1573
<i>David W. Corne</i>	
47 An Introduction to Artificial Immune Systems	1575
<i>Mark Read · Paul S. Andrews · Jon Timmis</i>	

48	Swarm Intelligence	1599
	<i>David W. Corne · Alan Reynolds · Eric Bonabeau</i>	
49	Simulated Annealing	1623
	<i>Kathryn A. Dowsland · Jonathan M. Thompson</i>	
50	Evolvable Hardware	1657
	<i>Lukáš Sekanina</i>	
51	Natural Computing in Finance – A Review	1707
	<i>Anthony Brabazon · Jing Dang · Ian Dempsey · Michael O'Neill · David Edelman</i>	
52	Selected Aspects of Natural Computing	1737
	<i>David W. Corne · Kalyanmoy Deb · Joshua Knowles · Xin Yao</i>	
Section VII: Broader Perspective – Alternative Models of Computation		1803
	<i>David W. Corne</i>	
53	Artificial Life	1805
	<i>Wolfgang Banzhaf · Barry McMullin</i>	
54	Algorithmic Systems Biology — Computer Science Propels Systems Biology	1835
	<i>Corrado Priami</i>	
55	Process Calculi, Systems Biology and Artificial Chemistry	1863
	<i>Pierpaolo Degano · Andrea Bracciali</i>	
56	Reaction–Diffusion Computing	1897
	<i>Andrew Adamatzky · Benjamin De Lacy Costello</i>	
57	Rough–Fuzzy Computing	1921
	<i>Andrzej Skowron</i>	
58	Collision-Based Computing	1949
	<i>Andrew Adamatzky · Jérôme Durand-Lose</i>	
59	Nonclassical Computation — A Dynamical Systems Perspective	1979
	<i>Susan Stepney</i>	
Index		2027

List of Contributors

Andrew Adamatzky

Department of Computer Science
University of the West of England
Bristol
UK
andrew.adamatzky@uwe.ac.uk

Paul S. Andrews

Department of Computer Science
University of York
UK
psa@cs.york.ac.uk

Masanori Arita

Department of Computational Biology,
Graduate School of Frontier Sciences
The University of Tokyo
Kashiwa
Japan
arita@k.u-tokyo.ac.jp

Wolfgang Banzhaf

Department of Computer Science
Memorial University of Newfoundland
St. John's, NL
Canada
banzhaf@cs.mun.ca

Hans-Georg Beyer

Department of Computer Science
Fachhochschule Vorarlberg
Dornbirn
Austria
hans-georg.beyer@fhv.at

Sander Bohte

Research Group Life Sciences
CWI
Amsterdam
The Netherlands
s.m.bohte@cwi.nl

Eric Bonabeau

Icosystem Corporation
Cambridge, MA
USA
eric@icosystem.com

Anthony Brabazon

Natural Computing Research and
Applications Group
University College Dublin
Ireland
anthony.brabazon@ucd.ie

Andrea Bracciali

Department of Computing Science and
Mathematics
University of Stirling
UK
braccia@cs.stir.ac.uk

Robert Brijder

Leiden Institute of Advanced Computer
Science
Universiteit Leiden
The Netherlands
robert.brijder@uhasselt.be

Časlav Brukner

Faculty of Physics
University of Vienna
Vienna
Austria
caslav.brukner@univie.ac.at

Anthony Bucci

Icosystem Corporation
Cambridge, MA
USA
anthony@icosystem.com

Ke Chen

Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB
Canada
kchen1@ece.ualberta.ca

Seungjin Choi

Pohang University of Science and Technology
Pohang
South Korea
seungjin@postech.ac.kr

Bastien Chopard

Scientific and Parallel Computing Group
University of Geneva
Switzerland
bastien.chopard@unige.ch

David W. Corne

School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh
UK
dwcorne@macs.hw.ac.uk

Mark Daley

Departments of Computer Science and Biology
University of Western Ontario
London, Ontario
Canada
daley@csd.uwo.ca

Jing Dang

Natural Computing Research and Applications Group
University College Dublin
Ireland
jing.dang@ucd.ie

Edwin D. de Jong

Institute of Information and Computing Sciences
Utrecht University
The Netherlands
dejong@cs.uu.nl

Kenneth De Jong

Department of Computer Science
George Mason University
Fairfax, VA
USA
kdejong@gmu.edu

Benjamin De Lacy Costello

Centre for Research in Analytical, Material and Sensor Sciences, Faculty of Applied Sciences
University of the West of England
Bristol
UK
ben.delacycostello@uwe.ac.uk

Kalyanmoy Deb

Department of Mechanical Engineering
Indian Institute of Technology
Kanpur
India
deb@iitk.ac.in

Pierpaolo Degano

Dipartimento di Informatica
Università di Pisa
Italy
degano@di.unipi.it

Marianne Delorme

Laboratoire d'Informatique Fondamentale de Marseille (LIF)
Aix-Marseille Université and CNRS
Marseille
France
delorme.marianne@orange.fr

Ian Dempsey
Pipeline Financial Group, Inc.
New York, NY
USA
ian.dempsey@pipelinefinancial.com

Alberto Dennunzio
Dipartimento di Informatica
Sistemistica e Comunicazione, Università
degli Studi di Milano-Bicocca
Italy
dennunzio@disco.unimib.it

Kathryn A. Dowsland
Gower Optimal Algorithms, Ltd.
Swansea
UK
k.a.dowsland@btconnect.com

Jérôme Durand-Lose
LIFO
Université d'Orléans
France
jerome.durand-lose@univ-orleans.fr

David Edelman
School of Business
UCD Michael Smurfit Graduate Business
School
Dublin
Ireland
david.edelman@ucd.ie

Gary B. Fogel
Natural Selection, Inc.
San Diego, CA
USA
g fogel@natural-selection.com

Enrico Formenti
Département d'Informatique
Université de Nice-Sophia Antipolis
France
enrico.formenti@unice.fr

Xinbo Gao
Video and Image Processing System Lab,
School of Electronic Engineering
Xidian University
China
xbgao@ieee.org

Shuzhi Sam Ge
Social Robotics Lab
Interactive Digital Media Institute, The
National University of Singapore
Singapore
elegesz@nus.edu.sg

Masami Hagiya
Department of Computer Science
Graduate School of Information Science
and Technology
The University of Tokyo
Tokyo
Japan
hagiya@is.s.u-tokyo.ac.jp

Tero Harju
Department of Mathematics
University of Turku
Finland
harju@utu.fi

Mika Hirvensalo
Department of Mathematics
University of Turku
Finland
mikhirve@utu.fi

Satoshi Hiyama
Research Laboratories
NTT DOCOMO, Inc.
Yokosuka
Japan
hiyama@nttdocomo.co.jp

Guang-Bin Huang

School of Electrical and Electronic
Engineering
Nanyang Technological University
Singapore
egbhuang@ntu.edu.sg

Joshua Knowles

School of Computer Science and
Manchester Interdisciplinary
Biocentre (MIB)
University of Manchester
UK
j.knowles@manchester.ac.uk

Thomas Jansen

Department of Computer Science
University College Cork
Ireland
t.jansen@cs.ucc.ie

Satoshi Kobayashi

Department of Computer Science
University of Electro-Communications
Tokyo
Japan
satoshi@cs.uec.ac.jp

Nataša Jonoska

Department of Mathematics
University of South Florida
Tampa, FL
USA
jonoska@math.usf.edu

Ken Komiya

Interdisciplinary Graduate School of
Science and Engineering
Tokyo Institute of Technology
Yokohama
Japan
komiya@dis.titech.ac.jp

Jarkko J. Kari

Department of Mathematics
University of Turku
Turku
Finland
jkari@utu.fi

Takeshi Koshiba

Graduate School of Science and
Engineering
Saitama University
Japan
koshiba@mail.saitama-u.ac.jp

Lila Kari

Department of Computer Science
University of Western Ontario
London
Canada
lila@csd.uwo.ca

Tim Kovacs

Department of Computer Science
University of Bristol
UK
kovacs@cs.bris.ac.uk

Sungchul Kim

Data Mining Lab, Department of Computer
Science and Engineering
Pohang University of Science and
Technology
Pohang
South Korea
subright@postech.ac.kr

Natalio Krasnogor

Interdisciplinary Optimisation Laboratory,
The Automated Scheduling, Optimisation
and Planning Research Group, School of
Computer Science
University of Nottingham
UK
natalio.krasnogor@nottingham.ac.uk

Lukasz A. Kurgan

Department of Electrical and Computer
Engineering
University of Alberta
Edmonton, AB
Canada
lkurgan@ece.ualberta.ca

Petr Kůrka

Center for Theoretical Studies
Academy of Sciences and Charles
University in Prague
Czechia
kurka@cts.cuni.cz

Thomas H. LaBean

Department of Computer Science and
Department of Chemistry and Department
of Biomedical Engineering
Duke University
Durham, NC
USA
thomas.labean@duke.edu

Kang Li

School of Electronics, Electrical Engineering
and Computer Science
Queen's University
Belfast
UK
k.li@ee.qub.ac.uk

Xuelong Li

Center for OPTical IMagery Analysis and
Learning (OPTIMAL), State Key Laboratory
of Transient Optics and Photonics
Xi'an Institute of Optics and Precision
Mechanics, Chinese Academy of Sciences
Xi'an, Shaanxi
China
xuelong_li@opt.ac.cn

Wen Lu

Video and Image Processing System Lab,
School of Electronic Engineering
Xidian University
China
luwen@mail.xidian.edu.cn

Jacques Mazoyer

Laboratoire d'Informatique Fondamentale
de Marseille (LIF)
Aix-Marseille Université and CNRS
Marseille
France
mazoyerj2@orange.fr

Barry McMullin

Artificial Life Lab, School of Electronic
Engineering
Dublin City University
Ireland
barry.mcmullin@dcu.ie

Silja Meyer-Nieberg

Fakultät für Informatik
Universität der Bundeswehr München
Neubiberg
Germany
silja.meyer-nieberg@unibw.de

Kenichi Morita

Department of Information Engineering,
Graduate School of Engineering
Hiroshima University
Japan
morita@iec.hiroshima-u.ac.jp

Michele Mosca

Institute for Quantum Computing and
Department of Combinatorics &
Optimization
University of Waterloo and St. Jerome's
University and Perimeter Institute for
Theoretical Physics
Waterloo
Canada
mmosca@iqc.ca

Nicolas Ollinger

Laboratoire d'informatique fondamentale
de Marseille (LIF)
Aix-Marseille Université, CNRS
Marseille
France
nicolas.ollinger@lif.univ-mrs.fr

Michael O'Neill

Natural Computing Research and
Applications Group
University College Dublin
Ireland
m.oneill@ucd.ie

Hélène Paugam-Moisy

Laboratoire LIRIS – CNRS
Université Lumière Lyon 2
Lyon
France
and
INRIA Saclay – Ile-de-France
Université Paris-Sud
Orsay
France
helene.paugam-moisy@univ-lyon2.fr
hpaugam@iri.fr

Gheorghe Păun

Institute of Mathematics of the Romanian
Academy
Bucharest
Romania
and
Department of Computer Science and
Artificial Intelligence
University of Seville
Spain
gpaun@us.es
george.paun@imar.ro

Ion Petre

Department of Information Technologies
Åbo Akademi University
Turku
Finland
ipetre@abo.fi

Riccardo Poli

Department of Computing and Electronic
Systems
University of Essex
Colchester
UK
rpoli@essex.ac.uk

Elena Popovici

Icosystem Corporation
Cambridge, MA
USA
elena@icosystem.com

Corrado Priami

Microsoft Research
University of Trento Centre for
Computational and Systems Biology
(CoSBI)
Trento
Italy
and
DISI
University of Trento
Trento
Italy
priami@cosbi.eu

Mark Read

Department of Computer Science
University of York
UK
markread@cs.york.ac.uk

John H. Reif

Department of Computer Science
Duke University
Durham, NC
USA
reif@cs.duke.edu

Alan Reynolds
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh
UK
a.reynolds@hw.ac.uk

Grzegorz Rozenberg
Leiden Institute of Advanced Computer Science
Universiteit Leiden
The Netherlands
and
Department of Computer Science
University of Colorado
Boulder, CO
USA
rozenber@liacs.nl

Günter Rudolph
Department of Computer Science
TU Dortmund
Dortmund
Germany
guenter.rudolph@tu-dortmund.de

Yasubumi Sakakibara
Department of Biosciences and Informatics
Keio University
Yokohama
Japan
yasu@bio.keio.ac.jp

Lukáš Sekanina
Faculty of Information Technology
Brno University of Technology
Brno
Czech Republic
sekanina@fit.vutbr.cz

Shinnosuke Seki
Department of Computer Science
University of Western Ontario
London
Canada
sseki@csd.uwo.ca

Ofer M. Shir
Department of Chemistry
Princeton University
NJ
USA
oshir@princeton.edu

Andrzej Skowron
Institute of Mathematics
Warsaw University
Poland
skowron@mimuw.edu.pl

Jamie Smith
Institute for Quantum Computing and
Department of Combinatorics &
Optimization
University of Waterloo
Canada
ja5smith@iqc.ca

Petr Sosík
Institute of Computer Science
Silesian University in Opava
Czech Republic
and
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
Spain
petr.sosik@fpf.slu.cz

Susan Stepney
Department of Computer Science
University of York
UK
susan.stepney@cs.york.ac.uk

Kalle-Antti Suominen
Department of Physics and Astronomy
University of Turku
Finland
kalle-antti.suominen@utu.fi

Andrew M. Sutton

Department of Computer Science
Colorado State University
Fort Collins, CO
USA
sutton@cs.colostate.edu

Jon Timmis

Department of Computer Science and
Department of Electronics
University of York
UK
jtimmis@cs.york.ac.uk

Siamak Taati

Department of Mathematics
University of Turku
Finland
siamak.taati@gmail.com

Joaquin J. Torres

Institute "Carlos I" for Theoretical and
Computational Physics and Department of
Electromagnetism and Matter Physics,
Facultad de Ciencias
Universidad de Granada
Spain
jtorres@ugr.es

Masahiro Takinoue

Department of Physics
Kyoto University
Kyoto
Japan
takinoue@chem.scphys.kyoto-u.ac.jp

Marc M. Van Hulle

Laboratorium voor Neurofysiologie
K.U. Leuven
Leuven
Belgium
marc@neuro.kuleuven.be

Fumiaki Tanaka

Department of Computer Science
Graduate School of Information Science
and Technology
The University of Tokyo
Tokyo
Japan
fumi95@is.s.u-tokyo.ac.jp

Leonardo Vanneschi

Department of Informatics, Systems and
Communication
University of Milano-Bicocca
Italy
vanneschi@disco.unimib.it

Dacheng Tao

School of Computer Engineering
Nanyang Technological University
Singapore
dacheng.tao@gmail.com

Pablo Varona

Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
Spain
pablo.varona@uam.es

Véronique Terrier

GREYC, UMR CNRS 6072
Université de Caen
France
veroniqu@info.unicaen.fr

Darrell Whitley

Department of Computer Science
Colorado State University
Fort Collins, CO
USA
whitley@cs.colostate.edu

Jonathan M. Thompson

School of Mathematics
Cardiff University
UK
thompsonjm1@cardiff.ac.uk

R. Paul Wiegand

Institute for Simulation and Training
 University of Central Florida
 Orlando, FL
 USA
 wiegand@ist.ucf.edu

Xin Yao

Natural Computation Group, School of Computer Science
 University of Birmingham
 UK
 x.yao@cs.bham.ac.uk

Takashi Yokomori

Department of Mathematics, Faculty of Education and Integrated Arts and Sciences
 Waseda University
 Tokyo
 Japan
 yokomori@waseda.jp

Hwanjo Yu

Data Mining Lab, Department of Computer Science and Engineering
 Pohang University of Science and Technology
 Pohang
 South Korea
 hwanjoyu@postech.ac.kr

Jean-Baptiste Yunès

Laboratoire LIAFA
 Université Paris 7 (Diderot)
 France
 jean-baptiste.yunes@liafa.jussieu.fr

G. Peter Zhang

Department of Managerial Sciences
 Georgia State University
 Atlanta, GA
 USA
 gpzhang@gsu.edu

Jie Zhang

School of Chemical Engineering and Advanced Materials
 Newcastle University
 Newcastle upon Tyne
 UK
 jie.zhang@newcastle.ac.uk

Shengyu Zhang

Department of Computer Science and Engineering
 The Chinese University of Hong Kong
 Hong Kong S.A.R.
 China
 syzhang@cse.cuhk.edu.hk

Eckart Zitzler

PHBern – University of Teacher Education,
 Institute for Continuing Professional Education
 Bern
 Switzerland
 eckart.zitzler@phbern.ch
 eckart.zitzler@tik.ee.ethz.ch

Marek Żukowski

Institute of Theoretical Physics and Astrophysics
 University of Gdańsk
 Poland
 marek.zukowski@univie.ac.at
 fizmz@univ.gda.pl

Section I

Cellular Automata

Jarkko J. Kari

1 Basic Concepts of Cellular Automata

Jarkko J. Kari

Department of Mathematics, University of Turku, Turku, Finland

jkari@utu.fi

1	<i>Introduction</i>	4
2	<i>Preliminaries</i>	5
3	<i>Classical Results</i>	10
4	<i>Injectivity and Surjectivity Properties</i>	14
5	<i>Algorithmic Questions</i>	19
6	<i>Dynamical Systems Concepts</i>	20
7	<i>Conclusions</i>	23

Abstract

This chapter reviews some basic concepts and results of the theory of cellular automata (CA). Topics discussed include classical results from the 1960s, relations between various concepts of injectivity and surjectivity, and dynamical system concepts related to chaos in CA. Most results are reported without full proofs but sometimes examples are provided that illustrate the idea of a proof. The classical results discussed include the Garden-of-Eden theorem and the Curtis–Hedlund–Lyndon theorem, as well as the balance property of surjective CA. Different variants of sensitivity to initial conditions and mixing properties are introduced and related to each other. Also, algorithmic aspects and undecidability results are mentioned.

1 Introduction

A cellular automaton (CA) is a discrete dynamical system that consists of an infinite array of cells. Each cell has a state from a finite state set. The cells change their states according to a local update rule that provides the new state based on the old states of the cell and its neighbors. All states use the same update rule, and the updating happens simultaneously at all cells. The updating is repeated over and over again at discrete time steps, leading to a time evolution of the system.

CA are among the oldest models of natural computing, dating back over half a century. The first CA studies by John von Neumann in the late 1940s were biologically motivated, related to self-replication in universal systems (Burks 1970; von Neumann 1966). Since then, CA have gained popularity as discrete models of physical systems. This is not surprising, considering that CA possess several fundamental properties of the physical world: they are massively parallel, homogeneous, and all interactions are local. Other important physical constraints such as reversibility and conservation laws can be added as needed by choosing the local update rule properly. See the chapter [Cellular Automata and Lattice Boltzmann Modeling of Physical Systems](#) by Chopard for more information on physical modeling using CA and related lattice Boltzmann methods, and the chapters [Reversible Cellular Automata](#) and [Conservation Laws in Cellular Automata](#) for details on the reversibility and conservation laws in CA.

CA have also been studied extensively as models of computation. Computational universality is common in CA, and even amazingly simple CA are capable of performing arbitrary computation tasks. Famous examples of this include Conway's Game-of-Life (Berlekamp et al. 1982; Gardner 1970) and the elementary rule 110 (Cook 2004; Wolfram 1986). The chapter [Universality in Cellular Automata](#) by Ollinger contains other examples and more information on universality. As CA have the advantage of parallelism while obeying natural constraints such as locality and uniformity, CA provide a framework for investigating realistic computation in massively parallel systems. Programming such systems requires special techniques. In particular, the concept of signals turns out to be most useful, see the chapters [Algorithmic Tools on Cellular Automata](#) and [Computations on Cellular Automata](#). The time complexity advantage gained from the parallelism under the locality/uniformity constraints of CA can be precisely analyzed in terms of language recognition. The time complexity classes of CA languages are among the classical research directions, see the chapter [Language Recognition by Cellular Automata](#) by Terrier.

A popular mathematical approach is to view CA as dynamical systems in the context of symbolic dynamics. Several interesting results in this area were reported already in 1969 in the seminal paper by Hedlund (1969), and until today this research direction has been among

the most fruitful sources of problems and new results in the theory of CA. The chapter **Cellular Automata Dynamical Systems** provides details on recent advances in this aspect.

The purpose of this chapter is to introduce the basic concepts and review some selected results in the CA theory in order to make the more advanced chapters of the handbook more accessible. In **Sect. 2**, the basic definitions and results are stated, including the Curtis–Hedlund–Lyndon characterization of CA as continuous, translation-commuting maps (**Theorem 1**). In **Sect. 3**, other classical results from the 1960s are reviewed. These include the balance property of surjective CA (**Theorem 2**) as well as the Garden-of-Eden theorem (**Theorems 3** and **4**). In **Sect. 4**, different injectivity and surjectivity properties are related to each other, and in **Sect. 5**, corresponding algorithmic questions are looked into. **Sect. 6** introduces concepts related to topological dynamics and chaos, including variants of sensitivity to initial conditions and mixing of the configuration space.

2 Preliminaries

Only the basic CA model is considered here: deterministic, synchronous CA, where the underlying grid is infinite and rectangular. The cells are hence the squares of an infinite d -dimensional checker board, addressed by \mathbb{Z}^d . The one-, two-, and three-dimensional cases are most common and, as seen below, one-dimensional CA behave in many respects quite differently from the higher dimensional ones.

2.1 Basic Definitions

The states of the automaton come from a finite *state set* S . At any given time, the *configuration* of the automaton is a mapping $c : \mathbb{Z}^d \rightarrow S$ that specifies the states of all cells. The set of all d -dimensional configurations over state set S is denoted by $S^{\mathbb{Z}^d}$. For any $s \in S$, the constant configuration $c(\mathbf{x}) = s$ for all $\mathbf{x} \in \mathbb{Z}^d$ is called *s-homogeneous*.

The cells change their states synchronously at discrete time steps. The next state of each cell depends on the current states of the neighboring cells according to an update rule. All cells use the same rule, and the rule is applied to all cells at the same time. The neighboring cells may be the nearest cells surrounding the cell, but more general neighborhoods can be specified by giving the relative offsets of the neighbors. Let $N = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$ be a vector of m distinct elements of \mathbb{Z}^d . Then the *neighbors* of a cell at location $\mathbf{x} \in \mathbb{Z}^d$ are the m cells at locations

$$\mathbf{x} + \mathbf{x}_i, \quad \text{for } i = 1, 2, \dots, m$$

The *local rule* is a function $f : S^m \rightarrow S$ where m is the size of the neighborhood. State $f(a_1, a_2, \dots, a_m)$ is the state of a cell whose m neighbors were at states a_1, a_2, \dots, a_m one time step before. This update rule then determines the global dynamics of the CA: Configuration c becomes in one time step the configuration e where, for all $\mathbf{x} \in \mathbb{Z}^d$,

$$e(\mathbf{x}) = f(c(\mathbf{x} + \mathbf{x}_1), c(\mathbf{x} + \mathbf{x}_2), \dots, c(\mathbf{x} + \mathbf{x}_m))$$

We denote $e = G(c)$, and $G : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ is called the *global transition function* of the CA.

In summary, CA are dynamical systems that are homogeneous and discrete in both time and space, and that are updated locally in space. A CA is specified by a quadruple (d, S, N, f) where $d \in \mathbb{Z}_+$ is the dimension of the space, S is the state set, $N \in (\mathbb{Z}^d)^m$ is the neighborhood vector, and $f : S^m \rightarrow S$ is the local update rule. One usually identifies a CA with its global

transition function G , and talks about CA function G , or simply CA G . In algorithmic questions, G is however always specified using the four finite items d , S , N , and f .

A CA is called *injective* if the global transition function G is one-to-one. It is *surjective* if G is onto. A CA is *bijective* if G is both onto and one-to-one.

Configuration c is *temporally periodic* for CA G , or *G -periodic*, if $G^k(c) = c$ for some $k \geq 1$. If $G(c) = c$ then c is a *fixed point*. Every CA has a temporally periodic configuration that is homogeneous: This follows from the facts that there are finitely many homogeneous configurations and that CA functions preserve homogeneity. Configuration c is *eventually periodic* if it evolves into a temporally periodic configuration, that is, if $G^m(c) = G^n(c)$ for some $m > n \geq 0$. This is equivalent to the property that the (*forward*) orbit $c, G(c), G^2(c), \dots$ is finite.

CA G is called *nilpotent* if $G^n(S^{\mathbb{Z}^d})$ is a singleton set for sufficiently large n , that is, if there is a configuration c and number n such that $G^n(e) = c$ for all configurations e . Since homogeneous configurations remain homogeneous, one can immediately see that configuration c has to be a homogeneous fixed point.

Let G_1 and G_2 be two given CA functions with the same state set and the same dimension d . The *composition* $G_1 \circ G_2$ is also a CA function, and the composition can be formed effectively. If N_1 and N_2 are neighborhoods of G_1 and G_2 , respectively, then a neighborhood of $G_1 \circ G_2$ consists of vectors $\mathbf{x} + \mathbf{y}$ for all \mathbf{x} in N_1 and \mathbf{y} in N_2 .

Sometimes it happens that $G_1 \circ G_2 = G_2 \circ G_1 = id$ where id is the identity function. Then CA G_1 and G_2 are called *reversible*, and G_1 and G_2 are the *inverse automata* of each other. One can effectively decide whether two given CA are inverses of each other. This follows from the effectiveness of forming the composition and the decidability of the CA equivalence. Reversible CA are studied more in the chapter [Reversible Cellular Automata](#) in this handbook.

Examples of one-dimensional CA dynamics are often depicted as *space–time* diagrams. The horizontal rows of a space–time diagram are consecutive configurations. The top row is the initial configuration. More generally, the d -dimensional space–time diagram for initial configuration c is the mapping $sp: \mathbb{N} \times \mathbb{Z}^d \rightarrow S$ such that $sp(t, \mathbf{x}) = G^t(c)(\mathbf{x})$ for all $t \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{Z}^d$.

2.2 Neighborhoods

Neighborhoods commonly used in CA are the *von Neumann* neighborhood N_{VN} and the *Moore* neighborhood N_M . The von Neumann neighborhood contains relative offsets \mathbf{y} that satisfy $\|\mathbf{y}\|_1 \leq 1$, where

$$\|(y_1, y_2, \dots, y_d)\|_1 = |y_1| + |y_2| + \dots + |y_d|$$

This means that a cell in location \mathbf{x} has $2d + 1$ neighbors: the cell itself and the cells at locations $\mathbf{x} \pm \mathbf{e}_i$ where $\mathbf{e}_i = (0, \dots, 0, 1, 0, \dots, 0)$ is the i th coordinate unit vector. The Moore neighborhood contains all vectors $\mathbf{y} = (y_1, y_2, \dots, y_d)$ where each y_i is $-1, 0$ or 1 , that is, all $\mathbf{y} \in \mathbb{Z}^d$ such that $\|\mathbf{y}\|_\infty \leq 1$, where

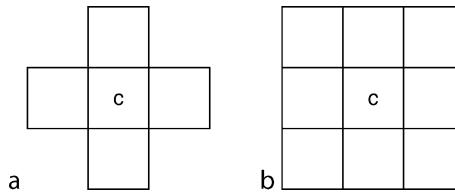
$$\|(y_1, y_2, \dots, y_d)\|_\infty = \max\{|y_1|, |y_2|, \dots, |y_d|\}.$$

Every cell has 3^d Moore neighbors. [Figure 1](#) shows the von Neumann and Moore neighborhoods in the case $d = 2$.

Generalizing the Moore neighborhood, one obtains *radius- r* CA, for any positive integer r . In radius- r CA, the relative neighborhood consists of vectors \mathbf{y} such that $\|\mathbf{y}\|_\infty \leq r$. The Moore neighborhood is of radius 1.

Fig. 1

Two-dimensional (a) von Neumann and (b) Moore neighbors of cell c.



By *radius- $\frac{1}{2}$* neighborhood it is meant the neighborhood that contains the offsets $\mathbf{y} = (y_1, y_2, \dots, y_d)$ where each y_i is 0 or 1. A one-dimensional, radius- $\frac{1}{2}$ CA is also called one-way, or OCA for short. Since the neighborhood $(0, 1)$ does not contain negative elements, information cannot flow to the positive direction. However, one can shift the cells as shown in Fig. 2 to obtain a symmetric trellis where information can flow in both directions.

2.3 Finite and Periodic Configurations

The *shift functions* are particularly simple CA that translate the configurations one cell down in one of the coordinate directions. More precisely, for each dimension $i = 1, 2, \dots, d$ there is the corresponding shift function σ_i whose neighborhood contains only the unit coordinate vector \mathbf{e}_i and whose local rule is the identity function. The one-dimensional shift function is the *left shift* $\sigma = \sigma_1$. *Translations* are compositions of shift functions and their inverses. Translation τ_y by vector \mathbf{y} is the CA with neighborhood $(-\mathbf{y})$ and the identity local rule. In $\tau_y(c)$ cell $\mathbf{x} + \mathbf{y}$ has the state $c(\mathbf{x})$, for all $\mathbf{x} \in \mathbb{Z}^d$.

Sometimes, one state $q \in S$ is specified as a *quiescent state*. It should be *stable*, meaning that $f(q, q, \dots, q) = q$. The quiescent configuration is the configuration where all cells are quiescent. A configuration $c \in S^{\mathbb{Z}^d}$ is called *q-finite* (or simply finite) if only a finite number of cells are non-quiescent, that is, the *q-support*

$$\text{supp}_q(c) = \{\mathbf{x} \in \mathbb{Z}^d \mid c(\mathbf{x}) \neq q\}$$

is finite. Let us denote by $\mathfrak{F}_q(d, S)$, or briefly \mathfrak{F}_q the subset of $S^{\mathbb{Z}^d}$ that contains only the *q-finite* configurations. Because of the stability of *q*, finite configurations remain finite in the evolution of the CA, so the restriction G_F of G on the finite configurations is a function $\mathfrak{F}_q \rightarrow \mathfrak{F}_q$.

A *periodic configuration*, or, more precisely, a *spatially periodic configuration* is a configuration that is invariant under d linearly independent translations. This is equivalent to the existence of d positive integers t_1, t_2, \dots, t_d such that $c = \sigma_i^{t_i}(c)$ for every $i = 1, 2, \dots, d$, that is,

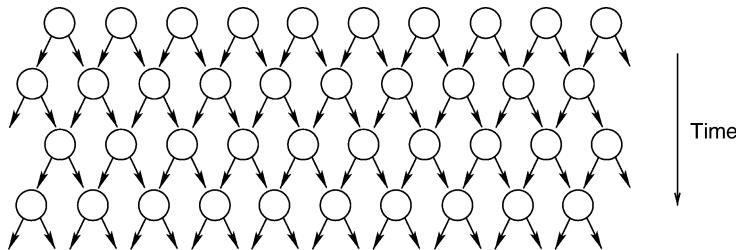
$$c(\mathbf{x}) = c(\mathbf{x} + t_i \mathbf{e}_i)$$

for every $\mathbf{x} \in \mathbb{Z}^d$ and every $i = 1, 2, \dots, d$. Let us denote by $\mathfrak{P}(d, S)$, or briefly \mathfrak{P} , the set of spatially periodic configurations. CA are homogeneous in space and consequently they preserve the spatial periodicity of configurations. The restriction G_P of G on \mathfrak{P} is hence a function $\mathfrak{P} \rightarrow \mathfrak{P}$.

Both \mathfrak{F}_q and \mathfrak{P} are dense in $S^{\mathbb{Z}^d}$, under the Cantor topology discussed in Sect. 2.5 below. Finite and periodic configurations are used in effective simulations of CA on computers.

Fig. 2

Dependencies in one-dimensional, radius- $\frac{1}{2}$ cellular automaton (CA).



Periodic configurations are often referred to as the periodic boundary conditions on a finite cellular array. For example, in the case $d = 2$, this is equivalent to running the CA on a torus that is obtained by “gluing” together the opposite sides of a rectangle. One should, however, keep in mind that the behavior of a CA can be quite different on finite, periodic, and general configurations, so experiments done with periodic boundary conditions may be misleading. Note that spatially periodic configurations are all temporally eventually periodic.

2.4 Elementary Cellular Automata

One-dimensional CA with state set $\{0, 1\}$ and the neighborhood $(-1, 0, 1)$ are called *elementary*. There are $2^3 = 8$ possible patterns inside the neighborhood of a cell, each of which may be mapped into 0 or 1. Hence, there are $2^8 = 256$ different elementary CA. Some of them are identical up to renaming the states or reversing right and left, so the number of essentially different elementary rules is smaller, only 88.

Elementary rules were extensively studied and empirically classified by Wolfram (1986). He introduced the following naming scheme: Each elementary rule is specified by an eight-bit sequence

$$f(111) \ f(110) \ f(101) \ f(100) \ f(011) \ f(010) \ f(001) \ f(000)$$

where f is the local update rule of the CA. The bit sequence is the binary expansion of an integer in the interval $0 \dots 255$, called the *Wolfram number* of the CA (Wolfram 1986).

Example 1 Rule 110 is the elementary CA where

$$\begin{aligned} f(111) &= 0, & f(110) &= 1, & f(101) &= 1, & f(100) &= 0 \\ f(011) &= 1, & f(010) &= 1, & f(001) &= 1, & f(000) &= 0 \end{aligned}$$

The name is obtained from the binary expansion $110 = (01101110)_2$. This rule will be used as a running example in the following sections.

2.5 Topology and CA Dynamics

A seminal paper in the dynamical systems approach to CA is by Hedlund (1969). In this paper, one-dimensional CA are studied as endomorphisms of the shift dynamical system, that is, as shift-commuting continuous transformations of the configuration space.

A metric on the configuration space $S^{\mathbb{Z}^d}$ is first defined. The distance between configurations c and e is

$$d(e, c) = \begin{cases} 0, & \text{if } e = c \\ 2^{-\min\{\|\mathbf{x}\|_\infty \mid c(\mathbf{x}) \neq e(\mathbf{x})\}}, & \text{if } e \neq c \end{cases}$$

where

$$\|(x_1, x_2, \dots, x_d)\|_\infty = \max\{|x_1|, |x_2|, \dots, |x_d|\}$$

It is easy to see that this function $d(\cdot, \cdot)$ is indeed a metric (even ultrametric) on $S^{\mathbb{Z}^d}$. Under this metric, two configurations that agree with each other on a large area around the origin are close to each other. The topology arising from metric $d(\cdot, \cdot)$ is the Cantor topology.

Next, a useful basis for the topology is built. Let $c \in S^{\mathbb{Z}^d}$ be an arbitrary configuration and let $D \subseteq \mathbb{Z}^d$ be a finite set of cells. The *cylinder*

$$\text{Cyl}(c, D) = \{e \in S^{\mathbb{Z}^d} \mid e(\mathbf{x}) = c(\mathbf{x}) \text{ for all } \mathbf{x} \in D\}$$

determined by c and D contains all those configurations that agree with c inside domain D .

Note that all open balls under the metric are cylinders, and that every cylinder is a union of open balls. Consequently, the cylinders form a basis of the topology. The cylinders with the same domain D form a finite partitioning of $S^{\mathbb{Z}^d}$, so all cylinders are also closed. Hence cylinders are “clopen” (closed and opened), and the space $S^{\mathbb{Z}^d}$ has a clopen basis, that is, the space is zero-dimensional. Clopen sets are exactly the finite unions of cylinders.

Another way to understand the topology is to consider convergence of sequences. Under this topology, a sequence c_1, c_2, \dots of configurations converges to $c \in S^{\mathbb{Z}^d}$ iff for every $\mathbf{x} \in \mathbb{Z}^d$ there exists k such that $c_i(\mathbf{x}) = c(\mathbf{x})$ for all $i \geq k$. In other words, the state in every cell eventually stabilizes in the sequence. A standard argumentation (as in the proof of the weak König’s lemma stating that every infinite binary tree has an infinite branch) shows that every sequence of configurations has a converging subsequence. But this simply means that $S^{\mathbb{Z}^d}$ is compact. Compactness further implies that $S^{\mathbb{Z}^d}$ is a Baire space and complete.

Let $G : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a CA function, and let c_1, c_2, \dots be a converging sequence of configurations, with limit c . Due to convergence, the states in the neighborhood of any cell $\mathbf{x} \in \mathbb{Z}^d$ stabilize, so in the sequence $G(c_1), G(c_2), \dots$ the state of cell \mathbf{x} eventually stabilizes to value $G(c)(\mathbf{x})$. The sequence $G(c_1), G(c_2), \dots$ hence converges to $G(c)$. But this property simply states that G is continuous.

The following properties have been established:

Proposition 1 *The set $S^{\mathbb{Z}^d}$ is a compact metric space, and cylinders form a countable clopen basis. Every CA function $G : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ is continuous.*

Pairs (X, F) where X is compact and $F : X \rightarrow X$ is continuous are commonly called (topological) dynamical systems.

The continuity of G simply reflects the fact that all cells apply a local rule to obtain the new state. On the other hand, the fact that all CA functions G commute with translations τ (i.e., $G \circ \tau = \tau \circ G$) means that the local rule is the same at all cells. The Curtis–Hedlund–Lyndon theorem from 1969 shows that these two properties exactly characterize CA functions:

Theorem 1 (Hedlund 1969) *Function $G : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ is a CA function if and only if it is continuous and it commutes with translations.*

The proof is based on the fact that continuous functions on compact spaces are uniformly continuous. The following corollary of the theorem states that reversible CA are exactly the automorphisms of the full shift:

Corollary 1 (Hedlund 1969) *A cellular automaton function G is reversible if and only if it is a bijection.*

Proof By definition, a reversible CA function is bijective. Conversely, suppose that G is a bijective CA. The inverse of a continuous bijection between compact metric spaces is continuous, so G^{-1} is continuous. Trivially, G^{-1} commutes with translations, so, by [Theorem 1](#), the inverse G^{-1} is a CA function.

3 Classical Results

In addition to [Theorem 1](#), several significant combinatorial results were proved in the 1960s. These include the balance in surjective CA, and the Garden-of-Eden theorem by Moore and Myhill.

3.1 Balance in Surjective CA

A configuration c is a *Garden-of-Eden configuration* if it has no pre-images, that is, if $G^{-1}(c)$ is empty. A CA has Garden-of-Eden configurations iff the CA is not surjective.

Example 2 Consider the elementary CA number 110 from [Example 1](#). Among eight possible neighborhood patterns there are three that are mapped to state 0 and five that are mapped to state 1. It is demonstrated below how this imbalance automatically implies that there are Garden-of-Eden configurations.

Let k be a positive integer, and consider a configuration c in which

$$c(3) = c(6) = \dots = c(3k) = 0.$$

See [Fig. 3](#) for an illustration. There are

$$2^{2(k-1)} = 4^{k-1}$$

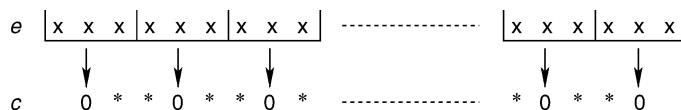
possible choices for the missing states between 0s in c (shown as “*” in [Fig. 3](#)).

In a pre-image e of c , the three state segments $e(3i-1)$, $e(3i)$, and $e(3i+1)$ are mapped into state 0 by the local rule f , for every $i = 1, 2, \dots, k$. Since $|f^{-1}(0)| = 3$, there are exactly 3^k choices of these segments. If k is sufficiently large then $3^k < 4^{k-1}$. This means that some choice of c does not have a corresponding pre-image e . Therefore, the CA is not surjective.

Alternatively, one could show the non-surjectivity of rule 110 by directly verifying that any configuration containing pattern 01010 is a Garden-of-Eden configuration.

Fig. 3

Illustration of the configuration c and its pre-image e in [Example 2](#).



In this section, the previous example is generalized. The concept of a (finite) pattern is defined first. A *pattern*

$$p = (D, g)$$

is a partial configuration where $D \subseteq \mathbb{Z}^d$ is the *domain* of p and $g: D \rightarrow S$ is a mapping assigning a state to each cell in the domain. Pattern p is *finite* if D is a finite set. The cylinder determined by finite pattern $p = (D, g)$ is the set of configurations c such that $c(\mathbf{x}) = g(\mathbf{x})$ for all $\mathbf{x} \in D$.

If $N = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$ is a neighborhood vector, the neighborhood of domain $D \subseteq \mathbb{Z}^d$ is defined as the set

$$N(D) = \{\mathbf{x} + \mathbf{x}_i \mid \mathbf{x} \in D, i = 1, 2, \dots, m\}$$

of the neighbors of elements in D .

Let G be a CA function specified by the quadruple (d, S, N, f) . Let $q = (E, h)$ be a pattern, and let $D \subseteq \mathbb{Z}^d$ be a domain such that $N(D) \subseteq E$. An application of the local rule f on pattern q determines new states for all cells in domain D . A pattern $p = (D, g)$ is obtained where for all $\mathbf{x} \in D$

$$g(\mathbf{x}) = f[h(\mathbf{x} + \mathbf{x}_1), h(\mathbf{x} + \mathbf{x}_2), \dots, h(\mathbf{x} + \mathbf{x}_m)]$$

The mapping $q \mapsto p$ will be denoted by $G^{(E \rightarrow D)}$, or simply by G when the domains E and D are clear from the context. With this notation, the global transition function of the CA is $G^{(\mathbb{Z}^d \rightarrow \mathbb{Z}^d)}$.

A finite pattern $p = (D, g)$ is called an *orphan* if there is no finite pattern q with domain $N(D)$ such that $G^{(N(D) \rightarrow D)}(q) = p$. In other words, orphan is a pattern that cannot appear in any configuration after an application of the CA. For example, pattern 01010 is an orphan for rule 110.

Proposition 2 *A CA has an orphan if and only if it is non-surjective.*

Proof It is clear that if an orphan exists then the CA is not surjective: Every configuration that contains a copy of the orphan is a Garden-of-Eden. To prove the converse, the compactness of the configuration space is applied (see [Proposition 1](#)). If G is not surjective then $G(S^{\mathbb{Z}^d}) \subsetneq S^{\mathbb{Z}^d}$ is closed, so its complement (the set of Garden-of-Eden configurations) is open and nonempty. As cylinders form a basis of the topology, there is a cylinder all of whose elements are Garden-of-Eden configurations. The finite pattern that specifies the cylinder is then an orphan.

The following balance theorem generalizes [Example 2](#). The result was proved in the one-dimensional case already in Hedlund (1969), but it holds in any dimension d (Maruoka and Kimura 1976). The theorem states that in surjective CA, all finite patterns of the same domain D must have the same number of pre-image patterns of domain $N(D)$. Any imbalance immediately implies non-surjectivity. As a special case, it can be seen that the local rule of a surjective CA must be balanced: each state appears in the rule table equally many times.

Theorem 2 *Let $A = (d, S, N, f)$ be a surjective CA, and let $E, D \subseteq \mathbb{Z}^d$ be finite domains such that $N(D) \subseteq E$. Then, for every pattern $p = (D, g)$ the number of patterns $q = (E, h)$ such that*

$$G^{(E \rightarrow D)}(q) = p$$

is $s^{|E| - |D|}$ where $s = |S|$ is the number of states.

Proof (sketch) The theorem can be proved analogously to [Example 2](#), by considering imbalanced mappings from d -dimensional hypercube patterns of size n^d , into hypercubes of size $(n - 2r)^d$ where r is the neighborhood radius of the CA. Considering an arrangement of k^d such hypercubes (see [Fig. 4](#)) one can deduce an orphan as in [Example 2](#), using the following technical Lemma.

Lemma 1 For all $d, n, s, r \in \mathbb{Z}_+$ the inequality

$$\left(s^{n^d} - 1\right)^{k^d} < s^{(kn - 2r)^d}$$

holds for all sufficiently large k .

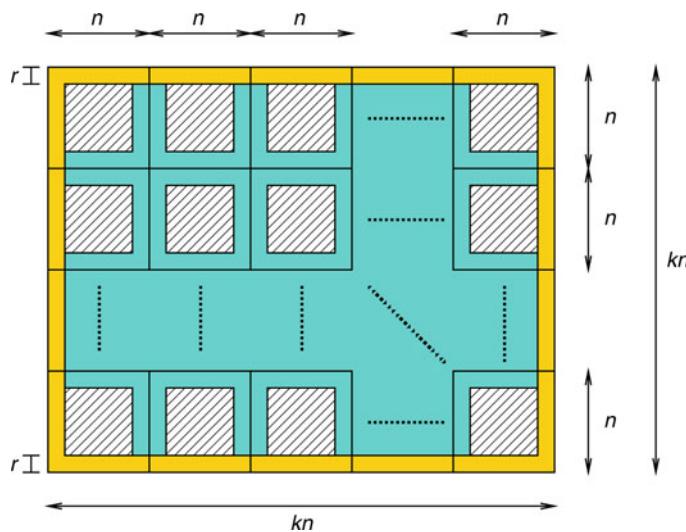
Example 3 The balance condition of the local rule table is necessary but not sufficient for surjectivity. Consider, for example, the elementary CA number 232. It is the *majority* CA: $f(a, b, c) = 1$ if and only if $a + b + c \geq 2$. Its rule table is balanced because 000, 001, 010, 100 map to 0, and 111, 110, 101, 011 map to 1.

However, the majority CA is not balanced on longer patterns and hence it is not surjective: Any word of length 4 that contains at most one state 1 is mapped to 00, so 00 has at least five pre-images of length 4. Balanceness would require this number of pre-images to be four. Pattern 01001 is a particular example of an orphan.

[Theorem 2](#) actually states the fact that the uniform Bernoulli measure on the configuration space is invariant under applications of surjective CA: Randomly (under the uniform probability distribution) picked configurations remain random under iterations of surjective CA.

Fig. 4

Illustration for the proofs of [Theorems 2](#), [3](#) and [4](#).



3.2 Garden-of-Eden Theorem

Two configurations c and d are *asymptotic* if they differ only in finitely many cells, that is, if

$$\text{diff}(c, d) = \{\mathbf{x} \in \mathbb{Z}^d \mid c(\mathbf{x}) \neq d(\mathbf{x})\}$$

is finite. CA G is called *pre-injective* if $G(c) \neq G(d)$ for any asymptotic c, d such that $c \neq d$.

Let $s \in S$ be an arbitrary state. Note that CA C is pre-injective iff it is injective among s -finite configurations. Among the earliest discovered properties of CA is the Garden-of-Eden theorem by Moore and Myhill from 1962 and 1963, respectively, which proves that pre-injectivity is equivalent to surjectivity.

Example 4 As in [Section 3.1](#), to start with, an illustration of the proof of the Garden-of-Eden theorem using elementary rule 110 is given. It demonstrates how non-surjectivity of rule 110 implies that it is not pre-injective.

By [Example 2](#), rule 110 is not surjective. In fact, finite pattern 01010 is an orphan. It is shown in the following that there must exist different 0-finite configurations c and e such that $G(c) = G(e)$.

Let $k \in \mathbb{Z}_+$ be arbitrary. Consider 0-finite configurations c whose supports are included in a fixed segment of length $5k - 2$, see [Fig. 5](#). There are

$$2^{5k-2} = 32^k/4$$

such configurations. The support of $G(c)$ is included in a segment of length $5k$. Partition this segment in k subsegments of length 5. It is known that pattern 01010 cannot appear anywhere in $G(c)$, so there are at most $2^5 - 1 = 31$ different patterns that can appear in the length 5 subsegments. Hence, there are at most 31^k possible configurations $G(c)$. For all sufficiently large values of k , one has

$$32^k/4 > 31^k$$

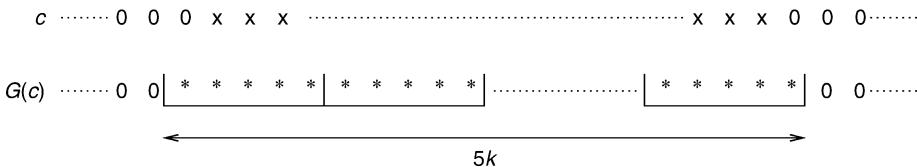
so there must be two 0-finite configurations with the same image.

Theorem 3 (Myhill 1963) *If G is pre-injective then G is surjective.*

Proof (sketch) The theorem can be proved along the lines of [Example 4](#). If G is not surjective, there is an orphan whose domain is a d -dimensional hypercube of size n^d , for some n . Arranging k^d copies of the hypercube as in [Fig. 4](#), one comes up with a domain in which at most $(s^{n^d} - 1)^{k^d}$ different non-orphans exist. Using [Lemma 1](#) one can now easily conclude the existence of two different asymptotic configurations with the same image.

Fig. 5

Illustration of [Example 4](#).



Injectivity trivially implies pre-injectivity, and therefore the following corollary is obtained.

Corollary 2 *Every injective CA is also surjective. Injectivity is hence equivalent to bijectivity and reversibility.*

The other direction of the Garden-of-Eden theorem is considered next. Again, one starts with a one-dimensional example that indicates the proof idea.

Example 5 Consider again rule 110. The asymptotic configurations

$$c_1 = \dots 000011010000 \dots$$

$$c_2 = \dots 000010110000 \dots$$

have the same image. In the following it is demonstrated how this implies that rule 110 is not surjective. (Of course, this fact is already known from prior examples.)

Extract patterns $p_1 = 011010$ and $p_2 = 010110$ of length 6 from c_1 and c_2 , respectively. Both patterns are mapped into the same pattern 1111 of length 4. Moreover, p_1 and p_2 have a boundary of width 2 on both sides where they are identical with each other. Since rule 110 uses radius-1 neighborhood, one can replace in any configuration c pattern p_1 by p_2 or vice versa without affecting $G(c)$.

Let $k \in \mathbb{Z}_+$, and consider a segment of $6k$ cells. It consists of k segments of length 6. Any pattern of length $6k - 2$ that has a pre-image of length $6k$ also has a pre-image where none of the k subsegments of length 6 contains pattern p_2 . Namely, all such p_2 can be replaced by p_1 . This means that at most $(2^6 - 1)^k = 63^k$ patterns of length $6k - 2$ can have pre-images. On the other hand, there are $2^{6k-2} = 64^k/4$ such patterns, and for large values of k

$$64^k/4 > 63^k$$

so some patterns do not have a pre-image.

Theorem 4 (Moore 1962) *If G is surjective then G is pre-injective.*

Proof (sketch) The proof of the theorem is as in [Example 5](#). If d -dimensional CA G is not pre-injective, there are two different patterns p_1 and p_2 with the same image such that the domain of the patterns is the same hypercube of size n^d , and the patterns are identical on the boundary of width r of the hypercube, where r is now twice the neighborhood radius of the CA. In any configuration, copies of pattern p_2 can be replaced by p_1 without affecting the image under G . Arranging k^d copies of the hypercubes as in [Fig. 4](#), one comes up with a domain in which at most $(s^{n^d} - 1)^{k^d}$ patterns have different images. Using [Lemma 1](#) again one sees that this number is not large enough to provide all possible interior patterns, and an orphan must exist.

4 Injectivity and Surjectivity Properties

The Garden-of-Eden theorem links surjectivity of CA to injectivity on finite configurations. The surjectivity and injectivity properties on $S^{\mathbb{Z}^d}$, on finite and on periodic configurations, are next related to each other, as reported in Durand ([1998](#)). On periodic configurations, the

situation is slightly different in the one-dimensional case than in the higher-dimensional cases. The following implications are easily seen to be valid, however, regardless of the dimension:

- If G is injective then G_F and G_P are also injective (trivial).
- If G_F or G_P is surjective then G is also surjective (due to denseness of \mathfrak{F}_q and \mathfrak{P} in $S^{\mathbb{Z}^d}$).
- If G_P is injective then G_P is surjective (because the number of periodic configurations with given fixed periods is finite).
- If G is injective then G_F is surjective (due to reversibility and the stability of the quiescent state).

4.1 The One-Dimensional Case

In this subsection, we concentrate on one-dimensional CA. The balance theorem of surjective CA has the following interesting corollary that is valid in the one-dimensional case only.

Theorem 5 (Hedlund 1969) *For every one-dimensional surjective CA, there is a constant m such that every configuration has at most m pre-images.*

Proof Let G be a one-dimensional surjective CA function defined using radius- r neighborhood. Let $s = |\mathcal{S}|$ be the number of states. In the following, it is proved that every configuration has at most s^{2r} different pre-images.

Suppose the contrary: there is a configuration c with $s^{2r} + 1$ different pre-images

$$e_1, e_2, \dots, e_{s^{2r}+1}$$

For some sufficiently large number $k > r$, each pair e_i and e_j of pre-images contains a difference inside the interval

$$E = \{-k, -k+1, \dots, k-1, k\}$$

But this contradicts **• Theorem 2** since one now has a pattern with domain

$$D = \{-k+r, -k+r+1, \dots, k-r\}$$

that has at least

$$s^{2r} + 1 > s^{|E|-|D|}$$

different pre-images in domain E .

Corollary 3 (Hedlund 1969) *Let G be a one-dimensional surjective CA function. The pre-images of spatially periodic configurations are all spatially periodic. In particular, G_P is surjective.*

Proof Suppose $G(c)$ is spatially periodic, so that $\sigma^n(G(c)) = G(c)$ for some $n \in \mathbb{Z}_+$. Then for every $i \in \mathbb{Z}$

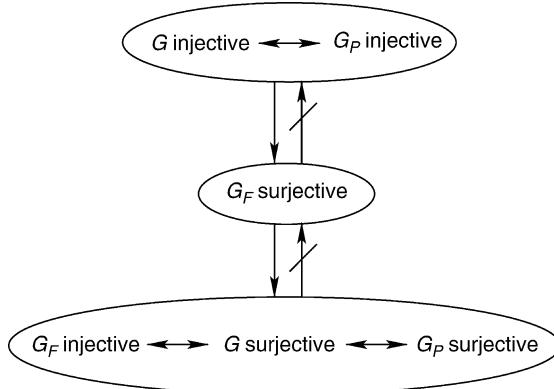
$$G(\sigma^{in}(c)) = \sigma^{in}(G(c)) = G(c)$$

so $\sigma^{in}(c)$ is a pre-image of $G(c)$. By **• Theorem 5**, configuration $G(c)$ has a finite number of pre-images so

$$\sigma^{i_1 n}(c) = \sigma^{i_2 n}(c)$$

Fig. 6

Implications between injectivity and surjectivity properties in one-dimensional CA.



for some $i_1 < i_2$. But then, c is periodic with period $(i_2 - i_1)n$. It follows that all pre-images of periodic configurations are periodic.

In the one-dimensional setting, the injectivity of G_P implies that G is injective. This can be seen easily, for example, using the de Bruijn graph technique (Sutner 1991). Combining the results of this and the previous sections, the positive implications in [Fig. 6](#) are obtained. The following two examples confirm the two negative implications in the figure.

Example 6 The XOR automaton is the elementary CA rule 102. Every cell adds together (mod 2) its state and the state of its right neighbor. This CA is surjective, but it is not surjective on 0-finite configurations because the only two pre-images of the finite configuration $\dots 001000\dots$ are the non-finite configurations $\dots 000111\dots$ and $\dots 111000\dots$.

Example 7 CONTROLLED-XOR is a one-dimensional radius- $\frac{1}{2}$ CA. It has four states 00, 01, 10, and 11. The first bit of each state is a control symbol that does not change. If the control symbol of a cell is 0, then the cell is inactive and does not change its state. If the control symbol is 1, then the cell is active and applies the XOR rule of [Example 6](#) on the second bit.

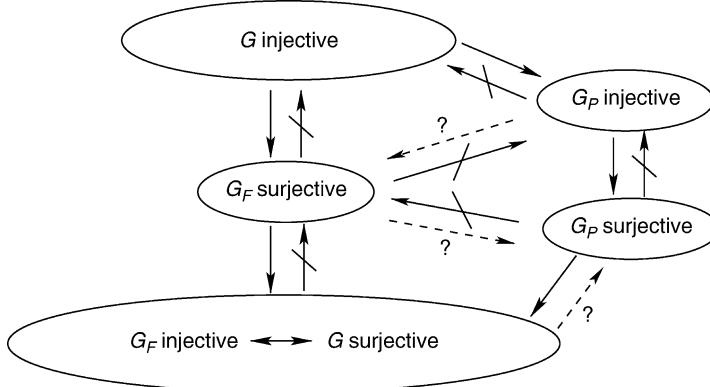
State 00 is the quiescent state. CONTROLLED-XOR is easily seen to be surjective on finite configurations. It is not injective on unrestricted configurations as two configurations, all of whose cells are active, have the same image if their second bits are complements of each other.

4.2 The Higher-Dimensional Cases

In the two- and higher-dimensional cases, injectivity on periodic configurations is no longer equivalent to general injectivity, and surjectivity on periodic configurations is not known to be equivalent to surjectivity. [Figure 7](#) summarizes the currently known implications (Durand 1998). Note that in three cases the relation is unknown.

Fig. 7

Implications between injectivity and surjectivity properties in two- and higher-dimensional CA.



The difference in the one- and two-dimensional diagrams can be explained using tilings. Plane tilings are also useful in obtaining undecidability results for two-dimensional CA, as will be explained in [Sect. 5](#). First, d -dimensional tilings are defined here using local matching rules. The definition closely resembles CA, the only difference being that tilings are static objects unlike dynamic CA. In symbolic dynamics terminology, tilings are d -dimensional subshifts of finite type.

A d -dimensional tile set $\mathfrak{T} = (d, T, N, R)$ consists of a finite set T whose elements are the tiles, a d -dimensional neighborhood vector N of size m (as defined in Sect. 2.1), and a local matching rule $R \subseteq T^m$ that gives a relation specifying which tilings are considered valid. Tilings are configurations $t \in T^{\mathbb{Z}^d}$. Configuration t is valid at cell $\mathbf{x} \in \mathbb{Z}^d$ iff

$$[t(\mathbf{x} + \mathbf{x}_1), t(\mathbf{x} + \mathbf{x}_2), \dots, t(\mathbf{x} + \mathbf{x}_m)] \in R$$

that is, the neighborhood of \mathbf{x} contains a matching combination of tiles. Configuration $t \in T^{\mathbb{Z}^d}$ is a *valid tiling* if it is valid at all positions $\mathbf{x} \in \mathbb{Z}^d$, and it is said that the tile set \mathfrak{T} then *admits tiling* t .

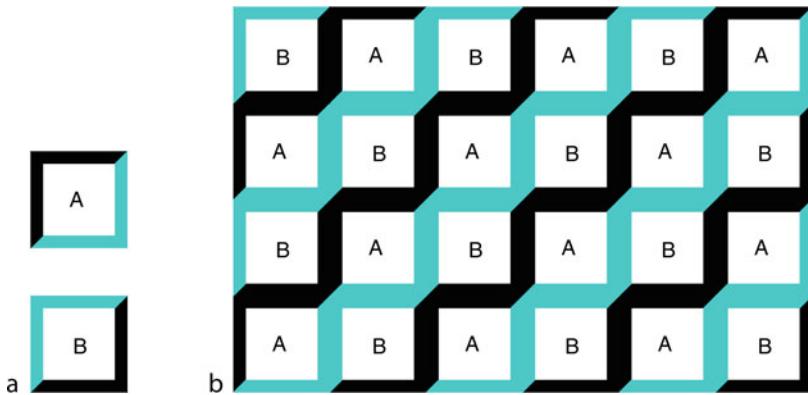
There is an apparent similarity in the definitions of tile sets and CA. The only difference is that instead of a dynamic local rule f , tilings are based on a static matching relation R .

A fundamental property of CA is that they commute with translations. The tiling counterpart states the obvious fact that $\tau(t)$ is a valid tiling for every valid tiling t and every translation τ . The second fundamental property of CA is the continuity of CA functions. The tiling counterpart of this fact states that the set of valid tilings is closed in the Cantor topology, that is, the limit of a converging sequence of valid tilings is also valid. Proofs of these facts are straightforward. Compactness also directly implies that a tile set that can properly tile arbitrarily large hypercubes admits a tiling of the whole space.

A particularly interesting case is $d = 2$ because two-dimensional tilings can draw computations of Turing machines, leading to undecidability results (Wang 1961). A convenient way to describe a two-dimensional tile set is in terms of *Wang tiles*. Wang tiles use the von Neumann neighborhood. The tiles are viewed as unit squares whose edges are colored, and the local matching rule is given in terms of these colors: A tiling is valid at cell $\mathbf{x} \in \mathbb{Z}^2$ iff each of the four edges of the tile in position \mathbf{x} has the same color as the abutting edge in the adjacent tiles.

Fig. 8

(a) Two Wang tiles, and (b) part of a valid tiling.



Example 8 Consider the two Wang-tiles A and B shown in [Fig. 8a](#). Since all four neighbors of tile A have to be copies of B , and vice versa, the only valid tilings are infinite checkerboards where A 's and B 's alternate, as shown in [Fig. 8b](#).

Tiling $t \in T^{\mathbb{Z}^d}$ is *periodic* if it is spatially periodic, as defined in [Sect. 2.3](#). In the two-dimensional case, this simply means that the tiling consists of a horizontally and vertically repeating rectangular pattern. An interesting fact is that there exist two-dimensional tile sets that only allow nonperiodic tilings. A tile set is called *aperiodic* if

1. It admits some valid tilings, but
2. It does not admit any valid periodic tilings.

The existence of such aperiodic tile sets was demonstrated by Berger in 1966.

Theorem 6 (Berger 1966) *There exist aperiodic sets of Wang tiles.*

In the following, a particular aperiodic tile set SNAKES from Kari (1994) is used to construct an example of a non-injective two-dimensional CA that is injective on periodic configurations. The tiles SNAKES have an arrow printed on them. The arrow is horizontal or vertical and it points to one of the four neighbors of the tile. Given any tiling t , valid or invalid, the arrows determine paths, obtained by following the arrows printed on the tiles. The tile that follows tile (x, y) on a path is the von Neumann neighbor of (x, y) in the direction indicated by the arrow on $t(x, y)$.

The tile set SNAKES of Kari (1994) has the following plane-filling property: Consider a tiling t and a path P that indefinitely follows the arrows as discussed above. If the tiling is valid at all tiles that P visits, then the path necessarily covers arbitrarily large squares. In other words, for every $N \geq 1$ there is a square of $N \times N$ tiles on the plane, all of whose tiles are visited by path P . Note that the tiling may be invalid outside path P , yet the path is forced to snake through larger and larger squares. In fact, SNAKES forces the paths to follow the well-known Hilbert curve (Kari 1994).

Example 9 Using SNAKES, the two-dimensional SNAKE-XOR CA is constructed in the style of the one-dimensional CONTROLLED-XOR of [Example 7](#). The states consist of two layers: a

control layer and a xor layer. The control layer does not change: it only indicates which cells are active and which neighbor cell provides the bit to the XOR operation. In SNAKE-XOR, the control layer consists of SNAKES tiles discussed above. Only cells where the tiling on the control layer is valid are active. Active cells execute addition (mod 2) on their xor layer. The arrow of the tile tells which neighbor provides the second bit to the operation.

SNAKE-XOR is not injective: Two configurations c_0 and c_1 whose control layer consist of the same valid tiling have the same image if their xor layers are complementary to each other. However, SNAKE-XOR is injective on periodic configurations, as the plane-filling property ensures that on periodic configurations any infinite path that follows the arrows must contain nonactive cells.

5 Algorithmic Questions

Many relevant algorithmic questions can be formulated. It would be nice to have algorithms to determine, for example, whether a given CA is injective, surjective, nilpotent, etc. The one-dimensional case is usually algorithmically more accessible than higher-dimensional cases.

Theorem 7 (Amoroso and Patt 1972) *There exist algorithms to determine if a given one-dimensional CA is injective, or if it is surjective.*

Elegant polynomial-time decision algorithms can be based on de Bruijn graphs (Sutner 1991).

Already in the two-dimensional case, the injectivity and the surjectivity questions turn out to be undecidable. For injectivity, the proof is easily based on the SNAKES-tiles of [Sect. 4.2](#), and the following classical result establishing the undecidability of the tiling problem:

Theorem 8 (Berger 1966) *It is undecidable if a given set of Wang tiles admits a valid tiling.*

The tiling problem can be reduced to the question concerning the injectivity of two-dimensional CA. Also, a proof in a similar style (but using a simpler variant of the tiling problem, known as the finite tiling problem) shows the undecidability of injectivity on finite configurations and, hence, by the Garden-of-Eden theorem, of surjectivity.

Theorem 9 (Kari 1994) *It is undecidable if a given two-dimensional cellular is injective. It is also undecidable whether it is surjective.*

Proof (sketch for injectivity) A reduction of the tiling problem to the injectivity question goes as follows. For a given set T of Wang tiles one effectively constructs a two-dimensional CA, similar to SNAKE-XOR of [Example 9](#). The CA has a control layer and a xor layer. The control layer in turn consists of two layers: one with tiles T and one with tiles SNAKES. A cell is active if and only if the tiling is valid at the cell on both tile components. Active cells execute the addition (mod 2) on their xor layer, and the arrow on the SNAKES tile tells which neighbor provides the second bit to the sum. The plane-filling property of SNAKES guarantees that if two different configurations have the same successor then arbitrarily large squares must have a valid tiling. Conversely, if a valid tiling exists then two different configurations with identical control layers can have the same successor. Hence the CA that is constructed is injective iff T does not admit a valid tiling, and this completes the proof.

Note the following fundamental difference in the injectivity problems of G and G_F : A semi-algorithm exists for the injectivity of G (based on an exhaustive search for the inverse CA) and for the non-injectivity of G_F (based on looking for two finite configurations with the same image).

Even though [Corollary 1](#) guarantees that the inverse function of every injective CA is a CA, [Theorem 9](#) implies that the neighborhood of the inverse CA can be very large: there can be no computable upper bound, as otherwise all candidate inverses could be tested one by one. In contrast, in the one-dimensional case, the inverse automaton can only have a relatively small neighborhood. In one-dimensional radius- $\frac{1}{2}$ CA, the inverse neighborhood consists of at most $s - 1$ consecutive cells where s is the number of states (Czeizler and Kari [2005](#)).

Analogously, [Theorem 9](#) implies that the size of the smallest orphan of a non-surjective CA has no computable upper bound. In the one-dimensional case, however, a polynomial upper bound for the length of the shortest orphan was recently reported (Kari et al. [2009](#)).

While injectivity and surjectivity are single time step properties, algorithmic questions on long-term dynamical properties have also been studied. Many such properties turn out to be undecidable already among one-dimensional CA. An explanation is that the space-time diagrams are two-dimensional and can be viewed as tilings.

Recall from [Sect. 2.1](#) that a CA G is called nilpotent if it has trivial dynamics: for some n , the n th iterate $G^n(c)$ of every configuration is quiescent. It turns out that there is no algorithm to determine if a given one-dimensional CA is nilpotent Kari ([1992](#)). This is true even if the quiescent state q is spreading, that is, if $f(a_1, a_2, \dots, a_m) \neq q$ implies that all $a_i \neq q$.

A reversible CA is called *periodic* if G^n is the identity function for some $n \geq 1$. Recently, it was shown that the periodicity of one-dimensional CA is undecidable (Kari and Ollinger [2008](#)).

Theorem 10 (Kari [1992](#); Kari and Ollinger [2008](#)) *It is undecidable if a given one-dimensional CA (with a spreading state) is nilpotent. It is also undecidable if a given reversible one-dimensional CA is periodic.*

Nilpotency is a simple property, and as such it can be reduced to many other dynamical properties. Some are indicated in [Sect. 6](#) below.

6 Dynamical Systems Concepts

As advertised in [Sect. 2.5](#), CA are rich examples of dynamical systems in the sense of topological dynamics. Chaotic behavior in dynamical systems means sensitivity to small perturbations in the initial state, as well as mixing of the configuration space. Sensitivity and mixing in CA under the Cantor topology have been investigated and related to each other.

Recall the metric defined in [Sect. 2.5](#). Observing a configuration approximately under this metric simply means that states inside a finite window can be seen, but the observer has no information on the states outside the window. Better precision in the observation just means a larger observation window.

6.1 Sensitivity

Informally, a dynamical system is called sensitive to initial conditions if small changes in the state of the system get magnified during the evolution. This means that the long-term behavior

of the system by simulation cannot be predicted unless the initial state is known precisely. In contrast, configurations whose orbits are arbitrarily well tracked by all sufficiently close configurations are called equicontinuous. Such orbits can be reliably simulated.

Precisely speaking, configuration $c \in S^{\mathbb{Z}^d}$ is an *equicontinuity point* for CA G if for every finite observation window $E \subseteq \mathbb{Z}^d$ there corresponds a finite domain $D \subseteq \mathbb{Z}^d$ such that for every $e \in \text{Cyl}(c, D)$ one has $G^n(e) \in \text{Cyl}(G^n(c), E)$ for all $n = 1, 2, 3, \dots$. In other words, no difference is ever seen inside the observation window E if states of c are changed outside of D . Let \mathcal{E}_G denote the set of equicontinuity points of G .

CA G is called *equicontinuous* if all configurations are equicontinuous, that is, if $\mathcal{E}_G = S^{\mathbb{Z}^d}$. It turns out that only eventually periodic CA are equicontinuous.

Theorem 11 (Blanchard and Tisseur; Kůrka 1997) A CA G is equicontinuous if and only if $G^{m+p} = G^m$ for some $p \geq 1$ and $m \geq 0$. A surjective CA is equicontinuous if and only if it is periodic.

It is undecidable to decide if a given one-dimensional CA is equicontinuous (Durand et al. 2003). In fact, equicontinuity is undecidable even among reversible CA (► Theorem 10).

CA G is *sensitive to initial conditions* (or simply *sensitive*) if there is a finite observation window $E \subseteq \mathbb{Z}^d$ such that for every configuration c and every domain D , there exists a configuration $e \in \text{Cyl}(c, D)$ such that $G^n(e) \notin \text{Cyl}(G^n(c), E)$, for some $n \geq 1$. In simple terms, any configuration can be changed arbitrarily far in such a way that the change propagates into the observation window E .

Example 10 Any nonzero translation τ is sensitive to initial conditions. The XOR-automaton of ► Example 6 is another example: For any configuration c and every positive integer n , a change in cell n propagates to cell 0 in n steps.

It is easy to see that a sensitive CA cannot have any equicontinuity points. In the one-dimensional case also the converse holds.

Theorem 12 (Kůrka 1997) A one-dimensional CA G is sensitive if and only if $\mathcal{E}_G = \emptyset$. If G is not sensitive then \mathcal{E}_G is a residual set (and hence dense).

The previous theorem is not valid in the two-dimensional case (Sablik and Theyssier 2008). The difference stems from the fact that equicontinuity in the one-dimensional case is characterized by the presence of *blocking words* (words that do not let information pass between their left and right sides), while in the two-dimensional case information can circumvent any finite block.

A very strong form of sensitivity is (*positive*) *expansivity*. In positively expansive CA, every small change to any configuration propagates everywhere. More precisely, there exists a finite observation window $E \subseteq \mathbb{Z}^d$ such that for any distinct configurations c and e there exists time $n \geq 0$ such that configurations $G^n(c)$ and $G^n(e)$ differ in some cell in the window E .

Example 11 The XOR automaton of ► Example 6 is not positively expansive because differences only propagate to the left, but not to the right. The three neighbor XOR (elementary CA 150) is positively expansive since differences propagate to both directions.

Positively expansive CA are clearly surjective and sensitive but not injective. It turns out that in the two- and higher-dimensional spaces, there are no positively expansive CA (Shereshevsky 1993). It is not known whether there exists an algorithm to determine if a given CA is positively expansive, and this remains an interesting open problem. However, sensitivity is known to be undecidable (Durand et al. 2003), even among reversible one-dimensional CA (Lukkarila 2010).

6.2 Mixing Properties

Mixing of the configuration space is another property associated to chaos. Like sensitivity, mixing also comes in different variants. CA G is called *transitive* if for all cylinders U and V there exists $n \geq 0$ such that $G^n(U) \cap V \neq \emptyset$. Clearly transitive CA need to be surjective.

Example 12 The XOR automaton of [Example 6](#) is easily seen to be transitive. For any fixed word w of length n , define the function $h_w : S^n \rightarrow S^n$ that maps $u \mapsto v$ iff $G^n(\dots w u \dots) = \dots v \dots$ where v is a word of length n starting in the same position as w , only n time steps later. Mapping h_w is injective, and therefore also surjective. It can be seen that for any two cylinders, U and V with domain $\{1, 2, \dots, n\}$ holds $G^n(U) \cap V \neq \emptyset$. This is enough to prove transitivity.

The following theorem illustrates that transitivity is a stronger property than sensitivity.

Theorem 13 (Codenotti and Margara 1996) *A transitive CA (with at least two states) is sensitive to initial conditions.*

Example 13 As an example of a surjective CA that is sensitive but not transitive, consider the product $\tau \times I$ of a nonzero translation τ and the identity function I , both over the binary state set $\{0, 1\}$. The product is simply the four-state CA with two independent binary tracks, where the tracks compute τ and I , respectively. This CA is sensitive because τ is sensitive, but it is not transitive because I is not transitive.

Transitivity can equivalently be characterized in terms of dense orbits. Let us denote by

$$\mathcal{T}_G = \{c \in S^{\mathbb{Z}^d} \mid \text{the orbit } c, G(c), G^2(c), \dots \text{ is dense}\}$$

the set of *transitive points* of G , that is, those points whose orbit visits every cylinder. It is not difficult to see that a CA is transitive iff it has transitive points.

Mixing is a more restrictive concept than transitivity: CA G is called *mixing* if for all cylinders U and V there exists positive integer n such that $G^k(U) \cap V \neq \emptyset$ for all $k \geq n$. It is immediate from the definition that every mixing CA is also transitive. Both transitivity and mixing are undecidable properties, even among reversible one-dimensional CA (Lukkarila 2010).

The following theorem relates positive expansivity to the mixing properties.

Theorem 14 (Blanchard and Maass 1997) *A positively expansive CA is mixing.*

We have the following implications between different variants of sensitivity and mixing in CA:

$$\text{pos. expansive} \implies \text{mixing} \implies \text{transitive} \implies \text{sensitive}$$

It is an open question whether every transitive CA is also mixing.

Notions of sensitivity and mixing of the space are related to chaotic behavior. Devaney (1986) defines a dynamical system to be chaotic if

1. It is sensitive to initial conditions.
2. It is transitive.
3. Temporally periodic points are dense.

As transitivity implies sensitivity, conditions 2 and 3 are sufficient. It remains a challenging open problem to show that periodic orbits are dense in every surjective CA, or even in every transitive CA. If this is the case, then Devaney's chaos is equivalent to transitivity.

7 Conclusions

In this chapter, some basic definitions, classical results, decidability issues, and dynamical system aspects of CA are covered. The goal was to provide background information that makes CA literature – and in particular other chapters in this handbook – more accessible. Full proofs were not provided, but in some cases a proof idea was given. For complete proofs, see the corresponding literature references.

References

- Amoroso S, Patt Y (1972) Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *J Comput Syst Sci* 6:448–464
- Berger R (1966) The undecidability of the domino problem. *Mem Amer Math Soc* 66:1–72
- Berlekamp ER, Conway JH, Guy RK (1982) Winning ways for your mathematical plays, vol II, chap 25. Academic, London
- Blanchard F, Maass A (1997) Dynamical properties of expansive one-sided cellular automata. *Israel J Math* 99:149–174
- Blanchard F, Tisseur P (2000) Some properties of cellular automata with equicontinuity points. *Ann Inst Henri Poincaré, Probab Stat* 36(5):569–582
- Burks A (1970) Von Neumann's self-reproducing automata. In: Burks A (ed) Essays on cellular automata, University of Illinois Press, Champaign, pp 3–64
- Codenotti B, Margara L (1996) Transitive cellular automata are sensitive. *Am Math Mon* 103(1):58–62
- Cook M (2004) Universality in elementary cellular automata. *Complex Syst* 15:1–40
- Czeizler E, Kari J (2005) A tight linear bound on the neighborhood of inverse cellular automata. In: Caires L, Italiano GF, Monteiro L, Palamidessi C, Yung M (eds) ICALP, Lecture notes in computer science, vol 3580, Springer, Berlin, pp 410–420
- Devaney RL (1986) An introduction to chaotic dynamical systems. Benjamin/Cummings, Menlo Park, CA
- Durand B (1998) Global properties of cellular automata. In: Goles E, Martinez S (eds) Cellular automata and complex systems. Kluwer, Dordrecht
- Durand B, Formenti E, Varouchas G (2003) On undecidability of equicontinuity classification for cellular automata. In: Morvan M, Remila E (eds) Discrete models for complex systems, DMCS'03, DMTCS Conference Volume AB, Lyon, France, pp 117–128
- Gardner M (1970) The fantastic combinations of John Conway's new solitaire game "Life". *Sci Am* 223(4): 120–123
- Hedlund GA (1969) Endomorphisms and automorphisms of the shift dynamical system. *Math Syst Theor* 3:320–375
- Kari J (1992) The nilpotency problem of one-dimensional cellular automata. *SIAM J Comput* 21(3):571–586
- Kari J (1994) Reversibility and surjectivity problems of cellular automata. *J Comput Syst Sci* 48(1): 149–182
- Kari J, Ollinger N (2008) Periodicity and immortality in reversible computing. In: Ochmanski E, Tyszkiewicz J (eds) MFCS, Lecture notes in computer science, vol 5162, Springer, Berlin, pp 419–430
- Kari J, Vanier P, Zeume T (2009) Bounds on non-surjective cellular automata. In: Královic R, Urzyczyn P (eds) MFCS, Lecture notes in computer Science, vol 5734, Springer, Berlin, pp 439–450

- Kůrka P (1997) Languages, equicontinuity and attractors in cellular automata. *Ergod Theor Dyn Syst* 17:417–433
- Lukkarila V (2010) Sensitivity and topological mixing are undecidable for reversible one-dimensional cellular automata. *J Cell Autom* 5:241–272
- Maruoka A, Kimura M (1976) Condition for injectivity of global maps for tessellation automata. *Inform Control* 32(2):158–162. doi: 10.1016/S0019-9958(76)90195-9, URL <http://www.sciencedirect.com/science/article/B7MFM-4DX4D7R-V/2/409f935b249fa44868bb569dffb5eb37>
- Moore EF (1962) Machine models of self-reproduction. In: Bellman RE (ed) *Proceedings of symposia in applied mathematics XIV: “Mathematical problems in the biological sciences”*, AMS, pp 17–33
- Myhill J (1963) The converse of Moore’s Garden-of-Eden theorem. *Proc Am Math Soc* 14:685–686
- von Neumann J (1966) Theory of self-reproducing automata. In: Burks AW (ed) University of Illinois Press, Urbana, London
- Sablik M, Theyssier G (2008) Topological dynamics of 2D cellular automata. In: CiE ’08: Proceedings of the 4th conference on computability in Europe, Springer, Berlin, Heidelberg, pp 523–532
- Shereshevsky MA (1993) Expansiveness, entropy and polynomial growth for groups acting on subshifts by automorphisms. *Indag Math* 4(2):203–210
- Sutner K (1991) De Bruijn graphs and linear cellular automata. *Complex Syst* 5:19–31
- Wang H (1961) Proving theorems by pattern recognition - ii. *Bell Syst Tech J* 40:1–42
- Wolfram S (ed) (1986) *Theory and applications of cellular automata*. World Scientific, Singapore
- Wolfram S (2002) *A new kind of science*. Wolfram Media, Champaign, IL, USA

2 Cellular Automata Dynamical Systems

Alberto Dennunzio¹ · Enrico Formenti² · Petr Kůrka³

¹Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Italy
dennunzio@disco.unimib.it

²Département d’Informatique, Université de Nice-Sophia Antipolis,
France
enrico.formenti@unice.fr

³Center for Theoretical Studies, Academy of Sciences and
Charles University in Prague, Czechia
kurka@cts.cuni.cz

1	<i>Introduction</i>	26
2	<i>Discrete Dynamical Systems</i>	26
3	<i>Symbolic Dynamical Systems</i>	28
4	<i>One-Dimensional Cellular Automata (1D CAs)</i>	30
5	<i>From 1D CA to 2D CA</i>	64

Abstract

We present recent studies on cellular automata (CAs) viewed as discrete dynamical systems. In the first part, we illustrate the relations between two important notions: subshift attractors and signal subshifts, measure attractors and particle weight functions. The second part of the chapter considers some operations on the space of one-dimensional CA configurations, namely, shifting and lifting, showing that they conserve many dynamical properties while reducing complexity. The final part reports recent investigations on two-dimensional CA. In particular, we report a construction (slicing construction) that allows us to see a two-dimensional CA as a one-dimensional one and to lift some one-dimensional results to the two-dimensional case.

1 Introduction

Cellular automata (CAs) comprise a simple formal model for complex systems. They are used in many scientific fields (e.g., computer science, physics, mathematics, biology, chemistry, economics) with different purposes (e.g., simulation of natural phenomena, pseudo-random number generation, image processing, analysis of universal models of computation, and cryptography (Farina and Dennunzio 2009; Chaudhuri et al. 1997; Chopard 2012; Wolfram 1986)).

The huge variety of distinct dynamical behaviors has greatly determined the success of CA in applications. The study of CA as dynamical systems was first systematized by Hedlund in his famous paper (Hedlund 1969). Subsequent research introduced new details, unexpected phenomena, or links with aspects of dynamical systems theory. Presenting all these results would go beyond the purposes of this chapter, so the interested reader is directed to other surveys (Formenti and Kůrka 2009; Kůrka 2008; Cervelle 2008; Kari 2008; Pivato 2008; Di Lena and Margara 2008, 2009; Cattaneo et al. 2009; Di Lena 2006). This chapter aims to introduce basic results of CA viewed as dynamical systems and some recent developments.

2 Discrete Dynamical Systems

A **discrete dynamical system** (DDS) is a pair (X, F) where X is a set equipped with a metric d and $F: X \rightarrow X$ is a map, which is continuous on X with respect to d . The n th **iteration** of F is denoted by F^n . If F is bijective (invertible), the negative iterations are defined by $F^{-n} = (F^{-1})^n$. The function F induces **deterministic dynamics** by its iterated application starting from a given element. Formally, the **orbit** of a point $x \in X$ is $\mathcal{O}_F(x) := \{F^n(x) : n > 0\}$. A point $x \in X$ is **periodic** with period $n > 0$, if $F^n(x) = x$, while it is **eventually periodic**, if $F^m(x)$ is periodic for some **preperiod** $m \geq 0$. A set $Y \subseteq X$ is **invariant**, if $F(Y) \subseteq Y$ and **strongly invariant** if $F(Y) = Y$.

A **homomorphism** $\varphi: (X, F) \rightarrow (Y, G)$ of DDS is a continuous map $\varphi: X \rightarrow Y$ such that $\varphi \circ F = G \circ \varphi$, that is, the following diagram commutes

$$\begin{array}{ccc} X & \xrightarrow{F} & X \\ \varphi \downarrow & & \downarrow \varphi \\ Y & \xrightarrow{G} & Y \end{array}$$

A **conjugacy** φ is a bijective homomorphism such that φ^{-1} is continuous. Two systems (X, F) and (Y, G) are topologically **conjugated**, if there exists a conjugacy between them. If a homomorphism φ is surjective, one can say that (Y, G) is a **factor** of (X, F) , while if φ is injective, (X, F) is a **subsystem** of (Y, G) . In that case $\varphi(X) \subseteq Y$ is a closed invariant set whenever X is compact. Conversely, if $Y \subseteq X$ is a closed invariant set, then (Y, F) is a subsystem of (X, F) . The orbit $\overline{\mathcal{O}_F(x)}$ of a point $x \in X$ is an invariant set, so its closure $(\overline{\mathcal{O}_F(x)}, F)$ is a subsystem of (X, F) .

Denote by $B_\delta(x) = \{y \in X : d(y, x) < \delta\}$ the open ball centered in $x \in X$ and with radius $\delta > 0$.

A point $x \in X$ of a DDS (X, F) is **equicontinuous** (or Lyapunov stable), if

$$\forall \varepsilon > 0, \exists \delta > 0, \forall y \in B_\delta(x), \forall n \geq 0, d(F^n(y), F^n(x)) < \varepsilon$$

The existence of an equicontinuous point is a condition of local **stability** for the system. There are also notions of global stability based on the “size” of the set \mathcal{E}_F of the equicontinuous points. A system is equicontinuous if

$$\forall \varepsilon > 0, \exists \delta > 0, \forall x, y \in X, (d(x, y) < \delta \Rightarrow \forall n \geq 0, d(F^n(x), F^n(y)) < \varepsilon)$$

If a system is equicontinuous then $\mathcal{E}_F = X$. The converse is also true in compact settings. A system is **almost equicontinuous** if \mathcal{E}_F is a **residual** set, that is, it contains a countable intersection of dense open sets.

Conversely to stable behavior, sensitivity and positive expansivity are elements of instability for a system. Sensitivity to initial conditions recalls the well-known butterfly effects, in which small perturbations in the initial conditions can lead to strong differences in the evolution. More formally, a system is **sensitive** if

$$\exists \varepsilon > 0, \forall x \in X, \forall \delta > 0, \exists y \in B_\delta(x), \exists n \geq 0, d(F^n(y), F^n(x)) \geq \varepsilon$$

while it is **positively expansive** if

$$\exists \varepsilon > 0, \forall x, y \in X, x \neq y, \exists n \geq 0, d(F^n(y), F^n(x)) \geq \varepsilon$$

Note that in perfect spaces, positively expansive systems are sensitive.

Sensitivity and expansivity are often referred to as indicators of chaotic behavior. However, the most popular definitions of chaos also include other dynamical properties such as denseness of periodic orbits, transitivity, mixing, etc.

A DDS has the **DPO** property if its set of periodic points is dense. The set of **transitive points** is defined as $\mathcal{T}_F := \{x \in X : \overline{\mathcal{O}(x)} = X\}$. A system is **transitive** if for any nonempty open sets $U, V \subseteq X$ there exists $n > 0$ such that $F^n(U) \cap V \neq \emptyset$. In perfect spaces, if a system has a transitive point then it is transitive. On the other hand, in compact settings, any transitive system admits a transitive point and \mathcal{T}_F is a residual set. A system (X, F) is **mixing** if for every nonempty open sets $U, V \subseteq X$, $F^n(U) \cap V \neq \emptyset$ for all sufficiently large n . A DDS is **strongly transitive** if for any nonempty open set U , $\bigcup_{n \in \mathbb{N}} F^n(U) = X$.

Using the notion of chain, the following weaker forms of transitivity and mixing can be introduced. A finite sequence $(x_i \in X)_{0 \leq i \leq n}$ is a **δ -chain** from x_0 to x_n if $d(F(x_i), x_{i+1}) < \delta$ for all $i < n$. A system is **chain-transitive** if for each x, y and for each $\varepsilon > 0$ there exists an ε -chain from x to y , and **chain-mixing**, if for any $x, y \in X$ and any $\varepsilon > 0$ there exist chains from x to y of arbitrary and sufficient length.

3 Symbolic Dynamical Systems

A **Cantor space** is any metric space that is **compact** (any sequence has a convergent subsequence), **totally disconnected** (distinct points are separated by disjoint **clopen**, that is, closed and open sets), and **perfect** (no point is isolated). Any two Cantor spaces are homeomorphic. A **symbolic space** is any compact, totally disconnected metric space, that is, any closed subspace of a Cantor space. A **symbolic dynamical system** (SDS) is any DDS (X, F) where X is a symbolic space.

The long-term behavior of a DDS can be conveniently described by the notions of limit set and attractor. Here, the definitions of these concepts adapted to SDS are given. Consider a SDS (X, F) . The **limit set** of a clopen invariant set $V \subseteq X$ is $\Omega_F(V) := \bigcap_{n \geq 0} F^n(V)$. A set $Y \subseteq X$ is an **attractor**, if there exists a nonempty clopen invariant set V such that $Y = \Omega_F(V)$. There exists always the largest attractor $\Omega_F := \Omega_F(X)$. The number of attractors is at most countable. The union of two attractors is an attractor. If the intersection of two attractors is nonempty, it contains an attractor. The **basin** of an attractor $Y \subseteq X$ is the set $\mathcal{B}(Y) = \{x \in X; \lim_{n \rightarrow \infty} d(F^n(x), Y) = 0\}$. An attractor $Y \subseteq X$ is a **minimal attractor** if no proper subset of Y is an attractor. An attractor is a minimal attractor iff it is chain-transitive. A periodic point $x \in X$ is **attracting** if its orbit $\mathcal{O}(x)$ is an attractor. Any attracting periodic point is equicontinuous. A **quasi-attractor** is a nonempty set that is an intersection of a countable number of attractors.

3.1 Subshifts

For a finite alphabet A , denote by $A^* := \bigcup_{n \geq 0} A^n$ the set of words over A and by $A^+ := \bigcup_{n > 0} A^n$ the set of words of positive length. The length of a word $u = u_0, \dots, u_{n-1} \in A^n$ is denoted by $|u| := n$ and the word of zero length is λ . One can say that $u \in A^*$ is a subword of $v \in A^*$ ($u \sqsubseteq v$) if there exists k such that $v_{k+i} = u_i$ for all $i < |u|$. One can denote by $u_{[i,j]} = u_i \dots u_{j-1}$ and $u_{[i,j]} = u_i \dots u_j$ subwords of u associated to intervals. One can denote by $A^\mathbb{Z}$ the space of **A -configurations**, or doubly infinite sequences of letters of A equipped with the metric

$$d(x, y) := 2^{-n}, \quad \text{where } n = \min\{i \geq 0 : x_i \neq y_i \text{ or } x_{-i} \neq y_{-i}\}$$

The **shift map** $\sigma : A^\mathbb{Z} \rightarrow A^\mathbb{Z}$ is defined by $\sigma(x)_i := x_{i+1}$. For any $u \in A^+$ one has a σ -periodic configuration $u^\infty \in A^\mathbb{Z}$ defined by $(u^\infty)_i = u_{|i| \bmod |u|}$. A **subshift** is a nonempty subset $\Sigma \subseteq A^\mathbb{Z}$, which is closed and strongly σ -invariant, that is, $\sigma(\Sigma) = \Sigma$. For a given alphabet A define the bijective **k -block code** $\alpha_k : A^\mathbb{Z} \rightarrow (A^k)^\mathbb{Z}$ by $\alpha_k(x)_i = x_{[i, i+k]}$. For a subshift $\Sigma \subseteq A^\mathbb{Z}$ denote by $\Sigma^{[k]} = \alpha_k(\Sigma) \subseteq (A^k)^\mathbb{Z}$ its **k -block encoding**. Then $\alpha_k : \Sigma \rightarrow \Sigma^{[k]}$ is a conjugacy as it commutes with the shift maps $\alpha_k \sigma = \sigma \alpha_k$. For a subshift Σ there exists a set $D \subseteq A^*$ of forbidden words such that

$$\Sigma = \Sigma_D := \{x \in A^\mathbb{Z} : \forall u \sqsubseteq x, u \notin D\}$$

A subshift is uniquely determined by its **language**

$$\mathcal{L}(\Sigma) := \{u \in A^* : \exists x \in \Sigma, u \sqsubseteq x\}$$

We denote by $\mathcal{L}^n(\Sigma) := \mathcal{L}(\Sigma) \cap A^n$. The language of **first offenders** of Σ is

$$\mathcal{D}(\Sigma) := \{u \in A^+ \setminus \mathcal{L}(\Sigma) : u_{[0, |u|-1]}, u_{[1, |u|]} \in \mathcal{L}(\Sigma)\}$$

so $\Sigma = \Sigma_{\mathcal{L}(\Sigma)}$. The **extended language** of Σ is

$$\tilde{\mathcal{L}}(\Sigma) = \{x|_I : x \in \Sigma, I \subseteq \mathbb{Z} \text{ is an interval}\}$$

A subshift $\Sigma \subseteq A^{\mathbb{Z}}$ is transitive, iff for any words $u, v \in \mathcal{L}(\Sigma)$ there exists $w \in A^*$ such that $uvw \in \mathcal{L}(\Sigma)$. A subshift $\Sigma \subseteq A^{\mathbb{Z}}$ is mixing if for any words $u, v \in \mathcal{L}(\Sigma)$ there exists $n > 0$ such that for all $m > n$ there exists $w \in A^m$ such that $uvw \in \mathcal{L}(\Sigma)$.

Definition 1 Given an integer $c \geq 0$, the **c -join** $\Sigma_0 \overset{c}{\vee} \Sigma_1$ of subshifts $\Sigma_0, \Sigma_1 \subseteq A^{\mathbb{Z}}$ consists of all configurations $x \in A^{\mathbb{Z}}$ such that either $x \in \Sigma_0 \cup \Sigma_1$, or there exist integers b, a such that $b - a \geq c$, $x_{(-\infty, b)} \in \tilde{\mathcal{L}}(\Sigma_0)$, and $x_{[a, \infty)} \in \tilde{\mathcal{L}}(\Sigma_1)$.

Proposition 1 (Formenti et al. 2010) *The c -join of two subshifts is a subshift and the operation of c -join is associative. A configuration $x \in A^{\mathbb{Z}}$ belongs to $\Sigma_1 \vee \dots \vee \Sigma_n$ iff there exist integers $k > 0$, $1 \leq i_1 < i_2 < \dots < i_k \leq n$, and intervals $I_1 = (a_1, b_1), I_2 = [a_2, b_2], \dots, I_k = [a_k, b_k]$ such that $a_1 = -\infty$, $b_k = \infty$, $a_j < a_{j+1}$, $b_j < b_{j+1}$, $b_j - a_{j+1} \geq c$, and $x|_{I_j} \in \tilde{\mathcal{L}}(\Sigma_{i_j})$.*

3.2 Sofic Subshifts

A subshift $\Sigma \subseteq A^{\mathbb{Z}}$ is **sofic** if its language $\mathcal{L}(\Sigma)$ is regular. Sofic subshifts are usually described by finite automata or by labeled graphs.

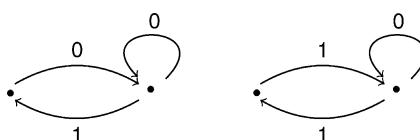
A **labeled graph** over an alphabet A is a structure $G = (V, E, s, t, l)$, where V is a finite set of vertices, E is a finite set of edges, $s, t: E \rightarrow V$ are the **source** and **target maps**, and $l: E \rightarrow A$ is a **labeling function**. A finite or infinite word $w \in E^* \cup E^{\mathbb{Z}}$ is a **path** in G if $t(w_i) = s(w_{i+1})$ for all i . The source and target of a finite path $w \in E^n$ are $s(w) := s(w_0)$, $t(w) := t(w_{n-1})$. The **label** of a path is defined by $l(w) := l(w_i)$. A subshift Σ is sofic iff there exists a labeled graph G such that $\Sigma = \Sigma_G$ is the set of labels of all doubly infinite paths in G . In this case, it can be said that G is a **presentation** of Σ (see e.g., Lind and Marcus (1995) or Kitchens (1998)).

A subshift $\Sigma \subseteq A^{\mathbb{Z}}$ is of **finite type** (SFT) if $\Sigma = \Sigma_D$ for some finite set $D \subseteq A^+$ of forbidden words. The words of D can be assumed to be all of the same length, which is called the order $\sigma(\Sigma)$ of Σ . A configuration $x \in A^{\mathbb{Z}}$ belongs to Σ iff $x_{[i, i+\sigma(\Sigma))} \in \mathcal{L}(\Sigma)$ for all $i \in \mathbb{Z}$. Any SFT is sofic: if $p = \sigma(\Sigma) - 1$, the **canonical graph** $G = (V, E, s, t, l)$ of Σ is given by $V = \mathcal{L}^p(\Sigma)$, $E = \mathcal{L}^{p+1}(\Sigma)$, $s(u) = u_{[0, p]}$, $t(u) = u_{[1, p]}$ and $l(u) = u_p$. If $\Sigma \subset A^{\mathbb{Z}}$ is a finite subshift, then it is of finite type and each its configuration is σ -periodic. The period $\mathfrak{p}(\Sigma)$ of Σ is then the smallest positive integer $\mathfrak{p}(\Sigma)$, such that $\sigma^{\mathfrak{p}(\Sigma)}(x) = x$ for all $x \in \Sigma$.

Example 1 The golden mean subshift is the SFT with binary alphabet $A = \{0, 1\}$ and forbidden word 11 (Fig. 1 left).

Fig. 1

Golden (left) mean subshift and (right) even subshift.



Example 2 The even subshift is the sofic subshift with binary alphabet and forbidden words $D = \{10^{2n+1}1 : n \geq 0\}$ (☞ Fig. 1 right).

A labeled graph $G = (V, E, s, t, l)$ is **connected** if for any two vertices $q, q' \in V$ there exists a path $w \in E^*$ from q to q' . An **induced subgraph** of a graph G is a graph $G' = (V', E', s', t', l')$, such that $V' \subseteq V$, $E' = \{e \in E : s(e) \in V' \& t(e) \in V'\}$, and s', t', l' coincide respectively with s, t, l on E' . A **strongly connected component** of G is a subgraph of G , which is connected and maximal with this property. The subshift of a connected graph is transitive. Conversely, every transitive sofic subshift $\Sigma \subseteq A^{\mathbb{Z}}$ has a connected presentation. A labeled graph $G = (V, E, s, t, l)$ is **aperiodic** if the set of vertices cannot be partitioned into disjoint union $V = V_0 \cup \dots \cup V_{p-1}$ such that $p \geq 2$ and if $s(e) \in V_i$, then $t(e) \in V_{(i+1) \bmod p}$. A sofic subshift is mixing iff it has a connected and aperiodic presentation.

For a labeled graph $G = (V, E, s, t, l)$ and $k > 1$ define the **k -block graph** $G^{[k]} := (V', E', s', t', l')$, where $V' \subseteq E^{k-1}$ is the set of paths of G of length $k-1$, $E' \subseteq E^k$ is the set of paths of G of length k , s' and t' are the prefix and suffix maps and $l' : E' \rightarrow A^k$ is defined by $l'(u)_i = l(u_i)$. Then $G^{[k]}$ is a presentation of the k -block encoding $(\Sigma_G)^{[k]}$ of Σ_G . From $G^{[k]}$ one may in turn get a presentation of Σ_G , if l' is replaced by its composition with a projection $\pi_i : A^k \rightarrow A$ defined by $\pi(u)_i := u_i$, where $i < k$.

Given two sofic subshifts $\Sigma_0, \Sigma_1 \subseteq A^{\mathbb{Z}}$, their union and intersection (provided nonempty) are sofic subshifts. Moreover there exists an algorithm that constructs a presentation of $\Sigma_0 \cup \Sigma_1$ and $\Sigma_0 \cap \Sigma_1$ from those of Σ_0 and Σ_1 . It is also decidable whether $\Sigma_0 \subseteq \Sigma_1$. Given a labeled graph G it is decidable whether Σ_G is an SFT (see Lind and Marcus 1995, p. 94). It is also decidable whether an SFT is mixing.

Proposition 2 Let $\Sigma_0, \Sigma_1 \subseteq A^{\mathbb{Z}}$ be sofic subshifts and $c \geq 0$. Then $\overset{c}{\Sigma_0 \vee \Sigma_1}$ is a sofic subshift. There exists an algorithm that constructs a presentation G of $\overset{c}{\Sigma_0 \vee \Sigma_1}$ from the presentations G_i of Σ_i . Moreover, G has the same strongly connected components as the disjoint union $G_0 \cup G_1$.

Proof Let $G_i = (V_i, E_i, s_i, t_i, l_i)$ be presentations of $\Sigma_i^{[c]}$, and assume that $V_0 \cap V_1 = \emptyset$ and $E_0 \cap E_1 = \emptyset$. Construct $G = (V, E, s, t, l)$, where $V = V_0 \cup V_1$

$$E = E_0 \cup E_1 \cup \{(e_0, e_1) \in E_0 \times E_1 : l_0(e_0) = l_1(e_1)\}$$

The source, target, and label maps extend s_i, t_i, l_i . For the new edges one has $s(e_0, e_1) = s_0(e_0)$, $t(e_0, e_1) = t_1(e_1)$, $l(e_0, e_1) = l_0(e_0) = l_1(e_1)$. Then $\Sigma_G = (\overset{c}{\Sigma_0 \vee \Sigma_1})^{[c]}$, and one gets a presentation of $\overset{c}{\Sigma_0 \vee \Sigma_1}$, if one composes l with a projection $\pi_i : A^c \rightarrow A$. Clearly, the strongly connected components of $G_0 \cup G_1$ are not changed by the new edges.

Proposition 3 The language $\mathcal{D}(\Sigma)$ of first offenders of a sofic subshift Σ is regular.

4 One-Dimensional Cellular Automata (1D CAs)

In this section, the basic notions and results about 1D CA as SDS are reviewed. To introduce them, a more general definition is used whose relevance has been argued by Boyle and Kitchens (1999).

Definition 2 A cellular automaton is a pair (X, F) , where $X \subseteq A^{\mathbb{Z}}$ is a mixing subshift of finite type, and $F : X \rightarrow X$ is a continuous mapping which commutes with the shift, i.e., $F\sigma = \sigma F$.

For a CA (X, F) there exists a **local rule** $f : \mathcal{L}^{a-m+1}(X) \rightarrow \mathcal{L}^1(X)$ such that $F(x)_i = f(x_{[i+m, i+a]})$. Here $m \leq a$ are integers called **memory** and **anticipation**. The **diameter** of (X, F) is $d := a - m$. If $a = -m \geq 0$, then it can be said that a is the **radius** of (X, F) . The local rule can be extended to a map $f : \mathcal{L}(X) \rightarrow \mathcal{L}(X)$ by $f(u)_i := f(u_{[i, i+d]})$ for $0 \leq i < |u| - d$. Thus $|f(u)| = \max\{|u| - d, 0\}$. The **cylinder set** of a word $u \in \mathcal{L}(X)$ located at $l \in \mathbb{Z}$ is $[u]_l := \{x \in X : x_{[l, l+|u|]} = u\}$. The cylinder set of a finite set $U \subset A^*$ is $[U]_l = \bigcup_{u \in U} [u]_l$. We also write $[u] := [u]_0$ and $[U] := [U]_0$. Every clopen set of X is a cylinder set of a finite set $U \subset A^*$.

Proposition 4 (Formentini and Kůrka 2007) *Let (X, F) be a CA. If $\Sigma \subseteq X$ is a sofic subshift, then $F(\Sigma)$ and $F^{-1}(\Sigma)$ are sofic subshifts and there exists an algorithm that constructs their graphs from the local rule f , the graph of Σ and the set of forbidden words of X .*

A word $u \in \mathcal{L}(X)$ with $|u| \geq s \geq 0$ is **s -blocking** for a CA (X, F) , if there exists an **offset** $k \in [0, |u| - s]$ such that

$$\forall x, y \in [u]_0, \forall n \geq 0, F^n(x)_{[k, k+s]} = F^n(y)_{[k, k+s]}$$

Theorem 1 (Kůrka 2008) *Let (X, F) be a CA with radius $r \geq 0$. The following conditions are equivalent.*

1. (X, F) is not sensitive.
2. (X, F) has an r -blocking word.
3. \mathcal{E}_F is **residual**, i.e., it contains a countable intersection of dense open sets.
4. $\mathcal{E}_F \neq \emptyset$.

For a nonempty set $B \subseteq A^*$ define

$$\begin{aligned}\mathcal{T}_\sigma^n(B) &:= \{x \in A^\mathbb{Z} : (\exists j > i > n, x_{[i,j]} \in B) \& (\exists j < i < -n, x_{[j,i]} \in B)\} \\ \mathcal{T}_\sigma(B) &:= \bigcap_{n \geq 0} \mathcal{T}_\sigma^n(B)\end{aligned}$$

Each $\mathcal{T}_\sigma^n(B)$ is open and dense, so the set $\mathcal{T}_\sigma(B)$ of **B -recurrent** configurations is residual. If B is the set of r -blocking words, then $\mathcal{E}_F \supseteq \mathcal{T}_\sigma(B)$.

Theorem 2 (Kůrka 2008) *Let (X, F) be a CA with radius $r \geq 0$. The following conditions are equivalent.*

1. (X, F) is equicontinuous, i.e., $\mathcal{E}_F = X$.
2. There exists $k > 0$ such that any $u \in \mathcal{L}^k(X)$ is r -blocking.
3. There exists a preperiod $q \geq 0$ and a period $p > 0$, such that $F^{q+p} = F^q$.

In particular, every CA with radius $r = 0$ is equicontinuous. A configuration is equicontinuous for F iff it is equicontinuous for F^n , that is, $\mathcal{E}_F = \mathcal{E}_{F^n}$.

Example 3 (a product rule ECA128) Consider the ECA given by the following local rule: $F(x)_i = x_{i-1}x_i x_{i+1}$

$$000:0, 001:0, 010:0, 011:0, 100:0, 101:0, 110:0, 111:1$$

The ECA128 is almost equicontinuous and 0 is a 1-blocking word (Fig. 2). It is not surjective since 101 has no preimage. The cylinder $[0]_0$ is a clopen invariant set whose omega-limit is the singleton $\Omega_F([0]_0) = \{0^\infty\}$ that contains stable fixed point 0∞ . The maximal attractor is larger. We have $F(A^\mathbb{Z}) = \Sigma_{\{101, 1001\}}$, $F^n(A^\mathbb{Z}) = \Sigma_{\{10^k 1: 0 < k \leq 2n\}}$ and

$$\Omega_F = \{u \in A^\mathbb{Z} : \forall n > 0, 10^n 1 \not\sqsubseteq u\}$$

Example 4 (the majority rule ECA232) $F(x)_i = \left\lfloor \frac{x_{i-1} + x_i + x_{i+1}}{2} \right\rfloor$

000:0, 001:0, 010:0, 011:1, 100:0, 101:1, 110:1, 111:1

The majority rule has 2-blocking words 00 and 11, so it is almost equicontinuous (Fig. 3). It is not surjective. The clopen sets $[00]_n$ and $[11]_n$ are invariant. More generally, let $E = \{u \in 2^* : |u| \geq 2, u_0 = u_1, u_{|u|-2} = u_{|u|-1}, 010 \not\sqsubseteq u, 101 \not\sqsubseteq u\}$. Then for any $u \in E$ and for any $i \in \mathbb{Z}$, $[u]_i$ is a clopen invariant set, so its limit set $\Omega_F([u]_i)$ is an attractor. These attractors are not subshifts. There exists a subshift attractor $\Omega_F(U)$, where $U := 2^\mathbb{Z} \setminus ([010]_0 \cup [101]_0)$. The maximal attractor is

$$\Omega_F = \Sigma_{\{010^k 1, 10^k 10, 01^k 01, 101^k 0: k > 1\}}$$

4.1 Combinatorial Properties for 1D CA

Some interesting dynamical properties can be conveniently expressed in a combinatorial manner allowing us to find decision algorithms. Such properties include surjectivity, injectivity and openness. A 1D CA is **surjective** (resp., **injective**, **open**) if its global function is surjective (resp., injective, open).

For the case of surjectivity, Hedlund gave the following characterization (Hedlund 1969), which will also be useful in what follows.

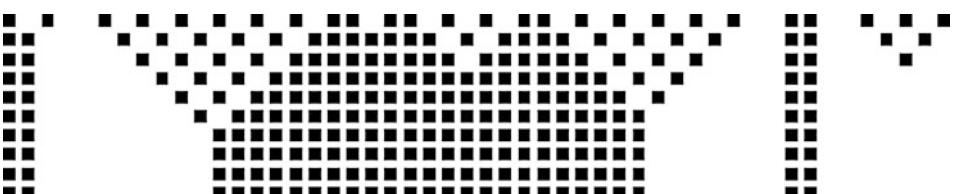
Fig. 2

A space-time diagram for ECA128 (see Fig. 2 Example 3).



Fig. 3

A space-time diagram for ECA232 (see Fig. 3 Example 4).



Theorem 3 Let $(A^{\mathbb{Z}}, F)$ be a CA with local rule $f : A^{d+1} \rightarrow A$. The following conditions are equivalent.

1. $F : A^{\mathbb{Z}} \rightarrow A^{\mathbb{Z}}$ is surjective.
2. For each $x \in A^{\mathbb{Z}}$, $F^{-1}(x)$ is a finite set.
3. $f : A^* \rightarrow A^*$ is surjective.
4. For each $u \in A^+$, $|f^{-1}(u)| = |A|^d$.

The above characterization (except for item (2)) has been extended to any dimension by Maruoka and Kimura (1976) via the notion of balance as shown in [Sect. 5.2](#).

Closingness is another interesting combinatorial property. A CA $(A^{\mathbb{Z}}, F)$ is **right** (resp., **left**) **closing** iff $F(x) \neq F(y)$ for any pair $x, y \in A^{\mathbb{Z}}$ of distinct left (resp., right) asymptotic configurations, that is, $x_{(-\infty, n]} = y_{(-\infty, n]}$ (resp., $x_{[n, \infty)} = y_{[n, \infty)}$) for some $n \in \mathbb{Z}$, where $z_{(-\infty, n]}$ (resp., $z_{[n, \infty)}$) denotes the portion of a configuration z inside the infinite integer interval $(-\infty, n]$ (resp., $[n, \infty)$). A CA is said to be **closing** if it is either left closing or right closing.

In Boyle and Kitchens (1999), it is shown that a CA is open iff it is both left closing and right closing. Moreover, in the same paper, they prove that a closing CA has the joint denseness of periodic orbits (**JDPO**) property; that is, it has a dense set of periodic points, which are also periodic for the shift map. In what follows, it can be seen that this last property is closely related to the solution of a long-standing open conjecture.

Permutivity is a combinatorial property, which can be easily decided using the CA local rule. A 1D CA of radius r is permutive in the variable of index i ($|i| \leq r$) if its local rule f is injective with respect to the i th variable. A 1D CA is **leftmost permutive** (resp., rightmost permutive) if it is permutive in the variable of index $-r$ (resp., r).

Permutivity is closely related to chaotic behavior. Indeed, a 1D CA that is both leftmost and rightmost expansive is also positively expansive and open (Shereshevsky and Afraimovich 1992; Nasu 1995).

Moreover, the following two results will be also useful for the 2D CA case.

Proposition 5 (Cattaneo et al. 2004) Let F be a 1D CA with local rule f on a possibly infinite alphabet A . If f is either rightmost or leftmost permutive, then F is topologically mixing.

Proposition 6 (Dennunzio and Formenti 2008, 2009) Let F be a 1D CA with local rule f on a possibly infinite alphabet A . If f is both rightmost and leftmost permutive, then F is strongly transitive.

4.2 Attractors

The notion of attractor is essential to understand the long-term behavior of a DDS. The following results are a first characterization of attractors and quasi-attractors in CA.

Theorem 4 (Hurley 1990)

1. If a CA has two disjoint attractors, then any attractor contains two disjoint attractors and an uncountably infinite number of quasi-attractors.

2. If a CA has a minimal (w.r.t. set inclusion) attractor, then it is a subshift, it is contained in any other attractor, and its basin of attraction is a dense open set.
3. If $x \in A^{\mathbb{Z}}$ is an attracting F -periodic configuration, then $\sigma(x) = x$ and $F(x) = x$.

The ECA232 of [Example 4](#) is of class (1) of [Theorem 4](#). A trivial example of this class is the identity CA $(A^{\mathbb{Z}}, \text{Id})$ in any alphabet. Each clopen set $U \subseteq A^{\mathbb{Z}}$ is invariant and its omega-limit set is itself: $\Omega_{\text{Id}}(U) = U$. The ECA128 of [Example 3](#) is of class (2) and satisfies also condition (3). A trivial example of class (2) is the zero CA in the binary alphabet $\{0, 1\}^{\mathbb{Z}}$ given by $F(x) = 0^{\infty}$.

Corollary 1 (Kůrka 2008) *For any CA, exactly one of the following statements holds.*

1. *There exist two disjoint attractors and a continuum of quasi-attractors.*
2. *There exists a unique quasi-attractor. It is a subshift and it is contained in any attractor.*
3. *There exists a unique minimal attractor contained in any other attractor.*

Both equicontinuity and surjectivity yield strong constraints on attractors.

Theorem 5 (Kůrka 2003b)

1. *A surjective CA has either a unique attractor or a pair of disjoint attractors.*
2. *An equicontinuous CA has either two disjoint attractors or a unique attractor that is an attracting fixed configuration.*
3. *If a CA has an attracting fixed configuration that is a unique attractor, then it is equicontinuous.*

4.3 Subshift Attractors

For non-surjective CA, attractors are often subshifts. In this section, this special type of attractor will be characterized. First, some definitions are needed.

Definition 3

1. A clopen F -invariant set $W \subseteq X$ is **spreading to the right** (or **left**) if $F^k(W) \subseteq \sigma^{-1}(W)$ (or $F^k(W) \subseteq \sigma(W)$) for some $k > 0$.
2. A clopen F -invariant set $W \subseteq X$ is **spreading** if it is spreading both to the right and to the left.

Proposition 7 (Formenti and Kůrka 2007) *Let (X, F) be a cellular automaton and $W \subseteq X$ a clopen F -invariant set. Then $\Omega_F(W)$ is a subshift attractor iff W is spreading.*

Theorem 6 (Formenti and Kůrka 2007) *Each subshift attractor is chain-mixing for the shift, contains a configuration which is both F -periodic and σ -periodic, and the complement of its language is recursively enumerable.*

It follows that every attractor that is a subshift of finite type is mixing.

Theorem 7 (Formenti and Kůrka 2007) *The only subshift attractor of a surjective CA $(A^{\mathbb{Z}}, F)$ is the full space $A^{\mathbb{Z}}$.*

Theorem 8 (Formenti and Kůrka 2007) *If $\Sigma \subseteq A^{\mathbb{Z}}$ is a mixing subshift of finite type, then there exists a cellular automaton $(A^{\mathbb{Z}}, F)$ such that Σ is an attractor of $(A^{\mathbb{Z}}, F)$ and $F(x) = x$ for every $x \in \Sigma$.*

Proposition 8 (Formenti and Kůrka 2007) *Let G be a labelled graph with two strongly connected components G_0 and G_1 such that $\Sigma_G \neq \Sigma_{G_0} = \Sigma_{G_1} = \{\sigma^i(u^\infty) : i < |u|\}$ for some $u \in A^+$. Then Σ_G is not a subshift attractor.*

The sofic subshifts in [Fig. 4](#) are not transitive. The left subshift is chain-transitive but not chain-mixing, so it is not subshift attractor by [Theorem 6](#). The right subshift is chain-mixing, but it is not subshift attractor by [Proposition 8](#). The union of two attractors is an attractor. The intersection of two attractors need not be an attractor, but if it is nonempty, it contains an attractor. The intersection of two subshift attractors is always nonempty but need not be an attractor either (see [Example 5](#)).

Example 5 (Kůrka 2007) The intersection of two subshift attractors need not to be an attractor.

Proof Take the alphabet $A = \{0, 1, 2, 3\}$ and local rule $f: A^3 \rightarrow A$ given by

$$x1v:0, z2x:1, x2z:1, y3x:1, x3y:1$$

where $x, v \in A$, $y \in \{0, 1, 2\}$, $z \in \{0, 1, 3\}$ and the first applicable production is used, otherwise the letter is left unchanged (see simulation in [Fig. 5](#)). Then $U_2 := [0] \cup [1] \cup [2]$, $U_3 := [0] \cup [1] \cup [3]$ are spreading sets whose attractors $\Sigma_2 := \Omega_F(U_2)$, $\Sigma_3 := \Omega_F(U_3)$ are sofic subshifts whose labeled graphs are in [Fig. 5](#). The subshifts consist of all labels of all (doubly infinite) paths in their corresponding graphs. Then $\Sigma_0 := \Sigma_2 \wedge \Sigma_3 = \{0^\infty\}$ and $0^\infty 10^\infty \in (\Sigma_2 \cap \Sigma_3) \setminus \Sigma_0$. There is one more subshift attractor $\Omega_F = \Sigma_2 \cup \Sigma_3$.

Example 6 (Kůrka 2007) There exists a CA with an infinite sequence of subshift attractors $\Sigma_{n+1} \subset \Sigma_n$ indexed by positive integers.

A CA is first constructed with an infinite number of clopen invariant sets $U_{n+1} \subset U_n$ that spread to the right. One has the alphabet $A = \{0, 1, 2, 3\}$ and a CA $F(x)_i = f(x_{[i-1, i+1]})$, where $f: A^3 \rightarrow A$ is the local rule given by

Fig. 4

A chain-transitive and a chain-mixing subshifts.



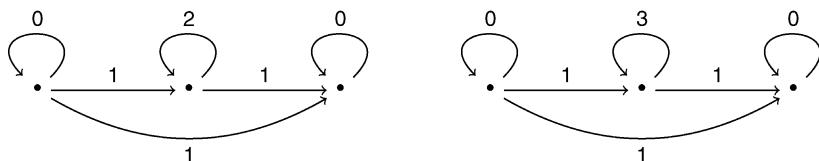
Fig. 5

Intersection of attractors.

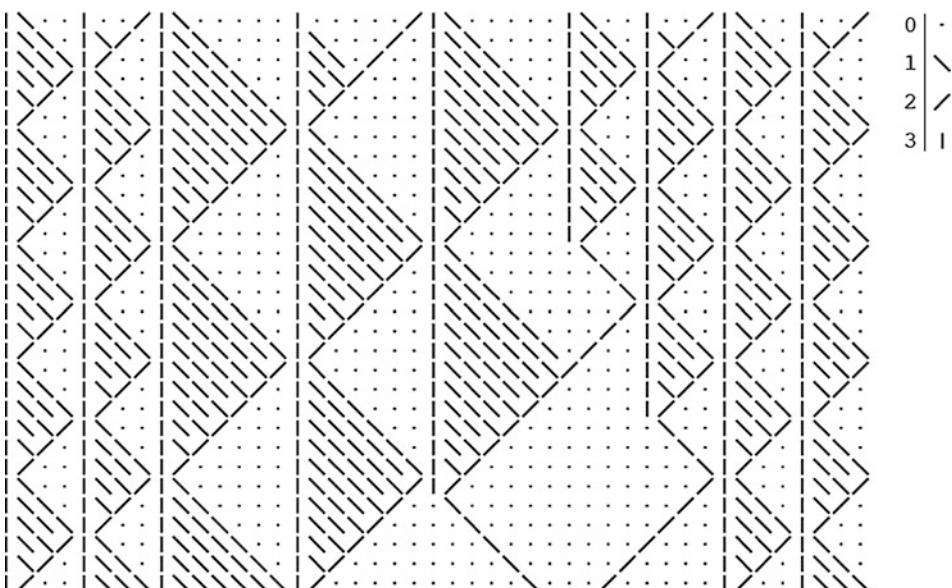
```

00022222211111333332222232323233333000
00012222100000133311222111111133331000
000012210000000131001210000000013310000
000001100000000010000100000000001100000
00000000000000000000000000000000000000000000

```

**Fig. 6**

Decreasing sequence of subshift attractors.



x33:0, 132:3, x32:0, xy2:2, x13:2

3xy:1, x2y:0, 10x:1, 11x:1, x1y:0

Here $x, y \in A$ and the first applicable production is used' otherwise the letter is left unchanged (see a simulation in [Fig. 6](#)). Letter 3 is a stationary particle, which generates right-going particles 1 via the production $3xx:1$. When a 1 particle reaches another 3, it changes to a left-going particle 2 via the production $x13:2$ and erases all 1 particles which it encounters. When this 2 particle reaches a 3 particle simultaneously with a 1 particle, the 3 particle is preserved by the production $132:3$. Otherwise 3 is destroyed by the production $x32:0$. Since the 3 particles are never created, they successively disappear unless they are distributed periodically. Set

$$U_n = \{x \in A^{\mathbb{Z}} : (0 \leq i < j < 2n - 2) \& (x_i = x_j = 3) \implies j - i \geq n\}$$

Each U_n is clopen and invariant, since the letter 3 is never created. We have $F^{4n-7}(U_n) \subseteq \sigma^{-1}(U_n)$ so U_n is spreading to the right. Define the mirror image CA G of F with local rule

$$\begin{aligned} 33x:0, 231:3, 23x:0, 2xx:2, 31x:2 \\ xx3:1, x2x:0, x01:1, x11:1, x1x:0 \end{aligned}$$

so that U_n is spreading to the left for G . Consider now the alphabet $B = \{(a, b) \in A^2 : a = 3 \iff b = 3\}$ with 10 letters, and a CA on B defined by

$$H(x, y)_i = \begin{cases} (0, 0) & \text{if } x_i = y_i = 3 \text{ and} \\ & F(x)_i \neq 3 \text{ or } G(y)_i \neq 3 \\ (F(x)_i, G(y)_i) & \text{otherwise} \end{cases}$$

Thus the components F and G of H share the same particles 3 both F and G can destroy them. Then $V_n := B^{\mathbb{Z}} \cap (U_n \times U_n)$ is a spreading set for H and $\Sigma_n := \Omega_H(V_n)$ is a subshift attractor. Since $V_{n+1} \subset V_n$ we have $\Sigma_{n+1} \subseteq \Sigma_n$. For $n > 2$ there exists a σ -periodic configuration $x = (31^{n-1}320^{n-2})^\infty \in U_n \setminus U_{n+1}$ that is periodic for F , so $x \in \Omega_F(U_n) \setminus \Omega_F(U_{n+1})$. The mirror image configuration $y = (30^{n-2}231^{n-1})^\infty$ is periodic for G , so $(x, y) \in \Sigma_n \setminus \Sigma_{n+1}$. For each configuration $(x, y) \in \Sigma_n \setminus \Sigma_{n+1}$ we have $H^{2n}(x, y) = (x, y)$.

4.4 The Small Quasi-attractor

When a CA F is initialized on a random configuration x and if each word has positive probability, then this configuration contains with probability one each spreading set, not only of F but also of all $F^q\sigma^p$. Therefore, the iterations of such a configuration converge to the intersection of all attractors of all $F^q\sigma^p$.

Definition 4 Let (Σ, F) be a cellular automaton. The intersection of all subshift attractors of all $F^q\sigma^p$, where $q > 0$ and $p \in \mathbb{Z}$, is called the **small quasi-attractor** of F and denoted by \mathcal{Q}_F .

Proposition 9 (Formenti and Kůrka 2007) *Let (Σ, F) be a cellular automaton. If $\Sigma_1, \dots, \Sigma_k \subseteq \Sigma$ are subshifts such that each Σ_i is an attractor of some $F^{q_i}\sigma^{p_i}$, where $q_i > 0$ and $p_i \in \mathbb{Z}$, then the intersection $\Sigma_1 \cap \dots \cap \Sigma_k$ is nonempty.*

Theorem 9 (Formenti and Kůrka 2007) *If (Σ, F) is a cellular automaton, then \mathcal{Q}_F is a nonempty F -invariant subshift and $F : \mathcal{Q}_F \rightarrow \mathcal{Q}_F$ is surjective. Moreover, (\mathcal{Q}_F, σ) is chain-mixing.*

The small quasi-attractor of ECA128 is $\mathcal{Q}_F = \{0^\infty\}$. The small quasi-attractor of ECA232 is

$$\mathcal{Q}_F = \Omega_F = \Sigma_{\{010^k 1, 10^k 10, 01^k 01, 101^k 0, \dots, k > 1\}}$$

Example 7 (A product rule ECA136) Consider the ECA given by the following local rule $F(x)_i = x_i x_{i+1}$.

The ECA136 is almost equicontinuous since 0 is a 1-blocking word (Fig. 7). As in Example 3, we have $\Omega_F = \Sigma_{\{10^k : k > 0\}}$. For any $m \in \mathbb{Z}$, $[0]_m$ is a clopen invariant

Fig. 7

Example of space–time diagram for ECA136 (see [Example 7](#)).



set, which is spreading to the left but not to the right. Thus $Y_m = \Omega_F([0]_m) = \{x \in \Omega_F : \forall i \leq m, x_i = 0\}$ is an attractor but not a subshift. We have $Y_{m+1} \subset Y_m$ and $\bigcap_m \geq_0 Y_m = \{0^\infty\}$ is the unique minimal quasi-attractor. Since $F^2\sigma^{-1}(x)_i = x_{i-1}x_ix_{i+1}$ is the ECA128, which has a minimal subshift attractor $\{0^\infty\}$, F has the small quasi-attractor $\mathcal{Q}_F = \{0^\infty\}$.

4.5 Signal Subshifts

Definition 5 Let (X, F) be a cellular automaton. A configuration $x \in X$ is **weakly periodic** if $F^q\sigma^p(x) = x$ for some $q > 0$ and $p \in \mathbb{Z}$. We call (p, q) the **period** of x and p/q its speed. Let $\mathcal{S}_{(p,q)}(X, F) := \{x \in X : F^q\sigma^p(x) = x\}$ be the set of all weakly periodic configurations with period (p, q) . A **signal subshift** is any non-empty $\mathcal{S}_{(p,q)}(X, F)$.

For fixed (X, F) we write $\mathcal{S}_{(p,q)} := \mathcal{S}_{(p,q)}(X, F)$. Note that $\mathcal{S}_{(p,q)}$ is closed and σ -invariant, so it is a subshift provided it is nonempty. Moreover, $\mathcal{S}_{(p,q)}$ is F -invariant and $F : \mathcal{S}_{(p,q)} \rightarrow \mathcal{S}_{(p,q)}$ is bijective, so $\mathcal{S}_{(p,q)} \subseteq \Omega_F(X)$. If $\mathcal{S}_{(p,q)}$ is finite, it consists only of σ -periodic configurations.

The ECA128 of [Example 3](#) has nontransitive signal subshifts

$$\begin{aligned}\mathcal{S}_{(1,1)} &= \{\sigma^n(0^\infty.1^\infty) : n \in \mathbb{Z}\} \cup \{0^\infty, 1^\infty\} \\ \mathcal{S}_{(-1,1)} &= \{\sigma^n(1^\infty.0^\infty) : n \in \mathbb{Z}\} \cup \{0^\infty, 1^\infty\}\end{aligned}$$

The ECA232 of [Example 4](#) has transitive signal subshift $\mathcal{S}_{(0,1)} = \Sigma_{\{010,101\}}$.

Example 8 (The traffic rule ECA184, [Fig. 8](#)) $F(x)_i = 1$ if $x_{[i-1,i]} = 10$ or $x_{[i,i+1]} = 11$.

000:0 001:0 010:0 011:1 100:1 101:1 110:0 111:0

The ECA184 has three infinite signal subshifts

$$\mathcal{S}_{(1,1)}(F) = \Sigma_{\{11\}} \cup \{1^\infty\}, \quad \mathcal{S}_{(0,1)}(F) = \Sigma_{\{10\}}, \quad \mathcal{S}_{(-1,1)}(F) = \Sigma_{\{00\}} \cup \{0^\infty\}$$

and a unique F -transitive attractor $\Omega_F = \Sigma_{\{1(10)^n: n > 0\}}$, which is sofic.

Theorem 10 Let (X, F) be a cellular automaton with memory m and anticipation a , and let $d = a - m$ be the diameter.

1. If $\mathcal{S}_{(p,q)}$ is nonempty, then it is a subshift of finite type.
2. If $\mathcal{S}_{(p,q)}$ is infinite, then $\text{o}(\mathcal{S}_{(p,q)}) \leq \max\{\text{o}(X), dq + 1\}$ and $-a \leq p/q \leq -m$.
3. If $p_0/q_0 < p_1/q_1$, then $\mathcal{S}_{(p_0,q_0)} \cap \mathcal{S}_{(p_1,q_1)}$ is finite and $\text{p}(\mathcal{S}_{(p_0,q_0)} \cap \mathcal{S}_{(p_1,q_1)})$ divides $(\frac{p_1}{q_1} - \frac{p_0}{q_0})q$, where $q := \text{lcm}(q_0, q_1)$ is the least common multiple.

Fig. 8

Example of space–time diagram for the traffic rule ECA184 (see [Example 8](#)).



Example 9 (Kůrka 2005) There exists a cellular automaton with infinitely many signal subshifts.

Consider a CA with alphabet $A = \{b, s, r, 1\}$, where b is blank, s is a stationary particle, r is a right-going particle, and 1 is a left-going particle. When either of these two particles meets s , it shifts it to the right and changes to the other moving particle. If an r collides with an 1 , one of them annihilates. The transition table $f: A^4 \rightarrow A$ is given by

xxbl	xblx	xxry	xryx	xxs1	xslx	s1xx	xxrs	xrsx	rsxx
------	------	------	------	------	------	------	------	------	------

where $x \in A$, $y \in \{b, r, 1\}$, and the first applicable rule is used. We have $m = 2$ and $d = 3$, so $F(x)_i = f(x_{[i-2, i+1]})$. Then, for every $n \geq 0$, $\mathcal{S}_{(1, 2n+3)}$ is a signal subshift. In [Fig. 9](#) one can see signals with speeds $1/3$, $1/5$, and $1/7$.

Theorem 11 (Formenti and Kůrka 2007) Let (X, F) be a cellular automaton with diameter d and assume that $\mathcal{S}_{(p_1, q_1)}, \dots, \mathcal{S}_{(p_n, q_n)}$ are signal subshifts with decreasing speeds, i.e., $p_i/q_i > p_j/q_j$ for $i < j$. Let $c > 0$ be an integer which satisfies the following inequalities:

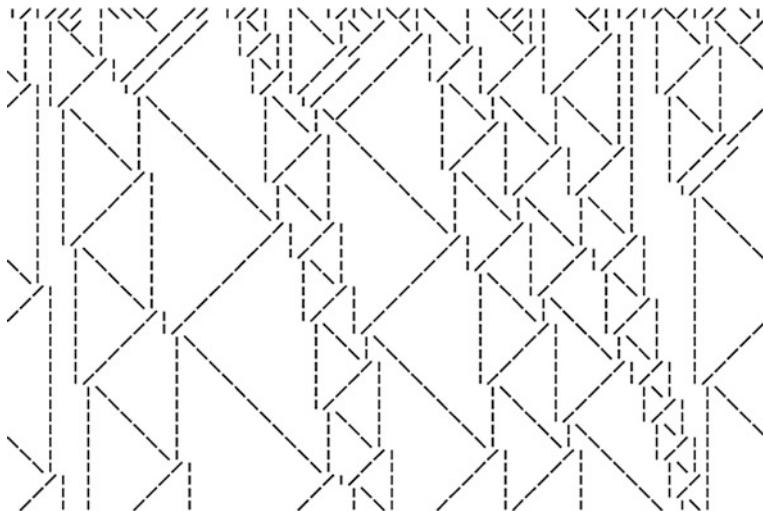
$$\begin{aligned} \mathfrak{o}(X) &\leq c \\ \forall i \leq n, \mathfrak{o}(\mathcal{S}_{(p_i, q_i)}) &\leq c \\ \forall i < j \leq n, (\mathcal{S}_{(p_i, q_i)} \cap \mathcal{S}_{(p_j, q_j)}) &\neq \emptyset \implies \mathfrak{p}(\mathcal{S}_{(p_i, q_i)} \cap \mathcal{S}_{(p_j, q_j)}) \leq c \\ \forall i < j \leq n, (\mathcal{S}_{(p_i, q_i)} \cap \mathcal{S}_{(p_j, q_j)}) &\neq \emptyset \implies \left(d - \frac{p_i}{q_i} + \frac{p_j}{q_j}\right)q \leq c \end{aligned}$$

where $q = \text{lcm}\{q_1, \dots, q_n\}$. Then for $\Sigma := \mathcal{S}_{(p_1, q_1)} \overset{c}{\vee} \dots \overset{c}{\vee} \mathcal{S}_{(p_n, q_n)}$ the following assertions hold:

1. $\Sigma \subseteq F^q(\Sigma)$ and therefore $\Sigma \subseteq \Omega_F(X)$.
2. If G is the graph of Σ constructed by [Proposition 2](#) from graphs of $\mathcal{S}_{(p_i, q_i)}$, and if H is a strongly connected component of G , then Σ_H is F^q -invariant.

Fig. 9

Infinitely many signal subshifts.



3. If G_k is the graph of $F^{kq}(\Sigma)$ constructed to \blacktriangleright [Proposition 4](#) from G , then for each strongly connected component H_k of G_k there exists a strongly connected component H of G such that $\Sigma_{H_k} = \Sigma_H$.

Corollary 2 Let (X, F) be a CA. If $\Omega_F(X) = F^k(\mathcal{S}_{(p_1, q_1)} \vee \dots \vee \mathcal{S}_{(p_n, q_n)})^c$ for some $(p_i, q_i) \in \mathbb{Z} \times \mathbb{N}^+$ and $k, c \geq 0$, then (X, F) has only a finite number of infinite transitive signal subshifts.

4.6 Decreasing Preimages

Notation

Given a language $L \subseteq A^*$ and $a, b \in \mathbb{N}$, $a < b$, define the following language:

$$L_{[a,b)} = \{v \in A^* : |v| \geq a + b, v_{[a,|v|-b)} \in L\}$$

Definition 6 Let $f : \mathcal{L}(X) \rightarrow \mathcal{L}(X)$ be a local rule of a cellular automaton (X, F) . We say that a subshift $\Sigma \subseteq X$ has **r -decreasing preimages** if for each $u \in \mathcal{L}(X) \cap \mathcal{D}(\Sigma)$, each $v \in f^{-r}(u)$ contains as a subword a word $w \in \mathcal{D}(\Sigma)$ such that $|w| < |u|$. We say that Σ has **decreasing preimages** if it has r -decreasing preimages for some $r > 0$.

The condition of \blacktriangleright [Definition 6](#) is satisfied trivially if $f^{-r}(u) = \emptyset$. Thus, for example, each $F^r(X)$ has r -decreasing preimages.

Theorem 12 (Formenti et al. 2010) If (X, F) is a CA and if a subshift $\Sigma \subseteq X$ has decreasing preimages, then $\Omega_F(X) \subseteq \Sigma$. If moreover $\Sigma \subseteq F^k(\Sigma)$ for some $k > 0$, then $\Omega_F(X) = \Sigma$.

Proposition 10 (Formenti and Kůrka 2007) *There exists an algorithm that decides whether for a given cellular automaton (X, F) and given $r > 0$, a given sofic subshift $\Sigma \subseteq X$ has r -decreasing preimages.*

The product CA (see **Example 3**) has two infinite nontransitive signal subshifts (see **Fig. 10**) $\mathcal{S}_{(1,1)} = \Sigma_{10}$, $\mathcal{S}_{(-1,1)} = \Sigma_{01}$, whose orders are $\sigma(\mathcal{S}_{(1,1)}) = \sigma(\mathcal{S}_{(-1,1)}) = 2$. Their intersection is $\{0^\infty, 1^\infty\}$, so, by **Theorem 11**, $p(\mathcal{S}_{(1,1)}) \cap \mathcal{S}_{(-1,1)} = 1$. Finally $\left(d - \frac{p_1}{q_1} + \frac{p_2}{q_2}\right)q = 0$, so, by **Theorem 11**, $c = 2$. The maximal attractor is constructed in

Fig. 10. In the first row, labeled graphs for 2-block encodings of $\mathcal{S}_{(1,1)}$ and $\mathcal{S}_{(-1,1)}$ are given. Their join $\Sigma := \mathcal{S}_{(1,1)} \vee \mathcal{S}_{(-1,1)} = \Sigma_{\{10^n: n>0\} \cup \{010\}}$ is constructed in the second row. In the third row, there is the minimal presentation of (2-block encoding of) $F(\Sigma) = \Sigma_{\{10^n: n>0\}}$. We have $F^2(\Sigma) = F(\Sigma)$ and $F(\Sigma)$ has 1-decreasing preimages: If $u \in \mathcal{D}(F(\Sigma))$, then $u = 10^n1$ for some $n > 0$. If $n \leq 2$, then u has no preimage. If $n > 2$, then the only preimage $110^{n-2}111$ of u contains a shorter forbidden word $10^{n-2}1$. Thus $f^{-1}(\mathcal{D}(F(\Sigma)) \cap \mathcal{L}(F(\Sigma))_{[2,1]}) = \emptyset$ as well as $f^{-1}(\mathcal{D}(F(\Sigma)) \cap \mathcal{L}(F(\Sigma))_{[1,2]}) = \emptyset$. Thus $F(\Sigma)$ has 1-decreasing preimages and therefore $F(\Sigma) = \Omega_F(A^\mathbb{Z})$.

The majority CA (see **Example 4**) has a spreading set $U := A^3 \setminus \{010, 101\}$, whose subshift attractor is the signal subshift $\Omega_F(U) = \mathcal{S}_{(0,1)} = \Sigma_{\{010, 101\}}$. There are two other infinite signal subshifts $\mathcal{S}_{(1,1)} = \Sigma_{\{011, 100\}}$, $\mathcal{S}_{(-1,1)} = \Sigma_{\{001, 110\}}$. The join Σ of all three signal subshifts is constructed in **Fig. 11**. We have $\sigma(\mathcal{S}_{(1,1)}) = \sigma(\mathcal{S}_{(0,1)}) = \sigma(\mathcal{S}_{(-1,1)}) = 3$, $\mathcal{S}_{(1,1)} \cap \mathcal{S}_{(-1,1)} = \{0^\infty, 1^\infty, (01)^\infty, (10)^\infty\}$, $\mathcal{S}_{(1,1)} \cap \mathcal{S}_{(0,1)} = \mathcal{S}_{(0,1)} \cap \mathcal{S}_{(-1,1)} = \{0^\infty, 1^\infty\}$, and $\left(d - \frac{p_1}{q_1} + \frac{p_2}{q_2}\right)q \leq 2$, so $c = 3$. The arrows from $\mathcal{S}_{(1,1)}$ to $\mathcal{S}_{(0,1)}$ and from $\mathcal{S}_{(0,1)}$

Fig. 10

The signal subshifts of ECA128.

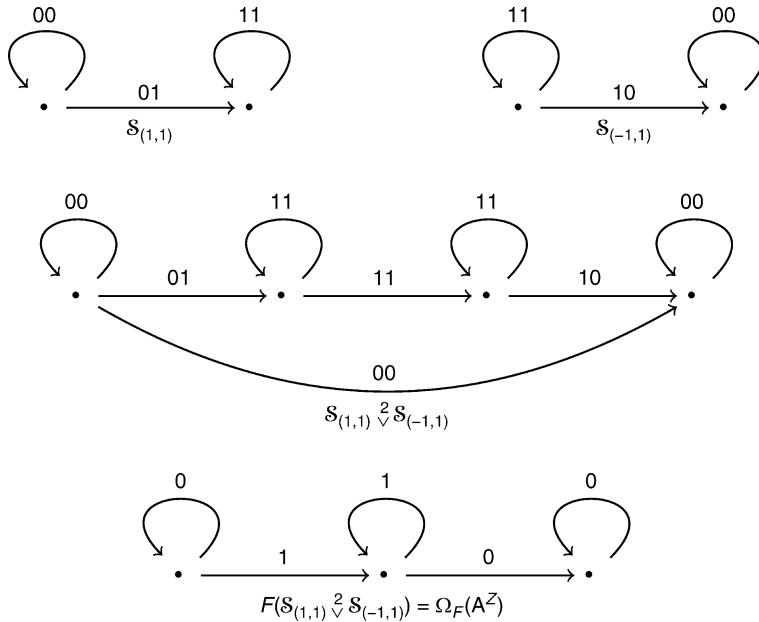
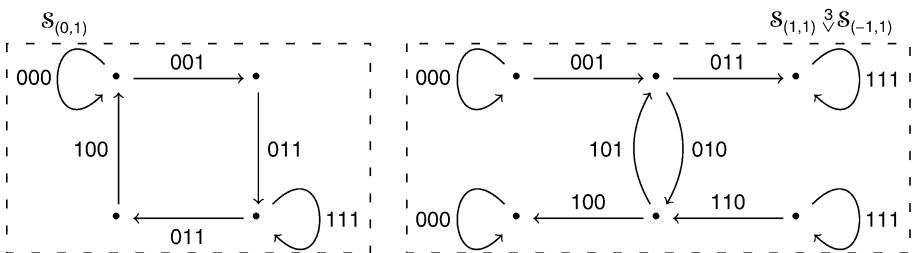
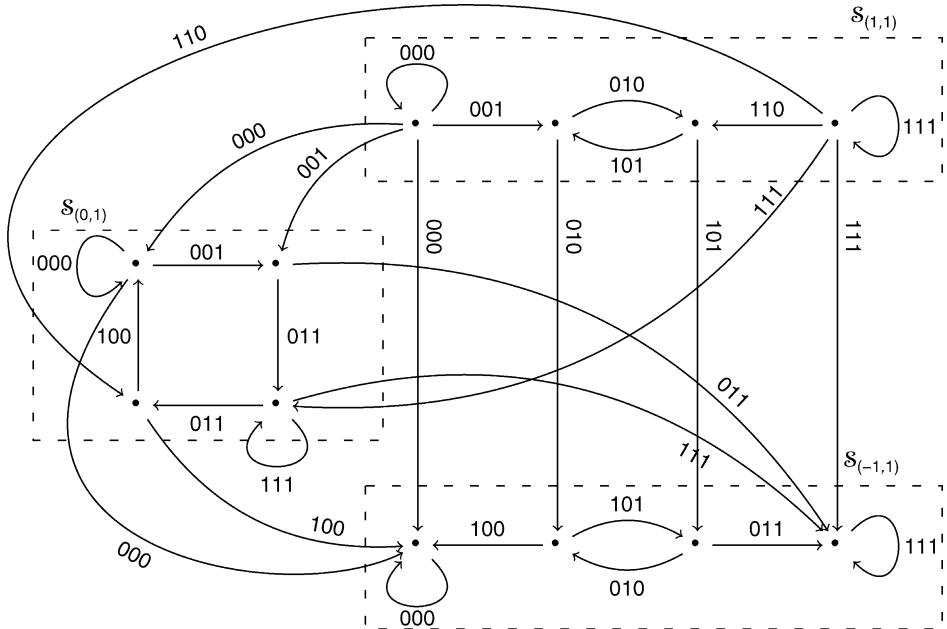


Fig. 11

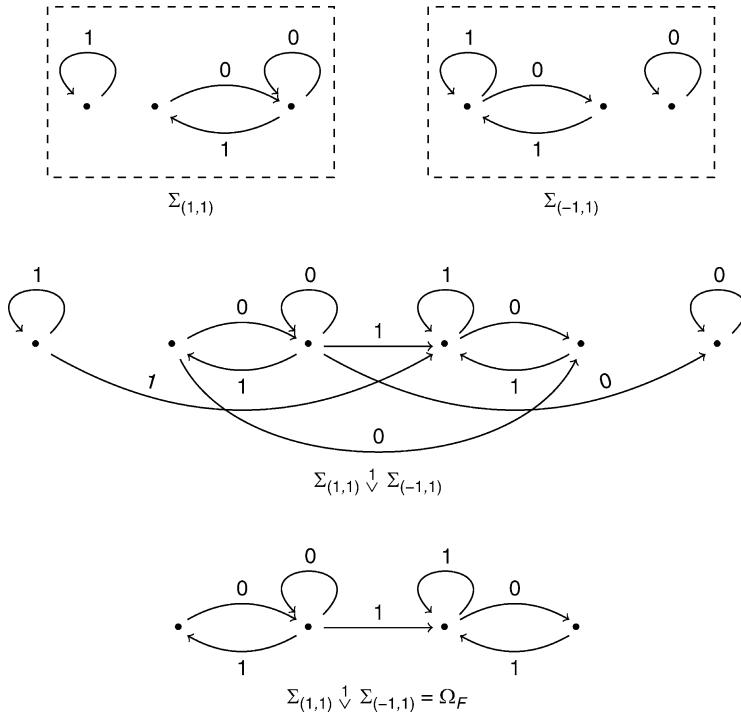
The join $\mathcal{S}_{(0,1)} \stackrel{3}{\vee} \mathcal{S}_{(1,1)} \stackrel{3}{\vee} \mathcal{S}_{(-1,1)}$ of the majority ECA232 (top) and the maximal attractor $\Omega_F = \mathcal{S}_{(0,1)} \cup (\mathcal{S}_{(1,1)} \stackrel{3}{\vee} \mathcal{S}_{(-1,1)})$ (bottom).



to $\mathcal{S}_{(-1,1)}$ can be omitted without changing the subshift. Thus we get $\Sigma := \mathcal{S}_{(1,1)} \stackrel{3}{\vee} \mathcal{S}_{(0,1)} \stackrel{3}{\vee} \mathcal{S}_{(-1,1)} = \mathcal{S}_{(0,1)} \cup (\mathcal{S}_{(1,1)} \stackrel{3}{\vee} \mathcal{S}_{(-1,1)})$ (see [Fig. 3](#)). First offenders are $\mathcal{D}(\Sigma) = \{010^n 1, 10^n 10, 01^n 01, 101^n 0 : n > 1\}$. We have $F(\Sigma) = \Sigma$ and Σ has 1-decreasing preimages. To show it, note that 010, 101 have unique preimages $f^{-1}(010) = \{01010\}$, $f^{-1}(101) = \{10101\}$. For the first offenders we get $f^{-1}(01001) = \emptyset$, $f^{-1}(010001) = \{01010011\}$. If $n \geq 4$, then each preimage of $010^n 1$ has one of the form $01010u0011$ or $01010u0101$, where $|u| = n - 4$ and in each case it contains a shorter forbidden word. We get the same result for other forbidden words. Thus $f^{-1}(\mathcal{D}(\Sigma)) \cap \mathcal{L}(\Sigma)_{[2,1]} \cap \mathcal{L}(\Sigma)_{[1,2]} = \emptyset$, Σ has 1-decreasing preimages, and $\Omega_F(A^{\mathbb{Z}}) = \Sigma$.

Fig. 12

Signal subshifts of ECA184.



The traffic rule of [Example 8](#) has two signal subshifts (see [Fig. 12](#)) representing holes and jams

$$\begin{aligned}\mathcal{S}_{(1,1)} &= \{x \in A^{\mathbb{Z}} : 11 \not\subseteq x\} \cup \{1^{\mathbb{Z}}\} \\ \mathcal{S}_{(-1,1)} &= \{x \in A^{\mathbb{Z}} : 00 \not\subseteq x\} \cup \{0^{\mathbb{Z}}\}\end{aligned}$$

Their intersection is the finite subshift $\{0^{\mathbb{Z}}, 1^{\mathbb{Z}}, (01)^{\mathbb{Z}}, (10)^{\mathbb{Z}}\}$. There is one more (nontransitive) signal subshift $\Sigma_{(0,1)} = \{x \in A^{\mathbb{Z}} : \forall i < j, x_i \leq x_j\}$. The maximal attractor is constructed in [Fig. 12](#). In the first row, 1-distinguishing presentations for $\Sigma_{(1,1)}$ and $\Sigma_{(-1,1)}$ are constructed. Their join is constructed in the second row. In the third row, the minimal presentation of $\Sigma_{(1,1)} \stackrel{1}{\vee} \Sigma_{(-1,1)}$ is given. As it has decreasing preimages, it equals Ω_F .

4.7 Measures

Given a metric space X , the family $\mathcal{B}(X) \subseteq \mathcal{P}(X)$ of Borel sets is the smallest family of sets, which contains open sets and is closed with respect to the operations of countable union and complementation. A **Borel probability measure** on a metric space X is a countably additive

function $\mu : \mathcal{B}(X) \rightarrow [0, 1]$ that assigns 1 to the full space. The (topological) **support** of a measure μ is

$$\Sigma_\mu := \bigcap \{V \subseteq X : \mu(V) = 1 \text{ and } V \text{ is closed}\}$$

If $\Sigma_\mu = X$, it can be said that μ has **full support**. A measure on $A^\mathbb{Z}$ is determined by its values on centered cylinders $\mu(u) := \mu([u]_{-n})$ for $u \in A^{2n} \cup A^{2n+1}$. The map $\mu : A^* \rightarrow [0, 1]$ satisfies the **Kolmogorov compatibility conditions**

$$\mu(\lambda) = 1, \sum_{a \in A} \mu(ua) = \mu(u) \text{ for } |u| \text{ even}, \quad \sum_{a \in A} \mu(au) = \mu(u) \text{ for } |u| \text{ odd}$$

Conversely, every function $\mu : A^* \rightarrow [0, 1]$ satisfying the Kolmogorov compatibility conditions determines a measure. Each point $x \in A^\mathbb{Z}$ determines the Dirac point measure δ_x defined by

$$\delta_x(U) = \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{if } x \notin U \end{cases}$$

For each positive probability vector $p = (p_a)_{a \in A}$, we have the Bernoulli measure defined by

$$\mu(u) = p_{u_0} \cdots p_{u_{n-1}}, \quad u \in A^n$$

A stochastic matrix $R = (R_{ab})_{a,b \in A}$ is a matrix with nonnegative entries such that the sum of each row is one: $\sum_{b \in A} R_{a,b} = 1$. A stochastic matrix R together with a probability vector p determines a Markov measure

$$\mu(u) = p_{u_0} \cdot R_{u_0, u_1} \cdots R_{u_{n-2}, u_{n-1}}, \quad u \in A^n$$

Given two measures μ_0, μ_1 , and positive real numbers a_0, a_1 with sum $a_0 + a_1 = 1$, then the convex combination $\mu(U) := a_0\mu_0(U) + a_1\mu_1(U)$ is a measure. If μ_0 and μ_1 are shift-invariant, then so is μ .

Denote by $\mathfrak{M}(A^\mathbb{Z})$ the set of all Borel probability measures on $A^\mathbb{Z}$. On $\mathfrak{M}(A^\mathbb{Z})$ we have the weak* topology. A sequence of measures μ_n converges to a measure μ if

$$\lim_{n \rightarrow \infty} \int f \, d\mu_n = \int f \, d\mu$$

for every continuous function $f : A^\mathbb{Z} \rightarrow \mathbb{R}$. The space of Borel probability measures with the weak* topology is compact and metrizable. A compatible metric is

$$\rho(\mu, v) := \sum_{n=0}^{\infty} \max\{|\mu(u) - v(u)| : u \in A^n\} \cdot 2^{-n}$$

Given two configurations $x, y \in A^\mathbb{Z}$ we have

$$\begin{aligned} d(x, y) &= 2^{-n} \& x_{-n} \neq y_{-n} \implies \rho(\delta_x, \delta_y) = 2^{-2n+1} \\ d(x, y) &= 2^{-n} \& x_{-n} = y_{-n} \implies \rho(\delta_x, \delta_y) = 2^{-2n} \end{aligned}$$

Thus the map $\delta : (A^\mathbb{Z}, d) \rightarrow (\mathfrak{M}(A^\mathbb{Z}), \rho)$ is continuous. Given two probability vectors p, q , denote by $d := \max_{a \in A} |p_a - q_a|$. Then the distance of the corresponding Bernoulli measures is

$$\rho(\mu, v) = \sum_{n=1}^{\infty} d^n 2^{-n} = \frac{d}{2-d}$$

If $F : A^{\mathbb{Z}} \rightarrow A^{\mathbb{Z}}$ is a continuous map and $\mu \in \mathfrak{M}(A^{\mathbb{Z}})$, then $F\mu$ is a Borel probability measure defined by $(F\mu)(U) = \mu(F^{-1}(U))$ for any Borel set U . A measure μ is **F -invariant**, if $F\mu = \mu$. The map $\mu \mapsto F\mu$ is continuous and **affine**, that is, it preserves convex combinations. Thus, we obtain a dynamical system $F : \mathfrak{M}(A^{\mathbb{Z}}) \rightarrow \mathfrak{M}(A^{\mathbb{Z}})$.

If a measure μ is σ -invariant, then the measure of a cylinder does not depend on its location, so $\mu(u) = \mu([u]_i)$ for every $u \in A^*$ and $i \in \mathbb{Z}$. Thus a σ -invariant measure is determined by a map $\mu : A^* \rightarrow [0, 1]$ subject to **bilateral compatibility conditions**

$$\mu(\lambda) = 1, \quad \sum_{a \in A} \mu(ua) = \sum_{a \in A} \mu(au) = \mu(u), \quad u \in A^*$$

A point measure is not σ -invariant unless x is a σ -fixed point of the form $x = a^\infty$. On the other hand, each Bernoulli measure is σ -invariant. A Markov measure is σ -invariant iff the vector p is a stationary vector of R , that is, if $p \cdot R = p$.

Denote by $\mathfrak{M}_\sigma(A^{\mathbb{Z}}) := \{\mu \in \mathfrak{M}(A^{\mathbb{Z}}) : \sigma\mu = \mu\}$ the closed subspace of σ -invariant measures. The support $\Sigma_\mu := \{x \in A^{\mathbb{Z}} : \forall u \sqsubseteq x, \mu(u) > 0\}$ of a σ -invariant measure μ is a subshift whose forbidden words are the words of zero measure. If $\Sigma \subseteq A^{\mathbb{Z}}$ is a subshift, then

$$\mathfrak{M}_\sigma(\Sigma) := \{\mu \in \mathfrak{M}_\sigma(A^{\mathbb{Z}}) : \Sigma_\mu \subseteq \Sigma\} = \{\mu \in \mathfrak{M}_\sigma(A^{\mathbb{Z}}) : \forall u \in A^* \setminus \mathcal{L}(\Sigma), \mu(u) = 0\}$$

is a closed subspace of $\mathfrak{M}_\sigma(A^{\mathbb{Z}})$. A measure $\mu \in \mathfrak{M}_\sigma(\Sigma)$ can be identified with a map $\mu : \mathcal{L}(\Sigma) \rightarrow [0, 1]$, which satisfies the bilateral compatibility conditions. It can be said that $\mu \in \mathfrak{M}_\sigma(\Sigma)$ is **σ -ergodic**, if for every strongly σ -invariant set $Y \subseteq \Sigma$ either $\mu(Y) = 0$ or $\mu(Y) = 1$. If $\mu \in \mathfrak{M}_\sigma(\Sigma)$, then the support of μ is contained in the **densely recurrent subshift**

$$\mathcal{D}(\Sigma) := \text{cl}\{x \in \Sigma : \forall k > 0, \limsup_{n \rightarrow \infty} \#\{|i| < n : x_{[i,i+k)} = x_{[0,k)}\}/n > 0\}$$

of Σ , that is, $\Sigma_\mu \subseteq \mathcal{D}(\Sigma)$ (see Akin (1991), Proposition 8.8, p. 164). On the other hand, there exists $\mu \in \mathfrak{M}_\sigma(\Sigma)$ such that $\Sigma_\mu = \mathcal{D}(\Sigma)$. If the support of a set $Y \subseteq \mathfrak{M}(A^{\mathbb{Z}})$ is defined as $\mathcal{S}(Y) := \text{cl}(\bigcup_{\mu \in Y} \Sigma_\mu)$, the following proposition can be arrived at.

Proposition 11 *Let $\Sigma \subseteq A^{\mathbb{Z}}$ be a subshift.*

1. $\mathcal{S}(\mathfrak{M}_\sigma(\Sigma)) = \mathcal{D}(\Sigma)$ and therefore, $\mathfrak{M}_\sigma(\Sigma) = \mathfrak{M}_\sigma(\mathcal{D}(\Sigma))$.
2. Let G be a labelled graph and let G_1, \dots, G_k be all its strongly connected components. Then $\mathcal{D}(\Sigma_G) = \Sigma_{G_1} \cup \dots \cup \Sigma_{G_k}$.

If $\mu \in \mathfrak{M}_\sigma(A^{\mathbb{Z}})$, then $(F\mu)(u) = \sum_{v \in f^{-1}(u)} \mu(v)$ is also a σ -invariant measure. Thus we get a dynamical system $F : \mathfrak{M}_\sigma(A^{\mathbb{Z}}) \rightarrow \mathfrak{M}_\sigma(A^{\mathbb{Z}})$, which is a subsystem of $F : \mathfrak{M}(A^{\mathbb{Z}}) \rightarrow \mathfrak{M}(A^{\mathbb{Z}})$. The properties of this dynamical system do not depend on the memory constant m . CA F and $F\sigma^p$ determine the same dynamical system on $\mathfrak{M}_\sigma(A^{\mathbb{Z}})$. If (X, F) is a CA, and $\mu \in \mathfrak{M}_\sigma(X)$, then $F\mu \in \mathfrak{M}_\sigma(X)$, so we get a dynamical system $(\mathfrak{M}_\sigma(X), F)$.

In ECA128 of **Example 3** if a measure μ gives positive probability to $\mu(0) > 0$, then it converges to the point measure δ_{0^∞} . This is, however, not an attractor in $\mathfrak{M}(A^{\mathbb{Z}})$, since any convex combination $a_0\delta_{0^\infty} + a_1\delta_{1^\infty}$ is a fixed point.

In ECA232 of **Example 4**, each measure with support included in the attractor $\Omega_F(A^{\mathbb{Z}} \setminus ([010]_0 \cup [101]_0))$ is a fixed point in $\mathfrak{M}(A^{\mathbb{Z}})$.

4.8 The Measure Attractor

Definition 7 Given a cellular automaton (X, F) , we say that an F -invariant subshift $\Sigma \subseteq X$ **attracts ergodic measures** if there exists $\varepsilon > 0$, such that for every σ -ergodic measure $\mu \in \mathfrak{M}_\sigma(X)$ such that $\rho(\mu, \mathfrak{M}_\sigma(\Sigma)) < \varepsilon$, we have $\rho(F^n\mu, \mathfrak{M}_\sigma(\Sigma)) \rightarrow 0$ as $n \rightarrow \infty$.

Proposition 12 Let (X, F) be a cellular automaton and let $D \subseteq \mathcal{L}(X)^*$ be a set of forbidden words. Then for every $\mu \in \mathfrak{M}_\sigma(X)$ we have

$$\lim_{n \rightarrow \infty} \rho(F^n\mu, \mathfrak{M}_\sigma(\Sigma_D)) = 0 \iff \forall u \in D, \quad \lim_{n \rightarrow \infty} (F^n\mu)(u) = 0$$

Proposition 13 Let (X, F) be a cellular automaton, and $\Sigma \subseteq X$ a subshift attractor with basin $\mathscr{B}_F(\Sigma) = \{x \in X : \lim_{n \rightarrow \infty} d(x, \Sigma) = 0\}$. If $\mu \in \mathfrak{M}_\sigma(X)$ and $\mu(\mathscr{B}_F(\Sigma)) = 1$, then $\rho(F^n\mu, \mathfrak{M}_\sigma(\Sigma)) \rightarrow 0$ as $n \rightarrow \infty$. In particular, for all $\mu \in \mathfrak{M}_\sigma(X)$, $\rho(F^n\mu, \mathfrak{M}_\sigma(\Omega_F(X))) \rightarrow 0$ as $n \rightarrow \infty$.

Proposition 14 (Kůrka 2005) A subshift attractor for F attracts ergodic measures.

Definition 8 Given a CA (X, F) and a measure $\mu \in \mathfrak{M}_\sigma(X)$, the μ -limit $\Lambda_\mu(F)$ is defined by the condition

$$u \notin \mathcal{L}(\Lambda_\mu(F)) \Leftrightarrow \lim_{n \rightarrow \infty} F^n\mu([u]_0) = 0, \quad \forall u \in \mathcal{L}(X)$$

Proposition 15 If $(A^\mathbb{Z}, F)$ is a CA, $\mu \in \mathfrak{M}(A^\mathbb{Z})$ a σ -ergodic measure, and $\Sigma \subseteq A^\mathbb{Z}$ a μ -attractor, then $\Lambda_\mu(F) \subseteq \Sigma$. In particular, $\Lambda_\mu(F) \subseteq \Omega_F(A^\mathbb{Z})$.

Proposition 16 (Kůrka 2005) If $\lim_{n \rightarrow \infty} F^n\mu = v$ in $\mathfrak{M}(X)$, then $\Lambda_\mu(F) = \Sigma_v$.

Proposition 17 (Kůrka 2005) If (X, F) is a cellular automaton, then $F : \mathfrak{M}_\sigma(X) \rightarrow \mathfrak{M}_\sigma(X)$ has a unique attractor $\Omega_F(\mathfrak{M}_\sigma(X)) = \mathfrak{M}_\sigma(\mathcal{D}(\Omega_F(X)))$.

Definition 9 Let (X, F) be a cellular automaton.

1. The **measure attractor** of F is $\mathcal{M}_F := \mathcal{S}(\Omega_F(\mathfrak{M}_\sigma(X))) = \mathcal{D}(\Omega_F(X))$, where the second equality is by [Proposition 17](#).
2. The **measure quasi-attractor** of F is $\mathcal{M}\mathcal{Q}_F := \mathcal{D}(\mathcal{Q}_F)$.

Proposition 18 (Pivato 2008) If (X, F) is a cellular automaton, then $\Lambda_\mu(F) \subseteq \mathcal{M}_F$ for every $\mu \in \mathfrak{M}_F(X)$,

$$\begin{aligned} \mathcal{L}(\mathcal{M}_F) &= \overline{\bigcup \{\Lambda_\mu(F) : \mu \in \mathfrak{M}_\sigma(X)\}} \\ \mathcal{M}_F &= \overline{\bigcup \{\Lambda_\mu(F) : \mu \in \mathfrak{M}_\sigma(X)\}} \end{aligned}$$

where $\mathfrak{M}_F(X)$ is the set of F -invariant measures in $\mathfrak{M}(X)$.

Corollary 3 (Kůrka 2003) Let (X, F) be a cellular automaton.

1. If $\mu \in \mathfrak{M}_\sigma(X)$, then $\rho(F^n\mu, \mathfrak{M}_\sigma(\mathcal{M}_F)) \rightarrow 0$ as $n \rightarrow \infty$.
2. \mathcal{M}_F attracts ergodic measures.

The maximal attractor of ECA128 is a nontransitive sofic subshift with strongly connected components for 0^∞ and 1^∞ . The measure attractor of ECA128 is therefore $\mathcal{M}_F = \{0^\infty, 1^\infty\}$. The measure quasi-attractor is $\mathcal{M}_F = \{0^\infty\}$. The measure attractor of ECA232 is $\mathcal{M}_F = \mathcal{M}_F = \{0^\infty, 1^\infty\}$. The measure attractor and quasi-attractor of ECA184 are $\mathcal{M}_F = \mathcal{M}_F = \mathcal{S}_{(-1,1)} \cup \mathcal{S}_{(1,1)}$.

4.9 Generic Space

The Besicovitch pseudometric on $A^\mathbb{Z}$ is defined by

$$d_B(x, y) = \limsup_{n \rightarrow \infty} \#\{i \in [-n, n] : x_i \neq y_i\} / 2n$$

For a configuration $x \in A^\mathbb{Z}$ and words $u, v \in A^+$ set

$$\begin{aligned} \varphi_v(u) &= \#\{i \in [0, |u| - |v|] : u_{[i, i+|v|]} = v\} \\ \varphi_v(x) &= \#\{i \in \mathbb{Z} : x_{[i, i+|v|]} = v\} \\ \varphi_v^n(x) &= \#\{i \in [-n, n] : x_{[i, i+|v|]} = v\} \\ \underline{\Phi}_v(x) &= \liminf_{n \rightarrow \infty} \varphi_v^n(x) / 2n \\ \overline{\Phi}_v(x) &= \limsup_{n \rightarrow \infty} \varphi_v^n(x) / 2n \end{aligned}$$

For every $v \in A^*$, $\underline{\Phi}_v, \overline{\Phi}_v : A^\mathbb{Z} \rightarrow [0, 1]$ are continuous in the Besicovitch topology. In fact we have

$$|\overline{\Phi}_v(x) - \overline{\Phi}_v(y)| \leq d_B(x, y)|v|, \quad |\underline{\Phi}_v(x) - \underline{\Phi}_v(y)| \leq d_B(x, y)|v|$$

Define the generic space (over the alphabet A) as

$$\mathcal{G}_A = \{x \in A^\mathbb{Z} : \forall v \in A^*, \underline{\Phi}_v(x) = \overline{\Phi}_v(x)\}$$

It is a closed subspace of $A^\mathbb{Z}$ (in the Besicovitch topology). For $v \in A^*$ denote by $\Phi_v : \mathcal{G}_A \rightarrow [0, 1]$ the common value of $\underline{\Phi}_v$ and $\overline{\Phi}_v$. Also for $x \in \mathcal{G}_A$ denote by $\Phi^x : A^* \rightarrow [0, 1]$ the function $\Phi^x(v) = \Phi_v(x)$. For every $x \in \mathcal{G}_A$, Φ^x is a shift-invariant Borel probability measure. The map $\Phi : \mathcal{G}_A \rightarrow \mathfrak{M}_\sigma(A^\mathbb{Z})$ is continuous with respect to the Besicovitch and weak topologies. In fact we have

$$d_M(\Phi^x, \Phi^y) \leq \sum_{n=1}^{\infty} d_B(x, y)n2^{-n} \leq 2d_B(x, y)$$

By the Kakutani theorem, Φ is surjective. Every shift-invariant Borel probability measure has a generic point. It follows from the Ergodic theorem that if μ is a σ -invariant measure, then $\mu(\mathcal{G}_A) = 1$ and for every $v \in A^*$, the measure of v is the integral of its density Φ_v ,

$$\mu(v) = \int \Phi_v(x) d\mu.$$

If $F : A^{\mathbb{Z}} \rightarrow A^{\mathbb{Z}}$ is a CA and $\tilde{F} : \mathfrak{M}_{\sigma}(A^{\mathbb{Z}}) \rightarrow \mathfrak{M}_{\sigma}(A^{\mathbb{Z}})$ its extension, then we have a commutative diagram $\Phi F = \tilde{F} \Phi$.

$$\begin{array}{ccc} \mathcal{G}_A & \xrightarrow{F} & \mathcal{G}_A \\ \Phi \downarrow & & \downarrow \Phi \\ \mathfrak{M}_{\sigma}(A^{\mathbb{Z}}) & \xrightarrow{\tilde{F}} & \mathfrak{M}_{\sigma}(A^{\mathbb{Z}}) \end{array}$$

4.10 Particle Weight Functions

For a function $p : A^+ \rightarrow \mathbb{N}$ denote by

$$\mathcal{S}_p = \{u \in A^+ : p(u) \neq 0\}, \quad \Sigma_p = \{x \in A^{\mathbb{Z}} : \forall v \sqsubseteq x, p(v) = 0\}$$

its support and the subshift with forbidden set \mathcal{S}_p , respectively.

Definition 10 A particle weight function over an alphabet A is a map $p : A^* \rightarrow \mathbb{N}$ such that $p(\lambda) = 0$, \mathcal{S}_p is finite and Σ_p is nonempty. The bound and order of p are defined by

$$b = \max\{p(u) : u \in A^*\}, \quad s = \max\{|u| : u \in \Sigma_p\}$$

The (total) weight of a word $u \in A^*$, or of a configuration $x \in A^{\mathbb{Z}}$ is defined by

$$\begin{aligned} \varphi_p(u) &= \sum_{v \in A^+} \varphi_v(u) \cdot p(v) = \sum_{i=0}^{|u|-1} \sum_{j=i+1}^{|u|} p(u_{[i,j]}) \\ \varphi_p(x) &= \sum_{v \in A^+} \varphi_v(x) \cdot p(v) = \sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} p(x_{[i,j]}) \\ \varphi_p^n(x) &= \sum_{v \in A^+} \varphi_v^n(x) \cdot p(v) = \sum_{i=-n}^n \sum_{j=i+1}^{\infty} p(x_{[i,j]}) \end{aligned}$$

The set of p -finite configurations is denoted by

$$\mathcal{F}_p = \{x \in A^{\mathbb{Z}} : \varphi_p(x) < \infty\}$$

The mean weight of a configuration $x \in \mathcal{G}_A$ and the weight of a measure $\mu \in \mathfrak{M}(A^{\mathbb{Z}})$ are defined by

$$\begin{aligned} \Phi_p(x) &= \sum_{v \in A^+} \Phi_v(x) \cdot p(v) = \lim_{n \rightarrow \infty} \varphi_p^n(x) / (2n + 1) \\ \Psi_p(\mu) &= \sum_{v \in A^+} \mu(v) \cdot p(v) \end{aligned}$$

In particular, if $x = u^\infty$ is a σ -periodic configuration, δ_u is the measure whose support is the σ -orbit of u , and if $k > 1$ is the least integer such that $(k - 1)|u| \geq s$, then

$$\Psi_p(\delta_u) = \Phi_p(u^\infty) = \sum_{i=0}^{|u|-1} \sum_{j=i+1}^{i+s} p((u^k)_{[i,j]}) / |u|$$

Lemma 1 For $u, v \in A^+$ and $x \in A^\mathbb{Z}$ we have

$$\begin{aligned}\varphi_p(u) &\leq bs|u|, \quad \varphi_p(uv) \leq \varphi_p(u) + \varphi_p(v) + bs^2, \\ \varphi_p(x_{[-n,n]}) &\leq \varphi_p^n(x) \leq \varphi_p(x_{[-n,n]}) + bs^2 \leq bs(2n+1+s)\end{aligned}$$

If $0 \leq k < l \leq |u| - s$ and $u_{[k,k+s]} = u_{[l,l+s]}$, then

$$\varphi_p(u) = \varphi_p(u_{[0,k]} u_{[l,|u|]}) + (l-k)\Phi_p((u_{[k,l]})^\infty)$$

The function $\Phi_p : \mathcal{G}_A \rightarrow [0, \infty)$ is continuous in the Besicovitch topology and the weight map $\Psi_p : \mathfrak{M}(A^\mathbb{Z}) \rightarrow [0, \infty)$ is bounded, affine, and continuous in the weak topology. We have a commutative diagram $\Psi_p \Phi = \Phi_p$. If $x \in \mathcal{G}_A$ and $\mu \in \mathfrak{M}(A^\mathbb{Z})$, then

$$\Phi_p(x) \leq bs, \quad \Psi_p(\mu) = \int \Phi_p(x) d\mu \leq bs$$

Theorem 13 We say that $p : A^* \rightarrow \mathbb{N}$ is a Lyapunov particle weight function for a cellular automaton $F : A^\mathbb{Z} \rightarrow A^\mathbb{Z}$, if one of the following equivalent conditions is satisfied:

1. There exists a local rule f for F such that $\forall u \in A^*, \varphi_p(f(u)) \leq \varphi_p(u)$.
2. For any $x \in \mathcal{F}_p$, $\varphi_p(F(x)) \leq \varphi_p(x)$.
3. For any $x \in \mathcal{G}_A$, $\Phi_p(F(x)) \leq \Phi_p(x)$.
4. For any σ -periodic $x \in A^\mathbb{Z}$, $\Phi_p(F(x)) \leq \Phi_p(x)$.
5. For any $\mu \in \mathfrak{M}(A^\mathbb{Z})$, $\Psi_p(F\mu) \leq \Psi_p(\mu)$.

Proof (1) \rightarrow (2): Let $f : A^{d+1} \rightarrow A$ be a local rule for F and assume that the memory is $m = 0$. If $x \in \mathcal{F}_p$ then there exists n such that all particles of x occur in the interval $[-n, n+d]$ and all particles of $F(x)$ occur in the interval $[-n, n]$. Then, by \bullet Lemma 1, it holds $\varphi_p(F(x)) = \varphi_p(F(x)_{[-n,n]}) \leq \varphi_p(x_{[-n,n+d]}) = \varphi_p(x)$.

(2) \rightarrow (3): For $n > 0$ let $y \in \mathcal{F}_p$ be a configuration such that $y_{[-n,n]} = x_{[-n,n]}$ and y contains no particles in $(-\infty, -n)$ and (n, ∞) . Then

$$\begin{aligned}\varphi_p^{n-d}(F(x)) &\leq \varphi_p(F(x)_{[-n+d,n-d]}) + bs^2 \leq \varphi_p(F(y)) + bs^2 \\ &\leq \varphi_p(y) + bs^2 \leq \varphi_p^n(x) + 3bs^2\end{aligned}$$

(3) \rightarrow (4) is trivial.

(4) \rightarrow (2): Let $x \in \mathcal{F}_p$ and choose n so that all particles of x are within the interval $[-n+s+d, n-s-d]$. For the σ -periodic configuration $y = (x_{[-n,n]})^\infty$ we have

$$\varphi_p(F(x)) = (2n+1)\Phi_p(F(y)) \leq (2n+1)\Phi_p(y) = \varphi_p(x)$$

(2) & (4) \rightarrow (1): Let $g : A^{e+1} \rightarrow A$ be a local rule for F . We show first that there exists $c \geq 0$ such that

$$\forall x \in \mathcal{F}_p, \forall j \leq |A|^{s+e}, \quad \varphi_p(F(x)_{[0,j]}) \leq \varphi_p(x_{[-c,j+c]})$$

Assume that the above condition does not hold. Then, for every $c > 0$ there exists $x^c \in \mathcal{F}_p$ and $j \leq |A|^{e+s}$ such that $\varphi_p(F(x^c)_{[0,j]}) > \varphi_p(x^c_{[-c,j+c]})$. Now, consider the sequence of x^c and extract a converging subsequence. Assume that this subsequence converges to $x \in A^\mathbb{Z}$. It is

clear that $\varphi_p(F(x)_{[0,|A|^{e+s})}) > \varphi_p(x)$. Thus $x \in \mathcal{F}_p$ and $\varphi_p(F(x)) > \varphi_p(x)$ and this is a contradiction. Next we prove that for all $x \in \mathcal{F}_p$ and all $i < j$ we have

$$\varphi_p(F(x)_{[i,j)}) \leq \varphi_p(x_{[i-c,j+c]})$$

We already proved that the condition holds for $j - i \leq |A|^{e+s}$. If the condition does not hold for some $j - i > |A|^{e+s}$, then there exist $i \leq k < l \leq j$ such that $x_{[k,k+e+s)} = x_{[l,l+e+s)}$ and $F(x)_{[k,k+s)} = F(x)_{[l,l+s)}$. Set $y = x_{(-\infty,k)}x_{[l,\infty)}$. Then

$$\begin{aligned} \varphi_p(F(y)_{[i,j-l+k]}) &= \varphi_p(F(x)_{[i,j]}) - (l - k)\Phi_p((F(x)_{[k,l]})^\infty) \\ &> \varphi_p(x_{[i-c,j+c]}) - (l - k)\Phi_p((x_{[k,l]})^\infty) \\ &\geq \varphi_p(y_{[i-c,j-l+k+c]}) \end{aligned}$$

and this is a contradiction. Set now $d = e + 2c$ and define $f: A^{d+1} \rightarrow A$ by $f(u) = g(u_{[c,|u|-c]})$. Then f is a local rule for F and $\varphi_p(f(u)) \leq \varphi_p(u)$.

(3) \rightarrow (5): By the Ergodic theorem,

$$\Psi_p(F\mu) = \int \Phi_p(F(x)) d\mu \leq \int \Phi_p(x) d\mu = \Psi_p(\mu)$$

(5) \rightarrow (4): If $x = u^\infty$, then $\Phi_p(Fx) = \Psi_p(F\delta_u) \leq \Psi_p(\delta_u) = \Phi_p(x)$

Definition 11 Let $F: A^\mathbb{Z} \rightarrow A^\mathbb{Z}$ be a cellular automaton and let $p: A^* \rightarrow \mathbb{N}$ be a Lyapunov particle weight function for F . A word $v \in A^+$ is (F, p) -reducing, if there exists $r(v) > 0$, such that for every $x \in \mathcal{F}_p$, $\varphi_p(F(x)) \leq \varphi_p(x) - r(v) \cdot \varphi_v(x)$. Denote by $\Sigma_{(F,p)}^0$ the subshift whose forbidden words are (F, p) -reducing words and

$$\Sigma_{(F,p)} = \bigcap_{n=-\infty}^{\infty} F^n(\Sigma_{(F,p)}^0)$$

Proposition 19 Let v be a (F, p) -reducing word. Then for every $\mu \in \mathfrak{M}(A^\mathbb{Z})$ we have $\rho(F^n\mu, \mathfrak{M}(\Sigma_{\{v\}})) \rightarrow 0$ as $n \rightarrow \infty$.

Proof We have $\Psi_p(F^{j+1}\mu) \leq \Psi_p(F^j\mu) - r(v) \cdot (F^j\mu)(v)$. Adding these inequalities for $j = 0, \dots, k$ we get

$$r(v) \cdot (\mu(v) + \dots + (F^k\mu)(v)) \leq \Psi_p(\mu) - \Psi_p(F^{k+1}\mu) \leq \Psi_p(\mu)$$

Thus $\sum_{k=0}^{\infty} (F^k\mu)(v)$ converges and therefore $\lim_{k \rightarrow \infty} (F^k\mu)(v) = 0$.

Corollary 4 If p is a Lyapunov particle weight function for F then $\mathcal{M}_F \subseteq \Sigma_{(F,p)}$.

Proof If $f^n(w)$ is a reducing word for some $n \geq 0$, then $F^m\mu(v) \rightarrow 0$ as $m \rightarrow \infty$, so $\mathcal{M}_F(A^\mathbb{Z}) \subseteq \bigcap_{n=-\infty}^0 F^n(\Sigma_{(F,p)}^0)$. Since $\mathcal{M}_F(A^\mathbb{Z})$ is F -invariant, we get $\mathcal{M}_F(A^\mathbb{Z}) \subseteq \Sigma_{(F,p)}$.

4.11 Particle Distribution

Assume that $(I, \lambda), (J, \lambda)$ are finite linearly ordered sets and let $x = (x_i)_{i \in I}$ $y = (y_j)_{j \in J}$ be nonnegative vectors such that $\sum_{i \in I} x_i \leq \sum_{j \in J} y_j$. A distribution for x and y is a nonnegative

matrix $z = (z_{ij})_{i \in I, j \in J}$ such that $\sum_{j \in J} z_{ij} = x_i$, $\sum_{i \in I} z_{ij} \leq y_j$. Let λ be the lexicographic order on $I \times J$ defined by

$$(i, j) \lambda (i', j') \iff i \lambda i' \vee (i = i' \wedge j \lambda j')$$

On the set of distributions a linear order can be defined by

$$z \lambda w \iff \exists (i, j) \in I \times J, (z_{ij} > w_{ij} \wedge \forall (k, l) \lambda (i, j), z_{kl} = w_{kl})$$

Then there exists a unique minimal distribution. If $I = \{0, 1, \dots, n-1\}$ and $J = \{0, 1, \dots, m-1\}$, then the minimal distribution is defined inductively by

$$z_{00} = \min\{x_0, y_0\}$$

and

$$z_{ij} = \min\{x_i - z_{i0} - \dots - z_{i,j-1}, y_j - z_{0j} - \dots - z_{j,i-1}\}$$

Assume now that p is a Lyapunov particle weight function for a local rule f , so $\varphi_p(f(u)) \leq \varphi_p(u)$ for every $u \in A^*$. If $p(f(u)) > 0$, one can try to distribute $p(f(u))$ among the particles of u . Denote by $q_{ij}(u) \leq p(u_{[i,j]})$ the share of a particle $u_{[i,j]}$. The sum of these shares is then $p(f(u))$. In a configuration $x \in A^{\mathbb{Z}}$, a particle $x_{[i,j]}$ may get shares from several particles of $F(x)$. If $k \leq i < j \leq l$, then a particle $F(x)_{[k-a,l-a-d]} = f(x_{[k,l]})$ requires the share $q_{i-k,j-k}(x_{[k,l]})$ from the particle $x_{[i,j]}$ (Fig. 13 right). The sum of these requirements cannot exceed the weight $p(x_{[i,j]})$.

Definition 12 A particle distribution for $f : A^{d+1} \rightarrow A$ and a particle weight function $p : A^* \rightarrow \mathbb{N}$ is a system $q = \{q_{ij}(u) : u \in A^+, 0 \leq i < j \leq |u|\}$ of integers such that for $u \in A^+$ and $x \in A^{\mathbb{Z}}$,

$$\begin{aligned} \sum_{i=0}^{|u|-1} \sum_{j=i+1}^{|u|} q_{ij}(u) &= p(f(u)), \\ q(x, i, j) &= \sum_{k=-\infty}^i \sum_{l=j}^{\infty} q_{i-k,j-k}(x_{[k,l]}) \leq p(x_{[i,j]}), \quad i < j \end{aligned}$$

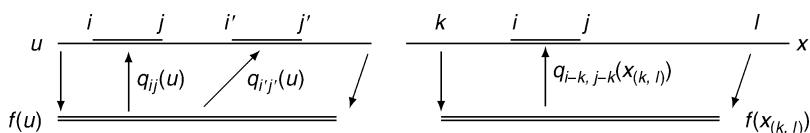
For $k = 0$ and $u = x_{[k,l]}$ we get in particular $q_{ij}(u) \leq p(u_{[i,j]})$.

Proposition 20 If p is a Lyapunov pwff for a CA F , then there exists d , local rule $f : A^{d+1} \rightarrow A$ for F and a distribution for p and f .

Proof By Proposition 13 there exists d and a local rule $f : A^{d+1} \rightarrow A$ such that for every $u \in A^*$, $\varphi_p(f(u)) \leq \varphi_p(u)$. We have the lexicographic order on the set of $I = \{[i, j] \in \mathbb{Z}^2 : i < j\}$ of intervals and also on the set I^2 of the pairs of intervals. Given $x \in \mathcal{F}_p$, we try to distribute

Fig. 13

Particle distribution.



the weight of a particle $F(x)_{[k,l-d]} = f(x_{[k,l]})$ in $F(x)$ among particles $x_{[i,j]}$. Call a distribution for x any system of nonnegative numbers $g_{kl}ij(x)$ indexed by intervals $[k, l]$ and $[i, j]$ such that

$$\sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} g_{kl}ij(x) = p(F(x)_{[k,l-d]})$$

$$\sum_{k=-\infty}^{\infty} \sum_{l=k+1}^{\infty} g_{kl}ij(x) \leq p(x_{[i,j]})$$

This implies in particular $g_{kl}ij(x) \leq p(x_{[i,j]})$, $g_{kl}ij(x) \leq p(F(x)_{[k,l-d]}) = p(f(x_{[k,l]}))$. There is only a finite number of distribution for any $x \in \mathcal{F}_p$ and there exists a unique minimal distribution. It can now be shown that there exist distributions that are nonzero only on the index set

$$I_4 = \{(k, l, i, j) \in I^2 : k \leq i < j \leq l\}$$

This means that the particle $F(x)_{[k,l-d]}$ is distributed only between particles in $x_{[k,l]}$. Take the first interval $[k, l]$ for which $F(x)_{[k,l-d]}$ is a particle and construct its minimal distribution into particles of $x_{[k,l]}$. Take now the second particle, that is, the least $[k', l'] \gg [k, l]$ such that $F(x)_{[k',l'-d]}$ is a particle. If a particle in $x_{[k',l']}$ has been charged in the first step by $F(x)_{[k,l-d]}$, we construct the minimal distribution for $F(x)_{[k,l-d]}$ into $x_{[k,l']}$. If no particle in $x_{[k',l']}$ has been charged in the first step, then we construct the minimal distribution for $F(x)_{[k',l'-d]}$ in $x_{[k',l'+d]}$. In either case, we get the minimal distribution for the first two particles indexed by I_4 . We continue and construct the minimal distribution of x indexed by I_4 . Consider now a word u such that $p(f(u)) > 0$. Set $k = 0$, $l = |u|$ and let $Y_u = \mathcal{F}_p \cap [u]_0$. For each $x \in Y_u$, let

$$\{g_{kl}ij(x) : 0 \leq i < j < |u|\}$$

be the restriction of the minimal distribution g of x to the interval $[k, l] = [0, |u|]$. The set of all these restricted distributions has a maximal element g and we set $q_{ij}(u) = g_{kl}ij(x)$. It can be now shown that q is a distribution for p and f . Given $x \in \mathcal{F}_p$, set

$$h_{kl}ij(x) = q_{i-k,l-k}(x_{[k,l+d]}).$$

Then h is a distribution for x so q satisfies the requirements.

Proposition 21 *Let F be a cellular automaton with a local rule $f: A^{d+1} \rightarrow A$. If a particle weight function $p: A^* \rightarrow \mathbb{N}$ has a distribution q , then p is a Lyapunov pwf for F .*

Proof Let $x \in \mathcal{F}_p$ be a p -finite configuration. For $k < l$ and $u = x_{[k,l]}$ we get

$$p(f(x_{[k,l]})) = \sum_{i=0}^{|u|-1} \sum_{j=i+1}^{|u|} q_{ij}(u) = \sum_{i=k}^{l-1} \sum_{j=i+1}^l q_{i-k,j-k}(x_{[k,l]})$$

Using the fact that $p(\lambda) = 0$ we get

$$\begin{aligned} \varphi_p(F(x)) &= \sum_{k=-\infty}^{\infty} \sum_{l=k+1}^{\infty} p(f(x_{[k,l]})) = \sum_{k=-\infty}^{\infty} \sum_{l=k+1}^{\infty} \sum_{i=k}^{l-1} \sum_{j=i+1}^l q_{i-k,j-k}(x_{[k,l]}) \\ &= \sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} \sum_{k=-\infty}^i \sum_{l=j}^{\infty} q_{i-k,j-k}(x_{[k,l]}) \leq \sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} p(x_{[i,j]}) = \varphi_p(x) \end{aligned}$$

By [Proposition 13](#), p is Lyapunov.

Proposition 22 Let $p : A^s \rightarrow \mathbb{N}$ be a particle weight function with order s , let $f : A^{d+1} \rightarrow A$ be a local rule and q a distribution for p . Then for every $x \in A^{\mathbb{Z}}$,

$$q(x, i, j) = \sum_{k=j-s-d}^i \sum_{l=j}^{i+s+d} q_{i-k, j-k}(x_{[k, l]}).$$

Proof If either $k < j - s - d \leq l - s - d$ or $k + s + d \leq i + s + d < l$, then $l - k > s + d$ and $q_{i-k, j-k}(x_{[k, l]}) \leq p(f(x_{[k, l]})) = 0$.

Proposition 23 For $|v| \leq 2(s + d)$ set $i_v = |v| - s - d$, $j_v = s + d$ and

$$r(v) = p(v_{[i_v, j_v]}) - \sum_{k=j_v-s-d}^{i_v} \sum_{l=j_v}^{i_v+s+d} q_{i_v-k, j_v-k}(v_{[k, l]}).$$

If $r(v) > 0$ then v is (F, p) -reducing and for every $x \in \mathcal{F}_p$, $\varphi_p(F(x)) \leq \varphi_p(x) - r(v) \cdot \varphi_v(x)$.

Proof For $x \in \mathcal{F}_p$ set $\xi_{ij}(x) = 1$ if $x_{[j-s-d, i+s+d]} = v$, and $\xi_{ij}(x) = 0$ otherwise. If $\xi_{ij}(x) = v$, then $x_{[i, j]} = v_{[i_v, j_v]}$ and

$$r(v) \cdot \xi_{ij}(x) \leq p(x_{[i, j]}) - \sum_{k=-\infty}^i \sum_{l=j}^{\infty} q_{i-k, j-k}(x_{[k, l]})$$

Similarly as in [Proposition 22](#) we get

$$\begin{aligned} \varphi_p(F(x)) &= \sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} \sum_{k=-\infty}^i \sum_{l=j}^{\infty} q_{i-k, j-k}(x_{[k, l]}) \\ &\leq \sum_{i=-\infty}^{\infty} \sum_{j=i+1}^{\infty} p(x_{[i, j]}) - r(v) \xi_{ij}(x) \leq \varphi_p(x) - r(v) \varphi_v(x) \end{aligned}$$

Definition 13 We say that a CA $(A^{\mathbb{Z}}, F)$ is p -attracting for a particle weight function $p : A^+ \rightarrow \mathbb{N}$ if there exists $s > 0$, such that each word $v \in A^+$ with $\varphi_p(v) > s$ is (F, p) -reducing.

Proposition 24 If a CA $(A^{\mathbb{Z}}, F)$ is p -attracting for a pwf $p : A^+ \rightarrow \mathbb{N}$, then $\mathcal{M}_F \subseteq \Sigma_p$.

Proof We get $\mathcal{M}_F \subseteq \Sigma_p^s := \{x \in A^{\mathbb{Z}} : \varphi_p(x) \leq s\}$. The densely recurrent subshift of Σ_p^s is Σ_p .

Example 10 (ECA62) Consider the ECA given by the following local rule: $F(x)_i = 0$ iff $x_{[i-1, i+1]} \in \{000, 110, 111\}$.

There exists a Lyapunov particle weight function given by

$$p(00) = 1, \quad p(111) = 1, \quad p(010) = 2, \quad p(u) = 0 \quad \text{for } u \notin \{00, 111, 010\}$$

A particle distribution for p is given in [Table 1](#). The strokes in the preimages give positions of i, j such that $q_{ij}(u) = 1$.

The only preimages of 0^6 are 0^8 , 01^7 , and 1^8 . However 1^7 is not in $\mathcal{L}(\Omega_F(A^{\mathbb{Z}}))$, so the only preimage of 0^6 in the omega-limit is 0^8 . Thus there exists an attractor that does not contain 0^∞ . Indeed, the set $V = [0^6]_2 \cup [1^7]_1 \cup \bigcup_{v \in f^{-1}(1^7)} [v]_0$ is inversely invariant, so $U = A^{\mathbb{Z}} \setminus V$ is a clopen invariant set and $\Omega_F(U)$ is a subshift attractor. It contains σ -transitive

■ **Table 1**
A particle distribution for ECA62

Image	Preimages
00	0 00 0, 111 0, 111 1
111	0 010 0, 0 010 1, 010 01, 010 11, 1 00 10, 1 00 11, 1 010 0, 1 010 1
010	11 00 0, 110 00

signal subshifts $\Sigma_{(1,2)}$ and $\Sigma_{(0,3)}$, whose intersection is $\sigma((110)^\infty)$, as well as their join. Similarly as in [Example 8](#) we show

$$\mathcal{Q}_F = \Omega_F(U) = F^2(\Sigma_{(1,2)} \vee \Sigma_{(0,3)}), \quad \mathcal{M}\mathcal{Q}_F = \Sigma_{(1,2)} \cup \Sigma_{(0,3)}$$

The only other signal subshifts are $\Sigma_{(4,4)}$ and $\Sigma_{(-1,1)}$. We get

$$\begin{aligned} \Omega_F(A^\mathbb{Z}) &= F^2(\Sigma_{(4,4)} \vee \Sigma_{(1,2)} \vee \Sigma_{(0,3)} \vee \Sigma_{(-1,1)}) \\ \mathcal{M}_F &= \{0^\infty\} \cup \Sigma_{(1,2)} \cup \Sigma_{(0,3)} \end{aligned}$$

Signal subshifts for ECA 62 are given in [Fig. 15](#). In [Fig. 14](#), a configuration whose left part belongs to $\Sigma_{(4,4)} \vee \Sigma_{(-1,1)}$ and whose right part belongs to $\Sigma_{(1,2)} \vee \Sigma_{(0,3)}$ can be seen. In the space–time diagram, the words 00, 111, and 010, which do not occur in $(110)^\infty$, are displayed in gray.

Example 11 (Gacs–Kurdyumov–Levin cellular automaton, [Fig. 16](#)) The alphabet $A = \{0, 1\}$ is binary, $d = 6$ and $a = -3$. The local rule is given by the table

0011	001	011	0010	1011	110	100	101	010	0	1
------	-----	-----	------	------	-----	-----	-----	-----	---	---

The first applicable rule is always used.

The original rule of Gacs et al. (1978), Gacs (2001), or de Sá and Maes (1992) was slightly modified, which reads

$$F(x)_i = \begin{cases} \mathbf{m}(x_{i-3}, x_{i-1}, x_i) & \text{if } x_i = 0 \\ \mathbf{m}(x_i, x_{i+1}, x_{i+3}) & \text{if } x_i = 1 \end{cases}$$

where \mathbf{m} is the majority function. The modification also homogenizes short regions. Note that the iterates of any finite perturbation of 0^∞ eventually attain 0^∞ and likewise the iterates of any finite perturbation of 1^∞ eventually attain 1^∞ .

This model has been considered to be an example of a computation that is stable with respect to random errors. Suppose that after each (deterministic) step, each site of the configuration is flipped with a given probability ε . This updating defines a dynamical system in the space $\mathfrak{M}(A^\mathbb{Z})$. It has been conjectured that for sufficiently small ε , the resulting dynamical system has two invariant measures, one close to $\mu_{\{0\}}$, the other close to $\mu_{\{1\}}$. The conjecture is still open but a much more complex CA with these properties has been constructed in Gacs (2001). Mitchell et al. (1994) consider more general class of CA, which

Fig. 14
ECA62.

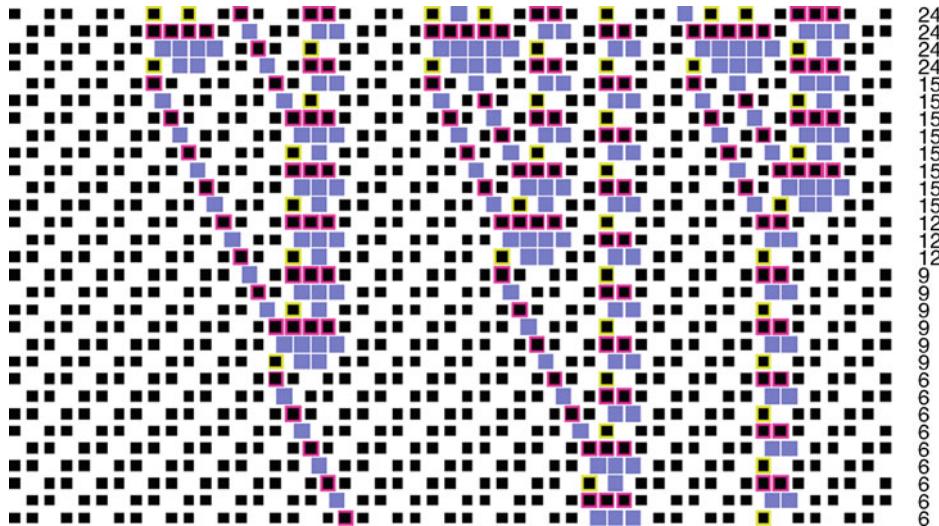


Fig. 15
Signal subshifts for ECA62.

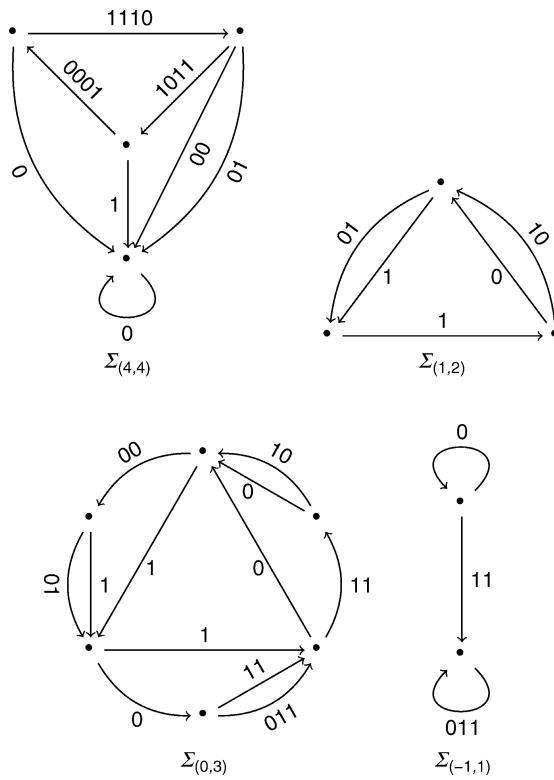
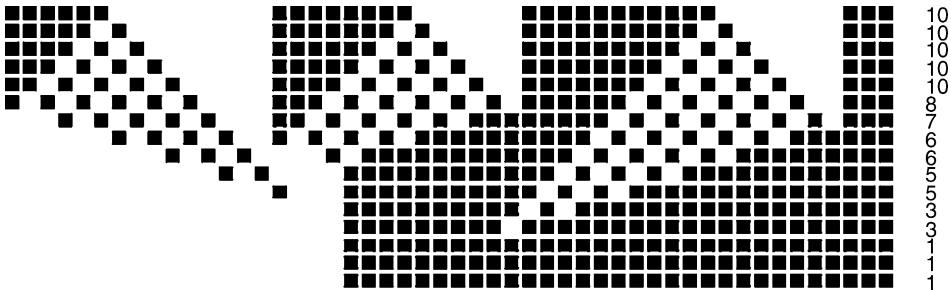


Fig. 16
Gacs–Kurdyumov–Levin CA.



solve the density classification task, that is, decide whether a given configuration has more zeros or ones. We have nontransitive signal subshifts

$$\begin{aligned}\mathcal{S}_{(3,1)} &= \{\sigma^n(0^\infty.(10)^\infty) : n \in \mathbb{Z}\} \cup \{0^\infty, (01)^\infty, (10)^\infty\} \\ \mathcal{S}_{(1,1)} &= \{\sigma^n((01)^\infty.0^\infty) : n \in \mathbb{Z}\} \cup \{0^\infty, (01)^\infty, (10)^\infty\} \\ \mathcal{S}_{(0,1)} &= \{\sigma^n(0^\infty.1^\infty) : n \in \mathbb{Z}\} \cup \{0^\infty, 1^\infty\} \\ \mathcal{S}_{(-1,1)} &= \{\sigma^n(1^\infty.(01)^\infty) : n \in \mathbb{Z}\} \cup \{1^\infty, (01)^\infty, (10)^\infty\} \\ \mathcal{S}_{(-3,1)} &= \{\sigma^n((10)^\infty.1^\infty) : n \in \mathbb{Z}\} \cup \{1^\infty, (01)^\infty, (10)^\infty\}\end{aligned}$$

There exists a particle weight function that charges these signal subshifts. It is given by $p(1100) = 2$,

$$\begin{aligned}p(1011) &= 1, p(1101) = 1, p(0011) = 1, p(0100) = 1, p(0010) = 1 \\ s(1011) &= -3, s(1101) = -1, s(0011) = 0, s(0100) = 1, s(0010) = 3\end{aligned}$$

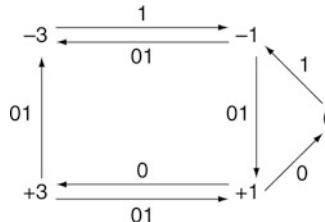
where s gives the speeds of these particles. Particle 1100 has no speed as it immediately changes to a pair of particles 1101 and 0100. The algorithm of [Proposition 22](#) shows that there exists no distribution for this particle weight function. However, there does exist a particle distribution restricted to the first image subshift with language $f(A^*) = \{u \in A^* : \forall v \in S, v \not\subseteq u\}$, where $S = \{1100, 100100, 110110\}$ is the set of forbidden words. One such distribution is given in the following table (the strokes in a preimage u give positions i, j such that $q_{ij}(u) = 1$).

Image	Preimages
1011	xxx110 1011 , 101 0011 1xx, x11010 1011 , 101 0011 011, 101010 1011
1101	xxx1 1101 0x, xx10 1101 0x
0011	xx 0010 11xx, x 0010 1011x, 0010 101011, u 0011 v
0100	x1 0100 0xxx, x1 0100 10xx

where $u \in \{xx0, 001\}$ and $v \in \{1xx, 011\}$. When two particles meet they annihilate or change as follows:

$$-1, -3 \rightarrow \lambda, \quad 0, -1 \rightarrow 3, \quad 3, -3 \rightarrow 0, \quad 1, 0 \rightarrow -3, \quad 3, 1 \rightarrow \lambda$$

It is also possible that three particles $1, 0, -1 \rightarrow \lambda$ meet simultaneously and disappear. These particles cannot occur in a configuration in an arbitrary order. Any possible sequence of their occurrence is represented by a path in the following graph. The particles are represented by their speeds and the edges between them represent one of the periodic words $0^\infty, 1^\infty, (01)^\infty$, which separate them.



The path $-3 \rightarrow -1 \rightarrow 1 \rightarrow 3$ is the longest path with increasing speeds and represents a configuration $(01)^\infty 1^i (01)^j 0^k (01)^\infty$ in which no particle ever disappears. Any configuration that contains at least $s = 5$ particles contains a pair of neighboring particles, which approach one another and finally either disappear or are transformed into another particle. This means that the CA is p -attracting, so $\mathcal{M}_F \subseteq \Sigma_p$. The subshift $\Sigma_p = \{0^\infty, 1^\infty, (01)^\infty, (10)^\infty\}$ is finite and $\mathfrak{M}(\Sigma_p)$ is a triangle consisting of convex combinations of $\mu_{\{0\}}, \mu_{\{1\}}$, and $\mu_{\{01\}}$. Every measure in this triangle is F -invariant. However, there are trajectories that go from an arbitrary neighborhood of any measure of $\mathfrak{M}(\Sigma_p)$ to any other measure in $\mathfrak{M}(\Sigma_p)$.

Example 12 (random walk) The alphabet is $\{0, 1\}^3$ and the rule is given by

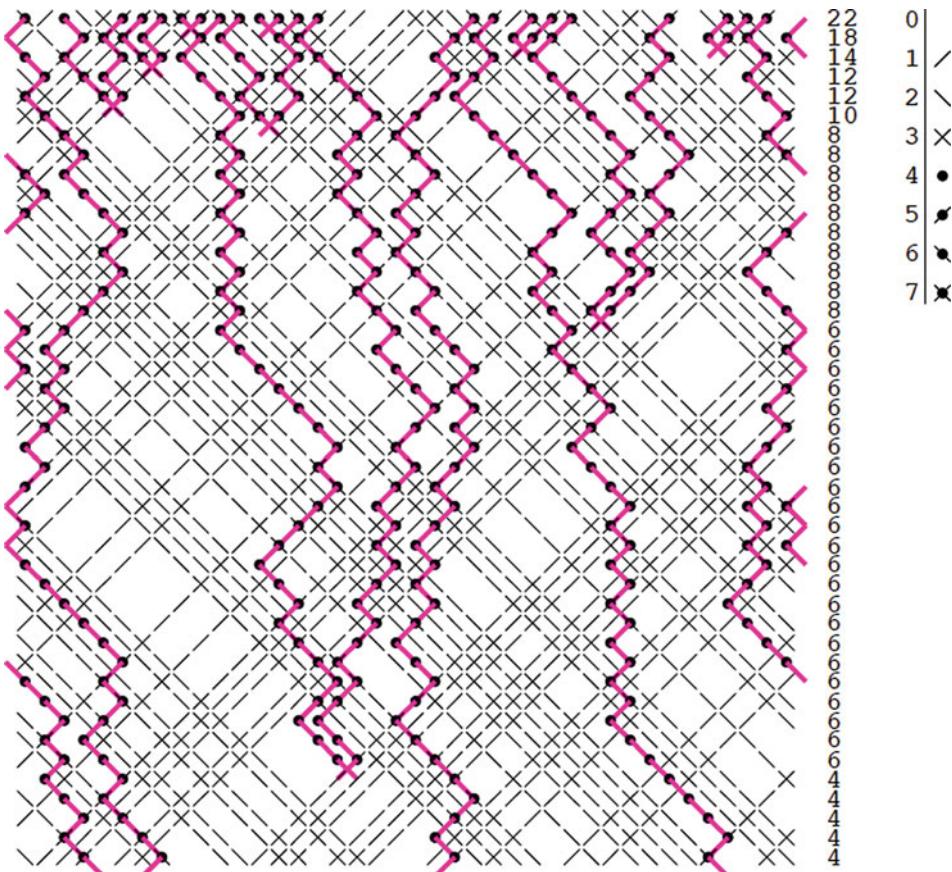
$$\begin{aligned} F(x, y, z)_i &= (1, y_{i-1}, z_{i+1}) \text{ if} \\ &x_{i-1} = 1, y_{i-1} = z_{i-1}, x_i \cdot (y_i - z_i) = 0 \text{ or} \\ &x_{i+1} = 1, y_{i+1} \neq z_{i+1} = 0, x_i \cdot (y_i + z_i - 1) = 0 \\ F(x, y, z)_i &= (0, y_{i-1}, z_{i+1}) \text{ otherwise} \end{aligned}$$

Denote by $\pi_i : A^\mathbb{Z} \rightarrow \{0, 1\}^\mathbb{Z}$, $i = 1, 2, 3$ the projections. Then $\pi_2 : (A^\mathbb{Z}, F) \rightarrow (\{0, 1\}^\mathbb{Z}, \sigma^{-1})$ and $\pi_3 : (A^\mathbb{Z}, F) \rightarrow (\{0, 1\}^\mathbb{Z}, \sigma)$ are factor maps. Ones in the first coordinate are particles, which move to the right if the second and third coordinates are equal and to the left otherwise. When two neighboring particles cross, they annihilate. When they meet, they merge into one particle. Suppose that μ is a Bernoulli measure on $A^\mathbb{Z}$, then $\pi_2\mu, \pi_3\mu$ are Bernoulli measures; so neighboring particles perform independent random walk until they annihilate or merge. These random walk need not be symmetric, but the distance between two neighboring particles performs a symmetric random walk with absorbing states 0 and -1 (which represents annihilation). Thus the random walk CA implements the behavior that has been observed in ECA18 or ECA54. We have a particle weight function $p : A \rightarrow \mathbb{N}$ given by $p(x, y, z) = x$, which counts the number of particles. It is clearly nonincreasing for F (see Fig. 17) and Σ_p is the subshift of particle-free configurations. It can be proved (see Kůrka and Maass 2000 or Kůrka 2003a) that for every connected measure μ we have $A_\mu \subseteq \Sigma_p$.

4.12 Particle Weight Functions with Infinite Support

In Kůrka (2003a) particle weight functions with infinite support are considered. They are functions $p : A^* \rightarrow \mathbb{N}$, which satisfy the following conditions.

Fig. 17
Random walk.



1. $p(\lambda) = 0$.
 2. There exists $b_0 > 0$, such that for every $u \in A^*$, $p(u) \leq b_0$.
 3. There exists $b_1 > 0$, such that if $x \in A^{\mathbb{Z}}$, $p(x_{[i,j)}) > 0$, and $p(x_{[k,l)}) > 0$, then the interval $[i, j) \cap [k, l)$ has less than b_1 elements.
 4. There exists $x \in A^{\mathbb{Z}}$ such that $p(x_{[i,j)}) = 0$ for all $i < j$.

Example 13 (ECA18, Fig. 18) $F(x)_i = 1$ iff $x_{[i-1, i+1]} \in \{001, 100\}$.

000:0, 001:1, 010:0, 011:0, 100:1, 101:0, 110:0, 111:0.

Consider a particle weight function given by

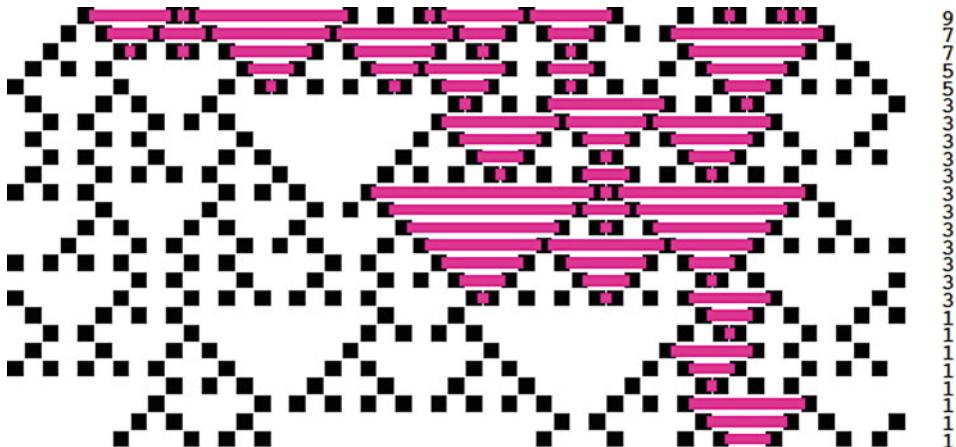
$$p(u) = \begin{cases} 1 & \text{if } u = 10^{2m}1 \text{ for some } m \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\Sigma_p = \{x \in A^{\mathbb{Z}} : \forall i, x_{2i} = 0\} \cup \{x \in A^{\mathbb{Z}} : \forall i, x_{2i+1} = 0\}$$

Fig. 18

Example of a space-time diagram for ECA18 (see [Example 13](#)). Particles are colored in red.



The dynamics on Σ_p is quite simple. If $x_{2i} = 0$ for all i , then $F(x)_{2i+1} = 0$ and $F(x)_{2i} = x_{2i-1} + x_{2i+1} \bmod 2$. It can be proved that $A_\mu \subseteq \Sigma_p$ for every connected measure μ (see Kůrka 2003a).

There exist also particle weight functions, which are decreasing only in the long run, while they oscillate in the short run.

Example 14 (equalizing CA) The alphabet is $A = \{0, 1, 2, 3\}$ and the local rule is given by

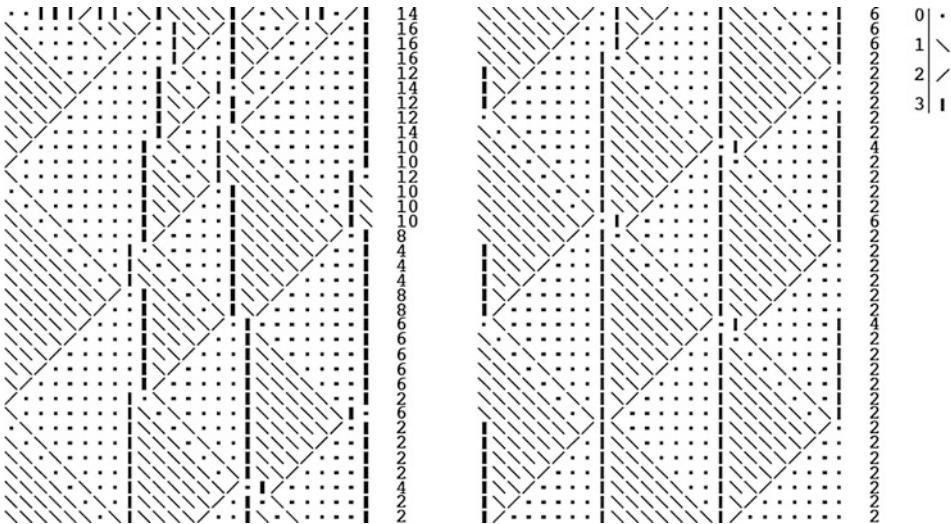
xx33x:0 xx032x:3 130xx:3 131xx:3 x132x:3 xx32x:0 x13xx:0
xxx2x:2 xx13x:2 x3xxx:1 xx2xx:0 x10xx:1 x11xx:1 xx1xx:0

Particles 1, 3 walk between two walls 3 similarly as in [Example 6](#), but when they reach a wall they push it forward. The intervals between neighboring walls grow and shrink but the shorter intervals grow faster, so the lengths of all intervals approach the same value. If $(x_i)_{0 \leq i \leq n}$ are lengths of neighboring intervals in a periodic configuration $u \in A^{\mathbb{Z}}$, we define weight function $p(u) = \sum_{i < n} |x_{i+1} - x_i|$ (see [Fig. 19](#)), which decreases in the long run and then oscillates with small values.

4.13 Operations on CA

The local rule of a 1D CA can be represented by a look-up table. However, given a look-up table one cannot uniquely define the corresponding CA since the CA memory has to be fixed. Indeed, for each memory value, a different CA is obtained. It is therefore natural to wonder what properties are conserved by the CA when changing the memory but keeping the same look-up table for the local rule. This problem can be equivalently rephrased as follows. Let F be a CA. If a property holds for a CA F , does it hold for the CA $\sigma^n \circ F$? Along the same direction, a look-up table defines a CA both on $A^{\mathbb{Z}}$ and $A^{\mathbb{N}}$ (under the constraint that the memory $m \geq 0$), but what properties does these two CA share? In this section, attempts have been made to

Fig. 19
Equalizing CA.



answer these questions. The obtained results help to shape a new scenario for the old-standing conjecture about the equivalence between surjectivity and DPO for CA: the study can be restricted to mixing CA (Acerbi et al. 2007, 2009).

4.13.1 Shifting

Throughout this section F_m stands for a CA F with memory $m \in \mathbb{Z}$. Given a CA $(A^{\mathbb{Z}}, F_m)$ and $h \in \mathbb{Z}$, consider the CA $(A^{\mathbb{Z}}, F_{m+h})$. Since $F_{m+h} = \sigma^h \circ F_m$, we say that the CA $(A^{\mathbb{Z}}, F_{m+h})$ is obtained by a *shifting operation*, which moves the memory of the originally given CA from m to $m+h$. In this section, properties that are preserved by the shifting operation are studied.

Proposition 25 (Acerbi et al. 2007, 2009) *Let $(A^{\mathbb{Z}}, F_m)$ be a CA. For any $h \in \mathbb{Z}$, $(A^{\mathbb{Z}}, F_{m+h})$ is surjective (resp., is injective, is right-closing, is left-closing, has JDPO) iff $(A^{\mathbb{Z}}, F_m)$ is surjective (resp., is injective, is right-closing, is left-closing, has JDPO).*

The following theorem establishes the behavior of the shift operation with respect to sensitivity, equicontinuity, and almost equicontinuity. Its proof is essentially contained in Sablik (2008).

Theorem 14 *For any CA $(A^{\mathbb{Z}}, F_m)$ one and only one of the following statements holds:*

- \mathcal{S}_0 : the CA $(A^{\mathbb{Z}}, F_{m+h})$ is nilpotent (and then equicontinuous) for any $h \in \mathbb{Z}$;
- \mathcal{S}_1 : there exists an integer \bar{h} with $\bar{h} + m \in [-d, d]$ such that the CA $(A^{\mathbb{Z}}, F_{m+h})$ is equicontinuous for $h = \bar{h}$ and it is sensitive for any $h \neq \bar{h}$;

- \mathcal{S}_2 : there is a finite interval $I \subset \mathbb{Z}$, with $I + m \subseteq [-d, d]$, such that the CA $(A^{\mathbb{Z}}, F_{m+h})$ is almost equicontinuous but not equicontinuous iff $h \in I$ (and then it is sensitive for any other $h \in \mathbb{Z} \setminus I$);
 \mathcal{S}_3 : the CA $(A^{\mathbb{Z}}, F_{m+h})$ is sensitive (ever-sensitivity) for any $h \in \mathbb{Z}$.

In the case of surjective CA, \blacktriangleright Theorem 14 can be restated as follows.

Theorem 15 (Acerbi et al. 2007, 2009) For any surjective CA $(A^{\mathbb{Z}}, F_m)$ one and only one of the following statements holds:

- \mathcal{S}'_1 : there exists an integer h' , with $h' + m \in [-d, d]$, such that the CA F_{m+h} is equicontinuous for $h = h'$ and it is mixing for $h \neq h'$;
 \mathcal{S}'_2 : there exists an integer h' , with $h' + m \in [-d, d]$, such that the CA F_{m+h} is almost equicontinuous but not equicontinuous for $h = h'$ and it is mixing for $h \neq h'$;
 \mathcal{S}'_3 : there is at most a finite set $I \subset \mathbb{Z}$, with $I + m \subseteq [-d, d]$, such that if $h \in I$ then the CA F_{m+h} is sensitive but not mixing, while it is mixing if $h \in \mathbb{Z} \setminus I$.

The next proposition assures that positively expansive CA are in class \mathcal{S}'_3 , in particular they are “ever-sensitive.”

Proposition 26 (Acerbi et al. 2007, 2009) If $(A^{\mathbb{Z}}, F_m)$ is a positively expansive CA, then for any $h \in \mathbb{Z}$ the CA $(A^{\mathbb{Z}}, F_{m+h})$ is sensitive.

The following is a long-standing conjecture in CA theory, which dates back at least to Blanchard and Tisseur (2000).

Conjecture 1 Any Surjective CA has DPO.

\blacktriangleright Proposition 25 states that the shift operation conserves surjectivity. Therefore, \blacktriangleright Conjecture 1 leads naturally to the following.

Conjecture 2 For any CA $(A^{\mathbb{Z}}, F_m)$ and any $h \in \mathbb{Z}$, $(A^{\mathbb{Z}}, F_{m+h})$ has DPO iff $(A^{\mathbb{Z}}, F_m)$ has DPO.

Recall that both surjective almost equicontinuous CA and closing CA have JDPO (Blanchard and Tisseur 2000; Boyle and Kitchens 1999). By \blacktriangleright Proposition 25, JDPO is preserved by the shifting operation, so all the CA in the classes \mathcal{S}'_1 and \mathcal{S}'_2 have JDPO. We conjecture that the same holds for (non closing) CA in \mathcal{S}'_3 :

Conjecture 3 If a CA has DPO then it has JDPO.

Remark that \blacktriangleright Conjectures 1 and \blacktriangleright 3 are also true for the class of additive CA and the class of number-conserving CA (Cattaneo et al. 2004; Cervelle et al. 2008; Formenti and Grange 2003).

A notion that will be useful in what follows is now introduced. A CA $(A^{\mathbb{Z}}, F_m)$ is *strictly right* (resp., *strictly left*) if $m > 0$ (resp., $a < 0$) while it is *one-sided* if $m \geq 0$.

Proposition 27 (Acerbi et al. 2007, 2009) A surjective strictly right (or strictly left) CA is mixing.

Proposition 28 (Cattaneo et al. 2000) Let $(A^{\mathbb{Z}}, F_m)$ be a strictly right CA. Any periodic configuration for the CA is also periodic for σ .

The following corollary is a trivial consequence of the previous proposition.

Corollary 5 (Acerbi et al. 2007, 2009) Consider a strictly right CA. If it has DPO then it has JDPO too.

Proposition 29 (Acerbi et al. 2007, 2009) \Rightarrow Conjecture 2 is equivalent to \Rightarrow Conjecture 3.

As a by-product of the previous results, we have the following.

Theorem 16 (Acerbi et al. 2007, 2009) In the CA settings, the following statements are equivalent

1. Surjectivity implies DPO.
2. Surjectivity implies JDPO.
3. For strictly right CA, topological mixing implies DPO.

\Rightarrow Theorem 16 illustrates that in order to prove \Rightarrow Conjecture 1 one can focus on mixing strictly right CA. Note that all known examples of topologically mixing CA have DPO. We now want to present a result that further supports the common feeling that \Rightarrow Conjecture 1 is true.

Proposition 30 (Acerbi et al. 2007, 2009) Any surjective CA (both on $A^{\mathbb{Z}}$ and on $A^{\mathbb{N}}$) has an infinite set of points that are jointly periodic for the CA and σ .

4.13.2 Lifting

Any CA with memory $m \geq 0$ is well defined also on $A^{\mathbb{N}}$. In that case, one can denote by $\Phi_m : A^{\mathbb{N}} \rightarrow A^{\mathbb{N}}$ its global rule. For a fixed local rule f and a memory $m \geq 0$, consider the two one-sided CA $(A^{\mathbb{Z}}, F_m)$ and $(A^{\mathbb{N}}, \Phi_m)$ on $A^{\mathbb{Z}}$ and $A^{\mathbb{N}}$, respectively. In this section, the properties that are conserved when passing from the CA on $A^{\mathbb{Z}}$ to the one on $A^{\mathbb{N}}$ and *vice-versa* are studied.

Consider the *projection* $P : A^{\mathbb{Z}} \rightarrow A^{\mathbb{N}}$ defined as follows: $\forall x \in A^{\mathbb{Z}}, \forall i \in \mathbb{N}, P(x)_i = x_i$. Then, P is a continuous, open, and surjective function. Moreover, the following diagram commutes

$$\begin{array}{ccc} A^{\mathbb{Z}} & \xrightarrow{F_m} & A^{\mathbb{Z}} \\ P \downarrow & & \downarrow P \\ A^{\mathbb{N}} & \xrightarrow{\Phi_m} & A^{\mathbb{N}} \end{array} \quad (1)$$

that is, $\Phi_m \circ P = P \circ F_m$. Therefore, the CA $(A^{\mathbb{N}}, \Phi_m)$ is a factor of $(A^{\mathbb{Z}}, F_m)$. For these reasons, we also say that the CA on $A^{\mathbb{Z}}$ is obtained by a *lifting (up) operation* from the CA on $A^{\mathbb{N}}$ (having the same rule and memory). As an immediate consequence of the fact that $\Phi_m \circ P = P \circ F_m$, the CA on $A^{\mathbb{N}}$ inherits from the CA on $A^{\mathbb{Z}}$ several properties such as surjectivity, transitivity, mixing, DPO, JDPO, left closingness, and openness.

The notion of left Welch index $L(f)$ for a local rule $f : A^{d+1} \rightarrow A$ of a surjective CA can be recalled now. This index is an upper bound for the number of the left possible extensions of a block u , which are mapped by f in the same word (where with an abuse of notation, f represents the extended application of the rule f on any word of length greater than $d+1$). Let $n \geq d$ and $k > 0$ be two integers and let $u \in A^n$ a block. A left k -extension of u is a word $v u$ where $v \in A^k$. A set of left k -extensions $v_1 u, v_2 u, \dots, v_n u$ of u is said to be compatible with f if

$f(v_1 u) = f(v_2 u) = \dots = f(v_n u)$. Set

$$L(u, k, f) = \max\{|W| : W \text{ set of left } k\text{-extensions of } u \text{ compatible with } f\}$$

and define $L(u, f) = \max_{k > 0} L(u, k, f)$. The left Welch index of the local rule f is $L(f) = L(u, f)$. Since the number $L(u, f)$ does not depend on the choice of u (see for instance Hedlund 1969), the left Welch index is well defined.

The following propositions reveal that the injectivity property is lifted down only under special conditions involving the Welch index, while the opposite case (lift up) is verified without additional hypothesis.

Proposition 31 (Acerbi et al. 2007, 2009) *Let $(A^{\mathbb{Z}}, F_m)$ be an injective one-sided CA. The CA $(A^{\mathbb{Z}}, \Phi_m)$ is injective if and only if $m = 0$ and the left Welch index $L(f) = 1$.*

Proposition 32 (Acerbi et al. 2007, 2009) *If $(A^{\mathbb{N}}, \Phi_m)$ is an injective (resp., surjective) CA, then the lifted CA $(A^{\mathbb{Z}}, F_m)$ is injective (resp., surjective).*

The following proposition directly follows from the definitions.

Proposition 33 *Left-closingness is conserved by the lifting up operation.*

The following results are immediately obtained from the fact that a word is blocking for a CA on $A^{\mathbb{N}}$ iff it is blocking for its lifted CA.

Proposition 34 *A CA $(A^{\mathbb{N}}, \Phi_m)$ is equicontinuous (resp., almost equicontinuous) (resp., sensitive) iff the CA $(A^{\mathbb{Z}}, F_m)$ is equicontinuous (resp., almost equicontinuous) (resp., sensitive).*

Positive expansivity is not preserved by the lifting operation. Indeed, there are no positively expansive one-sided CA on $A^{\mathbb{Z}}$ Acerbi et al. (2007, 2009). For a proof of this result, see (Blanchard and Maass 1997).

Proposition 35 *No one-sided CA $(A^{\mathbb{Z}}, F_m)$ is positively expansive.*

Proposition 36 (Acerbi et al. 2007, 2009) *If $(A^{\mathbb{N}}, \Phi_m)$ is mixing (resp., transitive), then its lifted CA $(A^{\mathbb{Z}}, F_m)$ is mixing (resp., transitive).*

The lifting (up) of DPO remains an open problem even if on the basis of the results of **Sect. 3** we conjecture that this should be true. This is also partially supported by the following proposition.

Proposition 37 (Acerbi et al. 2007, 2009) *If $(A^{\mathbb{N}}, \Phi_m)$ has JDPO, then $(A^{\mathbb{Z}}, F_m)$ has JDPO (i.e., JDPO is lifted up).*

The following result will be used later; it is the $A^{\mathbb{N}}$ version of **Proposition 27**.

Proposition 38 (Acerbi et al. 2007, 2009) *A surjective strictly right CA Φ_m on $A^{\mathbb{N}}$ is strongly transitive.*

• Proposition 28 and • Corollary 5 also hold for CA on $A^{\mathbb{N}}$. As a consequence we have the following result.

Proposition 39 (Acerbi et al. 2007, 2009) *The following statements are equivalent for CA on $A^{\mathbb{N}}$:*

1. *Surjectivity implies DPO.*
2. *Surjectivity implies JDPO.*
3. *Strong transitivity implies DPO.*

As a by-product of the previous results another equivalent version of • Conjecture 1 is obtained.

Proposition 40 (Acerbi et al. 2007, 2009) *The following statements are equivalent:*

1. *For CA on $A^{\mathbb{Z}}$, surjectivity implies (J)DPO.*
2. *For CA on $A^{\mathbb{N}}$, strong transitivity implies (J)DPO.*

5 From 1D CA to 2D CA

Von Neumann introduced CA as formal models for cells self-replication. These were two-dimensional objects. However, the study of the CA dynamical behavior focused essentially on the 1D case except for additive CA (Dennunzio et al. 2009a) and a few others (see, e.g., (Theyssier and Sablik 2008; Dennunzio et al. 2008, 2009c)). The reason of this gap is maybe twofold: from one hand, there is a common feeling that most of dynamical results are “automatically” transferred to higher dimensions; from the other hand, researchers mind the complexity gap. Indeed, many CA properties are dimension sensitive, that is, they are decidable in dimension 1 and undecidable in higher dimensions (Amoroso and Patt 1972; Durand 1993; Kari 1994; Durand 1998; Bernardi et al. 2005). In Dennunzio and Formenti (2008, 2009), in order to overcome this complexity gap, two deep constructions are introduced. They allow us to see a 2D CA as a 1D CA. In this way, well-known results of 1D CA can be lifted to the 2D case. The idea is to “cut” the space of configurations of a 2D CA into slices of dimension 1. Hence, the 2D CA can be seen as a new 1D CA operating on configurations made of slices. The only inconvenience is that this latter CA has an infinite alphabet. However, the constructions are refined so that slices are translation invariants along some fixed direction. This confers finiteness to the alphabet of the 1D CA allowing us to lift even more properties.

5.1 Notations and Basic Definitions

For a vector $\mathbf{i} \in \mathbb{Z}^2$, denote by $|\mathbf{i}|$ the infinite norm of \mathbf{i} . Let A be a finite alphabet. A *2D configuration* is a function from \mathbb{Z}^2 to A . The *2D configuration set* $A^{\mathbb{Z}^2}$ is equipped with the following metric, which is denoted for the sake of simplicity by the same symbol of the 1D case:

$$\forall x, y \in A^{\mathbb{Z}^2}, \quad d(x, y) = 2^{-k} \quad \text{where } k = \min\{|\mathbf{i}| : \mathbf{i} \in \mathbb{Z}^2, x_{\mathbf{i}} \neq y_{\mathbf{i}}\}$$

The 2D configuration set is a Cantor space.

For any $r \in \mathbb{N}$, let M_r be the set of all the two-dimensional matrices with values in A and entry vectors in the integer square $[-r, r]^2$. If $N \in M_r$, denote by $N(\mathbf{i}) \in A$ the element of the matrix N with entry vector \mathbf{i} .

A 2D CA is a structure $\langle 2, A, r, f \rangle$, where A is the alphabet, $r \in \mathbb{N}$ is the *radius* and $f : M_r \rightarrow A$ is the *local rule* of the automaton. The local rule f induces a *global rule* $F : A^{\mathbb{Z}^2} \rightarrow A^{\mathbb{Z}^2}$ defined as follows,

$$\forall x \in A^{\mathbb{Z}^2}, \forall \mathbf{i} \in \mathbb{Z}^2, \quad F(x)_{\mathbf{i}} = f(M_r^{\mathbf{i}}(x))$$

where $M_r^{\mathbf{i}}(x) \in M_r$ is the *finite portion* of x of reference position $\mathbf{i} \in \mathbb{Z}^2$ and radius r defined by $\forall \mathbf{k} \in [-r, r]^2, M_r^{\mathbf{i}}(x)_{\mathbf{k}} = x_{\mathbf{i}+\mathbf{k}}$. From now on, for the sake of simplicity, no distinction between a 2D CA and its global rule will be made.

For any $\mathbf{v} \in \mathbb{Z}^2$ the *shift map* $\sigma^{\mathbf{v}} : A^{\mathbb{Z}^2} \rightarrow A^{\mathbb{Z}^2}$ is defined by $\forall x \in A^{\mathbb{Z}^2}, \forall \mathbf{i} \in \mathbb{Z}^2, \sigma^{\mathbf{v}}(x)_{\mathbf{i}} = c_{\mathbf{i}+\mathbf{v}}$. A function $F : A^{\mathbb{Z}^2} \rightarrow A^{\mathbb{Z}^2}$ is said to be *shift-commuting* if $\forall \mathbf{k} \in \mathbb{Z}^2, F \circ \sigma^{\mathbf{k}} = \sigma^{\mathbf{k}} \circ F$. As in 1D case, the 2D CA are exactly the class of all shift-commuting functions, which are (uniformly) continuous with respect to the metric d (Hedlund's (1990) theorem).

For any fixed vector \mathbf{v} , we denote by $S_{\mathbf{v}}$ the set of all configurations $x \in A^{\mathbb{Z}^2}$ such that $\sigma^{\mathbf{v}}(x) = x$. Note that, for any 2D CA global map F and for any \mathbf{v} , the set $S_{\mathbf{v}}$ is F -invariant, that is, $F(S_{\mathbf{v}}) \subseteq S_{\mathbf{v}}$.

A *pattern* P is a function from a finite domain $Dom(P) \subseteq \mathbb{Z}^2$ taking values in A . The notion of cylinder can be conveniently extended to general patterns as follows: for any pattern P , let $[P]$ be the set $\{x \in A^{\mathbb{Z}^2} \mid \forall \mathbf{i} \in Dom(P), x_{\mathbf{i}} = P(\mathbf{i})\}$. As in the 1D case, cylinders form a basis for the open sets. In what follows, with a little abuse of notation, for any pattern P , $F(P)$ is the pattern P' such that $Dom(P') = \{\mathbf{i} \in Dom(P), \mathcal{B}_r(\mathbf{i}) \subseteq Dom(P)\}$ and for all $\mathbf{i} \in Dom(P'), P'(\mathbf{i}) = f(\mathcal{B}_r(\mathbf{i}))$, where $\mathcal{B}_r(\mathbf{i}) = \{\mathbf{j} \in \mathbb{Z}^2, |\mathbf{i} - \mathbf{j}| \leq r\}$. We denote by $F^{-1}(P) = |\{P' : F(P') = P\}|$ the set of the pre-images of a given pattern P .

5.2 Combinatorial Properties

Some results about injectivity and surjectivity of 2D CA are reviewed.

A r -radius 2D CA F is said to be k -balanced if $|F^{-1}(P)| = |A|^{(k+2r)^2-k^2}$ for any pattern P whose domain is a square of side k . A CA F is *balanced* if it is k -balanced for any $k > 0$.

Theorem 17 (Maruoka and Kimura 1976) A 2D CA is surjective if and only if it is balanced.

A 2D configuration x is *finite* if there are a symbol $a \in A$ and a natural n such that $x_{\mathbf{i}} = a$ for any \mathbf{i} with $|\mathbf{i}| > n$. A symbol $a \in A$ is said to be a *quiescent state* if $f(N_a) = a$ where $N_a \in M_r$ is the matrix with all the elements equal to a . The following theorem was originally proved in Moore (1962) and Myhill (1963) for CA in any dimension admitting a quiescent state. As remarked in Durand (1998), it is valid for any CA.

Theorem 18 A CA is surjective if and only if it is injective when restricted to finite configurations.

As an immediate consequence of the previous result, we have that injectivity and bijectivity are equivalent notions for CA.

Proposition 41 (Durand 1998) *Bijectivity and surjectivity are equivalent for CA restricted to finite configurations.*

Proposition 42 (Durand 1998) *If a CA is bijective, then its restriction to finite configurations is bijective too.*

Call $\bigcup_{v \in \mathbb{Z}^2} S_v$ the set of the *spatially periodic configurations*.

Proposition 43 (Durand 1998) *Bijectivity and injectivity are equivalent for CA restricted to spatially periodic configurations.*

As to decidability the following results are arrived at.

Theorem 19 (Kari 1994) *It is undecidable to establish if a 2D CA is injective.*

Theorem 20 (Kari 1994) *It is undecidable to establish if a 2D CA is surjective.*

Proposition 44 (Durand 1998) *It is undecidable to establish if a 2D CA restricted to spatially periodic configurations is injective.*

5.3 Stability Classification of 2D CA

❶ Theorems 1 and ❷ 2 lead to the well-known classification of 1D CA partitioned by their stability degree as follows: equicontinuous CA, non-equicontinuous CA admitting an equicontinuity configuration, sensitive but not positively expansive CA, positively expansive CA. This classification is no more relevant in the context of 2D CA since the class of positively expansive CA is empty (Shereshevsky 1993). Moreover, the following recent result holds.

Theorem 21 (Theyssier and Sablik 2008) *There exist nonsensitive 2D CA without any equicontinuity point.*

Corollary 6 (Theyssier and Sablik 2008) *Each 2D CA falls exactly into one among the following classes:*

1. Equicontinuous CA
2. Non-equicontinuous CA admitting an equicontinuity point
3. Nonsensitive CA without any equicontinuity point
4. Sensitive CA

Theorem 22 (Theyssier and Sablik 2008) *Each of the above classes 2, 3, and 4 is neither recursively enumerable nor co-recursively enumerable.*

5.4 2D CA as 1D CA: Slicing Constructions

In this section, the slicing constructions that are fundamental to prove the results involving 2D closingness and permutivity are illustrated.

5.4.1 v -Slicing

Fix a vector $v \in \mathbb{Z}^2$ and let $\mathbf{d} \in \mathbb{Z}^2$ be a normalized integer vector (i.e., a vector in which the coordinates are co-prime) perpendicular to v . Consider the line L_0 generated by the vector \mathbf{d} and the set $L_0^* = L_0 \cap \mathbb{Z}^2$ containing vectors of form $\mathbf{i} = t\mathbf{d}$ where $t \in \mathbb{Z}$. Denote by $\varphi : L_0^* \mapsto \mathbb{Z}$ the isomorphism associating any $\mathbf{i} \in L_0^*$ with the integer $\varphi(\mathbf{i}) = t$. Consider now the family \mathcal{L} constituted by all the lines parallel to L_0 containing at least a point of integer coordinates. It is clear that \mathcal{L} is in a one-to-one correspondence with \mathbb{Z} . Let l_a be the axis given by a direction \mathbf{e}_a , which is not contained in L_0 . The lines are enumerated according to their intersection with the axis l_a . Formally, for any pair of lines L_i, L_j it holds that $i < j$ iff $p_i < p_j$ ($p_i, p_j \in \mathbb{Q}$), where $p_i \mathbf{e}_a$ and $p_j \mathbf{e}_a$ are the intersection points between the two lines and the axis l_a , respectively. Equivalently, L_i is the line expressed in parametric form by $\mathbf{i} = p_i \mathbf{e}_a + t\mathbf{d}$ ($\mathbf{i} \in \mathbb{R}^2, t \in \mathbb{R}$) and $p_i = ip_1$, where $p_1 = \min\{p_i, p_i > 0\}$. Remark that $\forall i, j \in \mathbb{Z}$, if $\mathbf{i} \in L_i$ and $\mathbf{j} \in L_j$, then $\mathbf{i} + \mathbf{j} \in L_{i+j}$. Let $\mathbf{j}_1 \in \mathbb{Z}^2$ be an arbitrary but fixed vector of L_1 . For any $i \in \mathbb{Z}$, define the vector $\mathbf{j}_i = i\mathbf{j}_1$ that belongs to $L_i \cap \mathbb{Z}^2$. Then, each line L_i can be expressed in parametric form by $\mathbf{i} = \mathbf{j}_i + t\mathbf{d}$. Note that, for any $\mathbf{i} \in \mathbb{Z}^2$ there exist $i, t \in \mathbb{Z}$, such that $\mathbf{i} = \mathbf{j}_i + t\mathbf{d}$.

The construction can be summarized. We have a countable collection $\mathcal{L} = \{L_i : i \in \mathbb{Z}\}$ of lines parallel to L_0 inducing a partition of \mathbb{Z}^2 . Indeed, defining $L_i^* = L_i \cap \mathbb{Z}^2$, it holds that $\mathbb{Z}^2 = \bigcup_{i \in \mathbb{Z}} L_i^*$ (see Fig. 20).

Once the plane has been sliced, any configuration $c \in A^{\mathbb{Z}^2}$ can be viewed as a mapping $c : \bigcup_{i \in \mathbb{Z}} L_i^* \mapsto \mathbb{Z}$. For every $i \in \mathbb{Z}$, the slice c_i of the configuration c over the line L_i is the

Fig. 20

Slicing of the plane according to the vector $v = (1, 1)$.

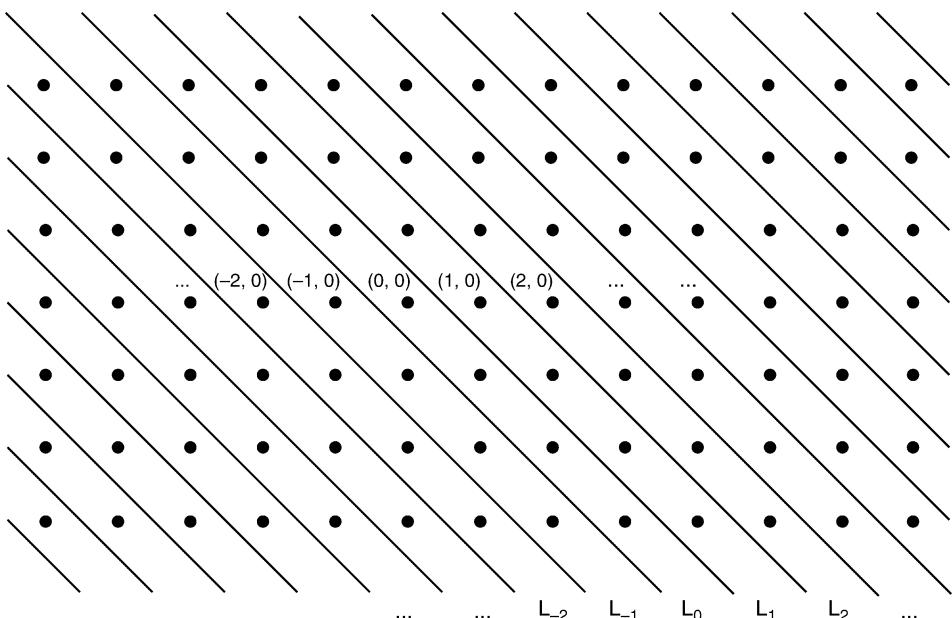
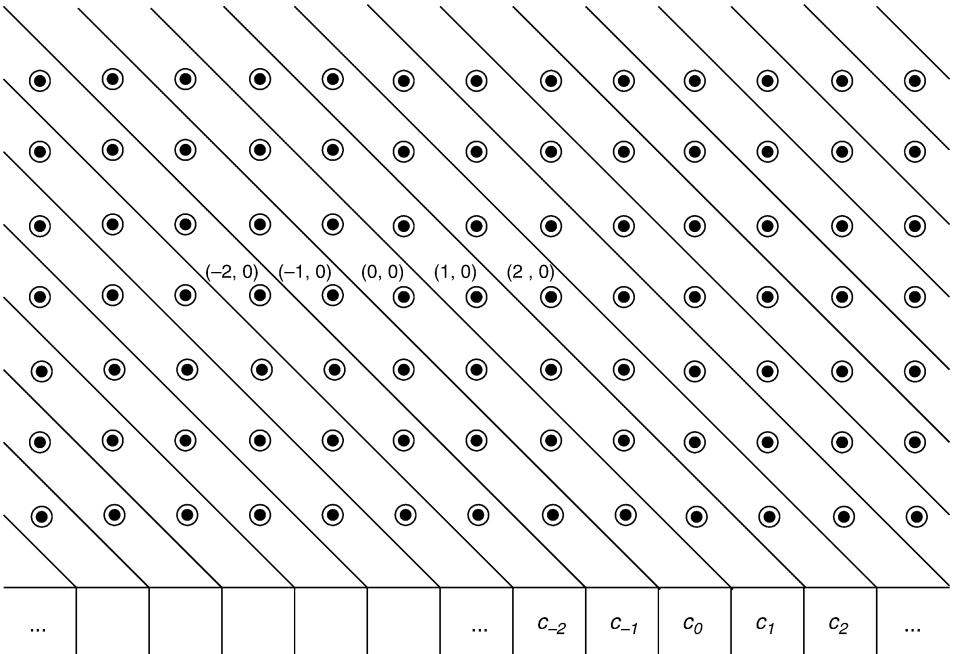


Fig. 21

Slicing of a 2D configuration c according to the vector $v = (1, 1)$. The components of c viewed as a 1D configuration are not from the same alphabet.



mapping $c_i : L_i^* \rightarrow A$. In other terms, c_i is the restriction of c to the set $L_i^* \subset \mathbb{Z}^2$. In this way, a configuration $c \in A^{\mathbb{Z}^2}$ can be expressed as the bi-infinite one-dimensional sequence $\prec c \succ = (\dots, c_{-2}, c_{-1}, c_0, c_1, c_2, \dots)$ of its slices $c_i \in A^{L_i^*}$ where the i -th component of the sequence $\prec c \succ$ is $\prec c \succ_i = c_i$ (see Fig. 21). One can stress that each slice c_i is defined only over the set L_i^* . Moreover, since $\forall i \in \mathbb{Z}^2, \exists! i \in \mathbb{Z} : \mathbf{i} \in L_i^*$, for any configuration c and any vector \mathbf{i} we write $c(\mathbf{i}) = c_i(\mathbf{i})$.

The identification of any configuration $c \in A^{\mathbb{Z}^2}$ with the corresponding bi-infinite sequence of slices $c \equiv \prec c \succ = (\dots, c_{-2}, c_{-1}, c_0, c_1, c_2, \dots)$ allows the introduction of a new one-dimensional bi-infinite CA over the alphabet $A^{\mathbb{Z}}$ expressed by a global transition mapping $F^* : (A^{\mathbb{Z}})^{\mathbb{Z}} \mapsto (A^{\mathbb{Z}})^{\mathbb{Z}}$, which associates any configuration $a : \mathbb{Z} \mapsto A^{\mathbb{Z}}$ with a new configuration $F^*(a) : \mathbb{Z} \mapsto A^{\mathbb{Z}}$. The local rule f^* of this new CA that is about to be defined will take a certain number of configurations of $A^{\mathbb{Z}}$ as input and will produce a new configuration of $A^{\mathbb{Z}}$ as output.

For each $h \in \mathbb{Z}$, define the following bijective map $\mathcal{T}_h : A^{L_h^*} \mapsto A^{L_0^*}$, which associates any slice c_h over the line L_h with the slice $\mathcal{T}_h(c_h)$

$$(c_h : L_h^* \rightarrow A) \xrightarrow{\mathcal{T}_h} (\mathcal{T}_h(c_h) : L_0^* \rightarrow A)$$

defined as $\forall \mathbf{i} \in L_0^*, \mathcal{T}_h(c_h)(\mathbf{i}) = c_h(\mathbf{i} + \mathbf{j}_h)$. Remark that the map $\mathcal{T}_h^{-1} : A^{L_0^*} \rightarrow A^{L_h^*}$ associates any slice c_0 over the line L_0 with the slice $\mathcal{T}_h^{-1}(c_0)$ over the line L_h such that

$\forall \mathbf{i} \in L_h^*, \mathcal{T}_h^{-1}(c_h)(\mathbf{i}) = c_0(\mathbf{i} - \mathbf{j}_h)$. Denote by $\Phi_0 : A^{L_0^*} \rightarrow A^{\mathbb{Z}}$ the bijective mapping putting in correspondence any $c_0 : L_0^* \rightarrow A$ with the configuration $\Phi_0(c_0) \in A^{\mathbb{Z}}$,

$$(c_0 : L_0^* \rightarrow A) \xrightarrow{\Phi_0} (\Phi_0(c_0) : \mathbb{Z}^2 \rightarrow A)$$

such that $\forall t \in \mathbb{Z}, \Phi_0(c_0)(t) := c_0(\varphi^{-1}(t))$. The map $\Phi_0^{-1} : A^{\mathbb{Z}} \rightarrow A^{L_0^*}$ associates any configuration $a \in A^{\mathbb{Z}}$ with the configuration $\Phi_0^{-1}(a) \in A^{L_0^*}$ in the following way: $\forall \mathbf{i} \in L_0^*, \Phi_0^{-1}(a)(\mathbf{i}) = a(\varphi(\mathbf{i}))$. Consider now the bijective map $\Psi : A^{\mathbb{Z}^2} \rightarrow (A^{\mathbb{Z}})^{\mathbb{Z}}$ defined as follows

$$\forall c \in A^{\mathbb{Z}^2}, \quad \Psi(c) = (\dots, \Phi_0(\mathcal{T}_{-1}(c_{-1})), \Phi_0(\mathcal{T}_0(c_0)), \Phi_0(\mathcal{T}_1(c_1)), \dots)$$

Its inverse map $\Psi^{-1} : (A^{\mathbb{Z}})^{\mathbb{Z}} \mapsto A^{\mathbb{Z}^2}$ is such that $\forall a \in (A^{\mathbb{Z}})^{\mathbb{Z}}$

$$\Psi^{-1}(a) = (\dots, \mathcal{T}_{-1}^{-1}(\Phi_0^{-1}(a_{-1})), \mathcal{T}_0^{-1}(\Phi_0^{-1}(a_0)), \mathcal{T}_1^{-1}(\Phi_0^{-1}(a_1)), \dots)$$

Starting from a configuration c , the isomorphism Ψ permits to obtain a one-dimensional configuration a whose components are all from the same alphabet (see Fig. 22).

Now we have all the necessary formalisms to correctly define the radius r^* local rule $f^* : (A^{\mathbb{Z}})^{2r^*+1} \rightarrow A^{\mathbb{Z}}$ starting from a radius r 2D CA F . Let r_1 and r_2 be the indexes of the lines passing for (r, r) and $(r, -r)$, respectively. The radius of the 1D CA is $r^* = \max\{r_1, r_2\}$. In other words, r^* is such that L_{-r^*}, \dots, L_{r^*} are all the lines that intersect the 2D r -radius Moore neighborhood. The local rule is defined as

$$\forall (a_{-r^*}, \dots, a_{r^*}) \in (A^{\mathbb{Z}})^{2r^*+1}, \quad f^*(a_{-r^*}, \dots, a_{r^*}) = \Phi_0(b)$$

where $b : L_0^* \rightarrow A$ is the slice obtained the simultaneous application of the local rule f of the original CA on the slices c_{-r^*}, \dots, c_{r^*} of any configuration c such that $\forall i \in [-r^*, r^*], c_i = \mathcal{T}_i^{-1}(\Phi_0^{-1}(a_i))$ (see Fig. 23). The global map of this new CA is

Fig. 22

All the components of the 1D configuration $a = \Phi(c)$ are from the same alphabet.

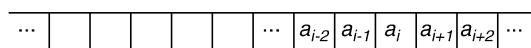
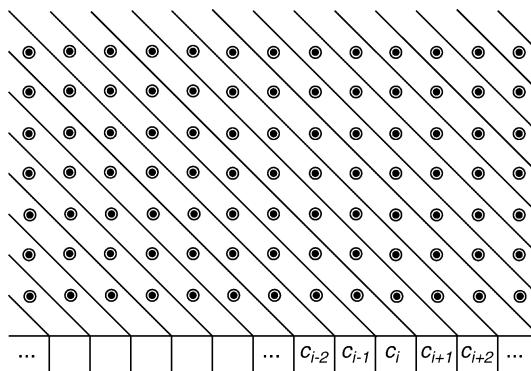
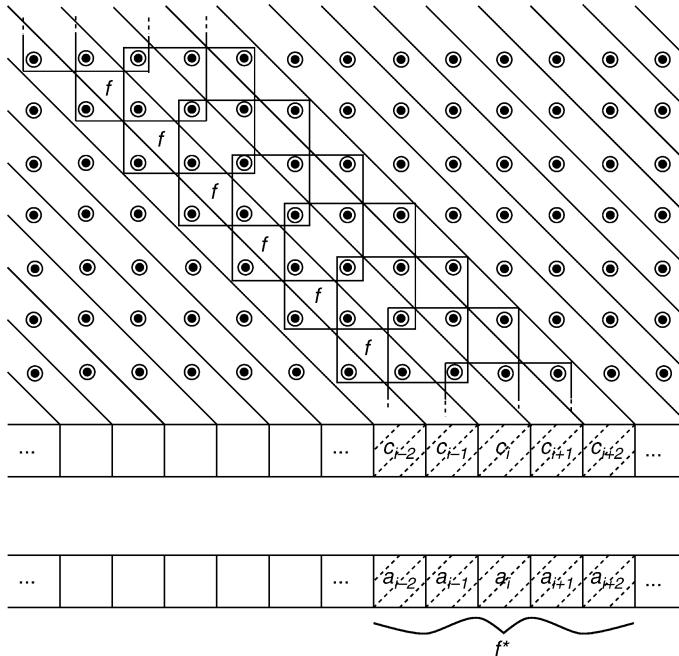


Fig. 23

Local rule f^* of the 1D CA as sliced version of the original 2D CA. Here $r=1$ and $r^*=2$.



$F^* : A^{\mathbb{Z}^2} \rightarrow A^{\mathbb{Z}^2}$ and the link between F^* and f^* is given, as usual, by $(F^*(a))_i = f^*(a_{i-r^*}, \dots, a_{i+r^*})$ where $a = (\dots, a_{-1}, a_0, a_1, \dots) \in (A^{\mathbb{Z}})^{\mathbb{Z}}$ and $i \in \mathbb{Z}$.

The slicing construction can be summarized by the following theorem.

Theorem 23 (Dennunzio and Formenti 2008, 2009) *Let $(A^{\mathbb{Z}^2}, F)$ be a 2D CA and let $((A^{\mathbb{Z}})^{\mathbb{Z}}, F^*)$ be the 1D CA obtained by the v -slicing construction of it, where $v \in \mathbb{Z}^2$ is a fixed vector. The two CA are isomorphic by the bijective mapping Ψ . Moreover, the map Ψ^{-1} is continuous and then $(A^{\mathbb{Z}^2}, F)$ is a factor of $((A^{\mathbb{Z}})^{\mathbb{Z}}, F^*)$.*

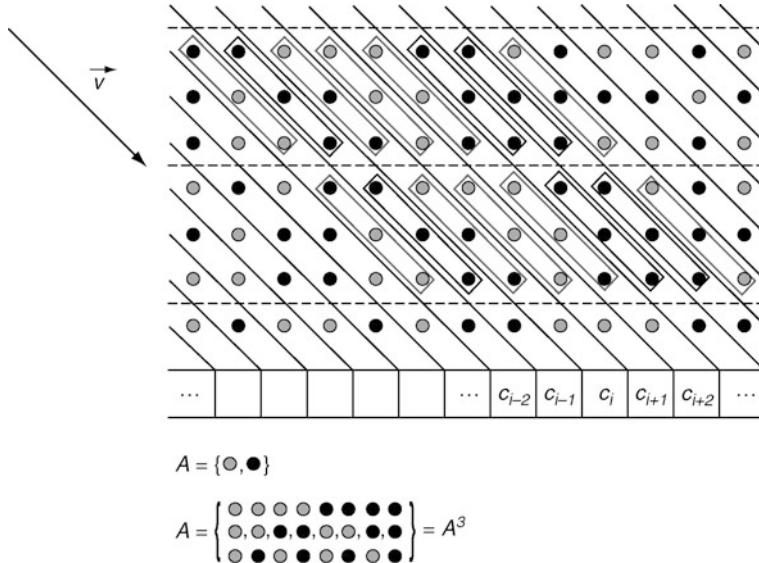
$$\begin{array}{ccc} (A^{\mathbb{Z}})^{\mathbb{Z}} & \xrightarrow{F^*} & (A^{\mathbb{Z}})^{\mathbb{Z}} \\ \downarrow \varphi^{-1} & & \downarrow \varphi^{-1} \\ A^{\mathbb{Z}^2} & \xrightarrow{F} & A^{\mathbb{Z}^2} \end{array}$$

5.4.2 v -Slicing with Finite Alphabet

Fix a vector $v \in \mathbb{Z}^2$. For any 2D CA F , an associated sliced version F^* with finite alphabet can be built. In order to obtain one, it is sufficient to consider the v -slicing construction of the 2D CA restricted on the set S_v , where v is any vector such that $v \perp v$. This is possible since the set S_v is F -invariant and so (S_v, F) is a DDS. The obtained construction leads to the following theorem.

Fig. 24

Example of v -slicing of a configuration $c \in S_v$ on the binary alphabet A where $v = (1, 1)$ and $v = (3, -3)$. The configuration $\Psi(c)$ is on the alphabet $B = A^3$.



Theorem 24 (Dennunzio and Formenti 2008, 2009) Let F be a 2D CA. Consider the v -slicing construction of F , where $v \in \mathbb{Z}^2$ is a fixed vector. For any vector $\bar{v} \in \mathbb{Z}^2$ with $v \perp \bar{v}$, the DDS (S_v, F) is topologically conjugated to the 1D CA $(B^{\mathbb{Z}}, F^*)$ on the finite alphabet $B = A^{|\bar{v}|}$ obtained by the v -slicing construction of F restricted on S_v (Fig. 24).

$$\begin{array}{ccc} B^{\mathbb{Z}} & \xrightarrow{F^*} & B^{\mathbb{Z}} \\ \downarrow \psi^{-1} & & \downarrow \psi^{-1} \\ S_v & \xrightarrow[F]{} & S_v \end{array}$$

The previous result is very useful since one can use all the well-known results about 1D CA and try to lift them to F .

5.5 2D Closingness

One can now generalize to 2D CA the notion of closingness.

Let $v \in \mathbb{Z}^2$ and denote $\bar{v} = -v$.

Definition 14 (v -asymptotic configurations) Two configurations $x, y \in A^{\mathbb{Z}^2}$ are v -asymptotic if there exists $q \in \mathbb{Z}$ such that $\forall i \in \mathbb{Z}^2$ with $v \cdot i \geq q$ it holds that $x_i = y_i$.

Definition 15 (v -closingness) A 2D CA F is v -closing is for any pair of \bar{v} -asymptotic configurations $x, y \in A^{\mathbb{Z}^2}$ we have that $x \neq y$ implies $F(x) \neq F(y)$. A 2D CA is *closing* if it is v -closing for some v . A 2D CA is *bi-closing* if it is both v and \bar{v} -closing for some $v \in \mathbb{Z}^2$.

Note that a 2D CA can be closing with respect to a certain direction but it cannot with respect to another one. For example, consider the radius $r = 1$ 2D CA on the binary alphabet whose local rule performs the xor operator on the four corners of the Moore neighborhood. It is easy to observe that this CA is $(1, 1)$ -closing but it is not $(1, 0)$ -closing.

Thanks to the v -slicing construction with finite alphabet, the following property holds.

Proposition 45 (Dennunzio and Formenti 2008, 2009) *Let F be a v -closing 2D CA. For any vector $\mathbf{v} \in \mathbb{Z}^2$ with $\mathbf{v} \perp v$, let $(B^\mathbb{Z}, F^*)$ be the 1D CA of [Theorem 24](#) which is topologically conjugated to (S_v, F) . Then F^* is either right or left closing.*

Using [Proposition 45](#) and the fact that closing 1D CA have DPO, it is possible to prove the following theorem.

Theorem 25 (Dennunzio and Formenti 2008, 2009) *Any closing 2D CA has DPO.*

Corollary 7 *Any closing 2D CA is surjective.*

Regarding decidability, we have the following result whose proof is based on tiling and plane filling curves as in Kari (1994).

Proposition 46 (Dennunzio et al. 2009b) *For any $v \in \mathbb{Z}^2$, v -closingness is an undecidable property.*

Recalling that closingness is decidable in dimension 1, it is determined to be another dimension sensitive property (see Kari 1994; Bernardi et al. 2005 for other examples).

5.5.1 2D Openness

The relation between openness and closingness can be now dealt with. Recall that in 1D case, a CA is open iff it is both left and right closing. In 2D settings, we have the following properties.

Theorem 26 (Dennunzio et al. 2009b) *If a 2D CA F is bi-closing, then F is open.*

Proposition 47 (Dennunzio and Formenti 2008a, 2009) *Any open 2D CA is surjective.*

5.6 2D Permutivity

The notion of 1D permutive CA can be conveniently extended to 2D CA as follows.

Let $\tau \in \{(1, 1)(1, -1)(-1, -1)(-1, 1)\}$

Definition 16 (permutivity) A 2D CA of local rule f and radius r is τ -permutive, if for each pair of matrices $N, N' \in M_r$ with $N(\mathbf{i}) = N'(\mathbf{i})$ in all vectors $\mathbf{i} \neq r\tau$, it holds that $N(r\tau) \neq N'(r\tau)$ implies $f(N) \neq f(N')$. A 2D CA is bi-permutive iff it is both τ permutive and $\bar{\tau}$ -permutive.

Several relationships among τ -permutivity, closingness, and dynamical properties are presented.

Firstly, as a consequence of the τ -slicing construction we have the following.

Proposition 48 (Dennunzio and Formenti 2008, 2009) *Consider a τ -permutive 2D CA F . For any v belonging either to the same quadrant or the opposite one as τ , the 1D CA $((A^{\mathbb{Z}})^{\mathbb{Z}}, F^*)$ obtained by the v -slicing construction is either rightmost or leftmost permutive.*

Theorem 27 (Dennunzio and Formenti 2008, 2009) *Any bi-permutive 2D CA is strongly transitive.*

Theorem 28 (Dennunzio and Formenti 2008, 2009) *Any τ -permutive 2D CA is topologically mixing.*

Theorem 29 (Dennunzio and Formenti 2008, 2009) *Consider a τ -permutive 2D CA F . For any v belonging to the same quadrant as τ , F is v -closing.*

Theorem 30 (Dennunzio and Formenti 2008, 2009) *Any τ -permutive 2D CA has DPO.*

Acknowledgments

The research was supported by the Research Program CTS MSM 0021620845, by the Interlink/MIUR project “Cellular Automata: Topological Properties, Chaos and Associated Formal Languages,” by the ANR Blanc “Projet Sycomore” and by the PRIN/MIUR project “Mathematical aspects and forthcoming applications of automata and formal languages.”

References

- Acerbi L, Dennunzio A, Formenti E (2007) Shifting and lifting of cellular automata. In: Third conference on computability in Europe (CiE 2007), Lecture notes in computer science, vol 4497. Springer, Siena, Italy, pp 1–10
- Acerbi L, Dennunzio A, Formenti E (2009) Conservation of some dynamical properties for operations on cellular automata. *Theor Comput Sci* 410 (38–40):3685–3693. <http://dx.doi.org/10.1016/j.tcs.2009.05.004>; <http://dblp.uni-trier.de>
- Akin E (1991) The general topology of dynamical systems. American Mathematical Society, Providence, RI
- Amoroso S, Patt YN (1972) Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *J Comput Syst Sci* 6:448–464
- Bernardi V, Durand B, Formenti E, Kari J (2005) A new dimension sensitive property for cellular automata. *Theor Comput Sci* 345:235–247
- Blanchard F, Maass A (1997) Dynamical properties of expansive one-sided cellular automata. *Isr J Math* 99:149–174
- Blanchard F, Tisseur P (2000) Some properties of cellular automata with equicontinuity points. *Ann Inst Henri Poincaré, Probabilité et Statistiques* 36:569–582
- Boyle M, Kitchens B (1999) Periodic points for cellular automata. *Indagat Math* 10:483–493
- Cattaneo G, Finelli M, Margara L (2000) Investigating topological chaos by elementary cellular automata dynamics. *Theor Comput Sci* 244:219–241
- Cattaneo G, Dennunzio A, Margara L (2002) Chaotic subshifts and related languages applications to one-dimensional cellular automata. *Fundam Inf* 52:39–80
- Cattaneo G, Dennunzio A, Margara L (2004) Solution of some conjectures about topological properties of linear cellular automata. *Theor Comput Sci* 325: 249–271

- Cattaneo G, Dennunzio A, Formenti E, Proville J (2009) Non-uniform cellular automata. In: Horia Dediu A, Armand-Mihai Ionescu, Martín-Vide C (eds) Proceedings of third international conference on language and automata theory and applications (LATA 2009), 2–8 April 2009. Lecture notes in computer science. Springer, vol 5457, pp 302–313. http://dx.doi.org/10.1007/978-3-642-00982-2_26; [conf/lata/2009](http://dblp.uni-trier.de); <http://dblp.uni-trier.de>; <http://dx.doi.org/10.1007/978-3-642-00982-2>; <http://dblp.uni-trier.de> doi:978-3-642-00981-5
- Cervelle J, Dennunzio A, Formenti E (2008) Chaotic behavior of cellular automata. In: Meyers B (ed) Mathematical basis of cellular automata, Encyclopedia of complexity and system science. Springer, Berlin, Germany
- Chaudhuri P, Chowdhury D, Nandi S, Chattopadhyay S (1997) Additive cellular automata theory and applications, vol 1. IEEE Press, Mountain View, CA
- Chopard B (2012) Cellular automata and lattice Boltzmann modeling of physical systems. Handbook of natural computing. Springer, Heidelberg, Germany
- de Sá PG, Maes C (1992) The Gacs-Kurdyumov-Levin automaton revisited. *J Stat Phys* 67(3/4):507–522
- Dennunzio A, Formenti E (2008) Decidable properties of 2D cellular automata. In: Twelfth conference on developments in language theory (DLT 2008), Lecture notes in computer science, vol 5257. Springer, New York, pp 264–275
- Dennunzio A, Formenti E (2009) 2D cellular automata: new constructions and dynamics, 410(38–40): 3685–3693
- Dennunzio A, Guillon P, Masson B (2008) Stable dynamics of sand automata. In: Fifth IFIP conference on theoretical computer science. TCS 2008, Milan, Italy, September, 8–10, 2008, vol 273, IFIP, Int. Fed. Inf. Process. Springer, Heidelberg, Germany, pp 157–179
- Dennunzio A, Di Lena P, Formenti E, Margara L (2009a) On the directional dynamics of additive cellular automata. *Theor Comput Sci* 410(47–49):4823–4833. <http://dx.doi.org/10.1016/j.tcs.2009.06.023>; <http://dblp.uni-trier.de>
- Dennunzio A, Formenti E, Weiss M (2009b) 2D cellular automata: expansivity and decidability issues. CoRR, abs/0906.0857 <http://arxiv.org/abs/0906.0857>; <http://dblp.uni-trier.de>
- Dennunzio A, Guillon P, Masson B (2009c) Sand automata as cellular automata. *Theor Comput Sci* 410 (38–40):3962–3974. <http://dx.doi.org/10.1016/j.tcs.2009.06.016>; <http://dblp.uni-trier.de>
- Di Lena P (2006) Decidable properties for regular cellular automata. In: Fourth IFIP conference on theoretical computer science. TCS 2006, Santiago, Chile, August, 23–24, 2006, IFIP, Int. Fed. Inf. Process. vol 209. Springer, pp 185–196
- Di Lena P, Margara L (2008) Computational complexity of dynamical systems: the case of cellular automata. *Inf Comput* 206:1104–1116
- Di Lena P, Margara L (2009) Undecidable properties of limit set dynamics of cellular automata. In: Albers S, Marion J-Y (eds) Proceedings of 26th international symposium on theoretical aspects of computer science (STACS 2009), 26–28 February 2009, Freiburg, Germany, vol 3, pp 337–347. <http://dx.doi.org/10.4230/LIPIcs.STACS.2009.1819>; [conf/stacs/2009](http://dblp.uni-trier.de); <http://dblp.uni-trier.de>
- Durand B (1993) Global properties of 2D cellular automata: some complexity results. In: MFCS, Lecture notes in computer science, vol 711. Springer, Berlin, Germany, pp 433–441
- Durand B (1998) Global properties of cellular automata. In: Goles E, Martinez S (eds) Cellular automata and complex systems. Kluwer, Dordrecht, The Netherlands
- Farina F, Dennunzio A (2008) A predator-prey cellular automaton with parasitic interactions and environmental effects. *Fundam Inf* 83:337–353
- Formenti E, Grange A (2003) Number conserving cellular automata II: dynamics. *Theor Comput Sci* 304(1–3):269–290
- Formenti E, Kůrka P (2007) Subshift attractors of cellular automata. *Nonlinearity* 20:105–117
- Formenti E, Kůrka P (2009) Dynamics of cellular automata in non-compact spaces. In: Robert A. Meyers (ed) Mathematical basis of cellular automata, Encyclopedia of complexity and system science. Springer, Heidelberg, Germany, pp 2232–2242. http://dx.doi.org/10.1007/978-0-387-30440-3_138; [reference/complexity/2009](http://dblp.uni-trier.de); <http://dblp.uni-trier.de>
- Formenti E, Kůrka P, Zahradník O (2010) A search algorithm for subshift attractors of cellular automata. *Theory Comput Syst* 46(3):479–498. <http://dx.doi.org/10.1007/s00224-009-9230-6>; <http://dblp.uni-trier.de>
- Gacs P (2001) Reliable cellular automata with self-organization. *J Stat Phys* 103(1/2):45–267
- Gacs P, Kurdyumov GL, Levin LA (1978) One-dimensional uniform arrays that wash out finite islands. *Perechi Informatiki* 14:92–98
- Hedlund GA (1969) Endomorphisms and automorphisms of the shift dynamical system. *Math Syst Theory* 3:320–375
- Hurley M (1990) Attractors in cellular automata. *Ergod Th Dynam Syst* 10:131–140
- Kari J (1994) Reversibility and surjectivity problems of cellular automata. *J Comput Syst Sci* 48:149–182
- Kari J (2008) Tiling problem and undecidability in cellular automata. In: Meyers B (ed) Mathematical

- basis of cellular automata, Encyclopedia of complexity and system science. Springer, Heidelberg, Germany
- Kitchens BP (1998) Symbolic dynamics. Springer, Berlin, Germany
- Kůrka P (1997) Languages, equicontinuity and attractors in cellular automata. *Ergod Th Dynam Syst* 17: 417–433
- Kůrka P (2003a) Cellular automata with vanishing particles. *Fundam Inf* 58:1–19
- Kůrka P (2003b) Topological and symbolic dynamics, Cours spécialisés, vol 11. Société Mathématique de France, Paris
- Kůrka P (2005) On the measure attractor of a cellular automaton. *Discrete Continuous Dyn Syst* 2005 (suppl):524–535
- Kůrka P (2007) Cellular automata with infinite number of subshift attractors. *Complex Syst* 17(3):219–230
- Kůrka P (2008) Topological dynamics of one-dimensional cellular automata. In: Meyers B (ed) Mathematical basis of cellular automata, Encyclopedia of complexity and system science. Springer, Heidelberg, Germany
- Kůrka P, Maass A (2000) Limit sets of cellular automata associated to probability measures. *J Stat Phys* 100 (5/6):1031–1047
- Lind D, Marcus B (1995) An introduction to symbolic dynamics and coding. Cambridge University Press, Cambridge
- Maruoka A, Kimura M (1976) Conditions for injectivity of global maps for tessellation automata. *Inf Control* 32:158–162
- Mitchell M, Crutchfield JP, Hraber PT (1994) Evolving cellular automata to perform computations: mechanisms and impediments. *Physica D* 75:361–391
- Moore EF (1962) Machine models of self-reproduction. *Proc Symp Appl Math* 14:13–33
- Myhill J (1963) The converse to Moore's Garden-of-Eden theorem. *Proc Am Math Soc* 14:685–686
- Nasu M (1995) Textile systems for endomorphisms and automorphisms of the shift, Memoires of the American Mathematical Society, vol 114. American Mathematical Society, Providence, RI
- Pivato M (2008) The ergodic theory of cellular automata. In: Meyers B (ed) Mathematical basis of cellular automata, Encyclopedia of complexity and system science. Springer, Heidelberg, Germany
- Sablik M (2008) Directional dynamics for cellular automata: a sensitivity to the initial conditions approach. *Theor Comput Sci* 400:1–18
- Shereshevsky MA (1993) Expansiveness, entropy and polynomial growth for groups acting on subshifts by automorphisms. *Indagat Math* 4:203–210
- Shereshevsky MA, Afraimovich VS (1992) Bipermutative cellular automata are topologically conjugate to the one-sided Bernoulli shift. *Random Comput Dyn* 1:91–98
- Theysier G, Sablik M (2008) Topological dynamics of 2D cellular automata. In: Computability in Europe (CIE'08), Lecture notes in computer science, vol 5028, pp 523–532
- Wolfram S (1986) Theory and applications of cellular automata. World Scientific, Singapore

3 Algorithmic Tools on Cellular Automata

Marianne Delorme¹ · Jacques Mazoyer²

¹Laboratoire d’Informatique Fondamentale de Marseille (LIF),
Aix-Marseille Université and CNRS, Marseille, France
delorme.marianne@orange.fr

²Laboratoire d’Informatique Fondamentale de Marseille (LIF),
Aix-Marseille Université and CNRS, Marseille, France
mazoyerj2@orange.fr

1	<i>Introduction</i>	78
2	<i>Signals</i>	80
3	<i>Cellular Translations of Finite Right Signals</i>	86
4	<i>Various Constructions of Signals Starting from Signals</i>	94
5	<i>Functions Constructions</i>	100
6	<i>Basic Tools and Methods for Moving Information</i>	108
7	<i>Conclusion</i>	121

Abstract

This chapter is dedicated to classic tools and methods involved in cellular transformations and constructions of signals and of functions by means of signals, which will be used in subsequent chapters. The term “signal” is widely used in the field of cellular automata (CA). But, as it arises from different levels of understanding, a general definition is difficult to formalize. This chapter deals with a particular notion of signal, which is a basic and efficient tool in cellular algorithmics.

1 Introduction

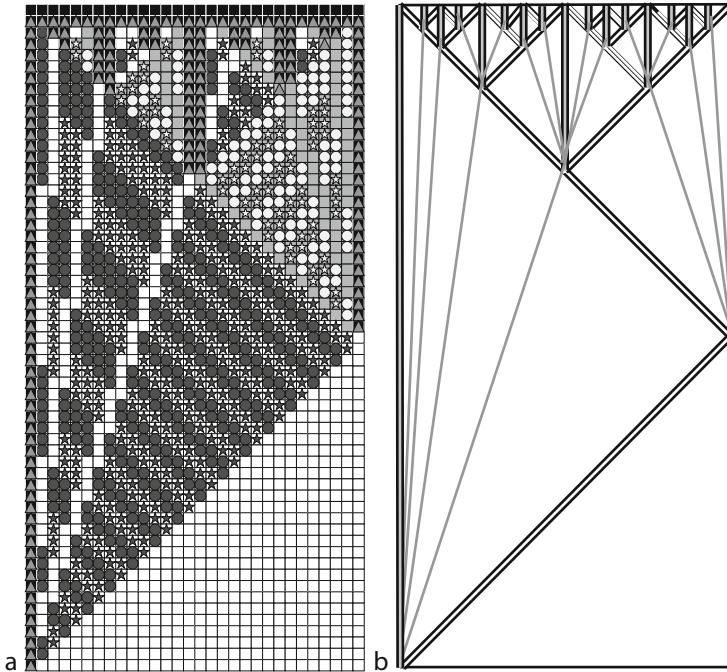
The first significant works that lead to the notion of cellular automaton were those of von Neumann (1966) and Ulam (1957). Von Neumann adopted an engineering point of view: how to build an object with a wanted global behavior by means of local interactions between different components. S. Ulam observed that iterations of simple local rules may lead to global complex behaviors and interesting geometric constructions. While von Neumann’s mechanical engineering project failed, the abstract cellular automaton (CA) object emerged and both processes or problematics – to look for bricks locally organized in order to obtain some wanted global result, or to discover global results from given local means – have since been at play in the history of cellular automata. Two powerful tools play a basic role in this history, especially in the emergence of “signals,” namely *space–time* and *geometric diagrams*. The space–time diagrams are a way to represent the evolution of configurations by piling them up in succession, as an illustration of orbits of configurations. Geometric diagrams are a way to represent, when possible, a realization of an algorithm: a kind of abstraction of space–time diagrams. From these representations emerge patterns that seem significant and are diversely interpreted and dealt with, depending on the observation field, for example, as particles and backgrounds, signals, or information moves.

➊ *Figure 1a* shows one space–time diagram and ➋ *Fig. 1b* one geometric diagram (actually a decorated geometric diagram: to get a “true” (underlying) geometric diagram, let us replace discrete straight lines by continuous lines by removing the colors represented by white squares or gray triangles in black squares) of a solution of the FSSP (Balzer 1966). It gives an immediate intuition of “signals” as segments of discrete or real lines, carrying and transmitting information inside some “universe.” Although this example may lead one to think there is some canonical correspondence between space–time and geometrical diagrams, it is not the case. Actually, a space–time diagram cannot necessarily be translated into a geometric one, and there are algorithms whose realizations cannot be drawn as geometric diagrams. Nevertheless, what follows will prove the power of these representations.

This chapter explains the classic tools and methods that lie at the foundations of cellular algorithmics in dimension 1 and that are essentially based on cellular constructions or transformations of signals. In the second section, after general definitions – in particular, the definition of a signal – an example of a construction of a signal is developed. This is done in order to introduce a more general notion of transformations by cellular automata, which is used in the following section. In the third section, the transformation formalism allows one to get the first negative result. The rest of the section is then dedicated to a special transformation: the translation of a finite right signal. The question about the uniformization of this translation is posed. A lot of techniques are used, such as extraction of signal moves, reconstruction of a signal, and data duplication. In the fourth section, other constructions are developed, concerning deformations of signals, and constructions of infinite families of

Fig. 1

(a) An evolution of the automaton. (b) A corresponding geometric diagram.



signals, discrete parabolas and exponentials. The fifth section is devoted to constructions of functions. Fischer's constructibility is explained and a limitation on signals slopes is presented and proved. In the sixth section, other tools and methods are presented on information transfers, arithmetic operations on segments, a technique called “riddle,” the technique of freezing, as well as its generalization that is called “clipping.” Finally, the seventh section is a conclusion, in which the question of signals in dimension 2 is also discussed.

Some definitions and notations are recalled here.

Definition 1 (one-dimensional cellular automata)

1. A 1D-CA is a 3-uplet $\mathcal{A} = (Q, V, \delta)$ where:
 - $Q = \{q_1, \dots, q_{|Q|}\}$ is a finite set (the *states* set of \mathcal{A}).
 - $V = \{v_1, \dots, v_{|V|}\}$ is a finite ordered subset of \mathbb{Z} , called the *neighborhood*, which from now on will be $\{-1, 0, +1\}$, unless clearly mentioned (so that \mathcal{A} becomes $\mathcal{A} = (Q, \delta)$).
 - δ is a mapping from $Q^{|V|}$ into Q (the *states* (or *local*) *transition function* of \mathcal{A}).
2. A *configuration* of \mathcal{A} is a mapping from \mathbb{Z} into Q . The set $Q^{\mathbb{Z}}$ of \mathcal{A} -configurations is denoted \mathcal{C} .

The *global transition function* of \mathcal{A} , $G : \mathcal{C} \rightarrow \mathcal{C}$, is defined by:

$$\forall x \in \mathcal{C}, \forall z \in \mathbb{Z}, G(x)(z) = \delta(x(z+v_1), \dots, x(z+v_{|V|}))$$

$G(x)(z)$ will also be denoted $G(x)_z$.

The *orbit* of a configuration x is the sequence $\mathcal{O}(x) = \{G^t(x) \mid t \in \mathbb{N}\}$.

3. A state q of Q is said to be *quiescent* when $\delta(q, \dots, q) = q$.

Restricting the neighborhood of CAs to $\{-1, 0, +1\}$ is not a problem. Indeed, let $\mathcal{A} = (Q, \{-2, -1, 0, +1, +2\}, \delta_{\mathcal{A}})$ be considered, for example. Then, there exists an automaton $\mathcal{B} = (Q^2, \{-1, 0, +1\}, \delta_{\mathcal{B}})$ that has the behavior of \mathcal{A} at the expense of some transformation. Here, $\delta_{\mathcal{B}}$ operates as follows:

- Starting from three states of \mathcal{B} , $(q_{\ell,-1}, q_{r,-1})$, $(q_{\ell,0}, q_{r,0})$, $(q_{\ell,+1}, q_{r,+1})$, one gets $q_{\ell} = \delta_{\mathcal{A}}(q_{\ell,-1}, q_{r,-1}, q_{\ell,0}, q_{r,0}, q_{\ell,+1})$ and $q_r = \delta_{\mathcal{A}}(q_{r,-1}, q_{\ell,0}, q_{r,0}, q_{\ell,+1}, q_{r,+1})$. Then $\delta_{\mathcal{B}}((q_{\ell,-1}, q_{r,-1}), (q_{\ell,0}, q_{r,0}), (q_{\ell,+1}, q_{r,+1})) = (q_{\ell}, q_r)$.

In fact, it is possible to build by grouping, for each cellular automaton \mathcal{A} and all (m, p) in $\mathbb{N} \times \mathbb{N}$ (where m is the space factor and p the time factor), the automaton $\mathcal{A}^{(m,p)}$, which is algorithmically equivalent to \mathcal{A} (Rapaport 1998; Ollinger 2002; Theyssier 2005). (See also the chapter [► Universalities in Cellular Automata](#) of this volume.) The neighborhood of $\mathcal{A}^{(m,p)}$ is $\{-1, 0, +1\}$ for all sufficiently large m .

Orbits of random configurations of some cellular automata show regular arrangements of states, sometimes looking like threads, which can be diversely interpreted, but are usually considered to be significant and carrying “information”. Here, we focus on some of them that we call signals. Understanding how pieces of information advance through a space-time diagram amounts to considering the states set Q as $Q_1 \cup Q_2$, Q_1 representing information which progresses through the states of Q_2 which can then be considered as equivalent relative to Q_1 . This justifies the identification of Q_2 states as a unique state, L , which is latent for Q_1 states. Therefore, this chapter considers configurations made of a finite segment of Q_1 states inside lines of $L : {}^\infty L q_0 \dots q_\ell L^\infty$. Using grouping, this can be seen as ${}^\infty L q L^\infty$. (In the following, we shall assume that the first non-quiescent state is on cell 0.)

2 Signals

2.1 Introduction

As the interest here is in cellular automata, the idea of signals is obtained by observing orbits; however, in reality it more generally concerns colorings of the discrete plane. Starting with a general idea, the attention here is restricted to signals that are constructible (or generated) by cellular automata.

2.2 Basic Definitions

Definition 2 (signal)

1. Let C be some finite set, $S \subset C$, and let $\mu : \mathbb{Z} \times \mathbb{N} \rightarrow C$ be some coloring of $\mathbb{Z} \times \mathbb{N}$. The coloring μ is said to present a *signal* \mathcal{S} on S if \mathcal{S} is a sequence of sites $(x_k, t_k)_{k < \alpha}$ (elements of $\mathbb{Z} \times \mathbb{N}$) with indices in an initial segment of \mathbb{N} (possibly whole \mathbb{N}), such that:
 - $\mu(x_k, t_k)$ is in S .
 - (x_{k+1}, t_{k+1}) is $(x_k, t_k + 1)$ or $(x_k + 1, t_k)$ or $(x_k - 1, t_k)$.

- If (x_{k+1}, t_{k+1}) is $(x_k, t_k + 1)$, then $\mu(x_k + 1, t_k) \notin S$ or $(x_k + 1, t_k)$ is (x_{k-1}, t_{k-1}) , and $\mu(x_k - 1, t_k) \notin S$ or $(x_k - 1, t_k)$ is (x_{k-1}, t_{k-1}) .
- If (x_{k+1}, t_{k+1}) is $(x_k + 1, t_k)$, then $\mu(x_k - 1, t_k) \notin S$.
- If (x_{k+1}, t_{k+1}) is $(x_k - 1, t_k)$, then $\mu(x_k + 1, t_k) \notin S$.

Usually, because the context is clear, one speaks of a signal S without mentioning the coloring that presents it. The state (in the signal) of a site (x_k, t_k) is sometimes denoted by $\langle x_k, t_k \rangle$.

2. A signal S presented by some coloring μ is said to be CA-generated if there exists some cellular automaton, $\mathcal{A} = (Q, \delta)$, and some element $c_{\mathcal{A}}$ of $Q^{\mathbb{Z}}$ such that μ is the orbit of the configuration $c_{\mathcal{A}}$.

A signal S presented by some coloring μ is said to be CA-constructible if there exists some cellular automaton $\mathcal{A} = (Q, \delta)$ with a quiescent state L , a distinguished state G and $S \subseteq Q \setminus \{L\}$ such that μ is the orbit of the configuration ${}^\infty L G L {}^\infty$. Such a configuration is called the *basic initial configuration*.

3. In what follows, sites (x_k, t_k) will represent both positions and states $\mu(x_k, t_k)$, and signals will be denoted by \mathcal{S}_S or \mathcal{S} .
4. A *signal* is said to be *right (left)* if, whatever k is, (x_{k+1}, t_{k+1}) is $(x_k, t_k + 1)$ or $(x_k + 1, t_k)$ ($(x_k, t_k + 1)$ or $(x_k - 1, t_k)$, respectively).
5. Let S be a right signal with $(x_0, t_0) = (0, 0)$ and let ξ be in $\{x_k | k < \alpha\}$. Let η_{ξ} denote the smallest time t for S to reach the cell ξ , that is, $\eta_{\xi} = t_{k_0}$ where $k_0 < \alpha$ is such that
 - $x_{k_0} = \xi$ and $t_{k_0} = \min\{t | \exists k (x_k = \xi \text{ and } t = t_k)\}$
 - For each k , $k < k_0$, $x_k \neq \xi$

The function $\Pi_S : \xi \mapsto \eta_{\xi}$ is called the *slope* of S . To the slope corresponds the *speed* of S that is the function $\mathcal{V}_S : t \mapsto (\max_{\eta_{\xi} \leq t} \xi)/t$.

Most of time, signals are discrete straight lines. Their slopes are then simply identified to the slopes of these straight lines.

6. If a signal S is finite,
 - Its site (x_0, t_0) is said to be its *origin* if for all $t < t_0$, all x , $(x, t) \notin S$ and $(x_0, t_0 + 1) \in S$ (then $(x_0 - 1, t_0) \notin S$, $(x_0 + 1, t_0) \notin S$ or $(x_0 + 1, t_0) \in S$ and $(x_0 + 1, t_0 + 1) \in S$ or $(x_0 - 1, t_0) \in S$ and $(x_0 - 1, t_0 + 1) \in S$).
 - Its site (x_f, t_f) is said to be its *end* if for all $t > t_f$, all x , $(x, t) \notin S$ and $(x_f, t_f - 1) \in S$ or $(x_f - 1, t_f)$ and $(x_f - 1, t_f - 1)$ belong to S or $(x_f + 1, t_f)$ and $(x_f + 1, t_f - 1)$ belong to S .

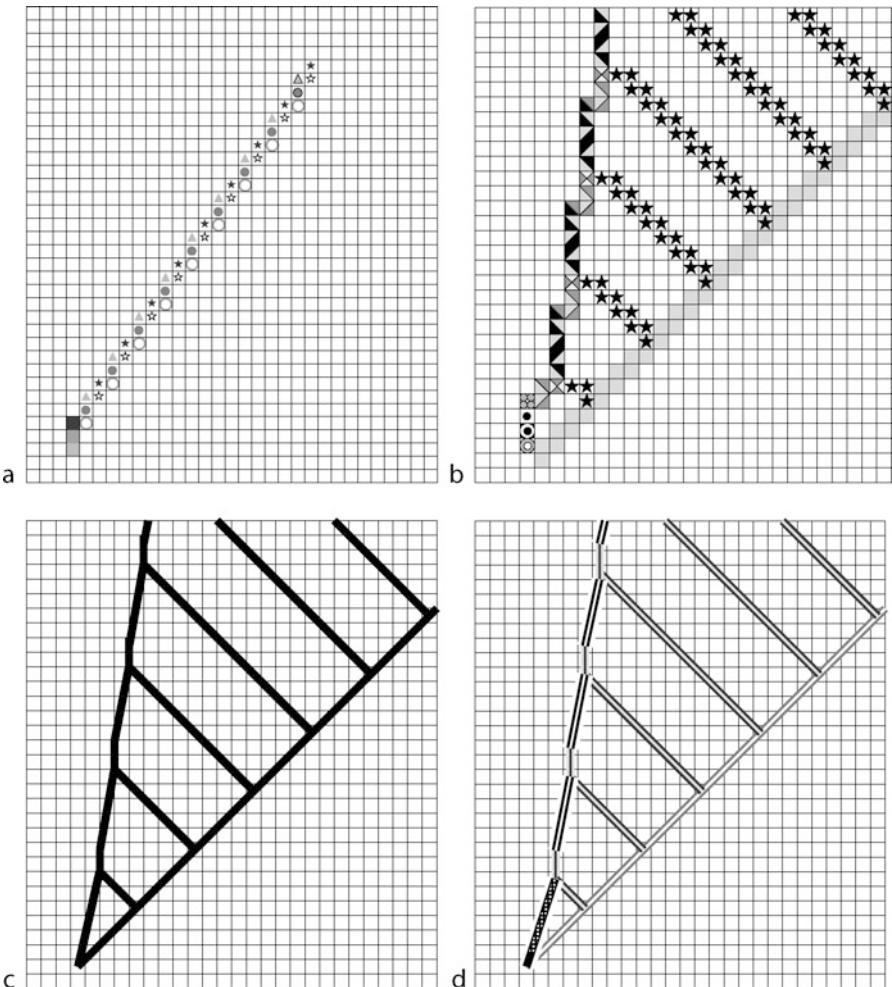
Due to the choice of the initial configuration, if a signal S is CA-constructed by some cellular automaton (Q, δ) , and is the only one to be constructed on ${}^\infty L G L {}^\infty$, then it is always an ultimately periodic sequence (of states in S , as well as in its slope and speed (Mazoyer 1989)). In [Fig. 2a](#), for example, the signal is of period $(2, 3)$, the slope is $\frac{3}{2}$ and the speed $\frac{2}{3}$ (“it advances of two cells in three time steps”).

Usually, several signals may appear in an orbit, as it is the case in [Fig. 2b](#). Clearly, the choice to use 4-connected signals is arbitrary. One may consider 8-connected signals as well: in this case, there exists a uniform way to translate a 8-connected signal into a 4-connected one (e.g., if $(x_{k+1}, t_{k+1}) = (x_k + 1, t_k + 1)$, add $(x_k, t_k + 1)$ to the signal and renumber it).

Definition 3 (signals emissions and interactions) Let \mathcal{S}_S be a signal generated by some cellular automaton $\mathcal{A} = (Q \cup \{L\}, \delta)$ ([Definition 2](#)) and let K be a subset of Q .

Fig. 2

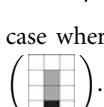
(a) An example of a single signal. (b) Several signals inside an orbit. (c) Geometric representation of a signal. (d) Geometric representation of a signal with indications.



1. \mathcal{S}_S is said to *send out* or *emit* a signal \mathcal{S}_K if there is some k such that:

- (x_k, t_k) belongs to \mathcal{S}_S
- $\langle x_k - 1, t_k + 1 \rangle \in K$
- $\langle x_k - 1, t_k + 2 \rangle \in K$

or the symmetrical layout where $(x_k + 1, t_k + 1)$ and $(x_k + 1, t_k + 2)$ belong to K , or the case where $(x_k, t_k) \in S$, $(x_k, t_k + 1)$ and $(x_k, t_k + 2)$ belong to K : $\begin{pmatrix} \text{grey} \\ \text{black} \end{pmatrix}$, $\begin{pmatrix} \text{black} \\ \text{grey} \end{pmatrix}$,



2. A signal \mathcal{S}_S is said to *interact* with some signal \mathcal{S}_K if (ξ, η) and $(\xi + 1, \eta)$ belong to S and K , respectively (or K and S) (), or if (ξ, η) and $(\xi + 2, \eta)$ belong to S and K , respectively (or K and S) ().

In [Fig. 2b](#), several signals may be distinguished. First, there is the signal on the only gray state, which goes to the right at speed 1. Second, there are the only star state signals, which are sent out to the left, at speed 1, by the gray signal. Finally, there is a right signal defined on all the other states that interacts with the star signals. But the previous figure can also be considered to be made of several finite signals. The site of the origin of the gray signal can be assumed to be $(1, 1)$. Then, one can distinguish a first finite signal \mathcal{S}_0 from the site $(0, 2)$ to $(2, 6)$, a second one \mathcal{S}_1 , sent out by the first star signal, from $(2, 7)$ to $(2, 9)$, which sends out a new one \mathcal{S}'_1 from $(2, 10)$ up to $(3, 13)$, which interacts with the second star signal, which in turn sends out a new signal \mathcal{S}_2 , and so on.

This example leads to at least two remarks about the notion of a signal. First, observing the star signals (as well as the \mathcal{S}_i and \mathcal{S}'_i , $i \geq 1$) one would like to consider them as different instances of a single one. Second, it appears that significant information moves seem not to essentially depend on the states themselves, but on state sets and their geometrical displays. This leads to abstract signals (or instances of signals) as segments of straight lines as shown in [Fig. 2c, d](#). One often says that [Fig. 2c](#) or [Fig. 2d](#) are *geometric diagrams* associated to the orbit in [Fig. 2b](#). Historically, when one wants to build a cellular automaton realizing a given task, one tries to draw a geometrical representation of an algorithm instance to get a geometrical diagram; when it is possible, one assigns sets of states to each isolated segment in identifying instances of a same signal and managing junctions (simple meetings or creations) in order to obtain the corresponding orbit.

An important question arises: starting from a geometrical diagram, is it possible to define a cellular automaton whose orbit would have the former as a geometrical diagram? The question is not simple, and it is necessary to clearly define the notions. An answer can be found in a specific framework in Richard (2008). Nevertheless, one is able to assert the following.

Proposition 1 *Starting from a finite family of geometric diagrams such that*

- *The number of different lines is finite.*
- *The number of different junctions (inputs and outputs) is finite.*
- *All the meetings and all the emissions are of finite size.*

Then, there is some cellular automaton with a family of orbits corresponding to these geometric diagrams.

In what follows, geometrical representations will be extensively used. When necessary, we will verify that the conditions of [Proposition 1](#) are satisfied. If they are not, proper justifications will be provided.

2.3 Translation of a Finite Right Signal: A First Example of Construction by Cellular Automaton

A first example of a construction of a signal is developed with the purpose of introducing the concept of (uniform) transformations of signals by a cellular automaton.

Let \mathcal{S} be a finite right signal, with origin (x_0, t_0) , and end (x_f, t_f) , generated by some cellular automaton \mathcal{B} , starting from an initial configuration $c_{\mathcal{B}}$ (see [Definition 2](#)). One wants to design a cellular automaton \mathcal{A} , with $T \subset Q_{\mathcal{A}}$, such that:

- The states of T make up a finite right signal \mathcal{T} in the \mathcal{A} -orbit of $c_{\mathcal{B}}$, beginning at $(x_f, t_f + 1)$, ending at $(2x_f - x_0, 2t_f - t_0 + 1)$, and such that $(\xi + x_f - x_0, \eta + t_f - t_0 + 1) \in \mathcal{T}$ for each $(\xi, \eta) \in \mathcal{S}$. In other words \mathcal{T} is obtained from \mathcal{S} by the translation $(x_f - x_0, t_f - t_0 + 1)$.
- $Q_{\mathcal{A}} = Q_{\mathcal{B}} \cup Q_{\mathcal{A}^*}$ with $Q_{\mathcal{B}} \cap Q_{\mathcal{A}^*} = S$ and $Q_{\mathcal{A}}$ contains a quiescent state $q_{\mathcal{A}}$. Moreover, if $\delta_{\mathcal{A}}|Q_{\mathcal{B}} = \delta_{\mathcal{B}}$, $\delta_{\mathcal{A}}$ is stable on $Q_{\mathcal{A}^*}$, the actions of all the states of $Q_{\mathcal{B}} \setminus S$ on $Q_{\mathcal{A}^*}$ are identical and the states in $Q_{\mathcal{A}^*}$ take into account only the states of S in $Q_{\mathcal{B}}$. So the only effective knowledge that $\mathcal{A}|_{Q_{\mathcal{A}^*}}$ has from \mathcal{B} is S .

One may observe that the previous conditions are satisfied if \mathcal{A} maintains in $Q_{\mathcal{A}^*}$ a “joke copy” of $Q_{\mathcal{B}}$.

The result is obtained by building a family of signals S_i . Because \mathcal{S} is a right signal, it is a sequence of stationary and rightward moves. Let St_i and Sd_i , for $i \geq 0$, be the successions of sites along vertical and diagonal segments, respectively.

- S_0 : This signal (see [Fig. 3a](#)) counts the sites of St_0 . Let st_0 be the result. If $st_0 \neq 0$, S_0 brings it along \mathcal{S} up to (x_f, t_f) , where it emits a signal which marks the sites $(x_f, t_f + 1), \dots, (x_f, t_f + st_0 + 1)$ (giving Tst_0) and then goes to the right at speed 1 along a diagonal DT_0 . If $st_0 = 0$, S_0 brings information up to (x_f, t_f) and generates DT_0 starting from $(x_f, t_f + 1)$.
- S_1 : First, this signal (see [Fig. 3b](#)) counts the moves to the right of Sd_0 . The result is assumed to be sd_0 . Then, from the first site (ξ_1, η_1) of St_1 it goes to the right at speed 1 for sd_0 time steps before becoming stationary. It remains stationary for time st_1 , which will be determined by means of signals at speed 1 starting from the sites of St_1 . Signal S_1 will memorize st_1 until it meets the speed 1 signal D_f , emitted from (x_f, t_f) . There, it will emit a stationary signal which will install the part Tst_1 of \mathcal{T} .

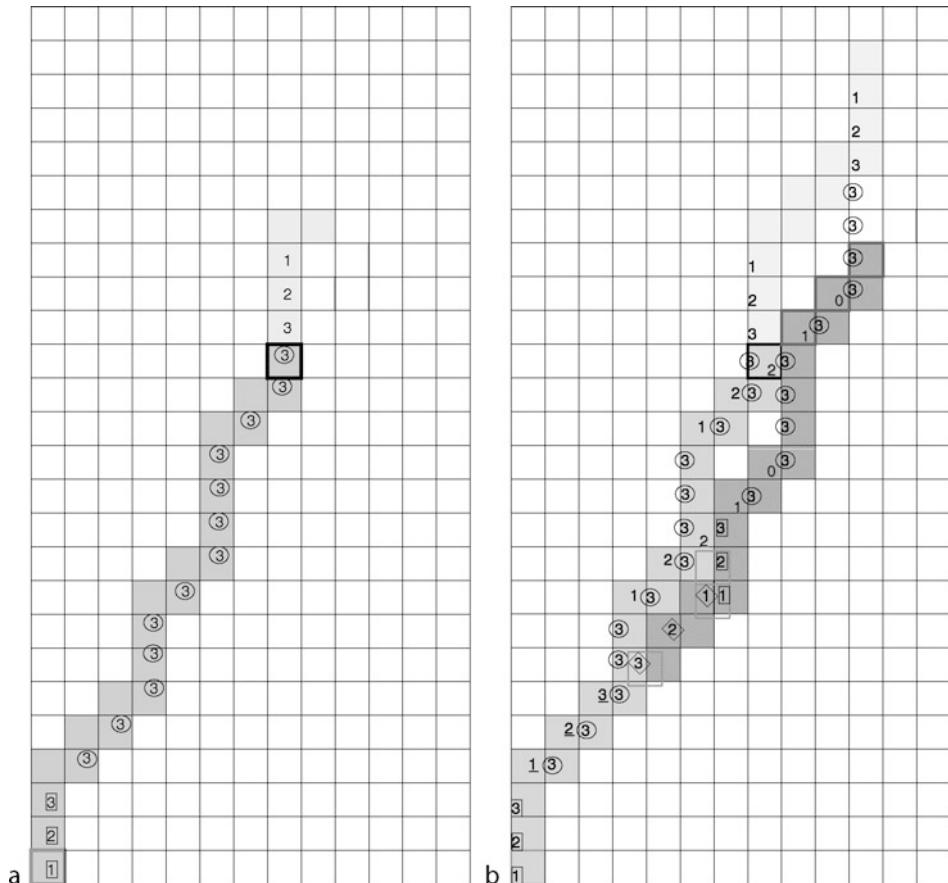
Moreover, a signal, at speed 1, will count the number sd_2 of sites of Sd_2 , and bring it to S_1 . This information will arrive at the end of the former stationary phase, and makes S_1 go to the right for sd_2 moves, and so on.

- S_2 : This will build the part Tst_2 of \mathcal{T} . It will start from the site of S_1 where S_1 becomes stationary for the second time, and will receive from S_1 the number of previous part of right moves of S_1 .
- A finite family of signals S_i is so obtained.
- When D_f does not get any more information from signals S_i , it becomes stationary and the process stops when the last DT diagonal meets it.

\mathcal{S} being finite, the number of described signals is finite, but several of them are superimposed over one another. Actually, if \mathcal{S} has at most h successive moves to the right and at most k successive stationary moves, then the states, outside S of a cellular automaton \mathcal{A} , which realizes the above algorithm, belong to $Q_{\mathcal{A}^*} = Q_{\mathcal{B}} \times (Q_h \times Q_h \times Q_k \times Q_s)^k$, where s represents the number of superimpositions. The space effectively used by all signals essentially lies between \mathcal{S} , D_f , and a signal \mathcal{P} which appears below all of them, and has vertical segments of the same length as vertical segments of \mathcal{S} and diagonal segments of length $2sd_i$, $i \in \{1, \dots, k\}$.

Fig. 3

(a) Translation of $S : S_0$. (b) Translation of $S : S_0$ and S_1 .



2.4 A Notion of Transformation by Cellular Automaton

In the construction of [Sect. 2.3](#), the automaton \mathcal{A} actually acts on the orbit $c_{\mathcal{B}}$ of \mathcal{B} , identifying all the states of \mathcal{B} which are not in S , otherwise being uniform in \mathcal{B} . This leads one to generalize the situation and to consider a cellular automaton that *works* on a colored half plane satisfying some property (presenting a finite right signal, e.g., as in [Sect. 2.3](#)) and transforms it into another colored half plane satisfying another property (e.g., containing a finite right signal the upper part being a translation of the lower one as in [Sect. 2.3](#)). The following definitions are proposed and will be used in the next section.

Definition 4 Let $C \cup \{\lambda\}$ be some finite set.

1. A cellular automaton \mathcal{A} is said to *work on* C if :
 - $Q_{\mathcal{A}} = \Theta_{\mathcal{A}} \times (C \cup \{\lambda\})$,

- $\delta_{Q_A} : Q_A^3 \rightarrow \Theta_A$,
 - $\{L\} \subseteq \Theta_A$ where L is quiescent
2. To each application μ from $\mathbb{Z} \times \mathbb{N}$ into $C \cup \{\lambda\}$ and to each configuration $c_A \in \Theta_A^\mathbb{Z}$ one can associate a sequence $(c_A^t)_{t \geq 0}$ of \mathcal{A} -configurations defined by $c_A^0 = c_A$

$$c_A^{t+1}(x) = \delta_{Q_A}((c_A^t(x-1), \mu(x-1, t)), (c_A^t(x), \mu(x, t)), (c_A^t(x+1), \mu(x+1, t)))$$

which is called the *orbit of c_A on μ* .

If $c_A^0 = {}^\infty L^\infty$, the *orbit of \mathcal{A} on μ* will be considered.

Moreover, one can associate to each orbit of a cellular automaton some coloring of the half plane.

Definition 5 Let \mathcal{A} be some cellular automaton such that $Q_A = Q_A^\sharp \times Q_A^\flat$ (with $Q_A^\sharp \cap Q_A^\flat = \emptyset$) and let c_A be some \mathcal{A} -configuration. To each pair of sets $C^\sharp \subseteq Q_A^\sharp$ and $C^\flat \subseteq Q_A^\flat$, one associates an application $\mu_{c_A, C^\sharp, C^\flat}$ from $\mathbb{Z} \times \mathbb{N}$ into $C^\sharp \cup C^\flat \cup \{\lambda\}$ as follows:

- If $\langle x, t \rangle \in Q_A^\sharp \times C^\flat$, then $\mu_{c_A, C^\sharp, C^\flat}(\langle x, t \rangle)$ is the component of $\langle x, t \rangle$ in C^\flat .
- If $\langle x, t \rangle \in C^\sharp \times Q_A^\flat$, then $\mu_{c_A, C^\sharp, C^\flat}(\langle x, t \rangle)$ is the component of $\langle x, t \rangle$ in C^\sharp .
- $\mu_{c_A, C^\sharp, C^\flat}(\langle x, t \rangle)$ is λ if both or neither of the two conditions above hold.

Then, one can see a cellular automaton that works on μ as an automaton which takes, as input, some picture $\mu : \mathbb{Z} \times \mathbb{N} \rightarrow C \cup \{\lambda\}$ defined by a finite part of sites in $\mathbb{Z} \times \mathbb{N}$ colored by C . As a result, one wants some finite picture colored by a new set C^* , and one wants the process to be uniform. This leads first to a notion of transformation ([Definition 6](#)) and the definition of a *uniformly working* cellular automaton ([Definition 7](#)).

Definition 6 Let C and $C \cup C^*$ be two finite sets, let M be the set of functions from $\mathbb{Z} \times \mathbb{N}$ into $C \cup \{\lambda\}$, and let M^* be the set of functions from $\mathbb{Z} \times \mathbb{N}$ into $C \cup C^* \cup \{\lambda\}$. A transformation \mathbb{T}_{C, C^*} means a partial function from M into M^* .

Definition 7 Let C and $C \cup C^*$ be two finite sets, $(C \cap C^* = \emptyset)$. Let \mathbb{T}_{C, C^*} be some transformation. A cellular automaton \mathcal{A} which works on C uniformly realizes \mathbb{T}_{C, C^*} if:

1. Θ_A contains C^*
2. For each μ belonging to the \mathbb{T}_{C, C^*} -domain, $\mathbb{T}_{C, C^*}(\mu) = \mu_{{}^\infty L^\infty, C, C^*}$.

Returning to the translation of finite right signals in this formal framework allows one to prove a first negative result.

3 Cellular Translations of Finite Right Signals

3.1 Setting Down the Problem

One considers two transformations:

$\mathbb{T}_{S, S \cup S^*}$ Its domain is made up of finite colorings μ of $\mathbb{Z} \times \mathbb{N}$ presenting finite right signals S , while their images $\mathbb{T}_{S, S \cup S^*}(\mu)$ present finite right signals S^* obtained by the translation $(x_f - x_0, t_f - t_0)$ applied to S as studied in [Sect. 2.3](#), without any other condition on S .

Then $\mathbb{T}_{S,S \cup S^*}$ transforms some finite right signal S , starting at (x_0, t_0) , ending at (x_f, t_f) , into the finite right signal starting at (x_0, t_0) , ending at $(2x_f - x_0, 2t_f - t_0 + 1)$ starting with S extended, via the above translation, by an instance of itself.

\mathcal{U}_{S,S^*} Its domain is made up of finite colorings μ of $\mathbb{Z} \times \mathbb{N}$ presenting finite right signals S (with origin (x_0, t_0) , end (x_f, t_f)). Their images $\mathbb{U}_{S,S^*}(\mu)$ present finite right signals S^* with origin (x_0^*, t_0^*) , end $(x_0^* + 2(x_f - x_0), t_0^* + 2(t_f - t_0) + 1)$ with the following properties:

- $\tau_1(S) \subseteq S^*$, where τ_1 denotes the translation defined by $(x_0^* - x_0, t_0^* - t_0)$.
- $S^* = \tau_1(S) \cup \tau_2(\tau_1(S))$, where τ_2 denotes the translation given by $(x_f - x_0, t_f - t_0 + 1)$.

Then, \mathbb{U}_{S,S^*} transforms some signal S into a finite right signal of the same shape as the signal S^* produced by $\mathbb{T}_{S,S \cup S^*}$ in the previous point, but with a starting site (x_0^*, t_0^*) elsewhere in $\mathbb{Z} \times \mathbb{N}$.

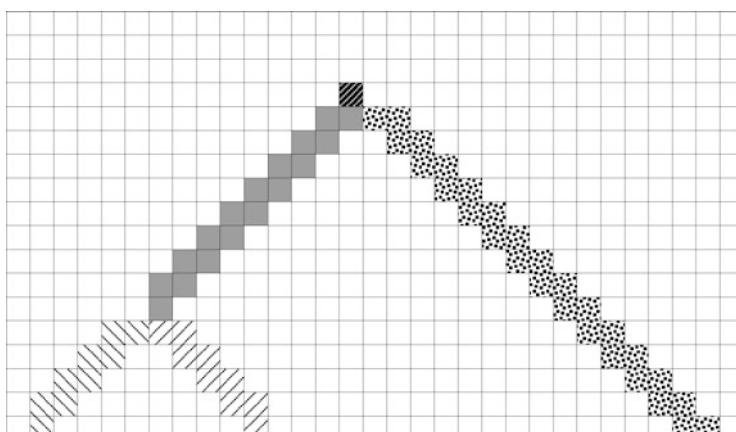
Proposition 2 *No cellular automaton can uniformly realize the transformation $\mathbb{T}_{S,S \cup S^*}$.*

Proof \blacktriangleright [Figure 4](#) can be seen as the significant part of an orbit μ of a cellular automaton \mathcal{B} , with six states including a quiescent one (the “white” one), all the transitions of which are represented on the diagram. The initial configuration is given on the lowest line. The orbit μ shows a finite right signal S in gray and black (made up of right moves followed by only one stationary move, beginning at (x_0, t_0) , ending at (x_f, t_f)), the length, ℓ , of which depends on the initial configuration.

Suppose there is some automaton \mathcal{A} that uniformly realizes the wanted transformation. As S is essentially diagonal, information that can start from its sites only goes above or on the diagonal D_S which carries it, and there is no way to get back information necessary for

Fig. 4

Example of a signal S (gray and black) the length of which depends on the initial line : it springs from the meeting of the two striped signals and finishes a step after the meeting with the signal in stipple, arriving at speed 1 from the right.



delimiting the translated signal \mathcal{S}^* on $D_{\mathcal{S}^*}$. The states of the sites $(x_f + \zeta, t_f - 1 + \zeta)$ ($\zeta \in \mathbb{N}^*$) define an ultimately periodic sequence of period at most $|\Theta_{\mathcal{A}}|$, after at most $|\Theta_{\mathcal{A}}|$ steps. As a consequence (see [Sect. 5.2](#)), the states of the sites $(x_f + \zeta, t_f + \zeta)$ are ultimately periodic of period at most $|\Theta_{\mathcal{A}}|^2$ and the sites $(x_f + \zeta, t_f + \zeta + 1)$ are ultimately periodic of period at most $|\Theta_{\mathcal{A}}|^3$. As a consequence, the signal image of \mathcal{S} in $\mathbb{T}_{\mathcal{S}, \mathcal{S}^*}$ has to be infinite or of length bounded by $2|\Theta_{\mathcal{A}}|^3$, a contradiction.

Then there is no cellular automaton which realizes $\mathbb{T}_{\mathcal{S}, \mathcal{S}^*}$. This shows that, in the construction of [Sect. 2.3](#), the automaton \mathcal{A} needs the knowledge of automaton \mathcal{B} .

3.2 Uniform Translation of Finite Right Signals

Proposition 3 *There is a cellular automaton which uniformly realizes the transformation $\mathbb{U}_{\mathcal{S}, \mathcal{S}^*}$.*

Proof The proof rests on a method of signal analysis and the possibility of rebuilding the signal starting with the analysis of its stationary and rightward moves. It is split into several parts dedicated to different algorithms.

- ▶ Point a : Analysis of a finite right signal.

The algorithm is illustrated with [Fig. 5](#). It consists in describing the succession of stationary and rightward moves of the original signal \mathcal{S} on a stationary signal that starts at the end of \mathcal{S} .

A family $(\mathcal{S}_i)_i$ of signals, parallel to \mathcal{S} (which is \mathcal{S}_0), is built. Signal \mathcal{S}_i contributes to the construction of \mathcal{S}_{i+1} only starting from its third move when it sends out to the left, at speed 1, a signal ς_3 that carries the nature (stationary or rightward) of this third move. Then, it indicates its moves to the right by means of a signal ς at speed 1 to the left. The signals \mathcal{S}_i end at $(x_{f_i}, t_{f_i}) = (x_f - i, t_f + i)$ on a signal \mathcal{E}_{\nwarrow} sent, at speed 1 to the left, from the end site (x_f, t_f) of \mathcal{S} .

The states of a signal \mathcal{S}_i are to be taken in a set $T \times \{0, 1\} \times \{a, b, \emptyset\}$. The value of the second component says whether the first move of \mathcal{S}_i is stationary (0) or rightward (1), while the value of the third component indicates whether the second move is rightward (a), stationary (b) or is unknown (\emptyset). These pieces of data are collected on \mathcal{E}_{\nwarrow} , and sent to the right and collected again on a stationary signal \mathcal{E}_{\uparrow} created at the end of \mathcal{S} . Let it be stressed that each signal \mathcal{S}_i sends pieces of information to be marked on signal \mathcal{E}_{\uparrow} .

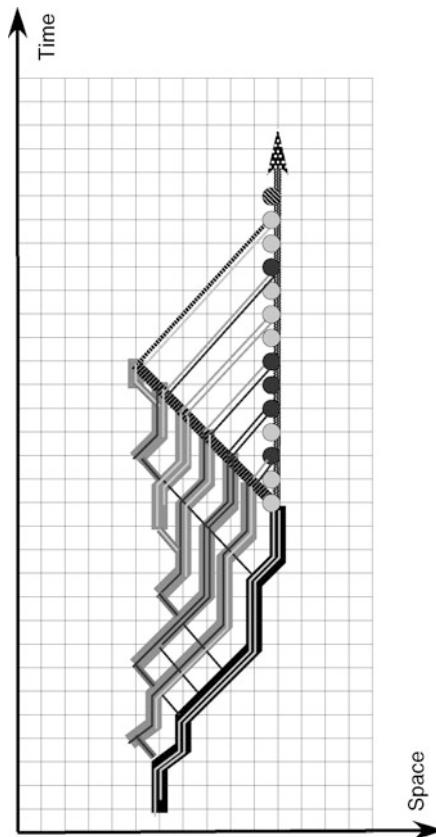
- ▶ Point b : Analysis of the stationary moves (rightward moves) of a finite right signal.

What one has to do now is to extract the stationary moves of \mathcal{S} , and to keep their number and the information of whether they follow a stationary or a rightward move, as shown in [Fig. 6a](#). The idea is the same as the one in point a.

Signal \mathcal{S}_{i+1} starts at the end of the second (from origin) stationary move on signal \mathcal{S}_i , $i \geq 0$, and $\mathcal{S}_0 = \mathcal{S}$. It carries pieces of information relative to these first stationary moves : a if the successor move on \mathcal{S}_i is stationary, b if it is rightward, \emptyset if there is no successor move. All this data is collected on the signal \mathcal{V} which goes to the left, at speed 1, from the end of \mathcal{S} . Then, they are redirected to the line $x = x_f$ at speed 1 and symbolized in [Fig. 6a](#) by circles. A dark gray circle means that the indicated move (which was stationary) is followed by a rightward

Fig. 5

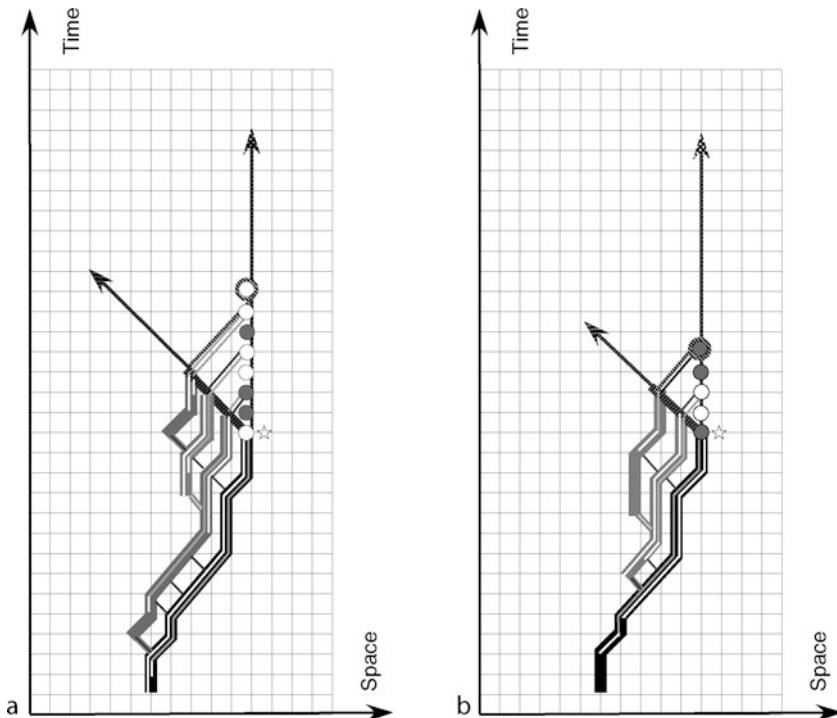
Extraction of the moves of a finite right signal ($0a1a1b1a0a1a0\emptyset$). The original signal \mathcal{S} is represented by an essentially thick black line. For $i \geq 1$, \mathcal{S}_i is represented by an essentially medium-gray line. The signals ς_3 are represented by thick medium-gray lines, with a fine darker gray line inside if the move is to the right, or a fine lighter gray line inside if the move is stationary. The signals ς are represented by fine dark gray lines. Inside the \mathcal{S}_i -lines appear two lines: the first one, starting from the starting point of \mathcal{S}_i , indicates the first move of \mathcal{S}_i stationary (0, light gray) or to the right (1, dark gray); the second one, starting one time unit later, indicates its second move stationary (a, light gray) or to the right (b, right gray), information which comes from \mathcal{S}_{i-1} ($i \geq 1$). Component \emptyset is indicated by the lack of a line. The signal \mathcal{E}_\nwarrow and \mathcal{E}_\nearrow are easily recognizable. Data sent from \mathcal{E}_\nwarrow to \mathcal{E}_\nearrow are in light gray for stationary moves and in dark gray for moves to the right. They are symbolized on \mathcal{E}_\nearrow by light or dark circles. The end of signal \mathcal{S} , denoted by \emptyset , is indicated by a dark striped line and symbolized on \mathcal{E}_\nearrow by a circle with stripes inside.



one (if any), while a white circle means that the indicated move (stationary) is followed by a stationary one (if any). As for the last circle with stripes, it also marks the end of \mathcal{V} . On \mathcal{V} , the information on successors of stationary moves is obtained, but not on their number, which appears on the vertical segment stemming from (x_f, t_f) .

Fig. 6

(a) Analysis of the stationary moves of a finite right signal (*abbaabaa*). The star marks the end of the initial signal S , the same as in [Fig. 5](#). (b) Analysis of the rightward moves of a finite right signal (*abbaa*), the same as in [Fig. 5](#).



The moves to the right of S are similarly analyzed ([Fig. 6b](#)). Finally, both algorithms acting in parallel bring the number of stationary and rightward moves onto the line $y - t_f = -(x - x_f)$ starting at (x_f, t_f) .

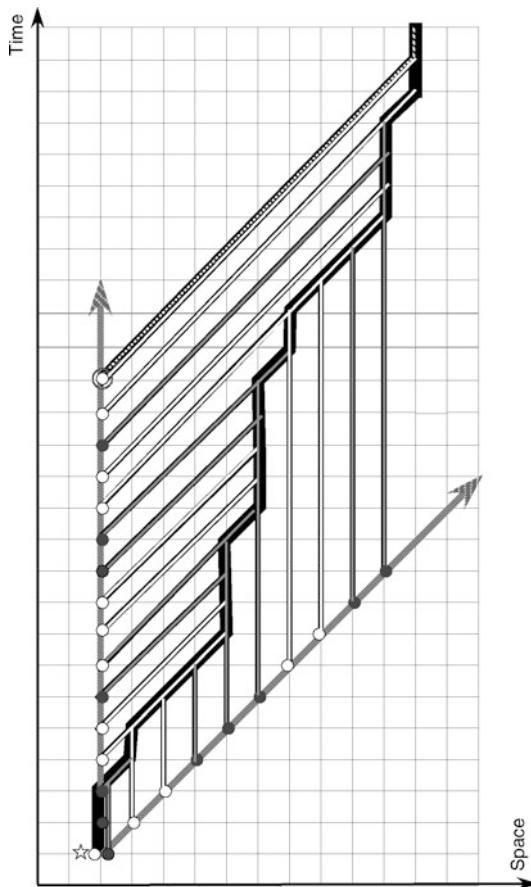
► Point c : Reconstruction of a finite right signal.

Suppose that one is able to gather, at some site (x_b, t_l) data obtained in point b in the following way : data about stationary moves on the line D_\uparrow ($x = x_l$), data about rightward moves on the line D_- ($y - t_l = x - x_l$), both lines starting from (x_b, t_l) . Then, it is possible to rebuild the analyzed signal starting from (x_b, t_l) , as shown in [Fig. 7](#). From each site on D_\uparrow a signal parallel to D_- is sent, which says whether the data indicates a change in the direction of the signal (*b*), a stationary move (*a*) or end of data. From each site on D_- is sent a signal parallel to D_\uparrow which says whether the data indicates a change of direction of the signal (case *a*) or not (case *b*) or end of the data. Now, it is easy to rebuild a signal S' which is a translated copy of the initial segment S .

- From point b, it is known whether the first sequence of moves is stationary or not.
- A sequence of stationary moves goes on as long as it does not get a signal that says that it has to change to a right move, which is the beginning of a sequence of right moves.

Fig. 7

Reconstruction of a finite right signal. Analysis of the stationary (rightward) moves of the studied signal are coded on D_{\uparrow} (D_{\rightarrow} , respectively). On D_{\uparrow} , a gray circle means “I was stationary and I become rightward,” a white circle means “I was stationary and I remains stationary.” On D_{\rightarrow} , a gray circle means “I was rightward and I become stationary,” a white circle means “I was rightward and I remain rightward.” The star marks the origin of the new signal. In fact, in the figure, the construction of the wanted signal S' is anticipated, in starting with data $abbaabaaabbaabaa$ represented on D_{\uparrow} and $abbaaabbaa$ represented on D_{\rightarrow} , which correspond to the duplicated results of the analyses represented in [Fig. 6a, b](#).



- A sequence of moves to the right goes on as long as it does not get a signal that says it has to change direction. Then, a stationary move is executed, which is the beginning of a sequence of stationary moves.
- The construction is achieved when both signals of end arrive.

It is clear that the new signal S' has the same sequences of moves as the original signal S . But it is not yet the wanted signal. In order to get the results, one has to duplicate the data

concerning stationary and rightward moves obtained in point b, as well as to find a suitable point (x_f, t_f) .

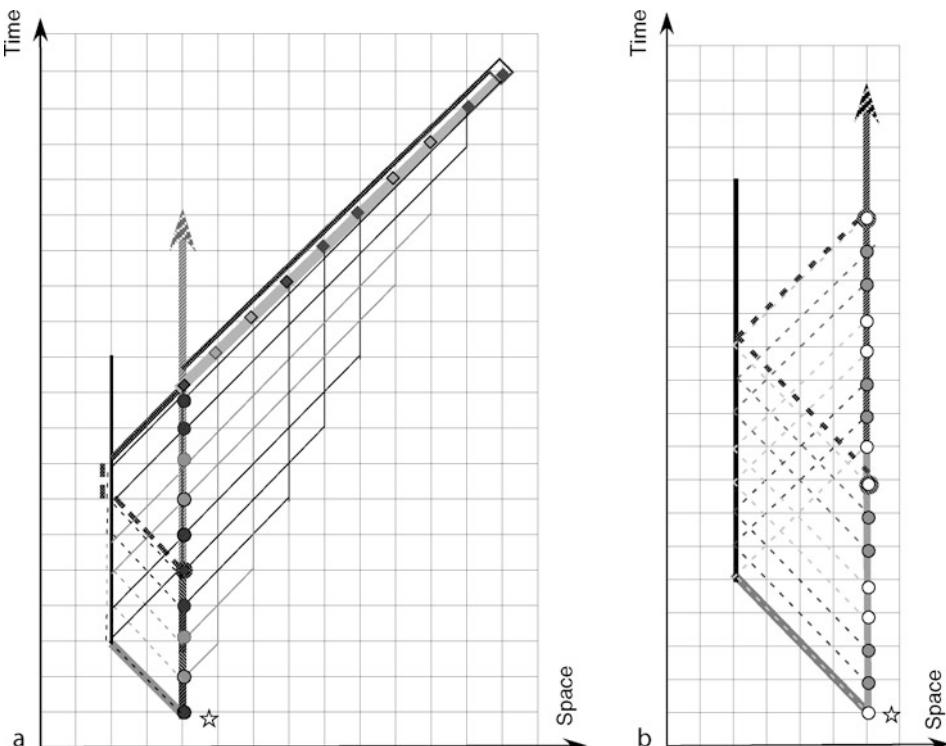
► Point d : The end of the proof, duplication, and transfer of sets of data.

The number of stationary (rightward) moves is denoted n_\uparrow (n_\rightarrow , respectively), so the length of signals \mathcal{V} in b) are $\lfloor \frac{n_\uparrow}{2} \rfloor + 1$ and $\lfloor \frac{n_\rightarrow}{2} \rfloor + 1$. Then several cases would have to be considered according to the parity of these numbers. As all the data resulting from the analyses of moves are on D_\uparrow , one wants to find (x_f, t_f) on this very line. Two cases have to be distinguished depending on whether $2n_\rightarrow \geq n_\uparrow$ or not. In the first case, the algorithm is the following one:

1. Data concerning the moves to the right, obtained, at the end of point b, on D_\uparrow starting at site (x_f, t_f) are sent to the left at speed 1. They rebound on the vertical line drawn from the end of the signal \mathcal{V} corresponding to rightward moves (actually, two cases are to be considered according to the parity of $\lfloor \frac{n_\uparrow}{2} \rfloor$, see [Sect. 6.1](#)) corresponding to the rightward moves and come back to the right, at speed 1, up to D_\uparrow , the vertical line starting from site (x_f, t_f) .

□ Fig. 8

(a) Duplication and setting up of the rightward moves. The sequence $abbaa$ has become $abbaaabbaa$. The star represents the suitable origin of the duplication. (b) Setting up of the stationary moves. The sequence $abbaabaaa$ has become $abbaabaabbaabaa$.



2. Then all the data corresponding to the rightward moves (on sites (x_f, t_f) up to $(x_f, t_f + n_{\rightarrow})$) are sent on the diagonal Δ_{\rightarrow} , parallel to D_{\rightarrow} , starting from site $(x_f, t_f + n_{\rightarrow})$ (see [Fig. 8a](#) and [Sect. 6.1](#)).
3. Data corresponding to stationary moves are collected on D_{\uparrow} starting from (x_f, t_f) . They are sent at speed 1 to the left, stay on the vertical line V_{\uparrow} created at the end of the signal \mathcal{V} corresponding to the stationary moves for a time d and come back at speed 1 on D_{\uparrow} where they rebound at speed 1 up to V_{\uparrow} , where they again rebound at speed 1 up to D_{\uparrow} and stop. The delay d is determined, by $2n_{\rightarrow} - n_{\uparrow}$ and its parity, in such a way that the duplication of the data starts at site $(x_f, t_f + n_{\rightarrow})$. See [Fig. 8b](#). Here two cases are to be considered, according to the parity of $\lfloor \frac{n_{\rightarrow}}{2} \rfloor$ ([Fig. 8b](#) shows the even case).
4. Finally, it is possible to build the wanted signal \mathcal{S}^* with origin $(x_f, t_f + n_{\rightarrow})$, as in point c.

The algorithm for the second case is analogous. Actually, the data concerning rightward signal are treated the same way except that they are dispatched on the right diagonal stemming from site $(x_f, t_f + n_{\uparrow})$. Data on stationary moves are duplicated from this very site by the usual means of rebounding between V_{\uparrow} and D_{\uparrow} .

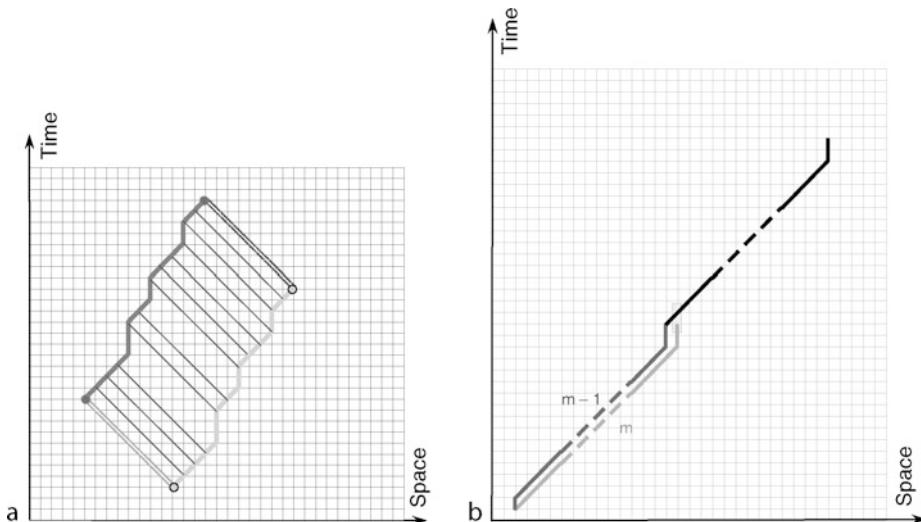
In the proof above, the origin of \mathcal{S}^* is not the best possible one because the data could be sent on \mathcal{V} not 2 by 2 but 2^k by 2^k for any $k \in \mathbb{N}$. Then the origin of \mathcal{S}^* would be $(x_f, t_f + \frac{n_{\rightarrow}}{k})$ or $(x_f, t_f + \frac{n_{\uparrow}}{k})$. But the origin can never be (x_f, t_f) as proved in [Proposition 2](#). One observes that computations are done in a part of the space-time diagram which is above the given signal, and never below. This is necessary according to [Proposition 2](#).

Of course the same process can be applied to left signals (diagonals become D_{\leftarrow} parallel to the straight line $y = -x$).

For any signal, the process can be easily adapted: it suffices to cut the signal into left and right signals and to shift the signals D_{\uparrow} parallel to \mathcal{S} .

Fig. 9

(a) Translation of a finite right signal according to a left diagonal. (b) The impossibility of translating a finite right signal according to some signal.



4 Various Constructions of Signals Starting from Signals

4.1 Translations and Other Modifications

Figure 9a shows how to shift a finite right signal according to a diagonal to the left. Let \mathcal{S} be a finite right signal (light gray in Fig. 9a) and let P be an arriving point on the diagonal $\{x_0 - \ell, t_0 + \ell | \ell \in \mathbb{N}\}$ stemming from the origin (x_0, t_0) of \mathcal{S} (light stripes in Fig. 9a). Each time \mathcal{S} “advances to the right”, a signal \mathcal{D} is emitted at speed 1 to the left (fine lines in Fig. 9a). From P is emitted a new signal \mathcal{T} (dark gray in Fig. 9a) that stays stationary except when it gets a signal \mathcal{D} , then it goes to the right. A particular signal \mathcal{D}^* (in dark stripes in Fig. 9a) is created on the end site of \mathcal{S} ; it goes at speed 1 to the left and kills \mathcal{T} when they meet.

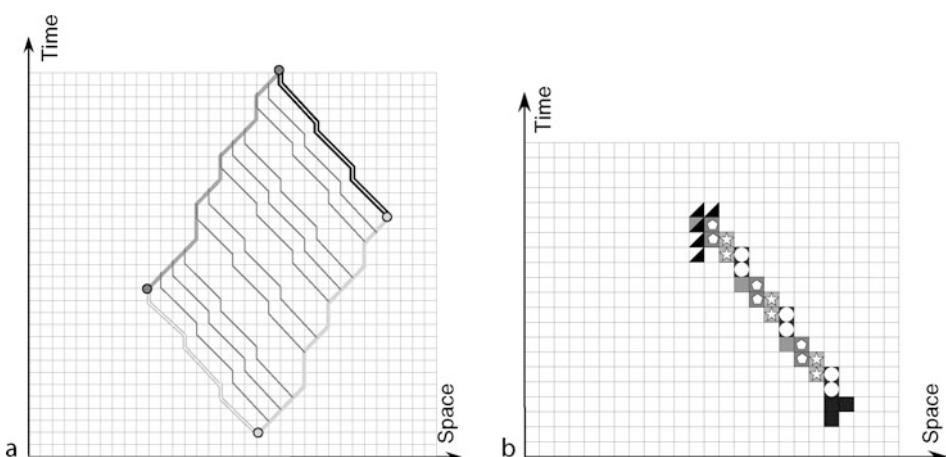
However, it is not trivial to generalize the previous construction. In the framework of Sect. 3.1, consider, for example, the signal \mathcal{S} (in light gray in Fig. 9b) defined by m moves at speed 1 to the right followed by two stationary moves, and the signal \mathcal{T} defined by one stationary move followed by $m-1$ moves at speed 1 to the right, followed by two stationary moves (dark gray in Fig. 9b). If one wants to translate \mathcal{S} according to \mathcal{T} , the three sites $(x_f, t_f + \ell), \ell \in \{0, \dots, 3\}$ (where (x_f, t_f) is the end site of \mathcal{T}) have to know the encoding of m , which is impossible.

But it is always possible to translate any finite right signal according to a *periodic* left signal (for a number of times equal to an integer multiple of the period). Figure 10a shows an example of a corresponding geometric diagram. A left periodic signal \mathcal{T} is defined by its period π , which can be seen as a sequence $(\epsilon_0, \dots, \epsilon_{\pi-1})$ of elements of $\{-1, 0, 1\}$ (0 coding a stationary move, -1 a left one and 1 a right one). In this case, the signals are no more coded by one state, but by π states as it is shown in Fig. 10b.

Fig. 10

(a) Translation of a finite right signal according to a periodic left signal, geometric diagram.

(b) Translation of a finite right signal according to a periodic left signal, representation of the left signal by states.

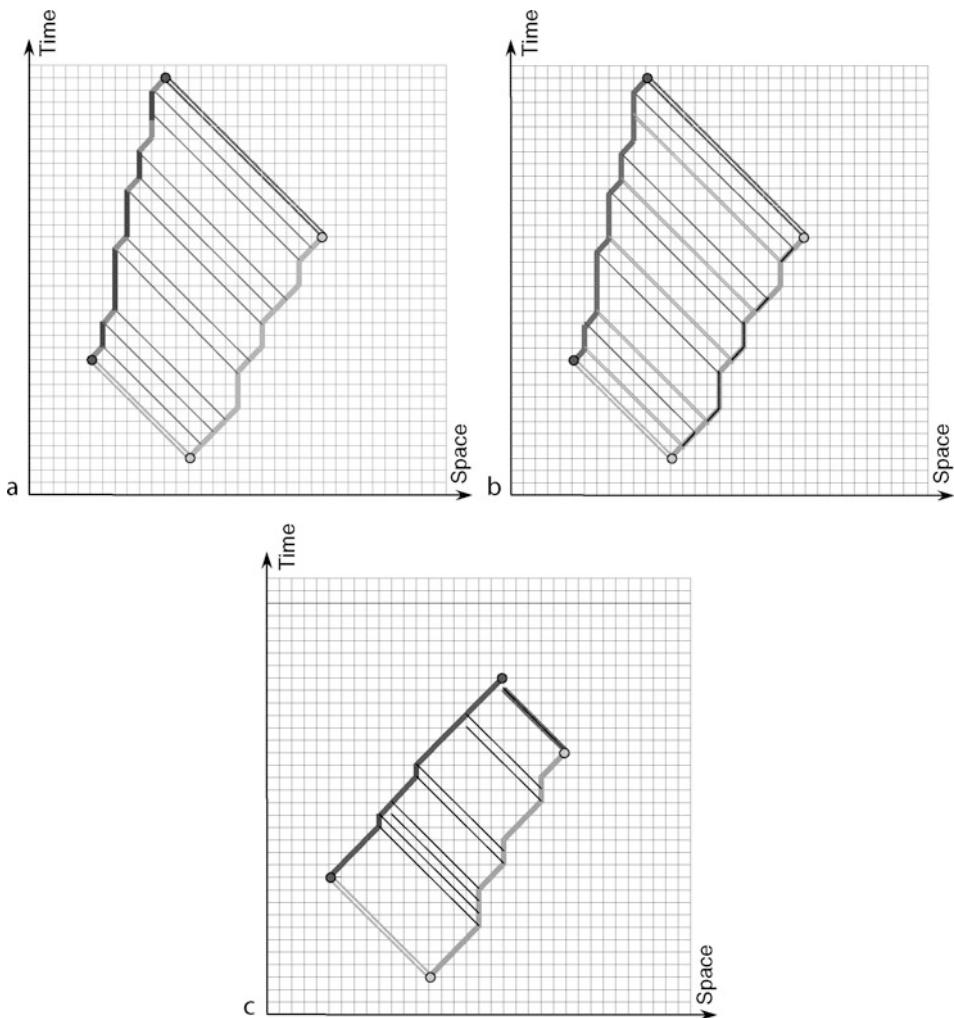


Numerous techniques exist in order to build signals by means of other signals. Some examples have already been seen. The most straightforward method is illustrated in [Fig. 11a](#). Some signal \mathcal{S} (fine lines in [Fig. 11a](#)) sends out, each time it goes to the right, a signal \mathcal{D} which goes to the left at speed 1. When these signals \mathcal{D} meet another signal \mathcal{T} , they modify it.

The set of states on \mathcal{T} is partitioned into two subsets (black and dark gray in [Fig. 11a](#)). It is stationary except when in the dark gray state it receives a signal \mathcal{D} . Then it goes to the right. The color is swapped each time the signal receives \mathcal{D} . In this way, \mathcal{T} goes to the right “more or less” with half speed compared to \mathcal{S} . Instead of partitioning \mathcal{T} , it is also possible to distinguish

Fig. 11

- (a) Modification of a finite right signal using its own moves to the right. Parity is indicated on \mathcal{T} .
- (b) Modification of a right signal by means of its rightward moves, parity marked on \mathcal{S} .
- (c) Modification of a right signal by means of its stationary moves.



two types of states for S (light gray and light gray with a black line in [Fig. 11b](#)) and to consider two types of signals \mathcal{D} , say \mathcal{D}_1 and \mathcal{D}_2 (denoted by fine dark gray lines and light gray doubled lines in [Fig. 11b](#)). Then T only advances when it gets a signal \mathcal{D}_1 .

One can finally think of using the stationary moves of \mathcal{S} . It is possible as shown on [Fig. 11c](#). Only one stationary move of \mathcal{S} out of two induces a stationary move of T . In this case, one has to partition T into six parts because a signal to the right and a signal to the left can meet in two different ways (see [Sect. 2.1](#)). These cases are highlighted in very light gray on [Fig. 11c](#). One then gets a signal T which goes to the right “more or less” twice as fast as \mathcal{S} .

4.2 Infinite Families of Signals

One can iterate the process and build slower and slower signals as described in [Sect. 4.1](#) and in [Fig. 11b](#). (This idea appears for the first time in Waksman (1996)). The only difficulty is to define the origins of the signals T (here denoted \mathcal{S}_k with $\mathcal{S}_1 = \mathcal{S}$). Actually, it is possible to put these origins on any infinite left signal. In order to make the process more regular, one has to set these origins on a stationary signal (in black in [Fig. 12a, b](#)). One gets, for example, the geometric diagram in [Fig. 12a](#), where the initial signal is the diagonal signal at speed 1 to the right. (The partitions of S are omitted in the figure.) [Figure 12b](#) shows the necessary states. The slope of S is 1 and the slope of \mathcal{S}_k is $2^k - 1$, $k \geq 1$. If (x_0, t_0) denotes the origin of S , then the origin of \mathcal{S}_k is $(x_0, t_0 + \sum_{i=1}^{i=k} 2^i)$.

Fig. 12

(a) Infinite family of signals of slope $2^k - 1$: geometric diagram. (b) Infinite family of signals of slope $2^k - 1$: geometric diagram and states.

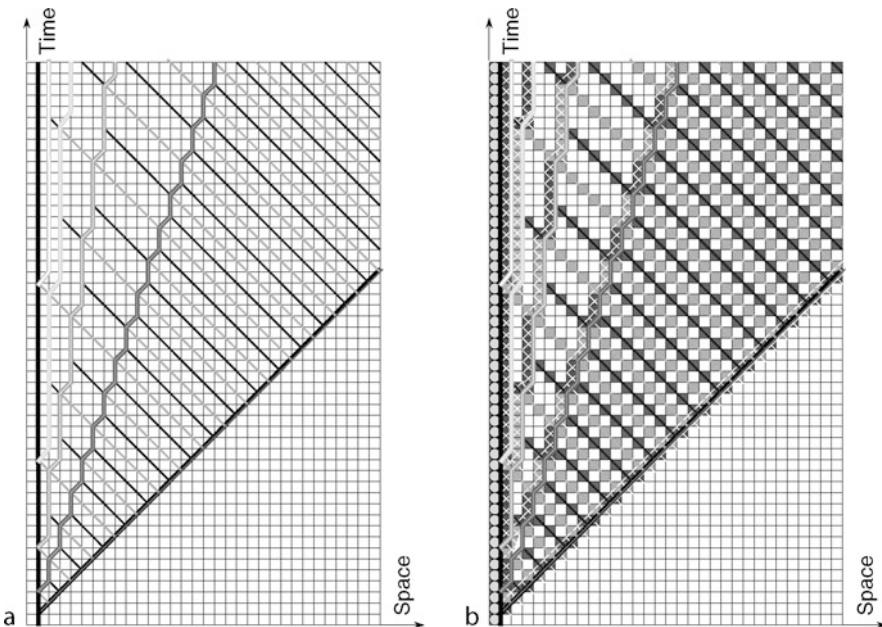
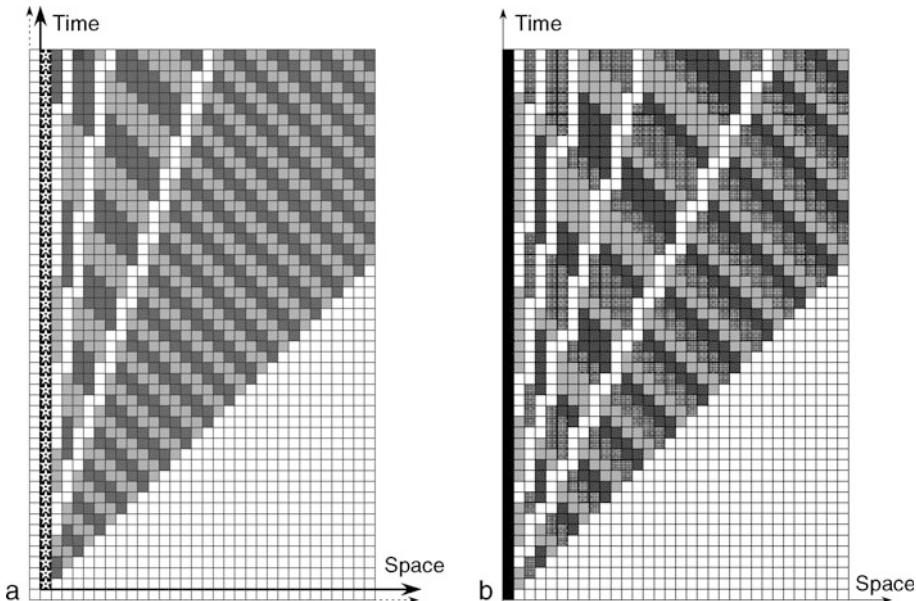


Fig. 13

(a) Minimization of the states number. (b) Another example.



It is possible to minimize the number of necessary states, by means of clever encodings, as shown in [Fig. 13a](#). A detailed study can be found in Delorme and Mazoyer (2002b).

Of course, it is possible to distinguish more than two types of diagonal signals to the left and then obtain other infinite families of signals as in [Fig. 13b](#). In fact, for any integers p and q , with $p > q$, it is possible to build an infinite family of signals of slopes $\left(\frac{p}{q}\right)^k$ (see Mazoyer (1989b)).

What is noticeable is that segments of exponential length can be built by a set of finite automata, with computation divided up among all the cells.

4.3 Parabolas and Exponentials

It appears, for the first time in Fisher (1965), that it is possible to build signals of nonlinear slopes on cellular automata. A discrete parabolic branch is constructed in Fisher (1965), and other examples are also exhibited in the chapter [Computations on Cellular Automata](#) of this volume.

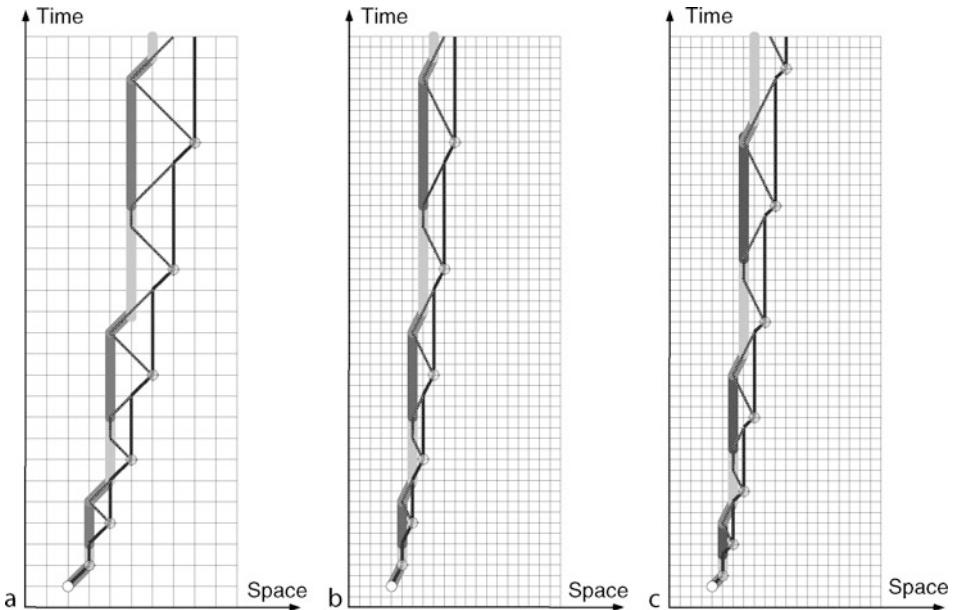
4.3.1 Parabolas

How to build a signal S of slope $\Pi_S(t) = \lfloor x^2 \rfloor$ is shown here. The basic idea is to use the formula

$$1 + 2 + \dots + k = \frac{k(k+1)}{2}$$

Fig. 14

(a) Construction of $y = \frac{x(x+1)}{2}$. The signal \mathcal{S} is in fine black lines. The signal \mathcal{Z} is in striped lines, it zigzags between the signals \mathcal{S} and \mathcal{T} . The signal \mathcal{T} is in thick lines and presents three sorts of states : medium gray corresponding to rightward moves, light and dark gray when it is stationary. After a right move, it is in light gray before receiving \mathcal{Z} and in dark gray afterward. (b) Construction of $y = x(x + 1)$. (c) Construction of $y = x^2$.



In a first step, one builds a signal \mathcal{S} of slope $\Pi_{\mathcal{S}}(x) = \left\lfloor \frac{x(x+1)}{2} \right\rfloor$ (see Fig. 14a). Signal \mathcal{S} (black in Fig. 14a) moves one time to the right, then waits k units of time. That requires an implementation of a counter for k . If k is even, $k = 2\ell$, a signal \mathcal{Z} , zigzagging as fast as possible between two stationary signals \mathcal{T} and \mathcal{S} on cells x and $x + \ell$, counts 2ℓ . If k is odd, $k = 2\ell + 1$, in order to count $2\ell + 1$, it suffices that signal \mathcal{Z} “waits” one time unit on \mathcal{T} . Then, from sites $P_{\mathcal{S}} = (x_0 + 2\ell + 1, t_0 + \frac{(2\ell+1)(2\ell+2)}{2})$ (on \mathcal{S}) a signal \mathcal{Z}_1 is sent along a diagonal to the left (black oblique stripes in Fig. 14a), it sends a signal \mathcal{Z}_3 which stays one unit of time on \mathcal{T} (black horizontal stripes in Fig. 14a) and comes back along a diagonal to the right (black oblique stripes in Fig. 14a) as \mathcal{Z}_2 , and meets \mathcal{S} at time $t_0 + \frac{(2\ell+1)(2\ell+2)}{2} + 2\ell + 1$ as wanted. Moreover, when \mathcal{Z}_2 meets \mathcal{S} , signal \mathcal{S} moves once to the right, then sends a new signal \mathcal{Z}_1 that immediately sends a signal \mathcal{Z}_2 when arriving on \mathcal{T} . The distance between \mathcal{S} and \mathcal{T} is then $2\ell + 2$ and \mathcal{Z}_2 arrives on \mathcal{S} after k times as wanted. In order to know whether \mathcal{Z}_1 waits one time or not, it is sufficient to distinguish two types of states in \mathcal{T} (light or dark gray in Fig. 14a). Then, in order to build \mathcal{S} , it is sufficient to move the counter “wall” \mathcal{T} of one cell each time \mathcal{S} has moved twice.

In a second step, one builds a signal \mathcal{S}^* of slope $\Pi_{\mathcal{S}^*}(x) = x(x + 1)$ (see Fig. 14b). The former signal \mathcal{S} has then to become another signal \mathcal{S}^* . When \mathcal{S} goes to the right, \mathcal{S}^* goes to the right and waits for one unit of time and when \mathcal{S} remains one unit of time on a cell, \mathcal{S}^* remains

two units of time on it. In order to establish this, it is sufficient to replace the speeds 1 and -1 of the counters in the former construction by speeds $\frac{1}{2}$ and $-\frac{1}{2}$ and to ask the signal \mathcal{Z}_1 to remain stationary two units of time instead of one. Then one gets the diagram shown in [Fig. 14b](#).

Finally, in order to get the signal \mathcal{S}^\sharp of slope $\Pi_{\mathcal{S}^\sharp}(x) = x^2$, it suffices to transform the moves at speed $\frac{1}{2}$ of \mathcal{S}^* into moves at speed 1 as shown in [Fig. 14c](#).

4.3.2 Signals of Exponential Slopes

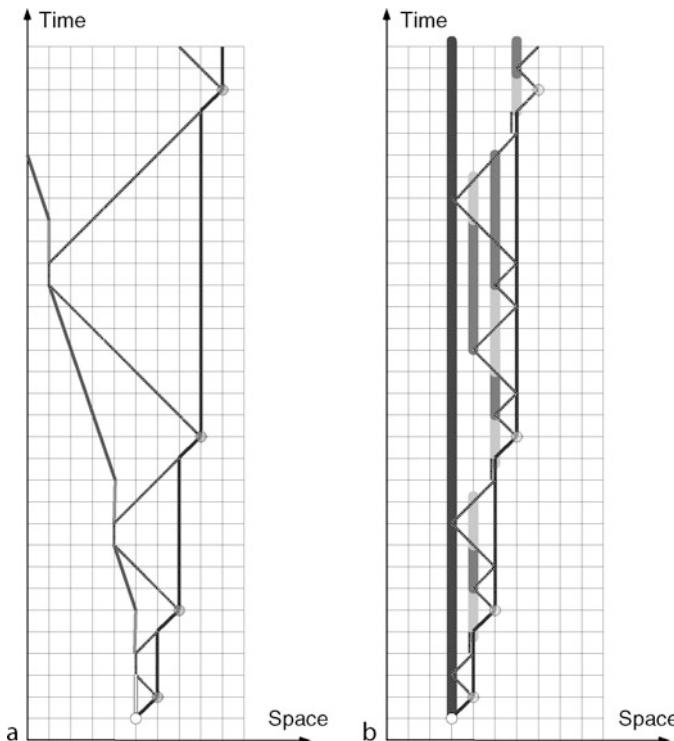
Suppose one wants to build signals of exponential slopes. The idea is given by the formula

$$1 + 2 + 2^2 + \dots + 2^{k-1} = 2^k - 1$$

One first builds a signal \mathcal{S} of slope $\Pi_{\mathcal{S}}(x) = 2^x - 1$ by following step-by-step the process of the previous construction (see [Fig. 15a](#)). Between site $(x_0 + k, t_0 + \sum_{j=0}^{j=k-1} 2^j)$ and site $(x_0 + k + 1, t_0 + \sum_{j=1}^{j=k} 2^j)$, \mathcal{S} has to remain stationary $2^k - 1$ units of time, before advancing by one step. This can be done with three signals \mathcal{Z}_1 , \mathcal{Z}_3 , and \mathcal{Z}_2 as follows. Signal \mathcal{Z}_1 has to go 2^{k-1} times to the left before giving birth to \mathcal{Z}_3 on the site

[Fig. 15](#)

(a) Construction of $y = 2^x - 1$ (first method). (b) Construction of $y = 2^x - 1$ (second method).



$M_k = (x_0 + k - 2^{k-1}, t_0 + \sum_{j=1}^{j=k-1} 2^j + 2^{k-1})$. With $\zeta = 2^{k-1}$ one has $M_k = (x_0 + k - \zeta, t_0 + 3\zeta - 1)$ and also $M_{k+1} = (x_0 + k + 1 - 2\zeta, t_0 + 6\zeta - 1)$. That leads to $M_{k+1} - M_k = (1 - \zeta, 3\zeta)$, that is also $M_{k+1} - M_k = (1 - \zeta, 3 + 3(\zeta - 1))$. Then, with $\eta = \zeta - 1$, $M_{k+1} - M_k = (-\eta, 3 + \eta)$. Between the sites M_k and M_{k+1} , the straight line $\{(-\eta, 3 + \eta) \mid \eta \in \mathbb{N}\}$ is built by a signal which stays three units of time on M_k followed by a signal of slope -3 (dark gray in [Fig. 15a](#)). One observes that the computing area arbitrarily stretches out to the right and to the left of cell x_0 .

There are of course other methods to count 2^k . In particular, [Fig. 15b](#) shows one in which the computing area does not stretch to the left of x_0 and is contained between S and the straight line $x = x_0$. In using the results of [Sect. 5.2](#), the computing area can be moved between S and the diagonal stemming from (x_0, t_0) .

To conclude, it must be noted that similar methods allow one to build signals of slopes k^p and p^k for $p \geq 2$.

5 Functions Constructions

5.1 Fisher's Constructibility

Fisher (1965) showed that increasing functions could be computed on cellular automata by marking times on a cell (prime numbers in Fisher's paper, see the chapter [Computations on Cellular Automata](#)). Later, it was proved that DOL systems could be generated on one-dimensional cellular automata as in Ćulik and Karhumäki (1983) and Ćulik and Dube (1993). This work has been resumed and completed in Terrier (1991). Finally, “marking times” were studied in more detail in Mazoyer and Terrier (1999).

Definition 8 (to mark times) A cellular automaton $\mathcal{A} = (Q, \delta)$ (with a quiescent state L and a distinguished state G) is said to *mark times* according to some increasing function, f , if there is some subset M of Q such that, in the evolution of \mathcal{A} on the basic initial configuration $(^\infty LGL^\infty)$, the state of cell 0 is in M at time t if and only if $t = f(n)$ for some $n \in \mathbb{N}$.

Many increasing functions can be “marked” by such cellular automata. [Figures 16a, b](#), [17a, b](#), and [18a, b](#) show how functions $n \rightarrow 2^n$, $n \rightarrow 3^n$, $n \rightarrow n^2$, and $n \rightarrow n^3$ can be marked. The algorithms are briefly described.

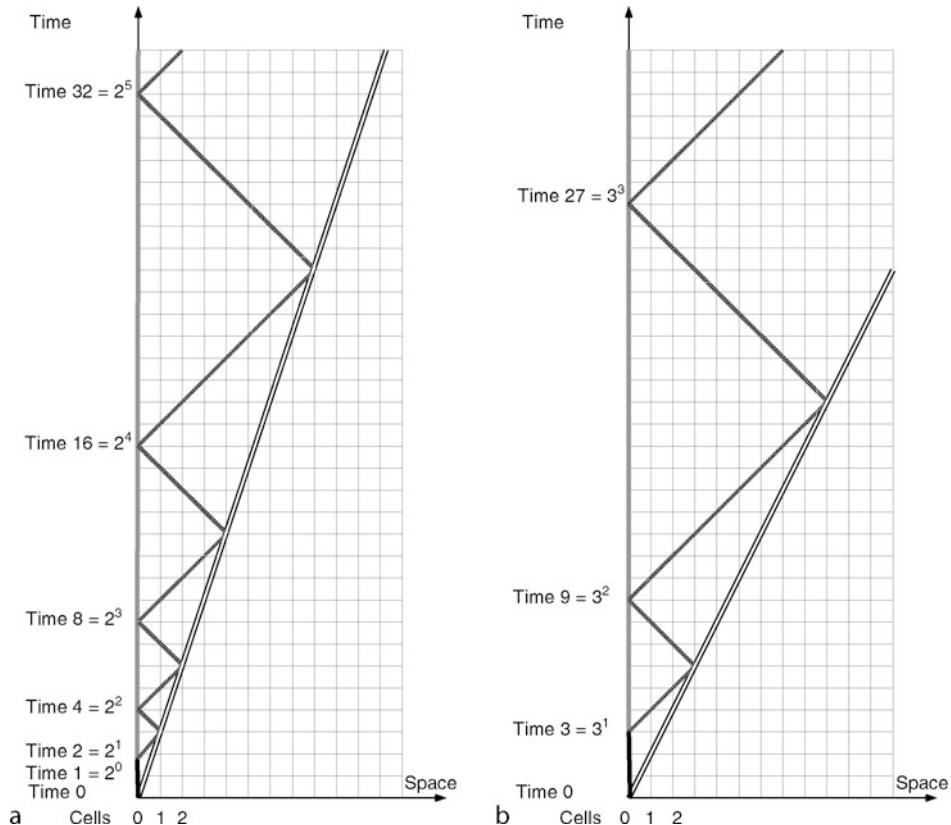
5.1.1 Marking Times k^n

In order to mark the times 2^n , one uses the formula

$$2^n - 1 = 1 + 2 + 2^2 + \dots + 2^{n-1}$$

Two signals (with two types of stripes in [Fig. 16a](#)) oscillate at speed one between a stationary signal that marks cell 0 (dark gray in [Fig. 16a](#)) and a signal of slope 3 (white with black edges in [Fig. 16a](#)) that progresses by i cells in $3i$ time steps.

Obviously, using the formula of the geometric series, all the functions $n \rightarrow k^n$ allow us to mark times k^n , by means of “black and white signals” of slope $\frac{k+1}{k-1}$. [Figure 16b](#) illustrates the case $k = 3$.

Fig. 16(a) Marking times 2^n . (b) Marking times 3^n .

5.1.2 Marking Times n^k

In order to mark sites $(0, n^2)$, one uses the formula

$$(n+1)^2 = n^2 + 2n + 1$$

An idea of the method in the case where one wants to distinguish the sites $P_n = (n, n^2)$ is given and, consequently, manages to mark times $n^2 + n$. One observes that vector $P_n P_{n+1}$ is $(1, 2n + 1)$. It is possible to go from site P_n to site P_{n+1} by means of a zigzag between the current cell and cell 0 (striped lines in [Fig. 17a](#)) followed by a shift of one cell to the right (dark gray lines edged in black in [Fig. 17a](#)).

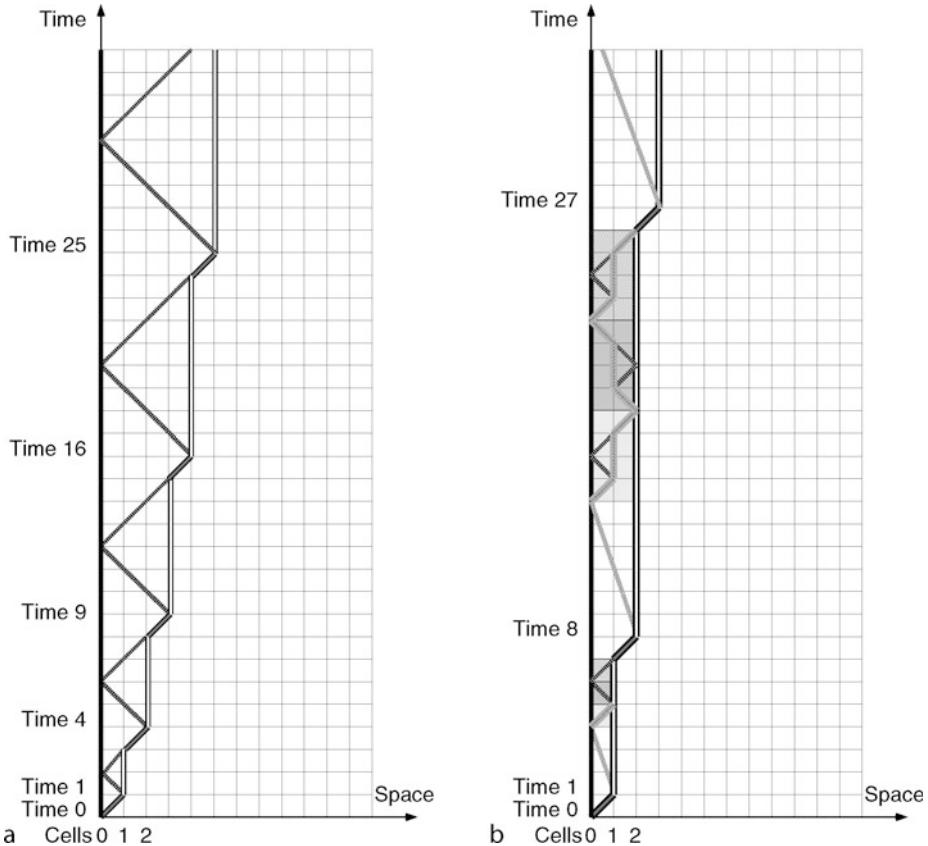
In a similar way, one can distinguish the sites (n, n^3) in using the formula

$$(n+1)^3 = n^3 + 3n^2 + 3n + 1$$

[Fig. 17b](#) illustrates the process, which is a little more complex. In order to build the vector $(n+1, (n+1)^3) - (n, n^3) = (1, 3n^2 + 3n + 1)$, one identifies the one step moves of the signal to be built. In [Fig. 17b](#), the signal joining the sites (n, n^3) is a white black edged line, except

Fig. 17

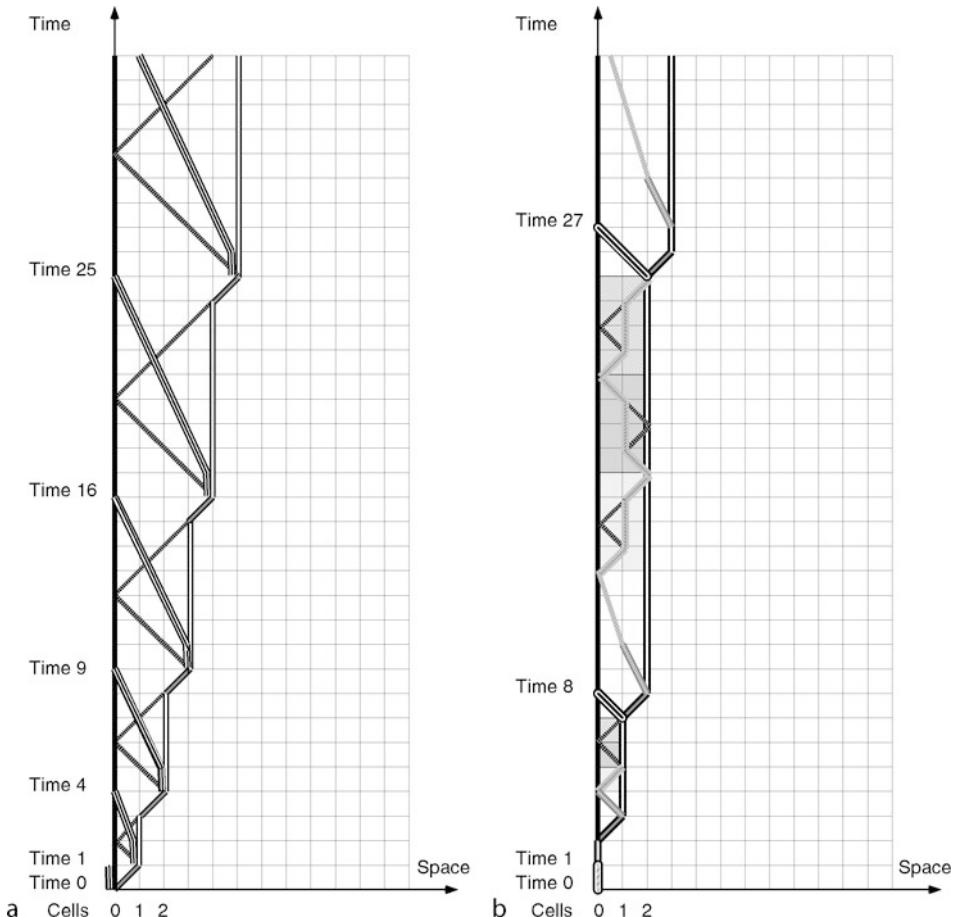
(a) Distinguishing times (n, n^2) and marking $n \rightarrow n^2 + n$. (b) Distinguishing times (n, n^3) and marking $n \rightarrow n^3 + n$.



when it moves to the right for one step where it becomes a gray black edged line. Cell 0 is marked by a stationary black signal. By means of a signal of slope 3 stemming from site (n, n^3) (light and medium striped lines in Fig. 17b), times $3n$ are marked. In order to mark $3n^2$, one uses three times the method to distinguish (n, n^2) (see Fig. 17a). The signal marking the sites (n, n^2) , starting from cell 0 (or from cell n) will reach cell n (or 0, respectively) in n^2 units of time. In Fig. 17b, n^2 is marked starting from the cell 0 (with a light gray background) then from the cell n (with a medium gray background) and finally, starting again from cell 0 (with a background striped in light and medium gray).

It is clear that by iterating the method, and using the binomial formula, one is able to mark (n, n^k) for all k .

Now, consider times n^2 . One observes that sending a signal to the left at speed 1 from site $(n-1, n^2 - (n-1)) = (n-1, n^2 - n + 1)$ ends in cell 0 where n^2 is then marked. So, one has to build signals that allow one to distinguish these sites, and this can be done as above: As $(n, (n+1)^2 - (n+1) + 1) - (n, n^2 - n + 1) = (1, 2n)$, from $(n-1, n^2 - (n-1))$ stems a signal \mathcal{S} up to site $(n, n^2 - (n-1) + 1)$, where it sends a signal \mathcal{C} to the left, which has to

Fig. 18(a) Marking times n^2 . (b) Marking times n^3 .

count $2n - 1$, and becomes itself stationary. In order to count $2n - 1$, \mathcal{C} goes at speed 1 to the left up to cell 1, stays there for one time, starts again at speed one to the right and meets \mathcal{S} on $(n, n^2 - n + 1)$. Of course the process has to be initiated (for $n = 1, 2$). This is illustrated in [Fig. 18a](#). In order to mark times n^3 (and more generally n^k), one uses a different method because times $3n$ have to be marked. One loses one time step in replacing the signal of slope 3 in [Fig. 17a](#) by a signal that moves from one cell to the left in two time steps (light and medium gray stripes, edged or not in dark gray in [Fig. 18b](#)). The process is initiated (for $n = 1$) as indicated in [Fig. 18b](#).

5.1.3 Some Properties of Functions That Can Be Marked

More generally, it has been proved (Mazoyer and Terrier 1999) that the class of functions that can be marked is closed under addition, subtraction (with some additional conditions of

growth), multiplication by a rational, multiplication, composition, and iteration. There is also an interesting characterization of these functions which links them to a class of complexity. (For the zoology of cellular classes, see Delorme and Mazoyer (1996).)

Proposition 4 (see Mazoyer and Terrier (1999)) *A function can be marked on the first cell if and only if the language $\{1^{f(n)} | n \in \mathbb{N}\}$ is recognizable in real time by some cellular automaton.*

Another interesting property is expressed in the following proposition, the proof of which is developed below.

Proposition 5 (see Mazoyer and Terrier (1999)) *If a function f can be marked on the first cell and if there is some integer k such that*

$$\forall n, (k-1)f(n) \geq kn$$

then there is a signal which characterizes the sites $(n, f(n))$, and is hence of slope f .

The condition that f has to satisfy is necessary – it is possible to mark 2^{2^n} and $n + \log \log n$ on the first cell, but impossible to mark $(n, n + \log \log n)$, because there is no signal with a slope between n and $n + \log n$ (see \blacktriangleright Proposition 6).

Proof A signal \mathcal{E} is sent at speed 1 to the right from each site $(0, f(n))$ (see \blacktriangleright Fig. 19). Let k be assumed to be even. One begins by marking the sites $(kn, kf(n))$.

- From site $(0, 0)$ starts a signal \mathcal{T} of slope $\frac{k+1}{k-1}$, a signal $\mathcal{D}_{\frac{k}{2}}$ to the right that progresses $\frac{k}{2}$ cells in $\frac{k}{2}$ steps and that becomes stationary on the cell $\frac{k}{2}$, and a signal \mathcal{D}_k that progresses to the right, of k cells in k times, then remains on cell k .
- When a signal \mathcal{E} and a signal \mathcal{T} meet, the signal \mathcal{E} disappears, a signal \mathcal{E}_{-1} of slope -1 is created and \mathcal{T} goes on with the same slope.
- When a signal \mathcal{E}_{-1} and a signal $\mathcal{D}_{\frac{k}{2}}$ meet, the signal \mathcal{E}_{-1} disappears, a signal \mathcal{E}_1 of slope 1 is created, the signal $\mathcal{D}_{\frac{k}{2}}$ goes on to the right for $\frac{k}{2}$ cells in $\frac{k}{2}$ times and then becomes stationary.
- When a signal \mathcal{E}_1 and a signal \mathcal{D}_k meet, the signal \mathcal{E}_1 disappears, the signal \mathcal{D}_k goes on to the right for k cells in k times and becomes again stationary.

It is not difficult to verify that the n^{th} signal \mathcal{E} meets signal \mathcal{T} on the site $\left(\frac{(k-1)f(n)}{2}, \frac{(k+1)f(n)}{2}\right)$. Starting from this site, \mathcal{E}_{-1} meets signal $\mathcal{D}_{\frac{k}{2}}$ on site $\left(\frac{kn}{2}, kf(n) - \frac{kn}{2}\right)$. Then signal \mathcal{E}_1 , which stems from this site, meets \mathcal{D}_k on site $(kn, kf(n))$.

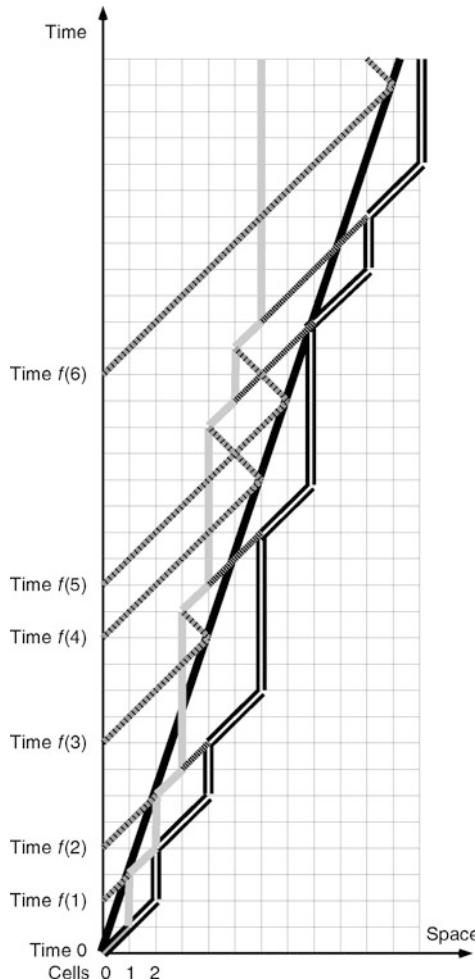
Now, in order to mark the sites $(n, f(n))$, one groups the cells k by k . That means that one considers a new cellular automaton such that the state of site (i, j) represents the states of the sites

$$\{(ki + u, kj + v) ; 0 \leq u < k ; 0 \leq v < k\}$$

It is known that the converse of \blacktriangleright Proposition 5 is true; it comes from the stability under subtraction. From the sites $(n, f(n))$, the sites $(0, n + f(n))$ can be marked. According to the \blacktriangleright Proposition 5 of Mazoyer and Terrier (1999), with $a = b = 1$, $m = 1$, $g(n) = n$, and $\hat{f}(n) = f(n) + n$, the required condition $\hat{f} \setminus g \geq (mb + 1) \setminus ma$ can be obtained to say that $\hat{f}(n) - g(n) = f(n)$ is Fisher's constructible. \blacktriangleright Proposition 5 can also be derived from the \blacktriangleright Proposition 5 of Mazoyer and Terrier (1999).

Fig. 19

Proof of \blacktriangleright Proposition 5. Signals \mathcal{E} are drawn in black stripes on a medium gray background as well as signals \mathcal{E}_{-1} (stripes in the other direction). Signal \mathcal{T} is black. Signal \mathcal{D}_k is light gray, while \mathcal{D}_k is black edged white. Signals \mathcal{E}_1 dark striped.



It should be noticed that although in the proof of \blacktriangleright Proposition 5 the space between the sites $(n, f(n))$ is used, it can be avoided in some cases. For example, \blacktriangleright Fig. 20a, b shows how to transform the algorithms of \blacktriangleright Fig. 16a, b, which marks times 2^n and 3^n into algorithms marking the sites $(n, 2^n)$ and $(n, 3^n)$. The idea is the same in both cases. The signal that was stationary on cell 0 progresses for one cell to the right each time it is marked (thick black edged medium or dark gray lines in \blacktriangleright Fig. 20a, b). This move is compensated by means of the zigzagging signal. The signal of slope 3 (or 2) is accelerated by one cell to the right; it progresses by two cells for three times (or by two cells for two times) (edged black medium gray in

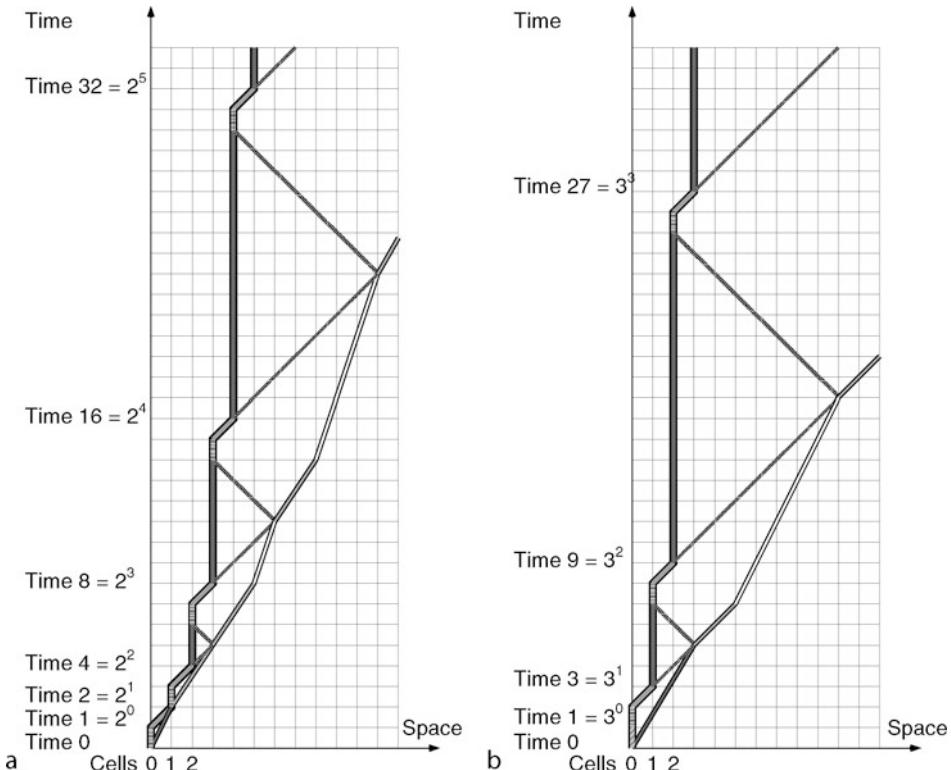
Fig. 20(a) Marking sites ($n, 2^n$). (b) Marking sites ($n, 3^n$).

Fig. 20a, b), on time before coming back to its *normal* advance. The zigzagging signal has to lose one unit of time, so it waits one time before going on.

5.2 Limitation of Slopes

Until now, signals have been constructed starting from a basic initial configuration $\infty LGL \infty$ where L is a quiescent state and G a distinguished one. It can be proved here that, in this framework, it is not possible to get signals of all slopes.

Proposition 6 (Mazoyer and Terrier 1999) *There does not exist any cellular automaton that, starting from a basic initial configuration $(\infty LGL \infty)$, builds a signal of slope between 1 and $1 + \mathcal{O}(\log n)$.*

Proof First, prove a preliminary result.

Let D_k denote the diagonal $\{(j, j+k) \mid j \in \mathbb{N}\}$. In the evolution of a cellular automaton $\mathcal{A} = (Q, \delta)$ on a basic configuration $\infty LGL \infty$ the sequence of the states on D_k is ultimately periodic of period $\pi_k \leq |Q|^{k+1}$ and its non-periodic part is of length $\iota_k \leq |Q|^{k+1}$. Moreover $\text{lcm}(\pi_k, \pi_{k-1})$ is π_k (or π_{k-1} divides π_k).

The proof is by induction on k and relies on the fact that the behavior of a cellular automaton on a diagonal D_k is the behavior of a transducer, the Fisher automaton (see Sutner (1997)).

- The sequence $(q_j)_j$ of D_0 -states is given by $q_0 = G$ and $q_{j+1} = \delta(q_j, L, L)$. One has then a dynamical system on a finite set with $|Q|$ elements. It becomes ultimately periodic with $\pi_0 \leq |Q|$, and $t_0 \leq |Q|$.
- The sequence $(q_j^{k+1})_j$ of D_{k+1} -states is given by $q_0^{k+1} = q$ and $q_{j+1}^{k+1} = \delta(q_j^{k+1}, q_{j+1}^k, q_{j+2}^{k-1})$. Let π be $\text{lcm}(\pi_k, \pi_{k-1})$. Then the dynamical system $q_j^{k+1} \rightarrow q_{j+\pi}^{k+1}$ on a finite set $|Q|$ becomes ultimately periodic of period less than $|Q|$. As a consequence, $\pi_{k+1} \leq |Q| \times |Q|^{k+1}$ and $\text{lcm}(\pi_{k+1}, \pi_k)$ is either π_{k+1} or π_k . The length of the transitory is, of course, at most $|Q|^{k+2}$.

Now, assume that the evolution of some cellular automaton shows a signal S of slope $1 + o(\log n)$ but not 1. Then, for k large enough, segments of consecutive states in S of length more than $|Q|^{k+1}$ appear on D_k . If not, the asymptotic slope of S would be greater than, or equal to, $1 + \log_{|S|} n$, which contradicts the above property.

Proposition 7 (see Mazoyer and Terrier (1999)) *There is a cellular automaton $\mathcal{A} = (Q, \delta)$ which, starting from a basic initial configuration ($^\infty LGL^\infty$), builds a signal of slope $1 + \log_B n$ (with $B \geq 2$).*

Proof Let B be some base. Let $\bar{n}_B = a_0^n|B|^0 + a_1^n|B|^1 + \dots + a_{[\log_B n]+1}^n|B|^{[\log_B n]+1}$ be the representation of n in base B . If one writes in an array the digits of the numbers in such a way that a_j^n is in square (n, j) (see Fig. 21a for $|B| = 2$), one gets an array that is locally constructible – the propagation of what to carry is local: the $(n+1)$ st column is obtained by adding 1 to the n th column from bottom to top. In square (n, j) , this induces a carry over β_j^n with $\beta_{-1}^n = 1$. More precisely, a_{j+1}^n (and β_{j+1}^n) is the remainder (the quotient, respectively) of the division of $a_j^{n-1} + \beta_j^n$ by B . Moreover, this array is such that the k^{th} line is periodic of period 2^{k+1} starting with $\sum_{i=0}^{i=k-1} 2^i = 2^k - 1$, and that the height of the n^{th} column representing n is $[\log_B n] + 1$.

The local nature of the above array allows one to easily design a cellular automaton $\mathcal{A} = (Q, \delta)$ that builds it. The digits of n are on a diagonal to the right (see Fig. 21b) and the state $\langle j, n+j \rangle$ represents a_j^n and β_j^n (the latter is not shown in Fig. 21b). The state transition is given by $a_{j+1}^n = a_j^{n-1} + \beta_j^n (\text{mod } B)$.

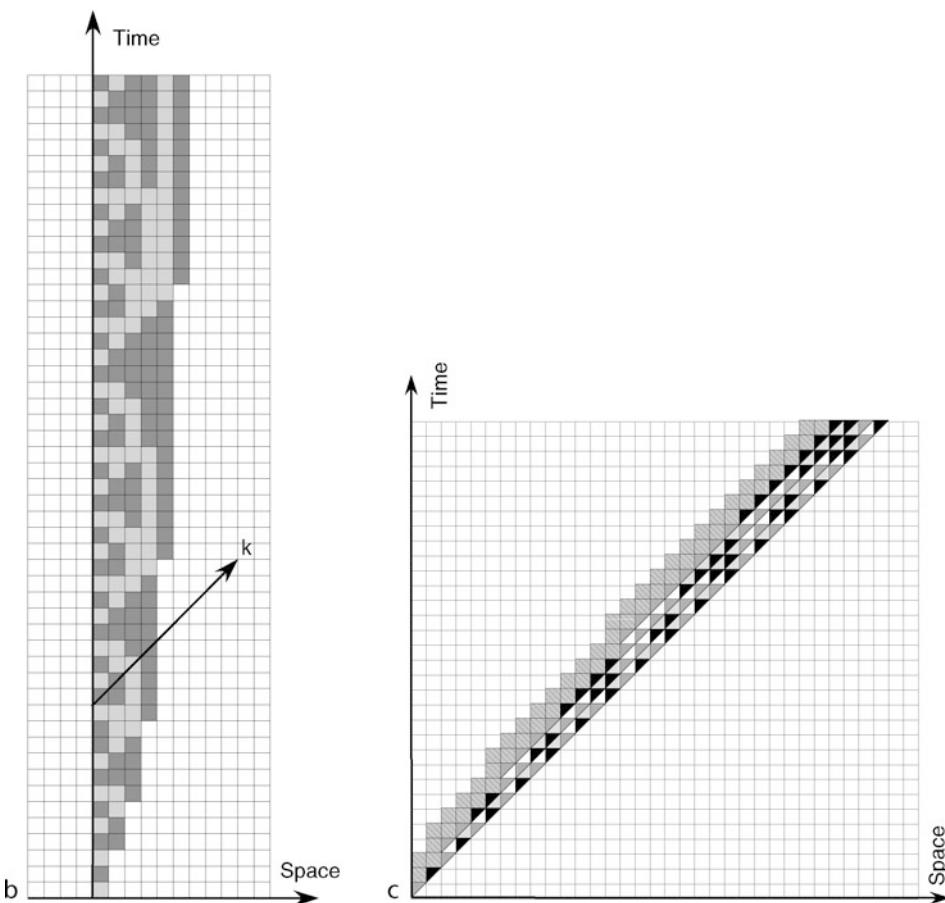
The idea of the proof is to use the former process. In order to get the wanted result, if n is written in base B as

$$\bar{n}_B = a_0^n B^0 + a_1^n B^1 + \dots + a_{[\log_B n]+1}^n B^{[\log_B n]+1},$$

it is sufficient to set a_j^n and β_j^n on site $(n, n+j)$. The former computations show that $\langle n, n+j+1 \rangle$ is the remainder and the quotient of the addition of the first part of $\langle n-1, n-1+j \rangle$ and the second part of $\langle n, n+j \rangle$ by $|B|$. But the arguments of δ are $\langle n-1, n+j \rangle$, $\langle n, n+j \rangle$ and $\langle n+1, n+j \rangle$, which means that the information contained in $\langle n-1, n-1+j \rangle$ must be kept for one unit of time, in order to find it again in $\langle n-1, n+j \rangle$. A way to do it is to choose $Q = B \cup \{0^{\text{with carry}}\} \cup \{L, S\}$ where L is quiescent and S represent the signal S of slope $1 + \log_B n$. The process is illustrated in Fig. 21c. In this case, $B = 2$, values of a_j^n are in dark gray (for 0) or black (for 1) while those of β_j^n are in light gray (for 0) or medium gray (for 1). Gray stripes represent S .

Fig. 21

(a) The sequence of natural numbers written in base 2. (b) Generation of all the representations of natural numbers in base 2. (c) Signal of slope $1 + \log_2 n$.



6 Basic Tools and Methods for Moving Information

6.1 Information Moves

In a space-time diagram, one identifies some states or some patterns of states as *information*. It is actually impossible to give a formal general definition of information. It is all the

more difficult because one can imagine that information is given to the network from outside (see the chapter [Computations on Cellular Automata](#), and also [Definition 4](#)). The following definition is used.

Definition 9 A signal S (in some orbit of a cellular automaton $\mathcal{A} = (Q, \delta)$) is said to carry pieces of information (or simply information) from Γ if $S = S^* \times (\Gamma \cup \{A\})$.

Starting from such a definition, every geometric transformation on signals (as in [Sect. 4.1](#)) amounts to moving information. In particular, one can see signal translation in [Sect. 4.1](#) (forgetting the stop point of the transformation, and considering signals as infinite “guides”) as a way to create an infinite stream of information that moves along a rational direction. This stream can be “collected” on a signal parallel to the one which carries it, as in [Sect. 4.1](#), or, a priori, on any other signal.

In certain cases, the process is simple, as in the examples of [Fig. 22](#). But it can be more complicated and needs the data to be compressed. Consider a signal S of slope $\frac{p}{q}$ that contains k elementary pieces of information on k cells. The stream of information from these cells (meaning the information contained in a strip determined by these k cells of S and left diagonals of some rational slope $\frac{p'}{q'} \geq 1$) can mix pieces of information originally contained in these k successive cells. It is clear enough that if one wants to collect the original information (from Γ) carried by S and sent via this stream, onto a signal S^* of slope $\frac{p'}{q'}$, some pieces of information coming from different cells will arrive on S^* in the same cell. (The “arrival” cell is assumed to be the greater cell that has knowledge of the information). This is the case when the ratio $\frac{p'}{q'} / \frac{p}{q}$ is strictly more than 1. Information is said to be compressed. Conversely there is a phenomenon of decompression if the same ratio is strictly less than 1. In this case, pieces of information coming from a same cell of S possibly arrive on different cells of S^* . Examples of rational compressions and decompression are illustrated in [Figs. 23a–c](#).

[Figure 23a](#) shows the case where the slope of S is 1 while the slope of S^* is 2. The data is compressed in a periodic but nonconstant way. In [Fig. 23b](#), signal S is of slope 0 and S^* of slope 1. In this case the data is compressed in a constant way. [Figure 23c](#) shows the case of decompression.

Fig. 22

Information moves collected in a simple way. The original signal is in gray. It sends information to the black signal, along lines that determine a “stream” of information.

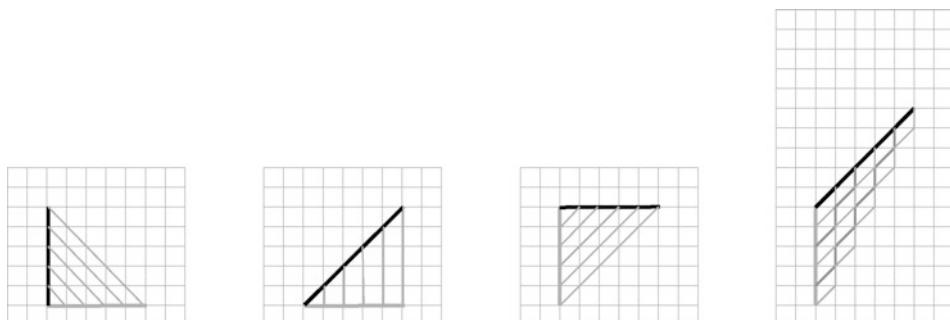
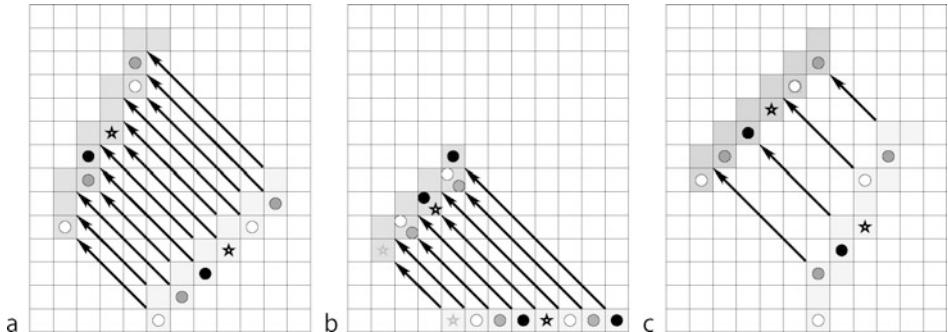


Fig. 23

(a) Rational compression. In [Figs. 23a–c](#), pieces of information are represented by circles or stars in a different level of gray. The signal \mathcal{S} from which information is sent out is light gray while the signal which collects them is dark gray. (b) Constant compression. (c) Decompression.



The phenomena of compression and decompression influence the general notions of “grouping” and “de-grouping”, which are developed in the chapter [Universality in Cellular Automata](#).

Sometimes, one has to keep pieces of information on the same cell for some time (which means moving it along a line of slope ∞), or to move it from some cell to another cell at some finite distance in some finite number of time units. This can be done by the means of “counters.” Two main types of counters appear in the literature: linear counters, as in [Sect. 4.3](#), or exponential counters (see [Fig. 27b](#)). It is worth noting that signals of polynomial slopes, briefly discussed in [Sect. 4.3](#), produce polynomial counters. Until now, these counters have not depended on the data size (that is the number of cells of \mathcal{S} which carry information in Γ). However, this will happen, for example, when arithmetical operations have to be performed on this size or when synchronization is needed. Next, such arithmetical operations will be developed, but first briefly recall the definition and one main result concerning synchronization.

Definition 10 A cellular automaton $\mathcal{A} = (Q, \delta)$ is a solution to the firing squad synchronization problem (FSSP) if

- Q is the disjoint union of Q_L , Q_X , $\{G\}$, and $\{F\}$.
- (Q_L, δ) and (Q_X, δ) are sub-automata of \mathcal{A} . (In other words, δ is stable on Q_L and Q_X).
- For each initial configuration ${}^\infty q G q^{*^{n-1}} q^\infty$, where states q are in Q_L and states q^* in Q_X , there exists some integer θ such that:
 - For all times $t \in \{0, \dots, \theta - 1\}$, configuration at time t is of the form ${}^\infty q q_1^* \dots q_n^* q^\infty$ where states q are in Q_L and states q_i^* in $Q_X \cup \{G\}$.
 - The configuration at time θ is of the form ${}^\infty q F^n q^\infty$ where the states q are in Q_L . \mathcal{A} is in optimal time if $\theta = 2n - 2$.

Synchronization is a well-known arithmetical problem (see Balzer (1967) and Mazoyer (1987), and a paper on the state of the art in 1989 (Mazoyer 1989a)).

Proposition 8

1. There are solutions to the FSSP in optimal time (Balzer 1967; Mazoyer 1987).
2. There are solutions in a wide variety of times (Gruska et al. 2006; Mazoyer 1989b).

It has to be noted that it is also possible to start from configurations $\infty q(q^*)^m G(q^*)^{n-m-1} q^\infty$ (Vaskhavsky et al. 1969) or from configurations $\infty qG(q^*)^{n-2}Gq^\infty$ (Mazoyer and Reimen 1992), and that it is possible to synchronize according to a given slope (this has been introduced in Čulik (1989)).

6.2 Arithmetical Operations on Segments

In order to organize a space–time diagram, it is sometimes necessary to extract information from some signal. If the signal is horizontal, it can be said that the information carried by the signal is not pieces of information on its states, but that the information is its length ℓ (as in [Sect. 4.3](#)). Then, knowing two segments (that means, horizontal segments with marked beginning and end of “lengths” p and q), one can be asked to compute (under the form of segments) things such as $p \div q$ or $p \bmod q$. Two methods are described. The first one is very general and uses the techniques encountered in [Sect. 6.1](#). However, it is not very fast. The second one, geometrical, comes down to cutting a segment in ratio $\frac{p}{q}$ where $\frac{p}{q} < 1$. In any case, the first step is to compute an integer multiple of q .

- The first operation : to put kq on a diagonal ($k, q \in \mathbb{N}$).

This task consists of duplicating k times the initial segment of length q . One has the choice to code k either by states or by a counter, and the result can be obtained with or without synchronization.

The general process is as follows:

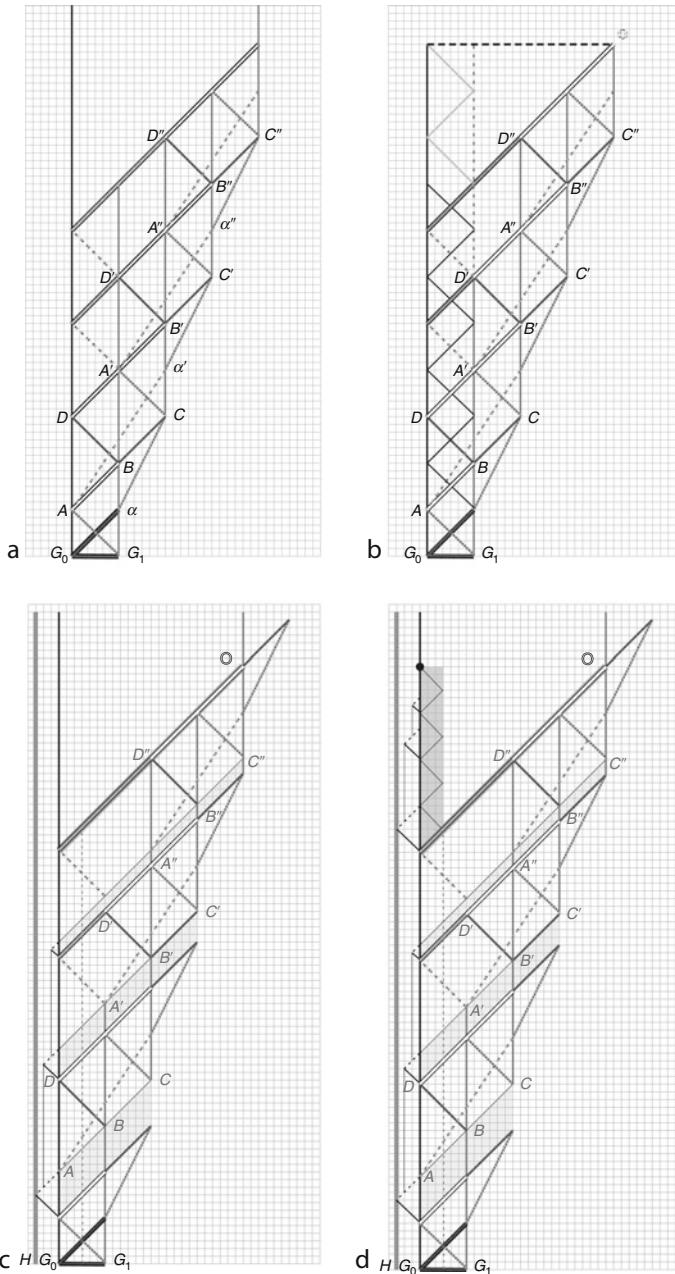
- In the first step, the data contained between two vertical signals (G_0 and G_1 in [Figs. 24a–d](#)) is carried on a diagonal between sites A and B . The sites A and α are marked by means of two signals at speeds 1 and -1 , stemming from G_0 and G_1 . The data is also duplicated between sites B and C by means of a signal of slope 2 sent out from site α (dark striped line in [Fig. 24a](#)).
- The process is then repeated: the abscissa of A' , the “new” site A , being the one of the former site B , and α' , the “new” site α , being on the cell of site C and found at the intersection of a signal of slope $\frac{3}{2}$ sent out from site A and of a stationary signal stemming from site C . The start D on G_0 is obtained in sending a signal of slope -1 from the site B on G_1 .

The algorithm is stopped after k repetitions of the process (see [Fig. 24a](#)) due to the encoding of k by states. If one wants to synchronize the process, one uses a signal that zigzags k times between G_0 and G_1 , counting the beginning on G_0 if k is even, and on G_1 if k is odd, as shown in [Fig. 24b](#). Of course, a general method of synchronizing in the sense of [Definition 10](#) can also be used.

When k is given by some counter (there are different possibilities: as in [Sect. 4.3](#), or in [Fig. 27b](#)), or in unary (as in [Figs. 24c, d](#)) a “freezing” technique is used. One assumes that the counter is situated at the left of G_0 , between H and G_0 (see [Figs. 24c, d](#)). Each time the initial length is added, cell G_0 launches the process of decreasing the counter by 1, taking θ

Fig. 24

(a) Duplication of a horizontal segment k times, k being coded by states, without synchronization. (b) Duplication of a horizontal segment k times, k being coded by states, with synchronization. (c) Duplication of a horizontal segment k times by means of a counter, without synchronization. (d) Duplication of a horizontal segment k times by means of a counter, with synchronization.



time units. During these θ time steps, the general algorithm is frozen (light areas in [Fig. 24c, d](#)).

Freezing an evolution of a sub-automaton $\mathcal{A}^* = (Q^*, \delta)$ of some automaton $\mathcal{A} = (Q_{\mathcal{A}}, \delta)$ consists in duplicating Q^* in $Q^* \cup Q^\sharp$. At the time of freezing, a state q^* of Q^* becomes the corresponding state q^\sharp of Q^\sharp . A state of Q^\sharp remains stationary during the freezing. At the time of unfreezing, it becomes again the state q^* of Q^* and interacts with the states of Q^\sharp as well as with the ones of Q^* . This makes the general algorithm to evolve with some delays ([Fig. 24c](#)).

If one wants to synchronize the result, one uses general synchronization in the sense of [Definition 10](#), starting from the site O in [Fig. 24d](#). Also other methods are possible, but they depend on the counter. For example, if the response time of the counter is less than the length to duplicate, then one can use a signal zigzagging between G_0 and $G_0 + \frac{G_1 - G_0}{2}$ as in [Fig. 24d](#) (see [Sect. 4.3](#)).

Now, see how to find the quotient and the remainder of the division of ℓ by q when $\ell \geq q$.

► The division.

The data ℓ is given, as before, as a horizontal segment delimited by two stationary signals (in medium gray in [Figs. 25a–c](#)). A signal \mathcal{D} is sent at speed 1. If q is coded by states ([Figs. 25a, c](#)), D contains q counter using states d_0, \dots, d_{q-1} . Otherwise, $D = \{d\}$, and q is given by a segment. In this case q is duplicated, as indicated in the first point, k times until $kq \geq \ell$. The duplication process is stopped when the diagonal that carries kq meets the stationary signal marking the end of the initial segment ℓ . In either case, the iq^{th} cells of \mathcal{D} become marked (by d_{q-1} or by the a signal marking q).

A more elegant method is illustrated in [Fig. 25c](#). It consists of using the states d_0, \dots, d_{q-1} to create a signal \mathcal{S} of slope q , by the means of the process in [Sect. 2.1](#). Then, the distance between the signal marking the beginning of segment ℓ and the signal \mathcal{S} at the time when \mathcal{S} meets the left diagonal indicating the end of ℓ , gives the quotient. A state of \mathcal{S} (or a state of \mathcal{D} if all is coded as in [Sect. 4.2](#)) gives the remainder of the division of ℓ by q .

We conclude with two remarks. First, a particular relative positioning of the segments coding ℓ , p , and q is assumed. But the data places can be modified by the techniques in [Sect. 6.1](#). Secondly, only unary counters have been considered. This also is general because the computation of the base change is realized in an optimal time on a cellular automaton with neighborhood $\{-1, 0, +1\}$ (see the chapter [Computations on Cellular Automata](#) of this volume).

6.3 A Riddle

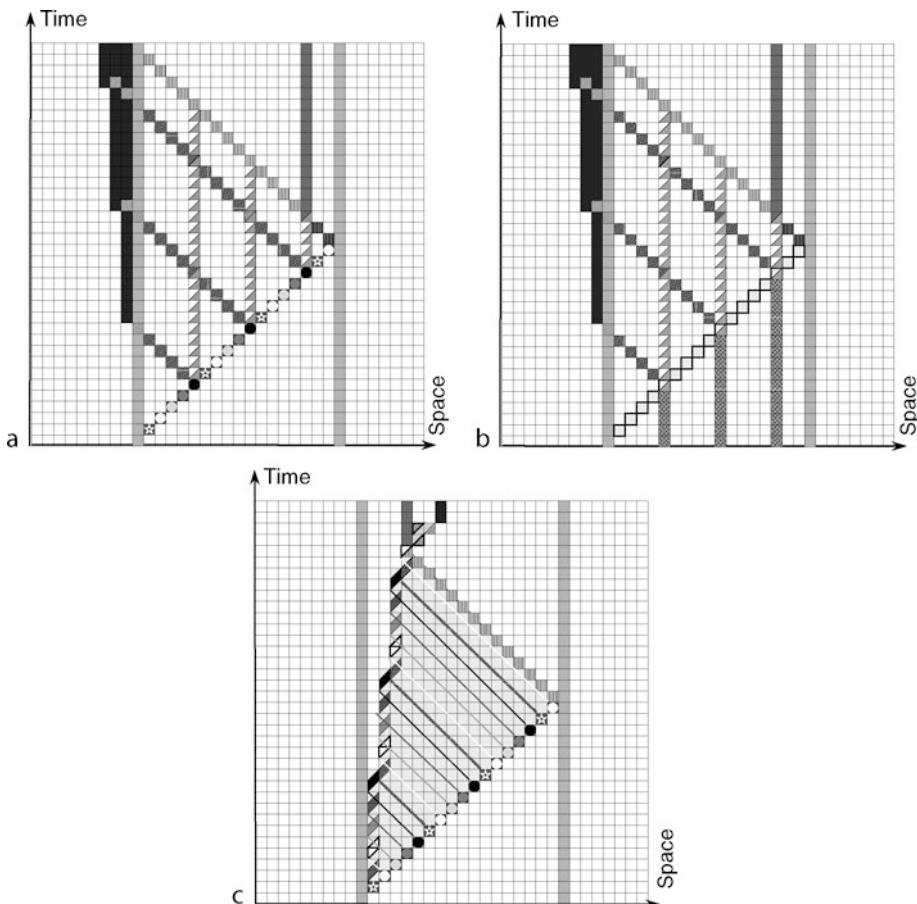
Starting from the possibility to mark times 2^k on cell 0, a riddle method is next developed, which allows one to get the binary expansion of each integer. This method can also be used with other functions.

[Figure 26a](#) reproduces [Fig. 16a](#). The initial configuration of the automaton that builds this geometric diagram is ${}^\infty LGL^{n-2}EL^\infty$, where L is quiescent, G is the state of cell 0 and a special state E is the initial state of cell $n - 1$. Cell $n - 1$ sends a signal \mathcal{E} , to the left, at speed 1. This signal meets one and only one signal of speed one stemming to the right from cell 0. If \mathcal{E} meets the k^{th} signal ($k \geq 1$), then $2^k \leq n < 2^{k+1}$. That allows one to know $\lfloor \log_2 n \rfloor$.

The process can be iterated (see [Fig. 26b](#)). Each time a signal at speed 1 to the right, coming from a site $(0, 2^k)$, meets the signal \mathcal{D} of slope 3 (white on black in [Fig. 26a](#) and

Fig. 25

(a) General method for the division of ℓ by q , where q is coded by states. In [Fig. 25a–c](#), data is represented by circles or stars. In [Fig. 25a, b](#), stationary signals are indicated by triangles, and the signals of slope -1 are striped. The counting of the quotient is realized in the very dark gray. The remainder is the segment delimited by the two stationary signals at the right (dark and medium gray). (b) General method, q coded by segments. (c) Method using a signal, q coded by states.

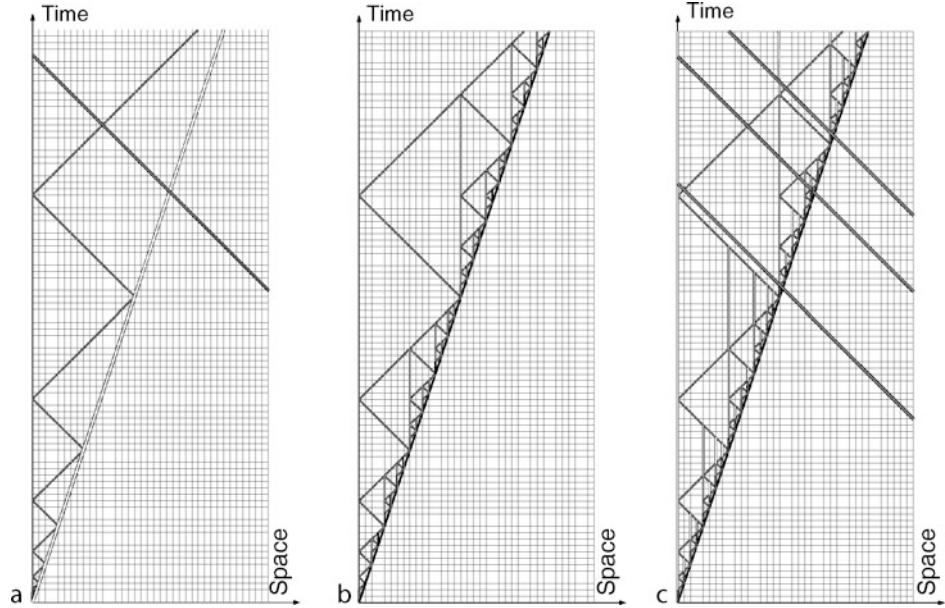


black in [Fig. 26b](#)) on site $(2^{k-1}, 3 \cdot 2^{k-1})$, a new exponential construction is started: a stationary signal is launched (in white on black and medium gray in [Fig. 26b](#)) and the process is iterated. It stops on the signal at speed 1 coming from the site $(0, 2^{k+1})$. Successive iterations are represented in [Fig. 26b](#). Let us call signals of type \mathcal{L} the signals at speed 1 to the right.

The process above easily allows one to know the number of bits 1 in the binary expansion of n (see Terrier (1991)). Signal \mathcal{E} , sent from cell $n - 1$, reaches cell 0 at time n and intersects several traces of right moving signals of type \mathcal{L} , in white and black vertical signals in [Fig. 26c](#). The last intersected signal is emitted from site $(0, \lfloor \log_2 n \rfloor)$. If $(i_0 = 0, i_1, \dots, i_v)$ is the sequence of the iteration levels of the intersected signals of type \mathcal{L} , one has $n = \sum_{j=0}^{j=v} \frac{2^{\lfloor \log_2 n \rfloor}}{2^{i_j}} + \epsilon$

Fig. 26

(a) $\lfloor \log_2 n \rfloor$. On [Fig. 26a–c](#), signals to the right (left) are striped to the right (to the left). Signal \mathcal{E} , representing the integer n , comes at speed one from the left. (b) Iterations. (c) Number of 1's in $\overline{n_2}$.



with $\epsilon \in \{0, 1\}$. One can determine ϵ by observing the intersection point of signals \mathcal{D} and \mathcal{E} . In

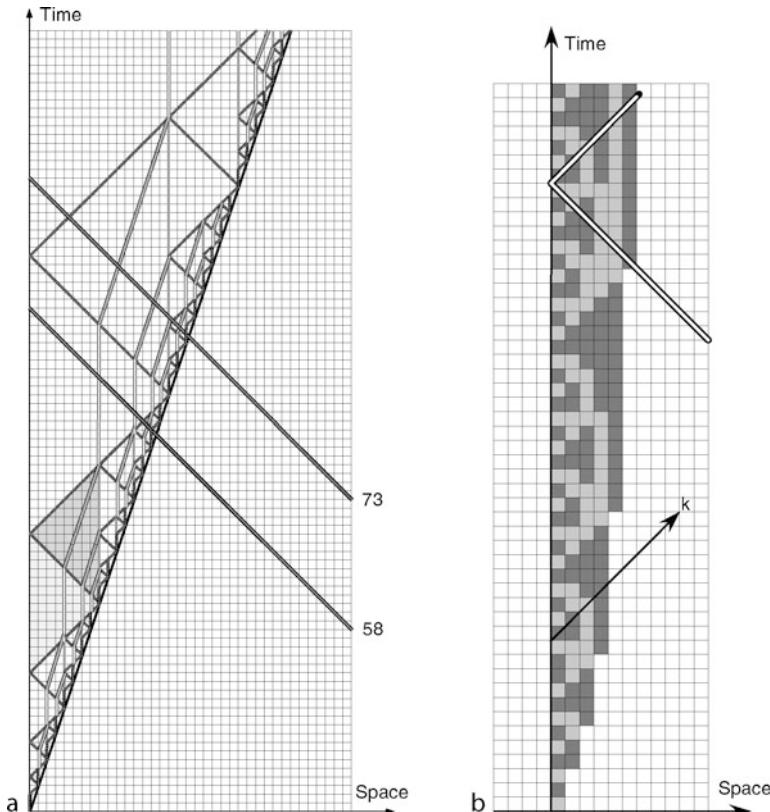
[Fig. 26c](#), the binary expansion of $n = 66$, $n = 86$ and $n = 98$ has 2, 4, and 3 digits 1.

All the signals in [Fig. 26b](#) building the iterated exponential appear as “particular grid patterns.” This allows one to uniformly determine some given properties, and can be used as a riddle.

The diagram in [Fig. 26b](#) can be complemented, as illustrated in [Fig. 27a](#), in order to get the whole binary expansion of any integer. One proceeds as follows:

- At each iteration $j \geq 2$ of the exponential computation, one considers the triangles $(0, 2^j)$, $(2^{j-1}, 3 \cdot 2^{j-1})$ and $(0, 2^{j+1})$ (T_j , light gray in [Fig. 27a](#)) and $(2^{j-1}, 3 \cdot 2^{j-1})$, $(0, 2^{j+1})$ and $(2^j, 3 \cdot 2^j)$ (T'_j , medium gray in [Fig. 27a](#)).
- At each iteration j , a stationary signal is sent from each site marking the end of a former iteration on T_j (white on gray in [Fig. 27a](#)), which stops when it enters T'_j and emits a new signal of slope 2 (light stripes on medium gray in [Fig. 27a](#)). It disappears at the end of the iteration (that means on the other side of the triangle where it meets a gray stationary signal, coming from the construction of the iterated exponential).
- The process is then iterated.

Finally, a signal of slope 1 to the left, starting from cell $n - 1$, with $n = 2^k + 2^{k-\ell} + \dots$ intersects $\ell - 1$ signals of slope 2 (light stripes on medium gray in [Fig. 27a](#)) after having crossed the signal indicating that $n - 2^k \geq 2^{k-\ell}$. So, one gets $\overline{n_2}$. For example, if one considers a line corresponding to $n = 58$ (white on black in [Fig. 27a](#)), starting from cell 0, it meets two

Fig. 27(a) Getting \bar{n}_2 by means of the riddle. (b) Getting \bar{n}_2 as in [Sect. 5.2](#).

successive stationary (white on gray) signals (so one has 111), then one signal of slope 2 (gray stripes on black), then another stationary (gray) signal (so one gets 01). Finally, as the line arrives on \mathcal{D} under the middle of the “little segment,” one has 0 as the last bit. Altogether, we have read 111010, which actually represents 58!

Of course, in order to find \bar{n}_2 , one also can use the method of generating binary words as in [Sect. 5.2](#), illustrated in [Fig. 27b](#). The difference between both methods is that the “riddle” allows one to *anticipate* the computation and to use, in what follows, the obtained information.

6.4 Folding Space–Time

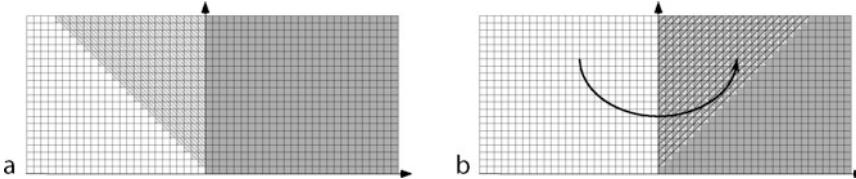
The interest here is in the evolution of an initial line of type c_{L^*} of some automaton \mathcal{A} where L^* is some quiescent state but possibly different from the usual quiescent state L . And the intention is to build an automaton \mathcal{C} that simulates \mathcal{A} and computes only on positive cells.

More precisely, if $\mathcal{A} = (Q_{\mathcal{A}}, \delta_{\mathcal{A}})$, one builds $\mathcal{C} = (\Theta, \Delta)$ as follows (see [Fig. 28a](#)).

1. $\Theta = Q_{\mathcal{A}} \cup (Q_{\mathcal{A}} \times Q_{\mathcal{A}})$, denoted as $\Theta = Q_{\mathcal{A}} \cup (Q_{\mathcal{A},+} \times Q_{\mathcal{A},-})$.

Fig. 28

(a) Zones in an evolution of \mathcal{A} . State L^* is in white. The influence of states different from L^* on cells to the left of cell 0 are marked by stripes. (b) Zones of computation in the evolution of \mathcal{C} simulating \mathcal{A} : states (X, \leftarrow) are represented in the lower triangle while the upper triangle represents (X, \leftarrow) and (X, \rightarrow) .



2. Initial configuration of \mathcal{C} is the “same” as the one of \mathcal{A} , that is ${}^\infty L^* q_0 q_1 \dots$, where the q_i are in the subset $Q_{\mathcal{A}}$ of Θ .
3. • If q_ℓ, q_c, q_r are in the subset $Q_{\mathcal{A}}$ of Θ and all different from L^* , $\Delta(q_\ell, q_c, q_r) = ((\delta_{\mathcal{A}}(q_\ell, q_c, q_r), \rightarrow), (L^*, \leftarrow))$.
• If q_c, q_r are in the subset $Q_{\mathcal{A}}$ of Θ and all different from L^* , $\Delta(L^*, q_c, q_r) = ((\delta_{\mathcal{A}}(L^*, q_c, q_r), \rightarrow), (\delta_{\mathcal{A}}(L^*, L^*, q_c), \leftarrow))$.
• If q_c, q_r are in the subset $Q_{\mathcal{A}}$ of Θ and if q_c or q_r is equal to L^* , $\Delta(L^*, q_c, q_r) = L^*$ (arbitrary choice).

Then, if at time 1, the configuration of \mathcal{A} has become ${}^\infty L^* q_{-1}^1 q_0^1 q_1^1 q_2^1 \dots$, the configuration of \mathcal{C} is

$${}^\infty L^*(q_0^1, \rightarrow), (q_{-1}^1, \leftarrow), (q_0^1, \rightarrow), (L^*, \leftarrow), (q_2^1, \rightarrow), (L^*, \leftarrow), \dots$$

4. • $\forall (q_\ell, \rightarrow), (q_c, \rightarrow), (q_r, \rightarrow) \in Q_{\mathcal{A}, \rightarrow}, \forall (q_\ell^*, \leftarrow), (q_c^*, \leftarrow), (q_r^*, \leftarrow) \in Q_{\mathcal{A}, \leftarrow}, \Delta(((q_\ell, \rightarrow), (q_\ell^*, \leftarrow)), ((q_c, \rightarrow), (q_c^*, \leftarrow)), ((q_r, \rightarrow), (q_r^*, \leftarrow))) = ((\delta_{\mathcal{A}}(q_\ell, q_c, q_r), \rightarrow), (\delta_{\mathcal{A}}(q_r, q_c, q_\ell), \leftarrow)).$
• $\forall (q_c, \rightarrow), (q_r, \rightarrow) \in Q_{\mathcal{A}, \rightarrow}, \forall (q_c^*, \leftarrow), (q_r^*, \leftarrow) \in Q_{\mathcal{A}, \leftarrow}, \Delta(L^*, ((q_c, \rightarrow), (q_c^*, \leftarrow)), ((q_r, \rightarrow), (q_r^*, \leftarrow))) = ((\delta_{\mathcal{A}}(q_c^*, q_c, q_r), \rightarrow), (\delta_{\mathcal{A}}(q_r^*, q_c, q_c), \leftarrow)).$

Then, if at time t the configuration of \mathcal{A} has become ${}^\infty L^* q_{-t}^t, \dots, q_{-1}^t q_0^t q_1^t q_2^t \dots$, the configuration of \mathcal{C} is

$${}^\infty L^*((q_0^t, \rightarrow), (q_{-1}^t, \leftarrow)), \dots, ((q_{t-1}^t, \rightarrow), (q_{-t}^t, \leftarrow)), ((q_t^t, \rightarrow), (L^*, \leftarrow)), ((q_{t+1}^t, \rightarrow), (L^*, \leftarrow)), \dots$$

5. Nonspecified transitions are arbitrarily fixed to give L^* .

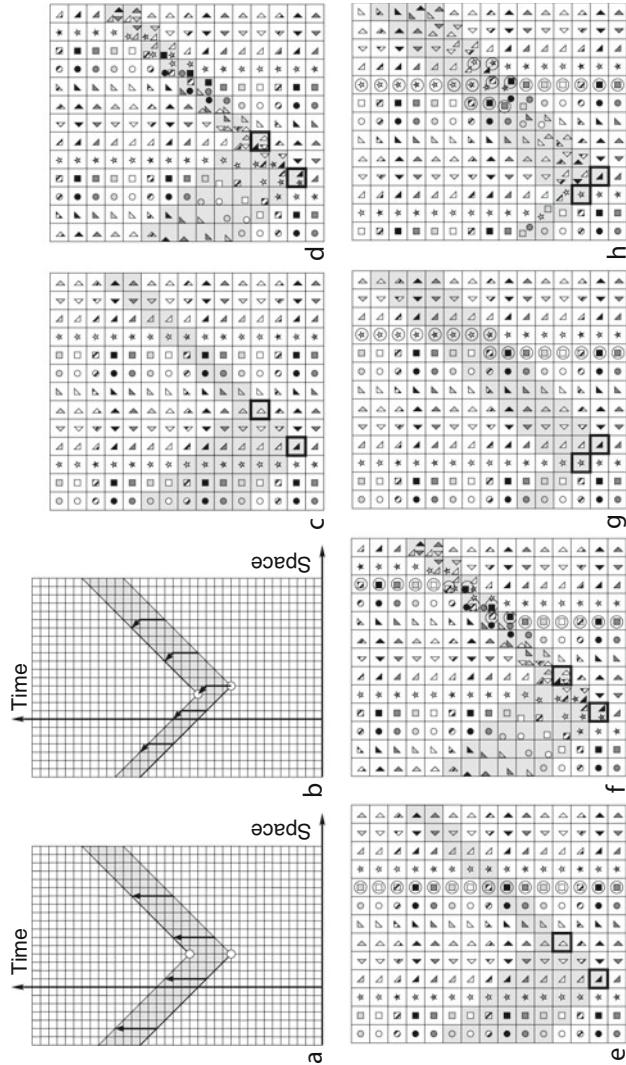
This technique of folding is used, for example, to prove that the class of languages recognized in linear time is exactly the class of languages recognized in linear time and space bounded by the input length (see Delorme and Mazoyer (1996) and Mazoyer (1999)).

6.5 Freezing and Clipping

The technique of “freezing” has already been evoked in [Sect. 6.2](#). Here, we recall and generalize it.

1. [Figure 29a](#) symbolized the freezing technique. Given some cellular automaton \mathcal{A} with a special state G and some integer k , there exists an automaton \mathcal{B}_k such that $Q_{\mathcal{A}} \subseteq Q_{\mathcal{B}_k}$ and

Fig. 29 (a) Freezing. (b) A generalization of freezing. (c) An orbit of \mathcal{A} . Cells are represented by geometric forms and times by patterns. This orbit is periodic of temporal period 3 and spatial period 7. Gray zones are the “clipping zones” of Fig. 29d. They appear here only to make the comparison between the figures easier. (d) Action of the vector $(1, 1)$ on the \mathcal{A} -orbit of Fig. 29c. States of \mathcal{B} , which are in $(Q_4 \cup \{\lambda\})^3$ (λ represents the absence of \mathcal{A} -states) are represented in the following way : state at the bottom, right, is the state of \mathcal{A} , if there was no clipping ; at the bottom, left, is a state of \mathcal{A} moved by $(1, 1)$, that is the state of the left neighbor, one time before in the \mathcal{A} -orbit ; the state at the top, left, is a state of \mathcal{A} moved by $(2, 2)$, that is the state of the second left neighbor two times before in the \mathcal{A} -orbit. (e) A stationary signal $S_{\mathcal{A}^t}$, marked by circled states, in some \mathcal{A} -orbit. (f) Action of the clipping vector $(1, 1)$ on the \mathcal{A} -orbit of Fig. 29e. (g) A signal with a right move $S_{\mathcal{A}^t}$, marked by circled states, in some \mathcal{A} -orbit. (h) Action of the clipping vector (-1) on the \mathcal{A} -orbit of Fig. 29g. Cells of $S_{\mathcal{A}^t}$ are marked by a circle. One or several components of \mathcal{B} may carry states of S .



- On an initial configuration $c_{\mathcal{A}}$, the orbit of which contains one – and only one – site (x_G, t_G) in state G , the evolution of \mathcal{B}_k is described as follows:
 - Let D_G and D_G^* denote the diagonal and the anti-diagonal stemming from site (x_G, t_G) , and let $D_{G,k}$ and $D_{G,k}^*$ denote the diagonal and the anti-diagonal stemming from $(x_G, t_G + k)$.
 - If site (ξ, η) is under or on $D_G \cup D_{G,k}^*$, then $\langle \xi, \eta \rangle_{\mathcal{B}_k} = \langle \xi, \eta \rangle_{\mathcal{A}}$.
 - If site (ξ, η) is strictly above $D_{G,k} \cup D_{G,k}^*$ then $\langle \xi, \eta \rangle_{\mathcal{B}_k} = \langle \xi, \eta - k \rangle_{\mathcal{A}}$.
 - The other sites are used to carry, vertically and during k times, the states of the sites on $D_G \cup D_{G,k}^*$, as well as on D_G' and $D_G'^*$ (translations of D_G and D_G^* by $(0, 1)$). This is done (inside the gray stripes (see [Fig. 29a](#))) by means of new states added to $Q_{\mathcal{A}}$, in such a way that the evolution of \mathcal{A} can normally resume. (One has to consider the diagonals D_G' and $D_G'^*$ because of the dependencies implied by the neighborhood vector.)
2. The former method can be generalized by replacing integer k by some vector \vec{V} of $\mathbb{Z} \times \mathbb{N}$, called the “clipping” vector, which is represented, as in [Sect. 4.1](#), by a finite sequence $(\epsilon_0, \dots, \epsilon_k)$ with $\epsilon_i \in \{-1, 0, +1\}$. Then, for each automaton \mathcal{A} such that $Q_{\mathcal{A}}$ contains a special state G and for each vector $\vec{V} = (\epsilon_0, \dots, \epsilon_k)$ there exists an automaton $\mathcal{B}_{\vec{V}}$ with $Q_{\mathcal{A}} \subseteq Q_{\mathcal{B}_{\vec{V}}}$, such that,
- On an initial configuration $c_{\mathcal{A}}$, the orbit of which contains one – and only one – site (x_G, t_G) in state G , the behavior of $\mathcal{B}_{\vec{V}}$ is as follows:
 - Let D_G and D_G^* denote the diagonal and the anti-diagonal stemming from site (x_G, t_G) , and let $D_{G,\vec{V}}$ and $D_{G,\vec{V}}^*$ be the diagonal and the anti-diagonal stemming from $(x_G + \sharp_1(\vec{V}) - \sharp_{-1}(\vec{V}), t_G + \sharp_0(\vec{V}) + \sharp_1(\vec{V}) + \sharp_{-1}(\vec{V}))$, where $\sharp_\ell(\vec{V})$ is the number of ℓ 's in the sequence $(\epsilon_0, \dots, \epsilon_k)$.
 - If site (ξ, η) is under or on $D_G \cup D_G^*$, then $\langle \xi, \eta \rangle_{\mathcal{B}_{\vec{V}}} = \langle \xi, \eta \rangle_{\mathcal{A}}$.
 - If site (ξ, η) is strictly above $D_{G,\vec{V}} \cup D_{G,\vec{V}}^*$ then $\langle \xi, \eta \rangle_{\mathcal{B}_{\vec{V}}} = \langle \xi - \sharp_1(\vec{V}) + \sharp_{-1}(\vec{V}), \eta - \sharp_0(\vec{V}) - \sharp_1(\vec{V}) - \sharp_{-1}(\vec{V}) \rangle_{\mathcal{A}}$.
 - Otherwhere (inside the gray stripes in [Fig. 29b](#)), the process consists in carrying, according to \vec{V} , the states of $D_G \cup D_G^*$ and $D_G' \cup D_G'^*$, by means of states added to the original states of \mathcal{A} , in such a way that the evolution of \mathcal{A} can normally resume.
- [Figure 29d](#) shows the clipping determined by vector $(1, 1)$. In the left part of the diagram (cells of abscissa $x < x_G$), the states of sites on the diagonal stemming from (x_G, t_G) (the lowest gray site), as well as the states of the diagonal immediately below are moved from $(2, 2)$. In the right part of the diagram, states of sites on the diagonal to the right stemming from (x_G, t_G) as well as states of sites on the diagonal immediately above, are moved by $(2, 2)$. One also observes that the whole part of the \mathcal{A} -orbit, (sites (x, t) , $x > x_G$, $t \geq t_G + 4$, above the gray zone in [Fig. 29c](#)) lays above the gray zone in the \mathcal{B} -orbit shifted by $(2, 2)$ in [Fig. 29d](#).
3. An important particular case is the one in which, for each site (ξ, η) of $D_G \cup D_G'$ (resp. $D_G^* \cup D_G'^*$),

$$\langle \xi + \sharp_1(\vec{V}) - \sharp_{-1}(\vec{V}), \eta + \sharp_{-1}(\vec{V}) + \sharp_0(\vec{V}) + \sharp_1(\vec{V}) \rangle_{\mathcal{A}} = \langle \xi, \eta \rangle_{\mathcal{A}}.$$

Then, $\mathcal{B}_{\vec{V}}$ does not need to carry states of \mathcal{A} between D_G and $D_{G,\vec{V}}$ (resp. D_G' and $D_{G,\vec{V}}'$). So, the evolution of $\mathcal{B}_{\vec{V}}$ on $c_{\mathcal{A}}$ only differs from the evolution of \mathcal{A} on $c_{\mathcal{A}}$ “on the other side.”

4. In case of points 1 and 2, one may allow state G to appear on several sites (even infinitely many times) in the evolution of an initial configuration c_A provided it does not appear in the zone where $\mathcal{B}_{\tilde{V}}$ has to carry states of \mathcal{A} (gray parts in \blacktriangleright Fig. 29a, b).

In case of point 3, sites (x_{G_j}, t_{G_j}) , $j \in \mathbb{N}$, and $t_{G_j} < t_{G_{j+1}}$, where G appears, can be in the zone of the evolution which is not modified. A special case is the one where there is some integer k such that for all $j \in \mathbb{N}^*$, $(x_{G_j}, t_{G_j}) = (x_{G_0} + k, t_{G_0} + k)$, which means that the points (x_{G_j}, t_{G_j}) are regularly placed on D_{G_0} (resp. $(x_{G_j}, t_{G_j}) = (x_{G_0} - k, t_{G_0} + k)$ on $D_{G_0}^*$).

5. In the case of an initial line c_A , the evolution shown state G on one – and only one – site (x_G, t_G) . Assume now that the \mathcal{A} -orbit of c_A contains a signal S_A , defined by a sequence of moves η_0, \dots , which meets D_G and $D_{G, \tilde{V}}$ but does not enter the zone between D_G^* and $D_{G, \tilde{V}}^*$ (resp. meets D_G^* and $D_{G, \tilde{V}}^*$, but does not enter the zone between D_G and $D_{G, \tilde{V}}$). Processes of points 1 and 2 applied to c_A provides on the corresponding orbit of $\mathcal{B}_{\tilde{V}}$ a signal that coincides with S_A outside the band determined by D_G and $D_{G, \tilde{V}}$ (resp. D_G^* and $D_{G, \tilde{V}}^*$) and that follows \tilde{V} inside the corresponding diagonals. Of course, carrying the necessary data onto the diagonal on which the action of \mathcal{A} resumes requires new states. To give two examples:

- (a) Action of a clipping vector $(1, 1, \dots, 1)$ (sequence of k moves by 1) on a stationary original signal.

\blacktriangleright Figure 29f shows the result of the clipping vector $(1, 1)$ on an \mathcal{A} -orbit carrying a signal S_A (see \blacktriangleright Fig. 29e), which is transformed into a signal S^* on the corresponding $\mathcal{B}_{(1,1)}$ -orbit. Actually, S^* is S_A in which two moves to the right have been inserted in the clipping zone. More generally, one observes that, if the sequence of moves of S_A is $\eta_0, \dots, \eta_{j_0}, \eta_{j_0+1}, \dots$ and if η_{j_0} is the last move of S_A before reaching D_G , then the sequence of moves of S^* is $\eta_0, \dots, \eta_{j_0}, 1, 1, \dots, 1, \eta_{j_0+1}, \dots$, where the number of inserted 1's is k .

- (b) Action of a clipping vector (-1) on a move to the right followed by a stationary move.

Consider the situation where the signal S_A arrives on the diagonal D_G^* on site (x, t) , coming from site $(x - 1, t)$ and going to site $(x, t + 1)$. Then the crippling effect of vector -1 will be to transform site (x, t) of \mathcal{A} into site $(x - 1, t + 1)$ of \mathcal{B}_{-1} on the diagonal D_G^* and site $(x, t + 1)$ of \mathcal{A} into $(x - 1, t + 2)$ of \mathcal{B}_{-1} . The result is that a move to the right has been replaced by two stationary moves. \blacktriangleright Figures 29g, h illustrate this point. In this case, the clipping allows the choice between two signals: an indentation to the left or a straight vertical line (the choice in what follows).

6. Let it be assumed now that the signal S_A is periodic of period $(\eta_0, \dots, \eta_\ell)$. Then, by suitably implementing points 4 and 5, one is able to modify the period, and one gets the following result.

Proposition 9 *Let \mathcal{A} be a cellular automaton whose set of states Q_A contains a special state G and a subset S . Let p and q be in \mathbb{N}^* , and let Π be a sequence $(\epsilon_0, \dots, \epsilon_{\Pi-1})$, $\epsilon_i \in \{0, 1\}$. Then, a cellular automaton $\mathcal{B}_{G, S, \Pi, p, q}$ exists such that $Q_A \subseteq Q_{\mathcal{B}_{G, S, \Pi, p, q}}$ and*

- *On each \mathcal{A} -configuration c_A the evolution of which has exactly one site (x_G, t_G) in state G and a signal S with states in S , ultimately periodic with period Π and ultimately above the diagonal D_G stemming from (x_G, t_G) .*
- *The evolution of $\mathcal{B}_{G, S, \Pi, p, q}$ contains a signal S^* with $S \subseteq S^*$, of slope $\frac{p+q}{q}$.*

Proof If the period of \mathcal{S} is (0) , on the anti-diagonal D_G^* stemming from (x_G, t_G) , starting from a suitable site, a clipping by vector $\mathbf{1}$ is launched each two sites. Then one gets a new signal of period $(1, 0)$, see point 5a. If \mathcal{S} has period (1) , on the diagonal D_G stemming from (x_G, t_G) , starting from a suitable site, a clipping by vector $-\mathbf{1}$ is launched each two sites. Then one gets a new signal of period $(0, 0)$, see point 5b. By changing the starting point of the period, one may assume that $\epsilon_0 = 1$ and $\epsilon_1 = 0$. By regularly launching clippings by vector $-\mathbf{1}$ from the diagonal D_G , one may “transform” all ϵ_i of value 1 into 00 . One then gets a signal of period $(1, \underbrace{0, \dots, 0}_{\alpha \text{ times}}, 0)$. One also may assume that α is even using a new clipping by vector $\mathbf{1}$ launched from D_G^* .

Now, in launching from D_G^* a clipping by vector $(\underbrace{1, \dots, 1}_{\ell \text{ times}}, 0, \dots, 0)$, one gets a signal of period $(\underbrace{1, \dots, 1}_{\ell+1 \text{ times}}, \underbrace{0, \dots, 0}_{\alpha \text{ times}}, 0, \dots, 0)$. Finally one regularly launches k clippings by vector $-\mathbf{1}$ from D_G in order to get a period $(\underbrace{1, \dots, 1}_{\ell+1-k \text{ times}}, \underbrace{0, \dots, 0}_{\alpha+2k \text{ times}}, 0, \dots, 0)$.

One can choose ℓ , k and β such that $\ell + 1 - k = \beta p$ and $\alpha + 2k = \beta q$. As α is even, let us say $\alpha = 2\alpha^*$, one can use $\beta = 2\alpha^*$, $k = \alpha^*(q - 1)$ and $\ell = \alpha^*(2p + q - 1) - 1$. In this way one gets a new signal, the slope of which is ultimately $\frac{p+q}{q}$.

It has to be noted that, at each main step of transformation, a new automaton is built.

7 Conclusion

Some methods in algorithmics on one-dimensional cellular automata have been shown. Other methods also exist, in particular, methods that are based on different possibilities of compressing states of some cellular automaton (see Mazoyer (1992) and Heen (1996)). Different methods have arisen in the course of studies of specific questions such as, for example, the synchronization of a firing squad, the recognition or computing power of cellular automata (see the chapter [Language Recognition by Cellular Automata](#)) or the computation of functions as in Fisher (1965) (see also the chapter [Computations on Cellular Automata](#)). But few general studies exist except for that by Delorme and Mazoyer (2002b). And if one chooses the geometric point of view, as in [Sect. 3](#), a lot remains to be done.

In the case of two-dimensional cellular automata, things are still more complicated because it is very difficult to draw space-time diagrams. The notion of a signal may be generalized in two ways. Either signals can be viewed as surfaces (see, e.g., Delorme and Mazoyer (2002a, 2004)), or they can be thought of as lines (see, e.g., Delorme and Mazoyer (1999)). Some methods of dimension 1 can be used. The method that can be found in Terrier (2006) or the synchronization method of Balzer (1967) is mentioned, which is extended to the plane in Szwierinski (1982). Of course other specific methods exist (very close to discrete geometry), such as “thinness” Romani (1976).

The notion of a signal is a core piece of the tools presented in this chapter. But algorithms without signals also exist, and these are minimally investigated. A characteristic example is the proof that XOR synchronizes each exponential line (Kari 1994; Yunès 2008).

References

- Balzer R (1966) Studies concerning minimal time solutions to the firing squad synchronization problem. Ph.D. thesis, Carnegie Institute of Technology
- Balzer R (1967) An 8-states minimal time solution to the firing squad synchronization problem. *Inform Control* 10:22–42
- Čulík K (1989) Variations of the firing squad synchronization problem. *Inform Process Lett* 30:153–157
- Čulík K, Dube S (1993) L-systems and mutually recursive function systems. *Acta Inform* 30:279–302
- Čulík K, Karhumäki J (1983) On the Ehrenfeucht conjecture for DOL languages. *ITA* 17(3):205–230
- Delorme M, Mazoyer J (1996) Languages recognition and cellular automata. In: Almeida J, Gomez GMS, Silva PV (eds) *World Scientific*, Singapore, pp 85–100
- Delorme M, Mazoyer J (2002a) Reconnaissance parallèle des langages rationnels sur automates cellulaires plans. *Theor Comput Sci* 281:251–289
- Delorme M, Mazoyer J (2002b) Signals on cellular automata. In: Adamatzky A (ed) *Springer*, London, pp 231–274
- Delorme M, Mazoyer J (2004) Real-time recognition of languages on a two-dimensional archimedean thread. *Theor Comput Sci* 322(2):335–354
- Delorme M, Mazoyer J, Tougne L (1999) Discrete parabolas and circles on 2D cellular automata. *Theor Comput Sci* 218(2):347–417
- Fisher PC (1965) Generation of primes by a one dimensional real time iterative array. *J ACM* 12:388–394
- Gruska J, Salvatore L, Torre M, Parente N (2006) Different time solutions for the firing squad synchronization problem. *Theor Inform Appl* 40(2):177–206
- Heen O (1996) *Economie de ressources sur automates cellulaires*. Ph.D. thesis, Université Paris Diderot. In French
- Kari J (1994) Rice's theorem for limit sets of cellular automata. *Theor Comp Sci* 127(2):227–254
- Mazoyer J (1987) A six states minimal time solution to the firing squad synchronization problem. *Theor Comput Sci* 50:183–238
- Mazoyer J (1989a) An overview on the firing squad synchronization problem. In: Choffrut C (ed) *Automata networks, Lecture notes in computer science*. Springer, Heidelberg
- Mazoyer J (1989b) Solutions au problème de la synchronisation d'une ligne de fusiliers. *Habilitation à diriger des recherches* (1989). In French
- Mazoyer J (1992) Entrées et sorties sur lignes d'automates. In: Cosnard MNM, Robert Y (eds) *Algorithmique parallèle*, Masson, Paris, pp 47–64. In French
- Mazoyer J (1999) Computations on cellular automata. In: Delorme M, Mazoyer J (eds) *Springer-Verlag*, London, pp 77–118
- Mazoyer J, Reimen N (1992) A linear speed-up theorem for cellular automata. *Theor Comput Sci* 101:59–98
- Mazoyer J, Terrier V (1999) Signals in one-dimensional cellular automata. *Theor Comput Sci* 217(1):53–80
- Ollinger N (2002) Automates cellulaires: structures. Ph.D. thesis, Ecole Normale Supérieure de Lyon. In French
- Rapaport I (1998) Ordre induit sur les automates cellulaires par l'opération de regroupement. Ph.D. thesis, Ecole Normale Supérieure de Lyon
- Richard G (2008) Systèmes de particules et collisions discrètes dans les automates cellulaires. Ph.D. thesis, Aix-Marseille Université. In French
- Romani F (1976) Cellular automata synchronization. *Inform Sci* 10:299–318
- Sutner K (1997) Linear cellular automata and Fisher automaton. *Parallel Comput* 23(11):1613–1634
- Szwierinski H (1982) Time optimal solution of the firing squad synchronization problem for n-dimensional rectangles with the general at any position. *Theor Comput Sci* 19:305–320
- Terrier V (1991) Temps réel sur automates cellulaires. Ph.D. thesis, Université Lyon 1. In French
- Terrier V (2006) Closure properties of cellular automata. *Theor Comput Sci* 352(1):97–107
- Theysier G (2005) Automates cellulaires: un modèle de complexité. Ph.D. thesis, Ecole Normale Supérieure de Lyon. In French
- Ulam S (1957) *The Scottish book: a collection of problems*. Los Alamos
- Vaskhavsky VI, Marakhovsky VB, Peschansky VA (1969) Synchronization of interacting automata. *Math Syst Theory* 14:212–230
- von Neumann J (1966) *Theory of self-reproducing*. University of Illinois Press, Urbana, IL
- Waksman A (1996) An optimum solution to the firing squad synchronization problem. *Inform Control* 9:66–78
- Yunès JB (2008) Goto's construction and Pascal's triangle: new insights in cellular automata. In: Proceeding of JAC08. Uzès, France, pp 195–202

4 Language Recognition by Cellular Automata

Véronique Terrier

GREYC, UMR CNRS 6072, Université de Caen, France
veroniqu@info.unicaen.fr

1	Preliminaries	124
2	Definitions and Examples	124
3	Positive Results and Simulation	136
4	Limitations	144
5	Comparison with Other Models	152
6	Questions	154

Abstract

Cellular automata (CA) comprise a simple and well-formalized model of massively parallel computation, which is known to be capable of universal computation. Because of their parallel behavior, CA have rich abilities of information processing; however, it is not easy to define their power limitations. A convenient approach to characterizing the computation capacity of CA is to investigate their complexity classes. This chapter discusses the CA complexity classes and their relationships with other models of computations.

1 Preliminaries

Cellular automata (CA) provide an ideal framework for studying massively parallel computation. Their description is very simple and homogeneous; however, as emphasized by various examples, they allow one to distribute and synchronize the information in a very efficient way. Their ability to do fast computation raised complexity issues. Early works drew much attention to CA as language recognizers.

Obviously, CA are of the same computational power as Turing machines. Hence the major motivation for CA complexity study is to gain knowledge on the way parallelism acts and to explicitly explain the gain that may be achieved with CA. In particular, much interest is devoted to their low-time complexity classes because they may provide significant complexity benefits. But, as for the other models of computation, it is not a simple task to deeply evaluate their power and their limitations.

In this chapter, the essential developments on CA in the field of language recognition are reviewed. The purpose is to give insight into the performance of different types of CA, with a major focus on the low-time complexity classes. Of course, it will not be possible to report every outcome in detail and some choices have been made. First, only space-bounded computations are discussed. Specifically, because of the interest in fast computation, one will just consider space bound defined as the space consumed by low time computation. And, despite the interest in general complexity issues, we decided not to include non-deterministic or alternating variants of CA. As a matter of fact, they are beyond the scope of realistic devices, because even their minimal time classes contain *NP*-complete problems. For the same reasons, CA with tree-like array structures shall be ignored.

The rest of this chapter is organized as follows. [Section 2](#) introduces the different variants of CA recognizers and their complexity classes. [Section 3](#) reviews the known inclusions and equalities among these complexity classes. [Section 4](#) recalls the limitations established on the recognition power of CA. [Section 5](#) relates to the comparison with other models of computation. As a conclusion, [Sect. 6](#) discusses some of the old open questions that remain up to now unsolved.

2 Definitions and Examples

Basically, a CA is a regular array of cells. These cells range over a finite set of states and evolve synchronously at discrete time steps. At each step, the new state of each cell is produced by a local transition rule according to the current states in its neighborhood. So a CA is completely

specified by a tuple $(d, S, \mathcal{N}, \delta)$ where d is the dimension of the array of cells indexed by \mathbb{Z}^d , S is the finite set of states, the neighborhood \mathcal{N} is a finite ordered subset of \mathbb{Z}^d , and δ is the transition function from $S^{|\mathcal{N}|}$ into S .

A cell is denoted by \mathbf{c} , $\mathbf{c} \in \mathbb{Z}^k$; a site (\mathbf{c}, t) denotes the cell \mathbf{c} at time t and $\langle \mathbf{c}, t \rangle$ denotes its state. And at each step $t \geq 0$, the state is updated in this way: $\langle \mathbf{c}, t+1 \rangle = \delta(\langle \mathbf{c} + \mathbf{v}_1, t \rangle, \dots, \langle \mathbf{c} + \mathbf{v}_r, t \rangle : \mathcal{N} = \{\mathbf{v}_1, \dots, \mathbf{v}_r\})$. Depending on the neighborhood, the flow of information may go in all directions of the cellular array as with the von Neumann neighborhood $\{\mathbf{v} \in \mathbb{Z}^k : \Sigma |v_i| \leq 1\}$ or with the Moore one $\{\mathbf{v} \in \mathbb{Z}^k : \max |v_i| \leq 1\}$ or it may be restricted to one-way in some directions as with the one-way von Neumann neighborhood $\{-\mathbf{v} : \mathbf{v} \in \mathbb{N}^k \text{ and } \Sigma v_i \leq 1\}$ or with the one-way Moore one $\{-\mathbf{v} : \mathbf{v} \in \mathbb{N}^k \text{ and } \max v_i \leq 1\}$. In what follows, one-way communication will refer to such restrictions whereas two-way communication will refer to the ability to transmit information everywhere in the cellular array.

In order for a CA to act as a language recognizer, one has to specify how the input is given to the cellular array and how the result of the computation is obtained.

The Input Mode

For the sake of simplicity, it is assumed that Σ the finite alphabet of input letters is a subset of the states set S of the CA. A quiescent state λ such that $\delta(\lambda, \dots, \lambda) = \lambda$ is also considered. Two modes are distinguished to give the input to the array: the parallel one and the sequential one. In the *parallel mode*, the whole input is supplied at initial time to the array. It implies an implicit synchronization at the beginning of the computation. All input symbols are fed into a distinct cell and are arranged in such a way as to keep the input structure. In the *sequential mode*, a specific cell is chosen to receive the input. All cells are initially quiescent and the input symbols are fed serially to the specific cell. When the whole input has been read by this input cell, an end-marker symbol $\$$ is infinitely fed to it. Because the input cell takes into account not only its neighborhood but also its received input symbol, it behaves in a particular way.

The Output Mode

The advantage of recognition problems is that the output is of yes/no type. Hence, on a CA, a specific cell is enough to indicate acceptance or rejection of the input. The choice of this output cell is arbitrary in the case of two-way communication. But, in the case of one-way communication, it is subject to constraint: the output cell must be able to get all information from the input.

Language Recognition

For the purpose of recognition, two subsets of the states set S are specified: the set S_{acc} of accepting states and the set S_{rej} of rejecting states. A language L is said to be *recognized* by a CA, if on input w the output cell enters at some time t_e an accepting state if $w \in L$ or a rejecting state if $w \notin L$; and for all time $t < t_e$ the output cell is neither in an accepting state nor in a rejecting state. A time complexity T refers to a function relating the input size to an amount of time steps. A CA recognizer works in time T , if it accepts or rejects each word w of size s within $T(s)$ steps.

Usually, the focus is on acceptance. In this case, only the set S_{acc} of accepting states is specified. A language L is said to be *accepted* by a CA, if on input w the output cell enters an accepting state if and only if $w \in L$. A CA acceptor works in time T , if it accepts each word $w \in L$ of size s within $T(s)$ steps.

In fact, the notion of acceptance and recognition turns out to be equivalent for current time complexities. Indeed, a CA, on an input of size s , is able to distinguish the output cell at

time $T(s)$ when T is any standard time complexity. Hence, the CA can reject at time $T(s)$ all non-accepted words of size s . So, except just below, one will not differentiate between CA recognizer and CA acceptor.

An example of a CA recognizer is now given.

Example 1 The majority language that consists of the words over the alphabet $\{a, b\}$ in which there are strictly more a 's than b 's is recognized by a one-dimensional CA with parallel input mode and one-way neighborhood $\{-1, 0\}$.

The CA is defined in this way. $\Sigma = \{a, b\}$ is the input alphabet, $S = \{a, b, A, B, 1, 2, n, x\}$ is the states set, $S_{acc} = \{A\}$ is the set of accepting states, $S_{rej} = \{B, x\}$ is the set of rejecting states, the cell -1 is assumed to remain in a persistent state $\#$, $\delta : S \cup \{\#\} \times S \rightarrow S$ is the transition function displayed below.

δ	a	b	A	B	1	2	n	x
a	a	b	A		a	n	A	
b	a	b		B	n	b	B	
A	a		A		a	n	A	
B		b		B	n	b	B	
1	1	2			1	2		
2	1	2			1	2		
n					1	2	n	
x	A	B	x	x	A	B	x	x
$\#$	1	2	x	x	A	B		x

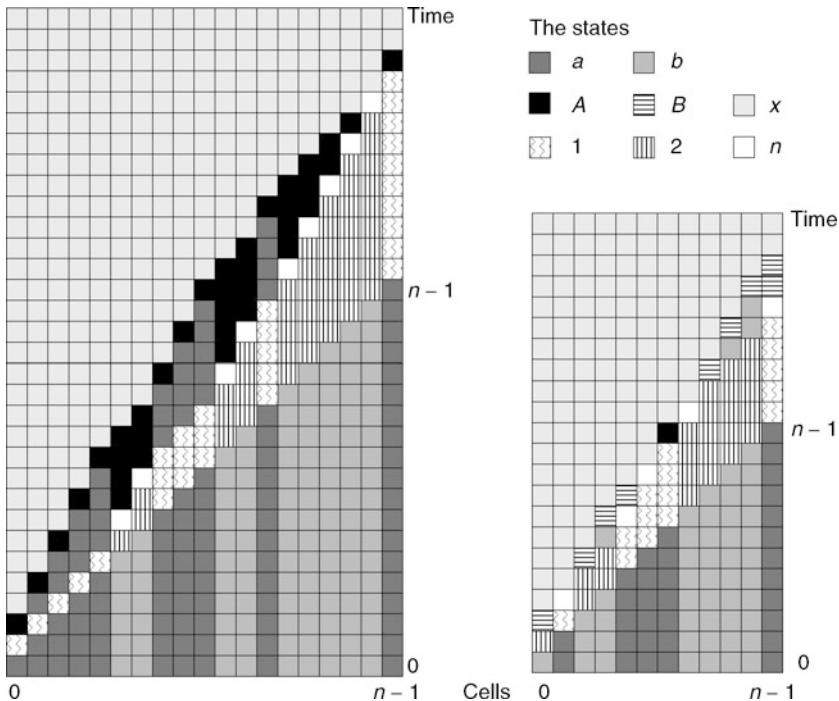
Since the neighborhood is one-way, the output cell is chosen to be the rightmost one. The computations on inputs $w_1 = aaaaabbaaabbbbbbba$ and $w_2 = babbaaabbbba$ are depicted in Fig. 1. As the input mode is parallel, the words are supplied at initial time. On each cell, above the diagonal, the length of the sequence of a 's and A 's or, b 's and B 's, records the difference between the numbers of a 's and b 's of the input. The input word w_1 is accepted since the output cell $n - 1$ enters the accepting state A , whereas the input word w_2 is rejected since it enters the rejecting state B . The CA recognizer works within time complexity $T(n) = 2n$. As a matter of fact, the cell 0 knows at time 1 that it is the leftmost one; so by mean of a signal, which moves one cell to the right every two steps, it is simple to mark the output cell $n - 1$ at time $2n$. Thus the status of the words not accepted may be decided at this time even though the set of rejecting states is not specified. More generally, notice that, on an input of length n , the output cell is able to know the whole input at time $n - 1$, whereas it is able to know that the input is completed at time n .

Time Complexities

Among the time complexities, two functions are of major interest: the real time and the linear time. The *real-time* complexity, denoted by rt , means that each word is accepted or rejected "as soon as possible." Here, only a slight difference between acceptance and recognition is

Fig. 1

Recognition of the majority language.



encountered. For acceptance, the real-time complexity corresponds to the minimal time for the output cell to read the whole input, whereas for recognition, one additional unit of time may be required in order that the output cell knows that the input is completed. In the sequel, we only deal with real-time acceptance. Practically, the real-time complexity is specified by the way the input is supplied, the choices of the output site and the neighborhood. The *linear-time* complexity, denoted by lt , is just defined as the real-time function multiplied by any constant strictly greater than 1.

Space-Bounded Computation

In this chapter, one will restrict oneself to space-bounded computation. During all the computation, only a fixed number of cells, depending on the size of the input, are active. All other cells remain in a persistent state $\#$ from the start to the end. One may imagine many space bounds. However, in practice, the bounded space is *uniquely* defined as the space required to perform real-time computation. It means that the active part of the array is identical whatever the effective time complexity may be. Then, in the following, all classes defined in terms of time complexity classes will be actually both time and space bounded. As a matter of fact, space-bounded and unbounded computations become the same for real-time and linear-time complexities. On the other hand, because the space is bounded, the evolution becomes periodical after a number of steps that is exponential in the size of the space. That establishes an upper bound on the relevant time complexities.

The Dependency Graph

In order to reflect the neighborhood constraints on the sites involved in the computation, one will consider the graph induced by the dependencies between them. Precisely, a dependency graph is defined according to a given type of CA, a fixed time complexity, and the input size. It is a directed graph. Its set of vertices consists of the sites, which both are influenced by the input and can have an effect on the output site—in other words, all relevant sites regarding the computation. Its edges are the couples of sites $((\mathbf{c} + \mathbf{v}, t), (\mathbf{c}, t + 1))$, for all \mathbf{v} belonging to the neighborhood \mathcal{N} .

The different types of CA recognizers are described in the following subsections. Their features are stated by the array dimension, the input dimension and, in the parallel mode, the way to place the input into the array.

2.1 One-Dimensional CA Language Recognizer

A one-dimensional CA recognizer is structured as a linear array and operates on words. The space is assumed to be linearly bounded: the number of active cells equals the length of the input. In practice, the active cells are numbered $0, \dots, n - 1$, for an input of length n . Because the other cells always remain in the persistent state $\#$, one-dimensional CA with two-way communication have the same computational power as Turing machines working in $O(n)$ space.

Different variants of one-dimensional CA are currently defined. On the one hand, it depends on the neighborhood, which allows either two-way or one-way communication. Actually, from Poupet (2005), it is known that there are somewhat only two kinds of neighborhoods, the two-way one $\{-1, 0, 1\}$ and the one-way one $\{-1, 0\}$. On the other hand, it depends on the input mode being either parallel or sequential. Hence four variants of one-dimensional CA are characterized according to the neighborhood and input mode choices. They are named PCA, SCA, POCA, and SOCA. The first letter “P” or “S” stands for parallel or sequential input mode and the “O” occurrence makes distinctions between one-way or two-way communication (☞ [Table 1](#)).

■ **Table 1**

The four variants of CA in dimension one

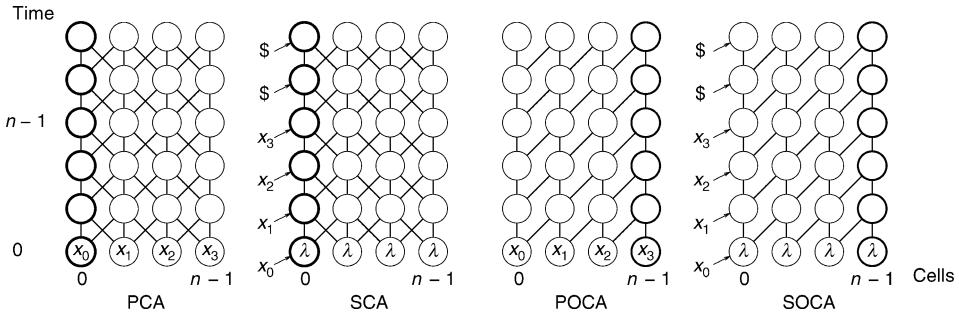
Automata	Neighborhood	Input mode	Output cell
PCA	$\{-1, 0, 1\}$	Parallel	0
SCA	$\{-1, 0, 1\}$	Sequential	0
POCA	$\{-1, 0\}$	Parallel	$n - 1$
SOCA	$\{-1, 0\}$	Sequential	$n - 1$

Different denominations have been used by other authors: SCA and SOCA are more often called iterative array (IA) and one-way iterative array (OIA); POCA are also called one-way cellular automata (OCA) or mentioned as trellis automata.

Let one describe the way to carry out the input in the two modes. In the parallel input mode, at initial time 0, the i th symbol of the input $w = x_0 \dots x_{n-1}$ of size n , is fed to the cell i : $\langle i, 0 \rangle = x_i$. In the sequential input mode, all active cells are initially quiescent (i.e., set to λ). The cell indexed by 0 is chosen to read the input: it gets the i th symbol x_i of the input $w = x_0 \dots x_{n-1}$.

Fig. 2

The space-time diagram of the four one-dimensional CA variants.



at time i ; then it gets an end-marker $\$$ at all time $t \geq n$. Note that the input cell 0 evolves according to a particular transition function $\delta_{\text{init}}: (\Sigma \cup \{\$\}) \times S^{|\mathcal{N}|} \rightarrow S$, so $\langle 0, t \rangle = \delta_{\text{init}}(x_p, \langle v_1, t-1 \rangle, \dots, \langle v_r, t-1 \rangle)$ where $\mathcal{N} = \{v_1, \dots, v_r\}$ refers to the neighborhood and $x_t = \$$ when $t \geq n$.

For the output cell that yields the result, the initial cell 0 in case of PCA and SCA that use two-way communication is chosen. The unique possibility for POCA and SOCA is the rightmost cell indexed $n-1$ since their neighborhood $\{-1, 0\}$ is one-way.

Figure 2 above shows the customary representation of the four models.

Some notation is useful for dealing with the various complexity classes of one-dimensional CA recognizers. For a time complexity function T from \mathbb{N} into \mathbb{N} , $\text{PCA}(T)$ (resp., $\text{SCA}(T)$, $\text{POCA}(T)$, and $\text{SOCA}(T)$) will denote the class of languages recognizable in time T by PCA (resp., SCA, POCA, and SOCA). With some liberty, PCA will refer to the class of languages recognized in unbounded time by PCA; and in the same way, SCA, POCA, and SOCA will indicate the device type as well their corresponding unbounded time complexity classes.

Particular attention is devoted to the low-time complexity classes, namely the real time and the linear time. The real-time complexity $rt(n)$ is defined as the earliest time for the output cell to read the whole input of length n . More precisely, it corresponds to $n-1$ in case of PCA, SCA, and POCA and to $2n-2$ in case of SOCA. And $lt(n) = \tau rt(n)$, where τ is any constant strictly greater than 1, gives rise to linear-time complexity. In the following, the classes of language recognized in real time by PCA, SCA, POCA, and SOCA will be denoted by RPCA , RSCA , RPOCA , and RSOCA . For linear-time complexity, the corresponding classes will be designated by LPCA , LSCA , LPOCA , and LSOCA .

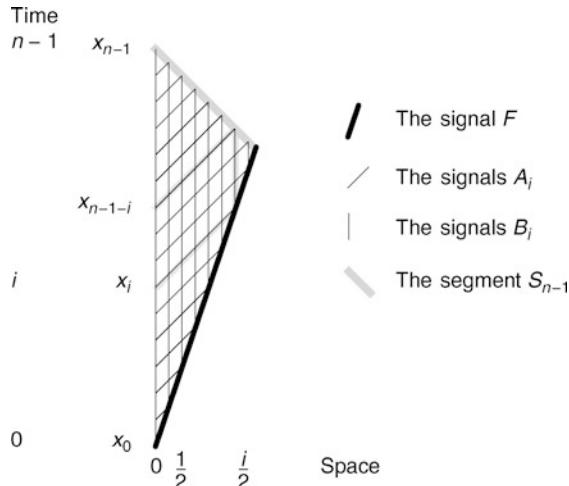
Some examples are now given to illustrate the computation ability of the real-time complexity classes.

Example 2 (Cole 1969) The palindrome language $\{w \in \Sigma^*: w = w^R\}$ is a real-time SCA language (w^R denotes w read backward).

The algorithm is described in a geometrical way. Its discretization to obtain the corresponding RSCA is straightforward. First of all, the input word received sequentially on the input cell 0 is sent at maximal speed to the right. Precisely the symbol x_p , which is fed on cell $c=0$ at time $t=i$ follows the line A_i of equation $t=c+i$ with $c \geq 0$. Simultaneously, a signal F of speed $1/3$ starts from the input cell 0 at initial time and draws the line $t=3c$. So the signals A_i and F

Fig. 3

Real-time recognition of the palindrome language by a SCA.



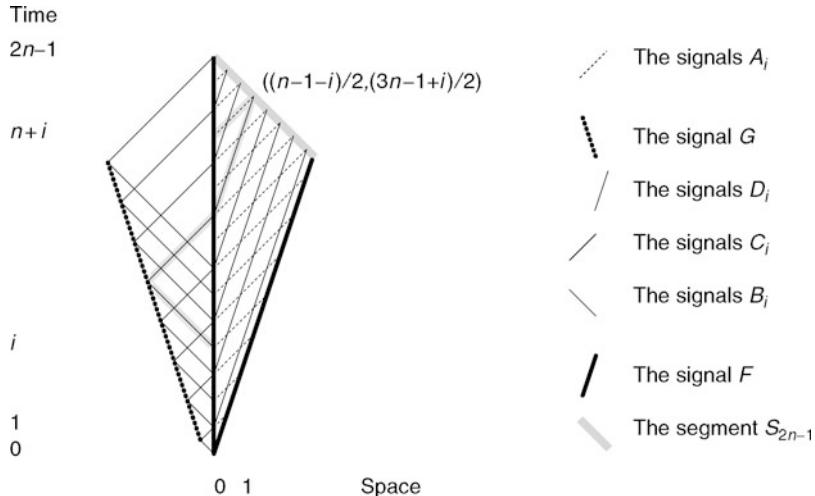
intersect on the point $(i/2, 3i/2)$. From this intersection, the symbol x_i carried by A_i goes further along the vertical line B_i : $c = i/2$ with $t \geq 3i/2$. In this way, the symbols x_j and x_i meet on the point $(i/2, j + i/2)$ where the signals A_j and B_i intersect. In particular, for any integer k , on the segment S_k : $c + t = k$ with $0 \leq c \leq k/4$, which runs at maximal speed to the left from the signal F to the output cell 0, one may compare all pairs of symbols $\{x_i, x_{k-i}\}$ with i such that $0 \leq i \leq k/2$. Now, on an input $x_0 \dots x_{n-1}$ of length n , the sequence of comparisons between x_i and x_{n-1-i} with $0 \leq i \leq (n-1)/2$ determines whether the input is a palindrome or not. This sequence of comparisons is exactly the one that occurs on the segment S_{n-1} : $c + t = n - 1$. Finally, observe that the segment S_{n-1} reaches the cell $c = 0$ at time $t = n - 1$. In other words, the result is obtained in real time (Fig. 3).

Example 3 (Cole 1969) *The square language $\{ww : w \in \Sigma^*\}$ is a real-time SCA language.*

On the one hand, each input symbol x_i is carried along the line A_i of equation $t = c + i$ with $c \geq 0$. On the other hand, each input symbol x_i is first sent at maximal speed to the left, following the signal B_i of equation $t = -c + i$ with $c \geq 0$. Simultaneously, initialized by the input cell c at time 0, a signal G of equation $t = -3c - 1$ starts from the point $(-1/2, 1/2)$ and moves with speed $1/3$ to the left. So the signals B_i and G intersect on the point $((-i+1)/2, (3i+1)/2)$. From this intersection, the symbol x_i carried by B_i goes further along the signal C_i : $t = c + 1 + 2i$ with $t \geq (3i+1)/2$. Then the signal C_i intersects the initial cell $c = 0$ on the point $(0, 1+2i)$. From this intersection, the symbol x_i follows the signal D_i : $t = 3c + 1 + 2i$. In this way, for any i, j with $i < j$, the symbols x_i and x_j meet at the point $((j-1)/2 - i, (3j-1)/2 - i)$ where the signals D_i and A_j intersect. Now, on an input $x_0 \dots x_{2n-1}$ of even length $2n$, the sequence of comparisons between x_i and x_{n+i} with $0 \leq i < n$ determines whether the input is a square word or not. These comparisons are performed on the points $((n-1-i)/2, (3n-1+i)/2)$. Hence they occur on the segment S_{2n-1} : $t + c = 2n - 1$ with $0 \leq c \leq (2n-1)/4$, which starts from the signal F : $t = 3c$, moves at maximal speed to the left and reaches the cell 0 at time $2n - 1$. Therefore the result is obtained in

Fig. 4

Real-time recognition of the square language by a SCA.



minimal time. With regard to space, note that negative cells are not necessarily active, since the negative side can be folded on the positive side, that is, all activity on the negative cells can be as well performed on the positive side (☞ Fig. 4).

Example 4 (Smith 1972) *The Dyck language is a real time POCA language.*

A RPOCA that recognizes the Dyck language over the alphabet $\{a, b\}$ can be defined in this way.

$\Sigma = \{a, b\}$ is the input alphabet, $S = \{a, b, d, o\}$ is the set of states, $S_{acc} = \{d\}$ is the set of accepting states and $\delta : S^2 \rightarrow S$ is the transition function displayed below.

δ	a	b	d	o
a	a	d	a	a
b	o	b		o
d		b	a	
o	o	b	b	o

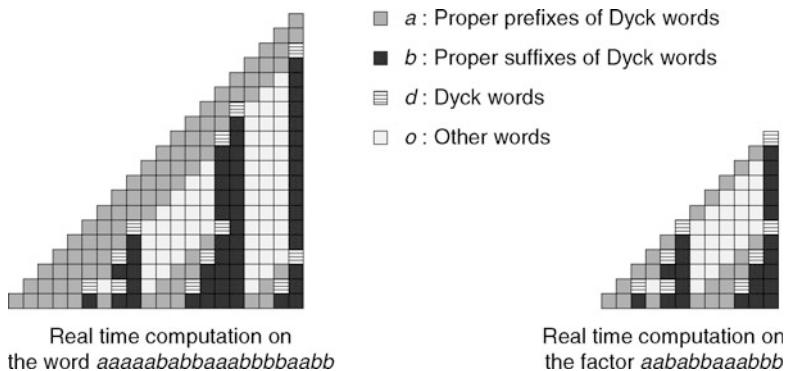
A significant feature of POCA is that the real-time computation on an input w contains the real-time computations of all its factors. An illustration is depicted in ☞ Fig. 5.

Example 5 (Culik 1989) *The language $L = \{a^i b^{i+j} a^j : i, j \in \mathbb{N}\}$ is a real-time POCA language.*

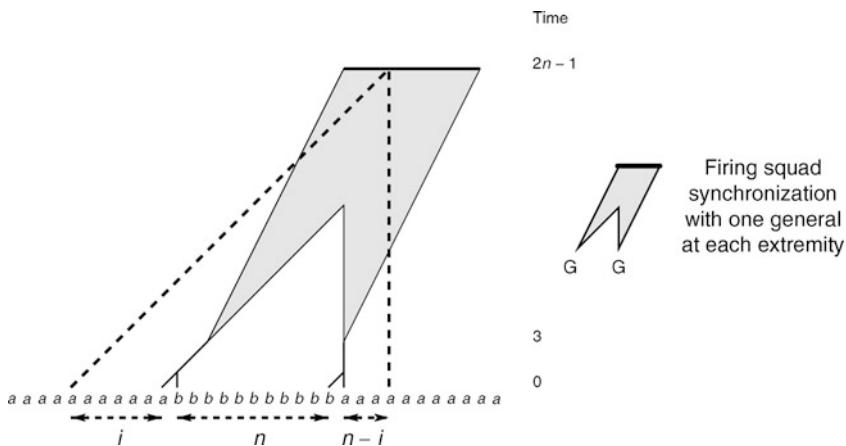
First, the RPOCA will reject all the words outside the regular language $\{a^i b^j a^k : i, j, k \in \mathbb{N}\}$. Second, notice that the factors of a word of shape $a^* b^n a^*$ that belong to L are on the one hand

Fig. 5

Recognition of the Dyck language by a RPOCA.

**Fig. 6**

Real-time recognition of the Culik language by POCA.



the words $a^i b^i$ and $b^i a^i$, which will be simply accepted by the RPOCA, and on the other hand the words $a^i b^n a^{n-i}$. Observe that to accept, in real time, all the factors $a^i b^n a^{n-i}$ with $i = 1, \dots, n-1$ of an input of shape $a^* b^n a^*$, means that $n-1$ consecutive cells simultaneously enter at time $2n-1$ and for the first time in an accepting state. In other words, a synchronization process is required to recognize this language. As sketched in **Fig. 6**, the synchronization can be set up using a firing squad synchronization process with two generals located respectively according to the boundary between the a 's and b 's and the boundary between the b 's and a 's.

2.2 Multidimensional CA Language Recognizer

Natural extensions to higher dimensional arrays have been early investigated in (Cole 1969; Chang et al. 1987; Ibarra and Palis 1988). In this framework, the space increases with the

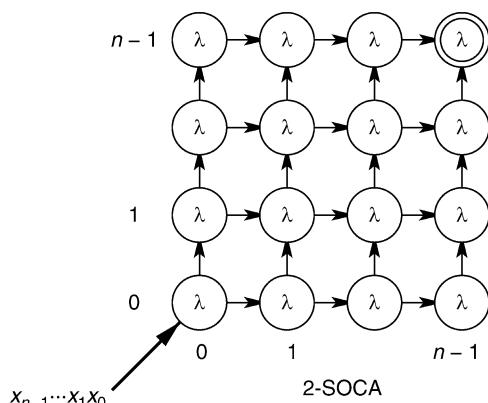
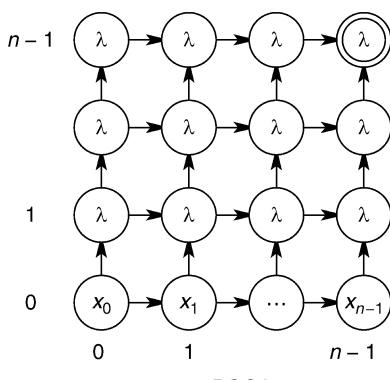
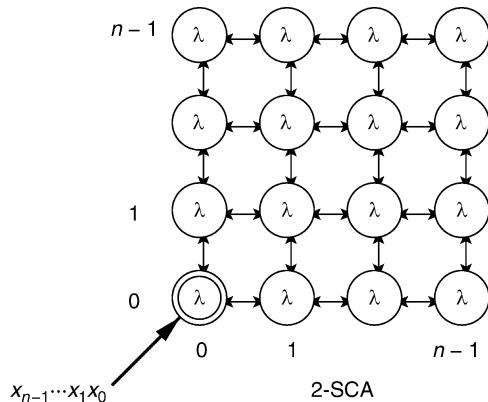
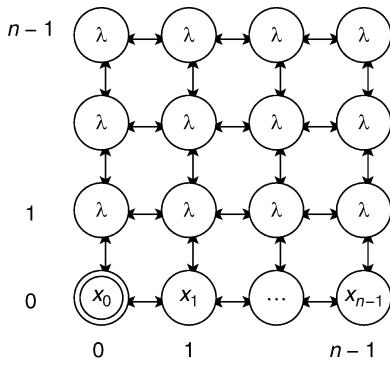
dimension, while the minimal time complexity remains linearly bounded by the length of the input word. Precisely, on an input of length n , the space of a d -dimensional CA recognizer is defined as a d -dimensional array of size n in each dimension. Hence, a d -CA with two-way neighborhoods has the same computation ability as Turing machines working in $O(n^d)$ space.

The diversity of neighborhoods also rises with the dimension. In the case of two-way neighborhoods and sequential input mode, it is known from Cole (1969) that the computation ability is preserved even though the neighborhood is restricted to the von Neumann one: $\{\mathbf{v} \in \mathbb{Z}^d : \sum |v_i| \leq 1\}$. Whether the same is true or not for the other variants is unclear. However, the impact of the neighborhood choice appears less crucial for language recognizers where almost all cells are initially quiescent than for picture language recognizers that will be defined below. Hence, as neighborhood issues are currently studied in this last context, we shall only regard the von Neumann neighborhood and its one-way counterpart $\{-\mathbf{v} : \mathbf{v} \in \mathbb{N}^d \text{ and } \sum v_i \leq 1\}$ in the case of a multidimensional CA language recognizer.

For an input of length n , the space area consists of the cells $\{\mathbf{c} : 0 \leq c_1, \dots, c_d < n\}$. The two-dimensional case is depicted in Fig. 7. The cells outside this area remain in a persistent

Fig. 7

The space array of the four 2-CA variants.



state $\#$ during all the computation and each cell inside the area is assumed to remain quiescent until the step when it may be affected by the input. In the parallel mode, the input is supplied on the first dimension of the array: the i th input symbol of the input $w = x_0 \dots x_{n-1}$ is fed to the cell $(i, 0, \dots, 0)$ at time 0. In the sequential mode, the specific cell that gets the input serially from time 0 is chosen to be the cell indexed by $\mathbf{0} = (0, \dots, 0)$. Of course, this input cell evolves according to a particular transition function δ_{init} , which takes into account its neighborhood and the received input symbol. The choice of the output cell depends on the neighborhood. One chooses the cell $\mathbf{0} = (0, \dots, 0)$ with the von Neumann neighborhood and the opposite corner $\mathbf{n} - \mathbf{1} = (n - 1, \dots, n - 1)$ with the one-way von Neumann neighborhood.

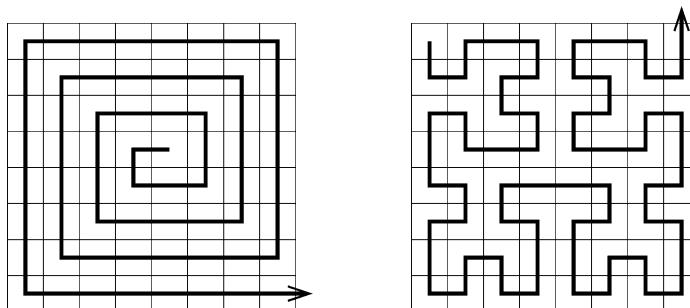
Analogously to one-dimensional CA recognizer, $d - \text{PCA}$, $d - \text{SCA}$, $d - \text{POCA}$, and $d - \text{SOCA}$ denote the four d -dimensional variants according to whether the input mode is parallel or sequential and whether the neighborhood is the two-way von Neumann one or its one-way counterpart. They are denominated variously in different papers: $k - \text{SCA}$ is named k -dimensional iterative array in Cole (1969); $2 - \text{SOCA}$ is named one-way two-dimensional iterative array in Chang et al. (1987) and OIA in Ibarra and Palis (1988); $k - \text{POCA}$ is named k -dimensional one-way mesh connected array in Chang et al. (1989).

The real-time function $rt(n)$, which is defined as the minimal time for the output cell to read the whole input of size n , corresponds to $n - 1$ in the case of $d - \text{PCA}$ and $d - \text{SCA}$, $d(n - 1)$ in the case of $d - \text{POCA}$ and $(d + 1)(n - 1)$ in the case of $d - \text{SOCA}$.

2.3 Two-Dimensional CA Language Recognizer Equipped with a Thread

With the parallel mode in order to save time, the input word may be supplied in a more compact way than the linear one. The difficulty is that there are many ways to set up the one-dimensional input words into multidimensional arrays. All ways are arbitrary and lead to distinct devices with their own complexity classes. A proper approach has been proposed by Delorme and Mazoyer (2002). To set up the inputs in a uniform way, they equip the CA with a thread along which the inputs are written. Thus the thread is a given parameter of the CA, which is independent of the inputs and their length. It is defined as an infinite sequence of adjacent positions in the two-dimensional array without repetition. \Rightarrow Figure 8 depicts the

Fig. 8
Archimedean and Hilbert threads.



Archimedean and Hilbert threads. In practice, the device array is provided with an additional layer that codes the thread. In this layer, each cell records how the thread enters and exits this cell. At initial time, the input symbols are placed consecutively along the thread. Then the evolution of each cell depends on both the states and the thread components of its neighborhood. The output cell is chosen to be the first cell of the thread. Now, the real-time complexity depends on both the neighborhood and the thread (precisely, its significant part according to the input length). No further details on these device types will be given, but the interested reader can refer to Delorme and Mazoyer (2002, 2004) for complete definitions and examples.

2.4 Two-Dimensional CA Picture Recognizer

For higher dimensions, another point of view is to process multidimensional data instead of simple strings. Motivated by image processing issues, much interest has been devoted to picture language recognized by two-dimensional CA. Another stimulation comes from the developments of picture language theory (Giammarresi and Restivo 1997). Hence, in the following, attention is limited to this context, although investigations of arbitrary dimensional CA with arbitrary dimensional inputs would be fairly instructive.

Let us recall some definitions related to picture languages. A picture p over an alphabet Σ is defined as a rectangular $m \times n$ array of symbols of Σ . The couple (m, n) refers to the size of the picture and $p(\mathbf{c})$ denotes the symbol at position \mathbf{c} . The set Σ^{**} denotes the set of all pictures over Σ . A picture language over Σ is any subset of Σ^{**} .

A two-dimensional CA picture recognizer (in short a PictCA) designates a CA recognizer that operates on pictures. On PictCA, the input mode is parallel: at initial time 0 the symbol $p(\mathbf{c})$ of the input picture p is fed to the cell \mathbf{c} . For an input of size (m, n) , the bounded space consists of the $m \times n$ cells $\mathbf{c} = (x, y)$ with $0 \leq x < m$ and $0 \leq y < n$. Outside, the cells remain in a persistent state $\#$ during all the computation.

A priori, the output cell is the cell indexed by $(0, 0)$. The time complexities T are functions defined from \mathbb{N}^2 to \mathbb{N} . And a PictCA is said to accept a picture language L in time T if it accepts the pictures $p \in L$ of size (m, n) in at most $T(m, n)$ steps. As usual, real time means “as soon as possible” and is conditional on the neighborhood. Precisely, the real-time function $rt_{\mathcal{N}}(m, n)$ is, for a PictCA with neighborhood \mathcal{N} , the minimal time needed by the output cell $(0, 0)$ to receive any particular part of a picture input of size (m, n) . That means $rt_{\mathcal{N}}(m, n) = m + n - 2$ when \mathcal{N} is the von Neumann neighborhood and $rt_{\mathcal{N}}(m, n) = \max(m, n) - 1$ when \mathcal{N} is the Moore neighborhood. The linear-time complexities for PictCA with the neighborhood \mathcal{N} are functions $lt_{\mathcal{N}}$ where $lt_{\mathcal{N}}(m, n) = \tau rt_{\mathcal{N}}(m, n)$ and τ is any constant strictly greater than 1. In what follows, the class of all pictures languages recognized by a PictCA with the neighborhood \mathcal{N} in real time (or linear time) will just be named as the real-time (or linear-time) PictCA with the neighborhood \mathcal{N} .

Various algorithms for pictures have been proposed in the general context of mesh-connected arrays of processors. But, their processing elements are not necessarily finite-state contrary to CA. And specific examples, which illustrate the possibilities of processing the data on PictCA, are scarce. Anyway, Beyer (1969) and Levialdi (1972) have independently exhibited two real-time PictCA with the Moore neighborhood that recognize the set of connected pictures. The majority language that consists of the pictures over the alphabet $\{0, 1\}$ in which there are more 1's than 0's has also been examined. Savage (1988) has shown that it is recognized in linear time by PictCA with one-way neighborhoods.

3 Positive Results and Simulation

In this section, the main known equalities and inclusions among CA complexity classes will be examined. These positive results are essentially based on the geometrical characteristics inherited from the regularity of the network structure. The proofs are established by simulations that widely use the tools presented in the chapter [Algorithmic Tools on Cellular Automata](#) by Delorme and Mazoyer and exploit the malleability of the dependency graphs.

3.1 Basic Equivalences Among the Low-Complexity Classes

As an introduction, the figure below gives a general overview of the main relationships among the complexity classes in dimension one.

$$\begin{array}{c}
 \text{PCA} = \text{SCA} = \text{DSpace}(n) \\
 \text{UI} \\
 \text{POCA} = \text{SOCA} \\
 \text{UI} \\
 \text{LPCA} = \text{LSOCA} = \text{LSCA} \\
 \text{UI} \\
 \text{RPCA} = \text{RSOCA} = \text{LPOCA} \\
 \swarrow \qquad \searrow \\
 \text{RPOCA} \neq \text{RSCA}
 \end{array}$$

In this section, we only focus on the various equivalences between the low-complexity classes. We shall return to the equality of POCA and SOCA in [Sect. 3.4](#), to the incomparability of RPOCA and RSCA and their proper inclusions in RPCA in [Sect. 4.1](#). Further discussions will also follow about the famous questions whether the inclusions $\text{RPCA} \subseteq \text{PCA}$ and $\text{RPCA} \subseteq \text{LPCA}$ are strict in [Sect. 6.1](#) and in [Sects. 3.5](#) and [6.2](#).

The original proofs of the positive relationships can be found in the following papers. The equality $\text{LPCA} = \text{LSOCA}$ comes from Ibarra et al. (1985) where it was observed that every PCA working in time $T(n)$ can be simulated by SCA in time $n + T(n)$. The equality $\text{LSOCA} = \text{LSCA}$ has been noticed in Ibarra and Jiang (1988). The equality $\text{LPOCA} = \text{RPCA}$ is from Choffrut and Culik (1984) and the equality $\text{RSOCA} = \text{LPOCA}$ from Ibarra and Jiang (1987). Further relationships between SOCA and POCA in higher dimension have been reported in Terrier (2006b), in particular the equality $d - \text{RSOCA} = d - \text{LPOCA}$ and the inclusions $d - \text{RSOCA} \subseteq (d+1) - \text{RPOCA}$ and $d - \text{LSOCA} \subseteq (d+1) - \text{RSOCA}$. One can also notice that restricted to unary languages (i.e., languages over a one-letter alphabet), RSCA is as powerful as RPCA.

All these equalities are easily obtained by basic simulations. To construct such simulations between one device \mathcal{A} and another device \mathcal{B} , a simple method consists in exhibiting a transformation, which maps the dependency graph of the initial device \mathcal{A} into another directed graph and to verify that this mapped graph, modulo slight modifications, fits the dependency graph of the device \mathcal{B} .

To illustrate this, one example can be considered: the inclusion of RPCA into RSOCA. The question is how to simulate the real-time computations of a given PCA \mathcal{A} by the real-time computations of a SOCA \mathcal{B} . The simulation is essentially based on the following

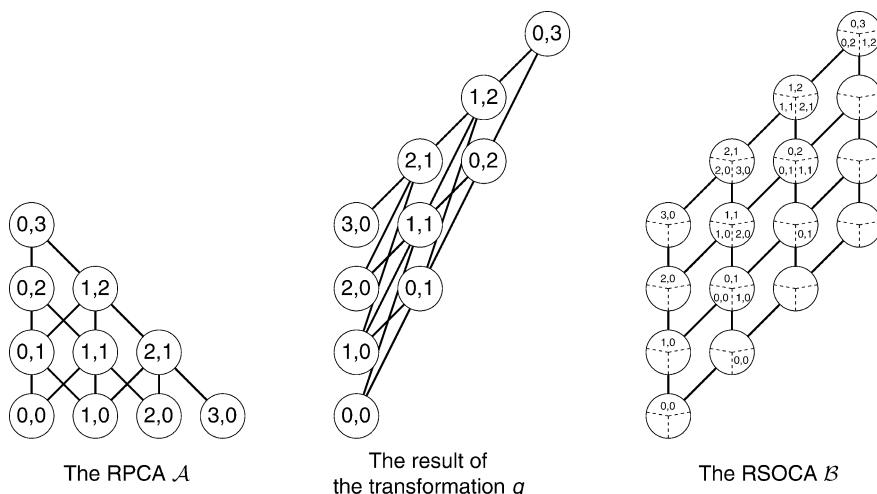
transformation $g(c, t) = (t, 2t + c)$. Hence we should verify that g allows one, with slight modifications, to convert a real-time computation of \mathcal{A} into a real-time computation of \mathcal{B} . For that, we must check that the conditions imposed by the features of the device \mathcal{B} are respected, namely the conditions relating to the structure of the array, the finite memory nature of each cell, the input and output modes, and the neighborhood. First, $g(i, 0) = (0, i)$ guarantees the conversion from the parallel input mode to the sequential input mode and $g(0, n - 1) = (n - 1, 2(n - 1))$ ensures the correspondence between the output sites. Second g maps all sites of \mathcal{A} into sites of \mathcal{B} , precisely a site of \mathcal{B} is mapped to by at most one site of \mathcal{A} . So a finite memory capacity is enough on each cell of \mathcal{B} . Finally, on \mathcal{A} governed by the neighborhood $\{-1, 0, 1\}$, the elementary data movements are $(-1, 1), (0, 1), (1, 1)$. They are converted into the movements $(1, 1), (1, 2), (1, 3)$, which satisfy the dependencies constraints on \mathcal{B} . Effectively, the data are transmitted through intermediate sites according to the elementary moves $(1, 1)$ and $(0, 1)$ and that without exceeding the finite memory capacity of each cell (Fig. 9).

An interesting link between one-dimensional devices and two-dimensional devices equipped with a thread has been observed in Delorme and Mazoyer (2004). It states that the class of languages recognized in real time by a two-dimensional CA with the Archimedean thread and the Moore neighborhood is strictly contained in the class of languages recognized in real time by one-dimensional PCA. Although the minimal time in $O(\sqrt{n})$ for two-dimensional CA with the Archimedean thread is lower than n the minimal time for PCA, this result is far from being straightforward. Notably, during the simulation, each cell must recover its particular position from the output cell and its neighbors in the initial Archimedean spiral. In fact, it enlightens one on the impact of the way input data are space-distributed on the recognition power.

Finally, notice that for picture languages recognition, PictCA with the von Neumann and Moore neighborhoods can simulate each other with a linear time overhead and thus are linear time equivalent. Unfortunately, finer inclusions concerning PictCA are ignored.

Fig. 9

Simulation of a RPCA \mathcal{A} by a RSOCA \mathcal{B} .

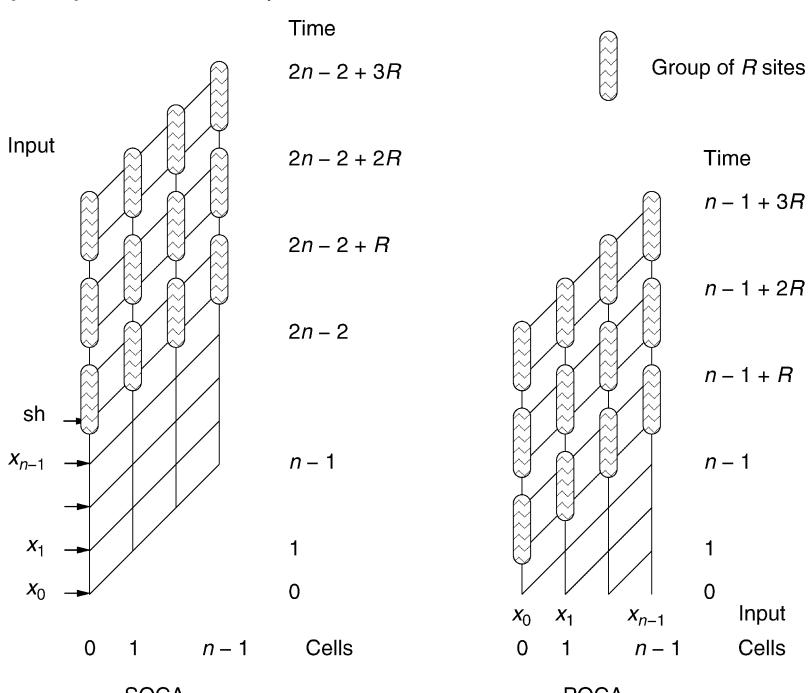


3.2 Linear Speedup

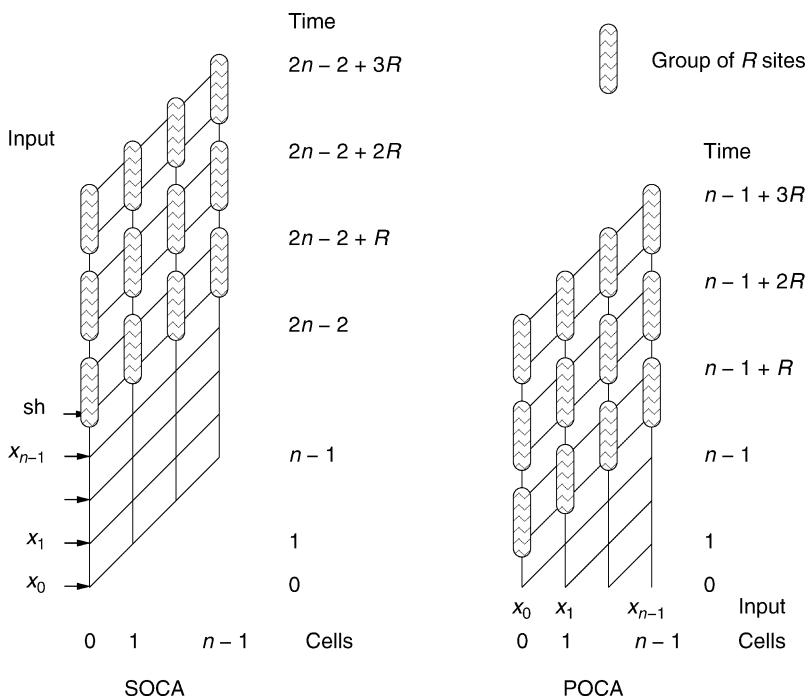
When investigating complexity classes, an immediate question concerns linear acceleration of the running time. For CA, because the recognition time never goes below real time, a linear speedup corresponds to an acceleration by a constant factor of the running time beyond real time.

Theorem 1 *Let f be a function from \mathbb{N} to \mathbb{N} and rt be the real time function for PCA (resp. SCA, POCA or SOCA). For any constant R , a language recognized in time $rt(n) + f(n)$ by a PCA (resp. SCA, POCA or SOCA) is also recognized in time $rt(n) + \lceil f(n)/R \rceil$ by another PCA (resp. SCA, POCA or SOCA).*

In an early work (Beyer 1969), Beyer demonstrated the speed up result for PCA (included in the case of dimension two). Because Beyer (1969) was not widely circulated, the same result can be also found in Ibarra et al. (1985a) and in more general settings in Mazoyer and Reimen (1992). A proof for POCA is in Bucher and Culik (1984) and for SCA and SOCA in Ibarra et al. (1985b). A generalization to SCA in dimension two is given in Ibarra and Palis (1988) and to SOCA and POCA in arbitrary dimension in Terrier (2006b).

One can now recall the usual methods applied to speed up the computation in a linear way. The simplest case is when communication is one-way. Because any two cells are not mutually interdependent, the communication graph is a directed graph that is acyclic. This characteristic allows one to speed up computation easily. 

 **Fig. 10**
Linear speedup in case of one-way communication.



one: once the cell gets the whole input part situated on its left, it can operate R times faster for any integer constant R . This principle can be generalized to higher dimension as long as communication is one-way.

When communication is two-way and all cells are interdependent, to reduce the running time requires compacting the space. In grouping cells into fewer ones, the time to exchange information between cells is reduced and so the computation can be achieved at a higher rate. The grouping operation differs according to the input mode.

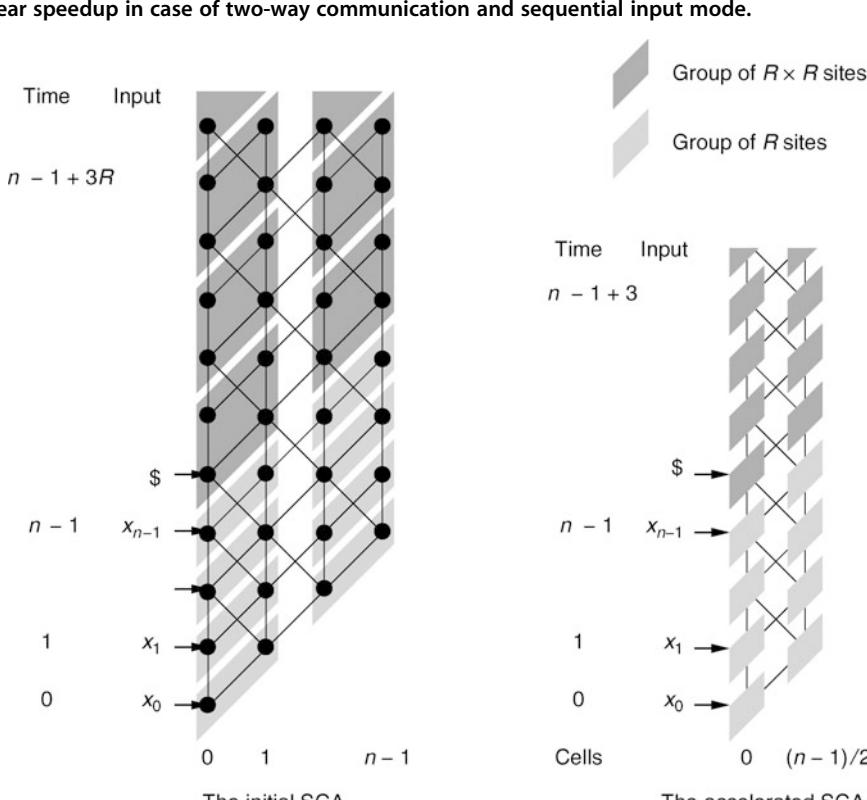
It is immediate when the input mode is sequential, as all cells have the same initial quiescent state. Initially there is no difficulty in grouping information into fewer cells. Then the accelerated computation can take place immediately when the distinguished cell has obtained the whole input. 

Figure 11 illustrates the situation in dimension one. The grouping operation is initially accomplished within each diagonal: for any integer constant R , the sites are grouped together by R . Once the input is read and the end marker is constantly fed, the information initially processed by R diagonals can be processed by a simple one and hence accelerate the computation by a factor R . The fact remains that the time space diagram is distorted. It involves keeping some redundant information in each cell but it preserves the data accessibility requirement while respecting the dependency constraints.

Fig. 11

Linear speedup in case of two-way communication and sequential input mode.

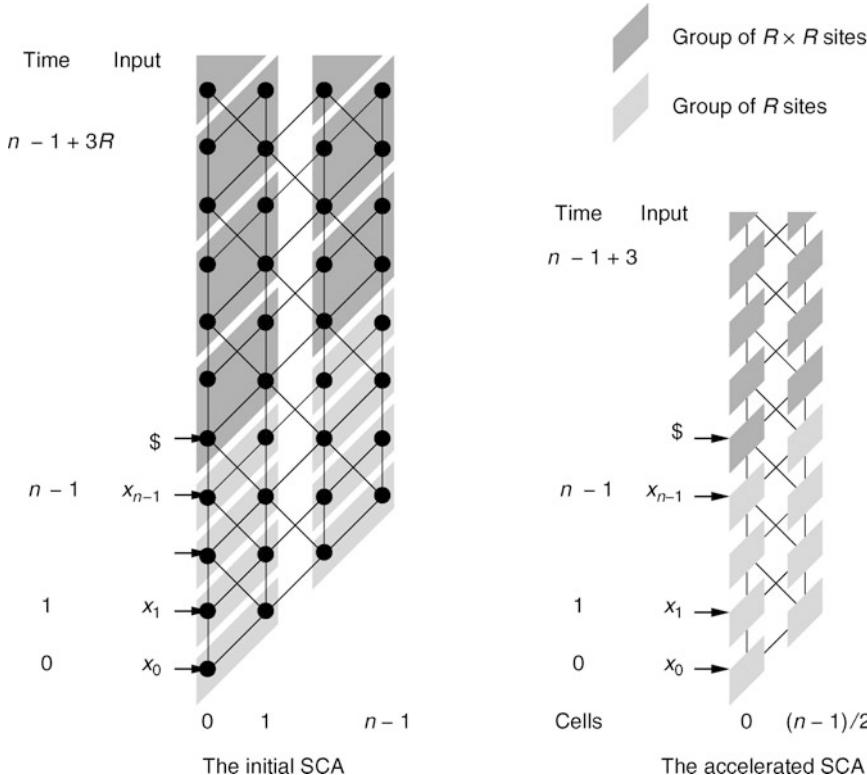
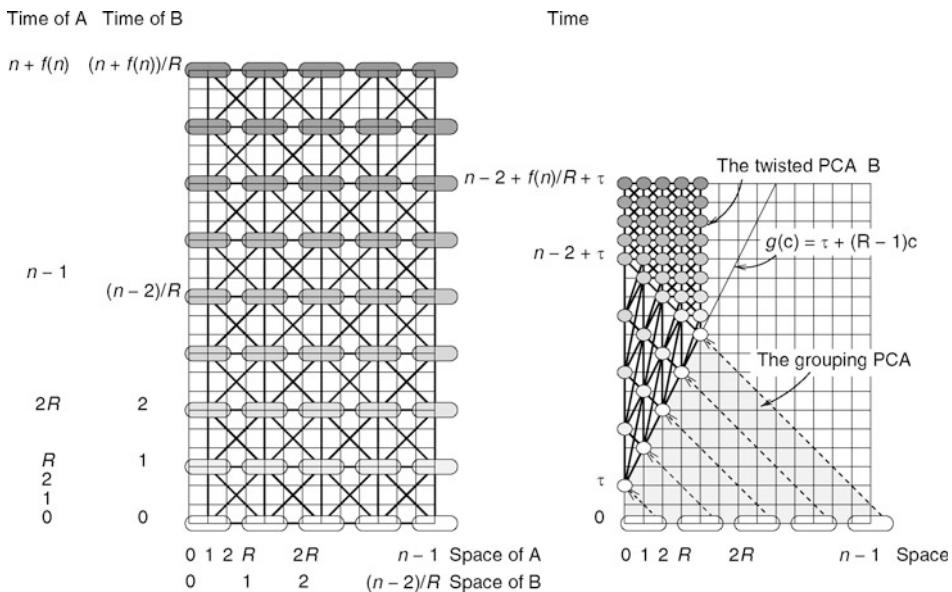


Fig. 12

Linear speedup in case of two-way communication and parallel input mode.



The case of parallel input mode (and two-way communication) is more tricky. The input data are initially fed into the array and some time must be spent grouping them together. In order to avoid losing more time, the accelerated computation must start as soon as possible.

Figure 12 illustrates the method for a PCA \mathcal{A} . On the one hand, observe that one can construct a PCA \mathcal{B} , which simulates R times faster the PCA \mathcal{A} provided the input given to \mathcal{A} is fed compacted with a factor R to \mathcal{B} . On the other hand, there is a grouping PCA that turns the initial ungrouped input into a grouped one. In this way, one can stick the grouping PCA with a twisted variant of the accelerated PCA \mathcal{B} to obtain the desired PCA. The accelerated computation starts on each cell as soon as its neighbor cells are grouped.

In developing such techniques for picture language recognizer, Beyer (1969) has shown linear speedup result for CA with the von Neumann neighborhood. These techniques work for the Moore neighborhood and similar kinds of neighborhoods as well (see Terrier 2004).

Theorem 2 *Let f be a function from $\mathbb{N} \times \mathbb{N}$ to \mathbb{N} , \mathcal{N} be the von Neumann or Moore neighborhood and $r_{\mathcal{N}}$ be the corresponding real time function. For any constant R , a picture language recognized in time $r_{\mathcal{N}}(m, n) + f(m, n)$ by a PCA with the neighborhood \mathcal{N} is also recognized in time $r_{\mathcal{N}}(m, n) + \lceil f(m, n)/R \rceil$ by another PCA with the same neighborhood.*

What is disturbing about picture language recognition is that some neighborhoods seem not to admit linear speedup. In the common method, each cell implicitly knows in which direction to send its content to achieve the grouping process. This direction corresponds to a shorter path toward the output cell. But for some neighborhoods, this direction differs according to the position of the cell into the array and no alternative efficient grouping process is currently known.

3.3 Constant Speedup

In addition to the speedup results of [Sect. 3.2](#), real-time complexity could also be defined modulo a constant. In dimension one, this property was first observed in Choffrut and Culik (1984) and a whole complete generalization has been done in Poupet (2005). Excluding pathological neighborhoods for which the output cell is not able to read the whole input, we have constant time acceleration for PCA.

Proposition 1 *For any neighborhood \mathcal{N} , any function satisfying $f(n) \geq rt_{\mathcal{N}}(n)$ and any constant τ :*

$$\text{PCA}_{\mathcal{N}}(f(n) + \tau) = \text{PCA}_{\mathcal{N}}(f(n))$$

An interesting consequence, which emphasizes the robustness of one-dimensional CA model, follows from this speedup (see Poupet 2005). The computation ability in dimension one is somewhat independent of the underlying communication graph.

Theorem 3 *In dimension one, all neighborhoods are real-time equivalent to either the one-way neighborhood $\{-1, 0\}$ or the standard one $\{-1, 0, 1\}$.*

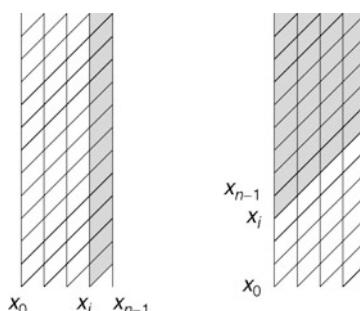
The situation turns out to be less satisfactory in higher dimensions. For the two classical von Neumann and Moore neighborhoods, the real-time complexity can also be defined modulo a constant. But a uniform approach does not yet exist to deal with various neighborhoods. And in spite of investigations about arbitrary neighborhoods reported in Delacourt and Poupet (2007), nothing much is known.

3.4 Equivalence Between the Parallel and Sequential Input Modes

A result that cannot be ignored says that, if one does not bother with the time comparison, the parallel and sequential input modes are equivalent. This result is fairly astonishing in the case of one-way communication. Indeed the parallel input mode combined with one-way communication induces strong limitations on the access to the data. As illustrated in [Fig. 13](#), a cell of a POCA has no access on the input letters applied on its right whereas every cell of a

Fig. 13

Influence of the i th input symbol on a POCA and on a SOCA.



SOCA has access to the whole input. Moreover, for SOCA, the end marker \$ supplied after the whole input letters provides information on the length of the input. On the other hand, such knowledge is impossible for POCA. Despite these outstanding differences, Ibarra and Jiang (1987) have shown that SOCA and POCA accept the same class of languages.

Theorem 4 $\text{POCA} = \text{SOCA}$.

Because of the advantages of SOCA over POCA, the simulation of a POCA by a SOCA is straightforward. In contrast, the reverse simulation is considerably more involved. Let one just present a rough idea of the construction and its difficulties. To simulate a SOCA by a POCA, the basic idea is to systematically generate each possible input, to simulate the SOCA on it, and check if it is actually the real input. For that, the working area is divided in two parts. The enumeration process takes place in the left part. First the inputs of length 1, then those of length 2, then those of length 3, etc., are generated in such a way that they can be obtained serially by the leftmost cell of the right part. Simultaneously, in the right part, the SOCA is simulated on the successive generated inputs. Moreover, the successive inputs are compared with the real input.

The delicate point is that the POCA does not have hints about the length of the real input and has to deal with inputs of all possible lengths. In particular, the time and the space required to simulate the SOCA on the current input depends on its length. Also this simulation time induces a delay between the generation of two inputs. And the demarcation of the two parts of the working area fluctuates according to the current length of the generated inputs. Then to distribute and to synchronize the different subcomputations into the working area entails many of the techniques described in the chapter [Algorithmic Tools on Cellular Automata](#). A complete description can be found in the original paper (Ibarra and Jiang 1987) and with a direct construction in Delorme and Mazoyer (1994).

The drawback of this construction is its exponential cost. Indeed, whatever the time complexity of the initial SOCA, the POCA simulates the behavior of the SOCA on an exponential number of inputs. Additional questions related to the simulation cost of SOCA by POCA will be discussed in [Sect. 6.2](#).

3.5 Closure Properties

Closure properties are naturally investigated in order to evaluate the computation ability of the different CA classes and to possibly achieve separation results. Obviously, all deterministic CA classes satisfy the closure under Boolean operations. More attention has been paid to the closure properties under other language operations as concatenation, reverse, or cycle. As you may recall, the reverse of a language L is $L^R = \{w^R : w \in L\}$ where w^R is the word w written backward and its cycle is $L^{\text{Cy}} = \{vu : uv \in L\}$. [Table 2](#) sums up the known results in dimension one (Y stands for yes, N for no, and ? for open).

Let us give further details about these results and the open questions. Because PCA has the same computation power as linear-space Turing machine, it satisfies various closure properties: reverse, concatenation, Kleene star, ε -free homomorphism, inverse homomorphism, and cycle. From Chang et al. (1988) it is known that the one-way counterparts POCA and SOCA also share the above closure properties. The proofs make essential use of similar constructions as the one outlined in [Sect. 3.4](#) to show that $\text{POCA} = \text{SOCA}$. In [Sect. 4.1](#), we will account for the negative closure properties of the RSCA class and its higher

Table 2

Closure properties in dimension one

Class	Reverse closure	Concatenation closure	Cycle closure
PCA	Y	Y	Y
POCA	Y	Y	Y
LPCA	Y	?	?
RPCA	?	?	?
RSCA	N	N	N
RPOCA	Y	N	N

dimension counterparts and then for the negative closure under concatenation of RPOCA. Curiously enough, this last result is the opposite in higher dimensions: for any dimension $d > 1$, d -RPOCA satisfies closure under concatenation as well as closure under Kleene star (Terrier 2006b). As observed in Choffrut and Culik (1984), the closure under reverse of RPOCA follows from the symmetry of its dependency graph. For the same structural reason, the result extends to the higher dimensions (Terrier 2006b). The closure under reverse of LPCA is an immediate consequence of the linear speedup results (Ibarra and Jiang 1988).

It is not known whether RPCA is closed under reverse or under concatenation. But a striking result due to Ibarra and Jiang (1988) relates the property of RPCA to be closed under reverse and its ability to be as powerful as LPCA. In a similar way, the cycle closure property of RPCA can be linked to the equality between RPCA and LPCA (Terrier 2006a). The following theorem gathers these relationships.

Theorem 5 *The following three statements are equivalent:*

- $\text{RPCA} = \text{LPCA}$.
- RPCA is closed under reverse.
- RPCA is closed under cycle.

This theorem is very interesting and deserves some explanations. Let one focus on the equivalence between the ability of RPCA to be as powerful as LPCA and its reverse closure property. The proof relating to the third statement makes essential use of similar arguments. As LPCA is closed under reverse, the equality $\text{RPCA} = \text{LPCA}$ directly implies the closure under reverse of RPCA. To show the converse implication is more tricky. How may a closure property involve speedup of linear-time computation? First the assertion $\text{LPCA} = \text{RPCA}$ can be restated in the following equivalent statement “Let c be any constant. If $\{\#^{c^2 \lceil \log |w| \rceil} w : w \in L\}$ is a RPCA language then L is a RPCA language.” The key argument developed in Ibarra and Jiang (1988) is that the reverse form of this statement is true.

Lemma 1 *Let c be any constant. If $\{\#^{c^2 \lceil \log |w| \rceil} w : w \in L\}$ is a RPCA language then L is a RPCA language.*

The proof of this meaningful lemma is fairly involved. A whole description can be found in the original paper (Ibarra and Jiang 1988) and with a direct construction in Delorme and

Mazoyer (1994). Now according to this [Lemma 1](#) and providing RPCA is closed under reverse, the following chain of implications is obtained:

$$\begin{aligned}
 L &\in \text{LPCA} \\
 \Rightarrow \widetilde{L} &= \{w\#^{c2^{\lceil \log |w| \rceil}} : w \in L\} \in \text{RPCA} && \text{(straightforward property)} \\
 \Rightarrow \widetilde{L}^R &= \{\#^{c2^{\lceil \log |w| \rceil}} w : w \in L^R\} \in \text{RPCA} && \text{(reverse closure)} \\
 \Rightarrow L^R &\in \text{RPCA} && \text{(Lemma 1)} \\
 \Rightarrow L &\in \text{RPCA} && \text{(reverse closure)}
 \end{aligned}$$

An immediate consequence of [Theorem 5](#) is that if RPCA is closed under reverse or under cycle then it is closed under concatenation (Ibarra and Jiang 1988). Whether the converse is true remains an open question. We just know a weaker implication also based on [Lemma 1](#) (Terrier 2006a).

Proposition 2 *If RPCA is closed under concatenation then*

- *LPCA unary languages are RPCA unary languages.*
- *LPCA is closed under concatenation.*

Here let us have a look at closure properties of picture language recognizers. Up to now the closure properties under concatenation have not been studied. In the matter of the reverse operation, its counterpart for picture languages is the 180° rotation operation. As in the one-dimensional case, the linear speedup results entail the closure under rotation of linear-time PictCA with the von Neumann or Moore neighborhoods. In contrast, it can be seen in [Sect. 4.1](#) that real-time PictCA with the Moore neighborhood, real-time PictCA and linear-time PictCA with the one-way von Neumann or one-way Moore neighborhood do not satisfy closure under rotation. Interestingly, the arguments developed by Ibarra and Jiang to relate, in dimension one, closure properties to computation ability can be extended to PictCA recognizer (Terrier 2003b).

Proposition 3 *Real time PictCA with the von Neumann neighborhood is closed under rotation if and only if real time and linear time PictCA with the von Neumann neighborhood are equivalent.*

At last a nice property pointed out by Szwerinski (1985) can be noticed. In dimension one, a PCA may confuse right and left without time loss. Formally, a local transition function $\delta : S^3 \rightarrow S$ is said symmetrical if for all $r, c, l \in S$ holds: $\delta(l, c, r) = \delta(r, c, l)$. So the following proposition, as shown in Szwerinski (1985) and in a more general setting in Kobuchi (1987), emphasizes that the orientation does not matter for one-dimensional PCA.

Proposition 4 *If a language L is recognized in time T by some PCA, then L is recognized in the same time T by another PCA whose transition function is symmetrical.*

4 Limitations

The purpose of this section is to present the known limitations on the recognition power of CA. It makes use of either algebraic arguments or diagonalization ones.

4.1 Algebraic Arguments

Let one start with a basic negative result. As observed in Culik et al. (1984), POCA operating in real time on languages over a one-letter alphabet are no more powerful than finite automata.

Proposition 5 *The class of unary languages recognized by RPOCA is the class of regular unary languages.*

This result has been strengthened in Buchholz and Kutrib (1998): it requires at least an amount of $n + \log n$ time to recognize non-regular unary languages on POCA.

An interesting consequence of [Proposition 5](#), observed in Choffrut and Culik (1984), derives from the existence of non-regular unary languages that are recognized by RSCA and therefore by RPCA. For instance, the languages $\{1^{2^n} : n \in \mathbb{N}\}$ and $\{1^p : p \text{ is a prime}\}$ belong to RSCA (see Choffrut and Culik (1984) and Fischer (1965) and the chapter [Computations on Cellular Automata](#) by Mazoyer and Yunès). It yields the following relationships.

Corollary 1

- RPOCA \subsetneq RPCA
- RSCA $\not\subseteq$ RPOCA

One can now concentrate on elaborate techniques to obtain lower bounds. These techniques were introduced by Hartmanis and Stearns (1965) for real-time Turing machines and adapted to real-time SCA by Cole (1969). Using counting arguments, they exploit limits on interaction between data.

4.1.1 The Method for Real-Time SCA

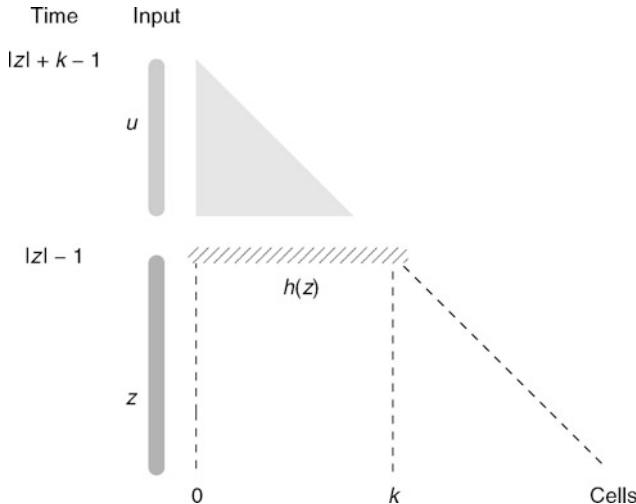
Let one recall the method used in Cole (1969) to get negative results on RSCA. As illustrated in [Fig. 14](#), a characteristic of the real-time SCA computation on a given input w is the following one. The suffix of arbitrary length k of w only has an impact on the computation during the k last steps. In particular, the first part z and the suffix u of length k of the input $w = zu$ interact on a number of cells independent of the length of z . Hence significant information on z may be lost before the k last steps of the computation. Precisely, consider the configuration of the SCA at time $|z| - 1$ when the last symbol of z is fed. At this time, the useful part consists of the $k + 1$ first sites, which may influence the result of the computation obtained on the output cell, k steps after. Let $h(z)$ denote the sequence of states of length $k + 1$ of this useful part. Thus the result of the computation of the RSCA on the input zu is completely specified by $h(z)$ and u . That sets an upper bound on the number of distinct behaviors and thus implies the following condition on the structure of RSCA languages.

Proposition 6 *Let L be a language over some alphabet Σ . Let X be a finite subset of Σ^* and let k be the maximal length of the words in this set X . Consider, for each word $z \in \Sigma^*$, the indicator function p_z from X into $\{0, 1\}$ defined as:*

$$p_z(u) = \begin{cases} 1 & \text{if } zu \in L \\ 0 & \text{otherwise} \end{cases}$$

Fig. 14

Real-time computation of a SCA on some input $w = zu$.



If the language L is recognized in real time by some SCA, then the number of distinct functions $\{p_z : z \in \Sigma^*\}$ is of order at most $2^{O(k)}$.

Proof Suppose that L is recognized by some RSCA \mathcal{A} . For any words $z, z' \in \Sigma^*$, observe that if $h(z) = h(z')$ then for all $u \in X$ either zu and $z'u$ are both accepted or both rejected by \mathcal{A} . Indeed the words u in X have length at most k and $h(z)$, as defined above, consists of the information accessible to the k last steps. Hence if $h(z) = h(z')$, then $p_z = p_{z'}$. Now notice that every $h(z)$ is a word of length $k+1$ on the finite set of states of \mathcal{A} . Therefore, the number of distinct functions p_z does not exceed the number of distinct sequences $h(z)$, which is of order $2^{O(k)}$.

A couple of examples are now given.

Example 6 $L = \{x_1 \# x_2 \# \dots x_t \# x : x_1, x_2, \dots, x_t, x \in \{0, 1\}^*\text{ and }x = x_j \text{ for some }j \text{ with }1 \leq j \leq t\}$ is not a real-time SCA language.

Proof Set $X = \{0, 1\}^k$. Associate to each subset A of X , the word $z_A = x_1 \# x_2 \# \dots x_t \#$ where x_1, x_2, \dots, x_t is some enumeration of the words in A . Namely, z_A is defined in such a way that $z_A u \in L$ if and only if $u \in A$. Thus, if A and B are two distinct subsets of X then $p_{z_A} \neq p_{z_B}$. So the number of functions $\{|p_z : z \in \Sigma^*\|$ is at least 2^{2^k} , the number of subsets of X . Therefore, according to [Proposition 6](#), L is not a RSCA language.

The following example differs from the previous one in the number of functions p_z .

Example 7 $L = \{01^{a_1} 01^{b_1} 01^{a_2} 01^{b_2} \dots 01^{a_t} 01^{b_t} 01^a 01^b : a_1, b_1, \dots, a_t, b_t \geq 0 \text{ and } a = a_j, b = b_j \text{ for some }j \text{ with }1 \leq j \leq t\}$ is not a real-time SCA language.

Proof Set $X = \{01^a 01^b : a, b \geq 0 \text{ and } a + b + 2 \leq k\}$. Associate to each subset A of X , the word $z_A = x_1 \dots x_t$ where x_1, \dots, x_t is some enumeration of the words in A . As z_A is defined,

$z_A u \in L$ if and only if $u \in A$. Thus, if A and B are two distinct subsets of X , then $p_{z_A} \neq p_{z_B}$. So the number of functions $|\{p_z : z \in \Sigma^*\}|$ is at least $2^{k(k-1)}$, the number of subsets of X . Hence L is not a RSCA language.

Several other languages are known not to belong to RSCA. Among them, the languages that have been shown not real time recognizable by multitape Turing machines in Hartmanis and Stearns (1965) and Rosenberg (1967) are also not real time recognizable by SCA. Indeed both devices share the same features on data accessibility, which are exploited to get limitations. Actually, all these languages are RPCA languages. As a consequence, RSCA is less powerful than RPCA. Furthermore, specific examples exhibited in Hartmanis and Stearns (1965), Rosenberg (1967), Cole (1969), Dyer (1980), and Kutrib (2001) yield several negative properties. A representative one is the language of words, which end with a palindrome of length at least three: $L = \{w \in \Sigma^* : w = uv, v = v^R, |v| > 2\}$. Yet L is linear context free and belongs to RPOCA; furthermore, the reverse of L and the palindrome language belong to RSCA. Therefore RSCA contains neither all linear context-free languages nor RPOCA languages and is not closed under reversal and under concatenation. The following corollary summarizes the various results obtained.

Corollary 2

- RSCA is strictly contained in RPCA.
- RSCA and RPOCA are incomparable.
- RSCA is not closed under reversal, concatenation, Kleene closure, sequential mapping, or the operations of taking derivatives and quotients.
- RSCA does not contain all deterministic linear context free languages.

One further noteworthy result of Cole (1969) is that the power of real-time SCA increases with the dimension of the space.

Proposition 7 *For any dimension d , $d - \text{RSCA} \subsetneq (d + 1) - \text{RSCA}$.*

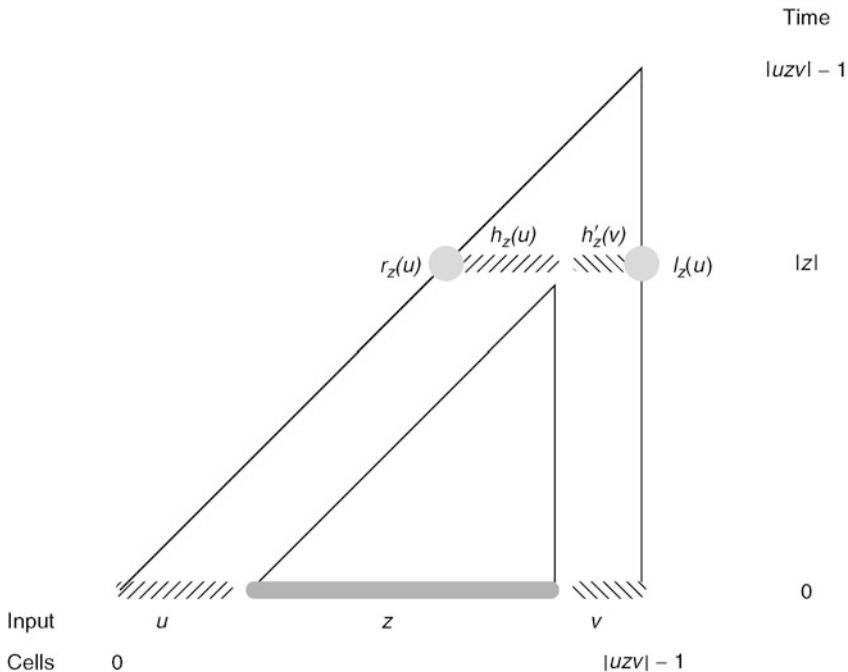
Proof The number of cells that influence the k last steps of any $d - \text{RSCA}$ computation increases polynomially with the dimension d . Precisely, the counterpart of $h(z)$ in dimension d , is a d -dimensional word of diameter $O(k)$ and volume $O(k^d)$. Hence the number of distinct words $h(z)$ is in $2^{O(k^d)}$. Furthermore, the condition stated in [Proposition 6](#) can be rewritten in this way: “If the language L is recognized in real time by some $d - \text{SCA}$ then the number of distinct functions $|\{p_z : z \in \Sigma^*\}|$ is of order at most $2^{O(k^d)}$.” As a consequence the language presented in [Example 6](#) is not a $d - \text{RSCA}$ language whatever the dimension d may be. On the other hand, the language given in [Example 7](#) is not a $1 - \text{RSCA}$ language but it may be, and in fact is, a $2 - \text{RSCA}$ language. Of course, languages with the same kind of structure as the one of [Example 7](#) can be built to separate $d - \text{RSCA}$ and $(d + 1) - \text{RSCA}$.

4.1.2 The Method for Real-Time POCA

One can now look at the case of RPOCA. A first characteristic noticed in Culik (1989) is that the real-time computation on an input w contains the real-time computations of all its factors. Recall the [Example 4](#) in [Sect. 2](#): the RPOCA, which tests whether an input is a Dyck word, processes together all its factors. So the evolution on an input of size n decides the memberships of $n(n + 1)/2$ words. This constraint has been exploited in Terrier (1995) to get a non-RPOCA language.

Fig. 15

Real-time computation of a POCA on some input uzv .



A second characteristic of RPOCA computation noticed in Terrier (1996) is the following one. As depicted in Fig. 15, on an input w , its prefixes u and suffixes v only interact during the $|uv| - 1$ last steps. More precisely, on input $w = u z v$, consider at time $|z|$ the useful part that consists of the $|uv|$ sites, which may have an impact on the result obtained $|uv| - 1$ steps after. This part subdivides into the first $|u|$ sites named $h_z(u)$ and the last $|v|$ sites named $h'_z(v)$. Note that the result of the computation on uzv is completely specified by $h_z(u)$ and $h'_z(v)$. Furthermore, according to the first characteristic, the results of all the factors, which contained z are determined by $h_z(u)$ and $h'_z(v)$. On the other hand, $h_z(u)$ is not influenced by the suffix v as well as $h'_z(v)$ by the prefix u . Hence, intuitively the information exchange between u and v takes place when significant information about z may be lost. This situation gives rise to the following condition on the structure of RPOCA languages.

Proposition 8 Let L be a language over some alphabet Σ . Let X, Y be two sets of words on Σ . Denote the set of all suffixes of words in X by $\text{Suff}(X)$ and the set of all prefixes of words in Y by $\text{Pref}(Y)$. Consider, for each word $z \in \Sigma^*$, the indicator function p_z defined as:

$$p_z: \text{Suff}(X) \times \text{Pref}(Y) \rightarrow \{0, 1\}$$

$$(u, v) \mapsto \begin{cases} 1 & \text{if } u z v \in L \\ 0 & \text{otherwise} \end{cases}$$

If L is recognized in real time by some POCA then the number of distinct functions $\{|p_z: z \in \Sigma^*\|$ is of order at most $2^{O(|\text{Suff}(X)| + |\text{Pref}(Y)|)}$.

Proof Suppose that L is recognized by some RPOCA \mathcal{A} . Denote S the set of states of \mathcal{A} . Consider the two functions l_z from $\text{Suff}(X)$ into S and r_z from $\text{Pref}(Y)$ into S defined in this following way. To every u in $\text{Suff}(X)$, l_z associates the state entered at time $|z|$ by the leftmost cell involved in the real-time computation of \mathcal{A} on input uz . Symmetrically, to every v in $\text{Pref}(Y)$, r_z associates the state entered at time $|z|$ by the rightmost cell involved in the real-time computation of \mathcal{A} on input zv . Observe that $h_z(u)$, as defined above, is the sequence of $r_z(x)$ where x ranges over the set of all suffixes of u : $h_z(u_1 u_2 \dots u_k) = r_z(u_1 u_2 \dots u_k) r_z(u_2 \dots u_k) \dots r_z(u_k)$. In a symmetric way, $h'_z(v)$ is the sequence of $l_z(y)$ where y ranges over the set of all prefixes of v : $h'_z(v_1 \dots v_{k-1} v_k) = l_z(v_1) \dots l_z(v_{k-1}) l_z(v_1 v_2 \dots v_k)$. Therefore, if $r_z = r'_{z'}$ and $l_z = l'_{z'}$, then for all $u \in \text{Suff}(X)$ and $v \in \text{Pref}(Y)$ either uzv and $uz'v$ are both accepted or both rejected by \mathcal{A} . In other words, if $r_z = r'_{z'}$ and $l_z = l'_{z'}$ then $p_z = p'_{z'}$. Therefore, the number of distinct functions p_z does not exceed the product of the number of distinct functions r_z with the number of distinct functions l_z , which is in $2^{O(|\text{Suff}(X)| + |\text{Pref}(Y)|)}$.

Example 8 $L = \{x \# x_1 \$ y_1 \# x_2 \$ y_2 \# \dots \# x_t \$ y_t \# y : x_1, y_1, x_2, y_2, \dots, x_t, y_t, x, y \in \{0, 1\}^*\text{ and }x = x_j, y = y_j \text{ for some } j \text{ with } 1 \leq j \leq t\}$ is not a real time POCA language.

Proof Set $X = Y = \{0, 1\}^k$. Associate to each subset A of $\text{Suff}(X) \times \text{Pref}(Y)$, the word $z_A = \# x_1 \$ y_1 \# x_2 \$ y_2 \# \dots \# x_t \$ y_t \#$ where $(x_1, y_1), \dots, (x_t, y_t)$ is some enumeration of the words in A . By construction, $x z_A y \in L$ if and only if $(x, y) \in A$. Thus, if A, B are two distinct subsets of $\text{Suff}(X) \times \text{Pref}(Y)$ then $p_{z_A} \neq p_{z_B}$. Therefore, the number of distinct functions $|\{p_z : z \in \Sigma^*\}|$ is at least $2^{2^{2^k}}$ the number of subsets of $\text{Suff}(X) \times \text{Pref}(Y)$. It is of greater order than $2^{O(|\text{Suff}(X)| + |\text{Pref}(Y)|)} = 2^{O(2^k)}$.

Example 9 Let us consider the linear context free language $L_1 = \{w : w = 1^u 0^u \text{ or } w = 1^u 0 y 1^u \text{ with } y \in \{0, 1\}^* \text{ and } u > 0\}$. The context free language $L = L_1 \cdot L_1$ is not a real time POCA language.

Proof Fix some integer k . Set $X = \{1^k\}$ and $Y = \{0^k\}$. Associate to each subset A of $\text{Suff}(X) \times \text{Pref}(Y)$, the word $z_A = 0^{i_1} 1^{j_1} \dots 0^{i_t} 1^{j_t}$ where $(1^{i_1}, 0^{j_1}), \dots, (1^{i_t}, 0^{j_t})$ is some enumeration of the words in A . As z_A is defined, $1^u z_A 0^v \in L$ if and only if $(1^u, 0^v) \in A$. Thus, if A, B are two distinct subsets of $\text{Suff}(X) \times \text{Pref}(Y)$ then $p_{z_A} \neq p_{z_B}$. Therefore, the number of distinct functions $|\{p_z : z \in \Sigma^*\}|$ is at least $2^{(k+1)^2}$ the number of subsets of $\text{Suff}(X) \times \text{Pref}(Y)$. It is of greater order than $2^{O(|\text{Suff}(X)| + |\text{Pref}(Y)|)} = 2^{O(k)}$. According to [Proposition 8](#) we conclude that L is not a RPOCA language.

Since linear context-free languages are RPOCA languages, an immediate consequence of [Example 9](#) is that RPOCA is not closed under concatenation and does not contain all context-free languages. With further results obtained in Klein and Kutrib (2003), the following corollary gives the main negative properties of RPOCA.

Corollary 3

- The class of real-time POCA languages is not closed under concatenation, Kleene closure, ϵ -free homomorphisms.
- The class of real-time POCA languages does not contain all context-free languages.

Furthermore, as shown in Buchholz et al. (2000) and Klein and Kutrib (2003), the previous arguments combined with padding techniques lead to an infinite hierarchy of separated classes between real-time SCA and linear-time SCA as well as another one between

real-time POCA and linear-time POCA. For the sake of [Proposition 9](#) below, the definition of Fischer's constructibility can be found in the chapter [Algorithmic Tools on Cellular Automata](#) by Delorme and Mazoyer. At least, the next function T can be viewed as a “reasonable” function of asymptotic order at most n .

Proposition 9

- Let T, T' be two functions from \mathbb{N} to \mathbb{N} such that the inverse of T is Fischer constructible and $T' \in o(T)$. We have the strict inclusion $\text{SCA}(n + T'(n)) \subsetneq \text{SCA}(n + T(n))$.
- Let T, T' be two functions from \mathbb{N} to \mathbb{N} such that the inverse of T is Fischer constructible and $T' \log(T') \in o(T)$. We have the strict inclusion $\text{POCA}(n + T'(n)) \subsetneq \text{POCA}(n + T(n))$.

Similarly, such algebraic techniques may be applied to exhibit limits on the computation ability of picture language recognizers. In this way, weakness of real-time computation with the Moore neighborhood has been observed in Terrier ([1999](#)). Precisely, there exists a picture language, which is real-time recognizable by no PictCA with the Moore neighborhood. Furthermore, this picture language is real-time recognizable by a PictCA with the von Neumann neighborhood and its corresponding language obtained by a rotation of 180° is accepted in real time with both the von Neumann and Moore neighborhoods.

In the same vein, restricted communication has been shown to reduce the computational power of low-complexity picture classes (Terrier [2006a](#)). Precisely, there exists a language recognized in real time by PictCA with both the von Neumann and Moore neighborhoods but not recognized in linear time with any one-way neighborhoods \mathcal{N} such that $a + b \geq 0$ for every (a, b) in \mathcal{N} . Furthermore, the corresponding language obtained by a rotation of 180° is recognized by PictCA with one-way neighborhoods $\mathcal{N}_1 = \{(0, 0), (0, 1), (1, 0)\}$, $\mathcal{N}_2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ or $\mathcal{N}_3 = \{(0, 0), (0, 1), (1, 0), (1, 1), (-1, 1), (1, -1)\}$.

The various consequences are summarized in the following proposition.

Proposition 10

- Real-time PictCA with the Moore neighborhood does not contain real-time PictCA with the von Neumann neighborhood.
- Real-time PictCA with the Moore neighborhood is not closed under rotation.
- Real-time PictCA with the Moore and von Neumann neighborhoods are not contained in linear time PictCA with a one-way neighborhood \mathcal{N} where $a + b \geq 0$ for all $(a, b) \in \mathcal{N}$.
- Real-time and linear time PictCA with the one-way neighborhoods \mathcal{N}_1 , \mathcal{N}_2 and \mathcal{N}_3 are not closed under rotation.

As emphasized by Cervelle and Formenti ([2009](#)), one can as well make use of Kolmogorov complexity to derive these results. This alternative approach expresses in a more direct manner the fact that significant information is lost. Here, we will not recall the formal definition of Kolmogorov complexity, but just give an example to illustrate the method. The reader is referred to Cervelle and Formenti ([2009](#)) for comprehensive definitions and for an example involving POCA device.

A basic example is the language not belonging to RSCA, which was presented in [Example 6](#): $L = \{x_1 \# x_2 \# \dots \# x_t \# x : x_1, x_2, \dots, x_t, x \in \{0, 1\}^*\text{ and }x = x_j \text{ for some }j \text{ with }1 \leq j \leq t\}$. Contrary to the algebraic method, we will not take into account all possible evolutions, but just focus on a “complex” one. Let one fix ω some Kolmogorov random number of length 2^k . Consider some bijection val between $X = \{0, 1\}^k$ and $\{0, \dots, 2^k - 1\}$, which codes each word of X by a distinct integer below 2^k , for instance $val(a_0 \dots a_{k-1}) = \sum a_i 2^i$. Consider the set $A_\omega = \{x \in X : \text{the}$

symbol of rank $val(x)$ in ω is a 1} and $z_\omega = x_1 \# \dots \# x_t \#$ where x_1, \dots, x_t is some enumeration of the words in A_ω . Now if L is recognized in real time by some SCA \mathcal{A} then ω can be reconstructed from the description of \mathcal{A} , the description of the bijection val and the sequence of states $h(z_\omega)$. Indeed, from \mathcal{A} and $h(z_\omega)$, we can decide, for each $x \in X$, whether $z_\omega x$ is in L or not, and so whether the symbol of rank $val(x)$ in ω is a 1 or a 0. The respective lengths of these descriptions are in $O(1)$ for \mathcal{A} , in $O(\log k)$ for val and in $O(k)$ for $h(z_\omega)$ and their sum is less than the 2^k bits of the word ω . Hence it leads to a contradiction on the incompressibility of ω .

Undoubtedly, algebraic arguments and also Kolmogorov complexity are powerful tools to establish limits on the computation ability of restricted devices with low complexity. But it should be admitted that many questions on the power of these devices are left open. For instance, the same arguments are exploited to prove that some languages do not belong to real-time Turing machines nor RSCA. Does it mean that real-time SCA are no more powerful than real-time Turing machines? Moreover, most of the witness languages, which enable one to derive negative properties, are usually built in an *ad hoc* manner. Then we fail to determine the status of more “natural” languages like, for RPOCA, the majority language $\{w \in \{0, 1\}^*: w \text{ has more 1's than 0's}\}$ or the square language $\{ww : w \in \{0, 1\}^*\}$.

4.2 Diagonalization Arguments

In computational complexity, many separation results use diagonalization techniques. Diagonalization also works to separate CA classes, but essentially in an indirect way via the Turing machines. It consists in exhibiting efficient simulations of CA by Turing machines, which allow one to translate Turing machine results into CA results. In this regard, Goldschlager (1982) has shown that whatever its dimension may be, a CA that works in time T can be simulated by a Turing machine in space T .

Fact 1 *For any dimension d and any complexity function T , $d - \text{PCA}(T) \subseteq \text{DSpace}(T)$.*

Because the PCA dependency graphs are regular, the simulation which consists of a depth first search in these graphs with height T , can be performed in space T by a Turing machine. In addition, this result holds as well as for CA as a picture recognizer. As a consequence, for CA in dimension two and higher, separation follows from the Turing machine space hierarchy. In particular, for language and picture recognition, in dimension greater than one, CA working in linear time are strictly less powerful than CA working in unrestricted time and that within the same bounded space.

In the case of restricted communication, a better simulation has been obtained for POCA as language recognizer (Chang et al. 1989).

Fact 2 *For any dimension d , $d - \text{POCA} \subseteq \text{DSpace}(n^{2-1/d})$.*

It allows one to show that, in dimension two and higher, restricted communication reduces the language recognition ability. Precisely, there is a language accepted by a $2 - \text{PCA}$ that cannot be accepted by any $d - \text{POCA}$ whatever the dimension d . Yet the question is still open in the case of PictCA.

5 Comparison with Other Models

In this section, CA is compared with other computational models. Such investigations may help to identify significant features of CA.

5.1 Sequential Models

In order to determine to what extent the use of parallelism provides significant advantages, one of the major concerns is the relationship between parallel computation and sequential computation.

First, efficient simulations of CA by sequential devices permit one to render explicit limitations on the CA computation ability. Notably, as seen in [Sect. 4.2](#), the simulations, which connect CA time complexities with Turing space complexities, entail separation results for CA. But to relate the CA time with the Turing time, there seems to be no better simulation than the trivial one, which states that the work of a CA performed within time T and space S can be done by a Turing machine within time $T \times S$ and space S . For the other representative sequential model, the random access machine, a lower overhead has been obtained in Ibarra and Palis ([1987](#)). By the way of a precomputation phase, it has been shown that any d – POCA working in time $T(n)$ can be simulated by a unit cost random access machine in time $n^d T(n) / \log^{1+1/d} T(n)$.

Conversely, the simulation of sequential devices by CA is a great challenge. What gain may be achieved with CA? For specific sequential problems, the CA computation power is manifest. For instance, it has been observed in Ibarra and Kim ([1984](#)) that real-time POCA contains a language that is P-complete and in Chang et al. ([1988](#)) that the language QBF of true quantified Boolean formulas that is PSpace-complete is in POCA.

But when we try to get general simulations, we come up against the delicate question of whether parallel algorithms are always faster than sequential ones. Indeed, there is no guarantee that efficient parallelization is always possible. Or there might exist a faster CA for each singular sequential solution whereas no general simulation exists. Besides, one can recall Fischer's algorithm to recognize the set $\{1^p : p \text{ is a prime}\}$ (Fischer [1965](#)) and Cole's and Culik's ones seen in [Sect. 2](#). They suggest that clever strategies in parallel are inadequate in sequential and *vice versa*. Then when the conception of efficient parallel algorithms makes use of radically different techniques from the sequential ones, automatic parallelization appears highly improbable.

Hence without surprise, the known simulation of Turing machines by CA provides no parallel speedup. As viewed in the chapter [Universality in Cellular Automata](#) by Ollinger, the early construction of Smith ([1971](#)) mimics one step of computation on a Turing machine (with an arbitrary number of tapes) in one step on a CA (with unbounded space). Furthermore, no effective simulations have been proposed, even for restricted variants. For instance, we do not know whether any finite automata with k heads can be simulated on CA in less than $O(n^k)$ steps, which is the sequential time complexity. And one wonders whether one-way multihead finite automata whose sequential time complexity is linear may be simulated in real time on CA.

In contrast to this great ignorance, the result of Kosaraju is noteworthy (Kosaraju [1979](#)).

Proposition 11 *Any picture language recognized by a four-way finite automata can be recognized in linear time by a PictCA.*

Unfortunately, a complete proof has never been published. Let one just outline the basic idea. The key point is to code the behavior of the finite automaton on a block of size $n \times n$ by a directed graph. A vertex of such a graph is a couple (q, n) where q is an automaton state and n a boundary node of the block. Then the directed graph records for each couple of vertices $((q_1, n_1), (q_2, n_2))$, if, when the automaton enters in state q_1 at boundary node n_1 , it exits in state q_2 at boundary node n_2 . The trick is that the space to record the adjacency matrix of this graph is of the same order as the corresponding block. Furthermore, the adjacency matrix of a block of size $2^i \times 2^i$ can be effectively computed from the four adjacency matrices of the four sub-blocks of size $2^{i-1} \times 2^{i-1}$. To this end, the four adjacency matrices are reorganized in one, the transitive closure is computed and then the new non-boundary nodes are eliminated. Now using this procedure recursively, the adjacency matrix of the whole initial pattern can be computed in linear time.

5.2 Alternating Automata and Alternating Grammars

The correspondence between CA and alternating finite automata was first pointed out by Ito et al. (1989), who showed the equivalence between a particular variant of two-dimensional CA and a restricted type of two-dimensional alternating finite automata. In Terrier (2003a, 2006b), alternating analogues of real-time CA with sequential input mode were given as follows.

Proposition 12

- Real time $d - \text{SCA}$ are equivalent through reverse to real time one-way alternating finite automata with d counters.
- Real time $d - \text{SOCA}$ are equivalent through reverse to one-way alternating finite automata with $d + 1$ heads.

Similarly, for CA with parallel input mode, which implicitly induces a synchronization at initial time, one might search a characterization in terms of one-way synchronized finite automata. Yet, these equivalences are somewhat unsatisfying in the sense that the corresponding types of alternating finite automata provide no further information about the computation power of CA. Moreover, writing algorithms is more intuitive for CA than for alternating devices.

Introducing the notion of alternating grammar, Okhotin (2002) exhibited a characterization of RPOCA, which gives some insight on the relationship between CA and the Chomsky hierarchy. First let one briefly present the alternating grammars. An alternating grammar is a grammar enhanced with a conjunctive operation denoted by $\&$. Each production is of the form $\alpha \rightarrow \alpha_1 \& \dots \& \alpha_k$ where $\alpha, \alpha_1, \dots, \alpha_k$ are strings over a set of variables and terminals. Such a production denotes that the language generated by α is the intersection of the languages generated by $\alpha_1, \dots, \alpha_k$. Analogously to linear context-free grammar, a linear conjunctive grammar is defined as an alternating grammar with the restrictions that for every production $\alpha \rightarrow \alpha_1 \& \dots \& \alpha_k$ α is a symbol and no α_i has more than one instance of variable.

From an algorithm given in Smith (1972), it was already known that POCA are able to recognize in real time every linear context-free languages. Finally, as shown in Okhotin (2002), extending linear context-free grammar with the conjunctive operation $\&$ leads to a complete characterization of RPOCA.

Proposition 13 *The languages recognized in real time by POCA are precisely the languages generated by linear conjunctive grammar.*

5.3 Other Massively Parallel Models

There exist other massively parallel computational models than the CA model. Among them, Boolean circuits, parallel random access machines (PRAM), and alternating Turing machines attract great attention. Curiously enough, CA and these parallel models seem not to recognize the existence of each other. Actually the way in which they modelize parallelism differs on several essential points. On CA, the network structure is homogeneous and the interactions are uniform and local. Furthermore, the d -dimensional array structures usually considered have a constant expansion rate: from a given cell, the number of cells accessible in $2t$ steps is linearly related to the number of cells accessible in t steps. In contrast, constraints on the network structures of uniform Boolean circuits are rather weak. Besides, most of the PRAM variants neglect the communication issue that is the bottleneck in physical machines. Another point of discord is the parallel computation thesis, which states that parallel time is polynomially equivalent to sequential space. This relationship satisfied by Boolean circuits, PRAM, and alternating Turing machines seems not to apply to CA whose network is structured as a d -dimensional array.

Hence, despite their common concern about massively parallel computation, very little is known about their links. At least, it has been proved in Chang et al. (1988) that POCA can simulate linear time bounded alternating Turing machine. As a consequence, remark also that POCA contains $\text{NSpace}(\sqrt{n})$.

6 Questions

Central questions about CA as language recognizer have emerged from the very beginning and up to now remain without answer. To end this chapter, we will go back over some emblematic ones.

6.1 The Linear Time Versus Linear Space Question

The first important issue concerning the recognition power of CA is whether, for one-dimensional space-bounded CA, minimal time is less powerful than unrestricted time (Smith 1972). According to the intuition that more time gives more power, the equality $\text{RPCA} = \text{PCA}$ between minimal time and unrestricted time CA seems very unlikely. But we fail to separate these classes. Actually, this flaw in knowledge is not specific to parallel computation when we think on similar questions in the computational complexity theory such that $L \stackrel{?}{\subseteq} P$ or $P \stackrel{?}{\subseteq} \text{PSpace}$ and more generally when we wonder how do time and space relate.

As a matter of fact, the equality $\text{RPCA} = \text{PCA}$ would imply the equality $P = \text{PSpace}$ since RPCA is included in P and some PSpace -complete language belongs to PCA . More precisely, using padding techniques, it is known from Poupet (2007) that if $\text{RPCA} = \text{PCA}$ (in other words, if $\text{PCA}(f) = \text{DSpace}(f)$ for $f(n) = n$) then for every space constructible function f : $\text{PCA}(f) = \text{DTIME}(f^2) = \text{DSpace}(f)$.

The common difficulty in establishing strict hierarchies lies in the fact that the amount of only one resource (here time) is varying while the amount of a second resource (here space) remains fixed. In that case, classical diagonalization arguments are of no help. And algebraic techniques only work for low-level complexity classes.

6.2 The Influence of the Input Mode

When the communication is two-way, the input mode, either parallel or sequential, does not have a great impact on the recognition time. Indeed, whatever the input mode may be, one is free to rearrange the input in various ways into the space–time diagram within linear time. So PCA and SCA are time-wise equivalent up to linear-time complexity.

The situation is not so simple when the communication is one-way. In this case, there exists a strong restriction for parallel input mode on the access to the input. The key point is that the i th cell of a POCA can only access to the first i symbols of the input, whereas every cell of a SOCA has access to the whole input within linear time. Despite this restriction, POCA and SOCA are equivalent. But taking time into account would make the difference. On the one hand, SOCA simulates POCA without time overhead. On the other hand, the only known algorithm to simulate a SOCA by a POCA is based on a brute-force strategy with an exponential cost even when the SOCA works in linear time (cf. [Sect. 3.4](#)).

Now we may wonder whether there exist less costly simulations by POCA for linear-time SOCA or polynomial-time SOCA. This question echoes another famous one: does RPCA equal LPCA? Indeed RPCA, RSOCA, and LPOCA are equivalent and also LPCA and LSOCA are equivalent. Thus the question whether RPCA is as powerful as LPCA is the same one as whether LPOCA can simulate LSOCA or not. Further, recall the result of Ibarra and Jiang, which relates this question to the closure properties under reverse and under cycle of RPCA (cf. [Sect. 3.5](#)). This result can be restated in this way: LPOCA is as powerful as LSOCA if and only if LPOCA is closed under reverse. Currently, the only idea to recognize on a POCA the reverse or the cycle of a language that is recognized by a LPOCA is the same one that for the simulation of a SOCA by a POCA: the exhaustive strategy, which consists in systematically generating all possible inputs. In other words, the same obstacle is encountered: an exponential cost, which is far from the expected linear cost.

Of course it reinforces the belief that LPCA is strictly more powerful than RPCA. Moreover, a close look at the method used by Ibarra and Jiang to link recognition ability and closure properties suggests that the result could be generalized in the following similar way. For POCA, the amount of time sufficient to simulate an arbitrary SOCA of complexity f equals the amount of time sufficient to recognize the reverse (or the cycle) of an arbitrary language accepted in time f by a POCA. The common difficulty is to make explicit the impact of the manner in which the input is supplied in case of one-way communication.

6.3 The Neighborhood Influence

Another major issue is the impact of the underlying communication graph on the computation ability. First of all, the difference between one-way communication and two-way communication is not well understood. We do not know whether POCA is as powerful as PCA. In fact, a strict inclusion between POCA and PCA would separate linear-time CA and

linear-space CA. But also, as it was stressed in Ibarra and Jiang (1987), it would improve Savitch's theorem. Indeed, $\text{NSpace}(\sqrt{n}) \subseteq \text{POCA} \subseteq \text{PCA} = \text{DSpace}(n)$. Hence to distinguish POCA and PCA would distinguish $\text{NSpace}(\sqrt{n})$ and $\text{DSpace}(n)$. On the other hand, we are far from claiming the equality of POCA and PCA since we do not even know whether POCA is able to simulate PCA working in quasi-linear time.

However, in dimension higher than one, one-way communication sets limits on the language recognition ability. There exists a language accepted by some 2 – PCA, which is accepted by no d – POCA, whatever the dimension d may be (cf. [Sect. 4.2](#)). Curiously enough, we fail to get such a result in the case of picture language recognition. The question of whether the inclusion between PictCA with one-way neighborhoods and PictCA with two-way neighborhoods is proper or not is still open, such as in dimension one.

In many aspects, the situation becomes much more complicated for computation on picture languages. Contrary to dimension one where all neighborhoods are real-time equivalent to either the one-way neighborhood $\{-1, 0\}$ or the two-way neighborhood $\{-1, 0, 1\}$, the recognition ability of PictCA appears more widely influenced by the choice of the neighborhood. For instance, there exists a picture language recognized in real time with the von Neumann neighborhood, which is not recognized in real time with the Moore neighborhood. On the other hand, it is an open question whether all picture languages recognized in real time with the Moore neighborhood are also recognized in real time with the von Neumann neighborhood. More generally, the precise relationships between the various neighborhoods are ignored. And worse, some neighborhoods seem not to admit linear speedup. Actually, the rules that lie behind the communication properties are not simple to grasp. One difficulty is the question of orientation in the two-dimensional underlying communication graph. Notably, Szwerinski's property, which states that a one-dimensional PCA may confuse right and left without time loss and emphasizes that the orientation does not matter in dimension one, seems not to apply to PictCA. And one factor, which might discriminate between the various neighborhoods, would be whether each cell of the array somehow knows the direction toward the output cell.

References

-
- Beyer WT (1969) Recognition of topological invariants by iterative arrays. Technical Report AITR-229, MIT Artificial Intelligence Laboratory, October 1, 1969
- Bucher W, Culik K II (1984) On real time and linear time cellular automata. *RAIRO Theor Inf Appl* 81:307–325
- Buchholz T, Kutrib M (1998) On time computability of functions in one-way cellular automata. *Acta Inf* 35(4):329–352
- Buchholz T, Klein A, Kutrib M (2000) Iterative arrays with small time bounds. In: Nielsen M, Rovan B (eds) MFCS, Lecture notes in computer science, vol 1893. Springer, Berlin, Heidelberg, pp 243–252
- Cervelle J, Formenti E (2009) Algorithmic complexity and cellular automata. In: Meyers RA (ed) Encyclopedia of complexity and system science. Springer, New York
- Chang JH, Ibarra OH, Palis MA (1987) Parallel parsing on a one-way array of finite state machines. *IEEE Trans Comput C-36(1)*:64–75
- Chang JH, Ibarra OH, Vergis A (1988) On the power of one-way communication. *J ACM* 35(3):697–726
- Chang JH, Ibarra OH, Palis MA (1989) Efficient simulations of simple models of parallel computation by time-bounded ATMs and space-bounded TMs. *Theor Comput Sci* 68(1):19–36
- Choffrut C, Culik K II (1984) On real-time cellular automata and trellis automata. *Acta Inf* 21(4): 393–407
- Cole SN (1969) Real-time computation by n-dimensional iterative arrays of finite-state machine. *IEEE Trans Comput* 18:349–365
- Culik K II (1989) Variations of the firing squad problem and applications. *Inf Process Lett* 30(3):153–157

- Culik K II, Gruska J, Salomaa A (1984) Systolic trellis automata. I. *Int J Comput Math* 15(3–4):195–212
- Delacourt M, Poupet V (2007) Real time language recognition on 2D cellular automata: Dealing with non-convex neighborhoods. In: Kucera L, Kucera A (eds) Mathematical foundations of computer science 2007, vol 4708 of Lecture Notes in Computer Science, pp 298–309
- Delorme M, Mazoyer J (1994) Reconnaissance de langages sur automates cellulaires. Research Report 94–46, LIP, ENS Lyon, France
- Delorme M, Mazoyer J (2002) Reconnaissance parallèle des langages rationnels sur automates cellulaires plans. [Parallel recognition of rational languages on plane cellular automata] Selected papers in honour of Maurice Nivat. *Theor Comput Sci* 281(1–2):251–289
- Delorme M, Mazoyer J (2004) Real-time recognition of languages on an two-dimensional Archimedean thread. *Theor Comput Sci* 322(2):335–354
- Dyer CR (1980) One-way bounded cellular automata. *Inf Control* 44(3):261–281
- Fischer PC (1965) Generation of primes by one-dimensional real-time iterative array. *J ACM* 12:388–394
- Giammarresi D, Restivo A (1997) Two-dimensional languages. In: Rozenberg G, Salomaa A (eds) Handbook of Formal Languages, vol 3. Springer, New York, pp 215–267
- Goldschlag LM (1982) A universal interconnection pattern for parallel computers. *J ACM* 29(4):1073–1086
- Hartmanis J, Stearns RE (1965) On the computational complexity of algorithms. *Trans Am Math Soc (AMS)* 117:285–306
- Ibarra OH, Jiang T (1987) On one-way cellular arrays. *SIAM J Comput* 16(6):1135–1154
- Ibarra OH, Jiang T (1988) Relating the power of cellular arrays to their closure properties. *Theor Comput Sci* 57(2–3):225–238
- Ibarra OH, Kim SM (1984) Characterizations and computational complexity of systolic trellis automata. *Theor Comput Sci* 29(1–2):123–153
- Ibarra OH, Palis MA (1987) On efficient simulations of systolic arrays of random-access machines. *SIAM J Comput* 16(2):367–377
- Ibarra OH, Palis MA (1988) Two-dimensional iterative arrays: characterizations and applications. *Theor Comput Sci* 57(1):47–86
- Ibarra OH, Kim SM, Moran S (1985a) Sequential machine characterizations of trellis and cellular automata and applications. *SIAM J Comput* 14(2):426–447
- Ibarra OH, Palis MA, Kim SM (1985b) Some results concerning linear iterative (systolic) arrays. *J Parallel Distrib Comput* 2(2):182–218
- Ito A, Inoue K, Takanami I (1989) Deterministic two-dimensional on-line tessellation acceptors are equivalent to two-way two-dimensional alternating finite automata through 180° rotation. *Theor Comput Sci* 66(3):273–287
- Kobuchi Y (1987) A note on symmetrical cellular spaces. *Inf Process Lett* 25(6):413–415
- Kosaraju SR (1979) Fast parallel processing array algorithms for some graph problems (preliminary version). In ACM conference record of the eleventh annual ACM symposium on theory of computing: papers presented at the symposium, Atlanta, Georgia, ACM Press, New York, 30 April–2 May 1979, pp 231–236
- Klein A, Kutrib M (2003) Fast one-way cellular automata. *Theor Comput Sci* 295(1–3):233–250
- Kutrib M (2001) Automata arrays and context-free languages. In Where mathematics, computer science, linguistics and biology meet, Kluwer, Dordrecht, The Netherlands, pp 139–148
- Leviadli S (1972) On shrinking binary picture patterns. *Commun ACM* 15(1):7–10
- Mazoyer J, Reimen N (1992) A linear speed-up theorem for cellular automata. *Theor Comput Sci* 101(1): 59–98
- Okhotin A (2002) Automaton representation of linear conjunctive languages. In International conference on developments in language theory (DLT), LNCS, vol 6. Kyoto, Japan
- Poupet V (2005) Cellular automata: real-time equivalence between one-dimensional neighborhoods. In Diekert V, Durand B (eds) STACS 2005, 22nd annual symposium on theoretical aspects of computer science, Stuttgart, Germany, February 24–26, 2005, Proceedings, vol 3404 of Lecture Notes in Computer Science. Springer, pp 133–144
- Poupet V (2007) A padding technique on cellular automata to transfer inclusions of complexity classes. In Diekert V, Volkov MV, Voronkov A (eds) Second international symposium on computer science in Russia, vol 4649 of Lecture Notes in Computer Science. Springer, pp 337–348
- Rosenberg AL (1967) Real-time definable languages. *J ACM* 14(4):645–662
- Savage C (1988) Recognizing majority on a one-way mesh. *Inf Process Lett* 27(5):221–225
- Smith AR III (1971) Simple computation-universal cellular spaces. *J ACM* 18(3):339–353
- Smith AR III (1972) Real-time language recognition by one-dimensional cellular automata. *J Comput Syst Sci* 6:233–253
- Szwierinski H (1985) Symmetrical one-dimensional cellular spaces. *Inf Control* 67(1–3):163–172
- Terrier V (1995) On real time one-way cellular array. *Theor Comput Sci* 141(1–2):331–335
- Terrier V (1996) Language not recognizable in real time by one-way cellular automata. *Theor Comput Sci* 156(1–2):281–285

- Terrier V (1999) Two-dimensional cellular automata recognizer. *Theor Comput Sci* 218(2):325–346
- Terrier V (2003a) Characterization of real time iterative array by alternating device. *Theor Comput Sci* 290 (3):2075–2084
- Terrier V (2003b) Two-dimensional cellular automata and deterministic on-line tessellation automata. *Theor Comput Sci* 301(1–3):167–186
- Terrier V (2004) Two-dimensional cellular automata and their neighborhoods. *Theor Comput Sci* 312(2–3): 203–222
- Terrier V (2006a) Closure properties of cellular automata. *Theor Comput Sci* 352(1–3):97–107
- Terrier V (2006b) Low complexity classes of multi-dimensional cellular automata. *Theor Comput Sci* 369(1–3):142–156

5 Computations on Cellular Automata

Jacques Mazoyer¹ · Jean-Baptiste Yunes²

¹Laboratoire d’Informatique Fondamentale de Marseille (LIF),
Aix-Marseille Université and CNRS, Marseille, France
mazoyerj2@orange.fr

²Laboratoire LIAFA, Université Paris 7 (Diderot), France
jean-baptiste.yunes@liafa.jussieu.fr

1	<i>Introduction</i>	160
2	<i>Usual Multiplication Algorithm and CAs</i>	160
3	<i>Real-Time Multiplication</i>	163
4	<i>Prime Numbers Sieve and CAs</i>	168
5	<i>CA Computations with Grids</i>	171
6	<i>Beyond Grids</i>	184

Abstract

This chapter shows how simple, common algorithms (multiplication and prime number sieve) lead to very *natural* cellular automata implementations. All these implementations are built with some *natural* basic tools: signals and grids. Attention is first focussed on the concept of signals and how simple and rich they are to realize computations. Looking closely at the space–time diagrams and the dependencies induced by the computations reveals the concept of grids, and shows how powerful they are in the sense of computability theory.

1 Introduction

This chapter shows how well-known, frequently used algorithms are fundamentally parallel and can be easily implemented on cellular automata (CAs). In fact, programming such simple algorithms with CAs reveals major issues in computability theory. How do inputs enter and outputs exit the machine? How does this affect the programming? What is the concept of time? It also raises questions about what exactly a CA is. Two main examples are considered.

First, the multiplication of two numbers is considered. Though one can consider this as a trivial task, it can be seen that implementing it with CAs is of great interest. It illustrates how CAs can catch the behavior at different scales of different kinds of machines: from circuits made of elementary gates to grids of processors.

Second, the CA implementation of the Eratosthenes prime number sieve is considered. This, unlike the preceding problem, is not a function mapping input values to a corresponding output, but is a “pure” computation, namely, a never ending loop, that produces beeps at times, which are prime numbers.

These considerations stress the importance of the dependency graph in cellular automata. It is sufficient to have a very simple graph to be able to compute any computable function (computable in the sense of Turing’s definition). We also show that any distortion applied to a grid can be useful to compute a functional composition or a function call (in the sense of traditional computer programming).

2 Usual Multiplication Algorithm and CAs

There exist many different algorithms to multiply two numbers. A very common algorithm, which is taught in elementary classrooms of many countries, is considered. It has a very straightforward implementation in cellular automata.

➊ *Figure 1* is a sample of that classical algorithm used to compute 148×274 . Each intermediate result is produced digit by digit from the right to left. The propagation of the necessary carries introduces “natural” dependencies between those digits, the horizontal arrows of ➋ *Fig. 2a*. One observes that in basis 2, no such carry is needed as the multiplicand is either multiplied by 1 or by 0, so that no delay is necessary for the computation of the horizontal lines.

The final result is also produced digit by digit from the right to left, propagating a carry (whatever the basis). A final digit is the last digit of the sum of all digits in its column and the carry of the preceding one – the vertical arrows of ➋ *Fig. 2a*. This induces a “natural” order

Fig. 1

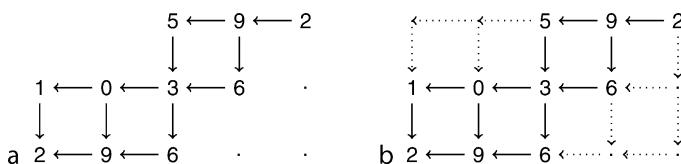
The usual multiplication.

$$\begin{array}{r}
 148 \\
 \times 274 \\
 \hline
 592 \\
 1036. \\
 296. \\
 \hline
 40552
 \end{array}$$

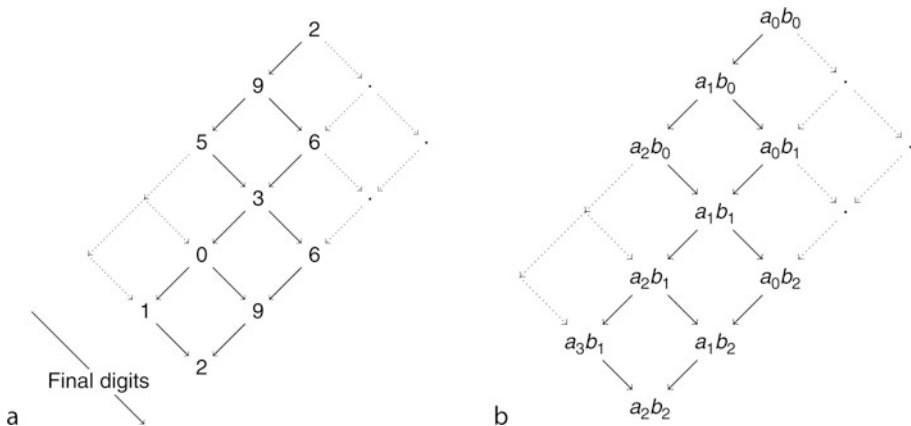
$= 148 \times 4$
 $= 148 \times 70$
 $= 148 \times 200$

Fig. 2

(a) Graph of dependencies. (b) Extended graph.

**Fig. 3**

(a) Example as a trellis CA. (b) Multiplication as a trellis.



on the production of the digits of the result: they are produced from the least to the most significant one.

All those dependencies make the trellis shown in [Figs. 2](#) and [3](#). [Figure 2b](#) shows the full dependency graph of which the only part in [Fig. 2a](#) is used for the particular multiplication. [Figure 2b](#) represents the hand execution of the algorithm. A rotation leads to [Fig. 3](#), which is a classical view of a space-time diagram of a trellis CA. Replacing sample digits by generic ones gives a good hint of what really happens during the process as seen in [Fig. 3b](#).

Of course, there still remains the problem of carry accumulation in the summing process to get a final digit. This leads to unbounded carries that contradict the finite nature of cells in cellular automata. To solve the problem, one can move elementary carries as soon as they are produced.

The details are left to the reader who can refer to [Sect. 3](#) to get an idea of how this can be done. The stress now is rather on what the above analysis tells us about more general concerns in CAs.

2.1 Input/Output Mode

First, one can remark that [Fig. 2b](#) with “vertical” time is not the space–time diagram of a CA, since it would require that the i th digit of the multiplicator be repeated all along the line at time i .

[Fig. 3a](#) shows the convenient trellis CA, which is obtained by changing the arrow of time: the vertical of [Fig. 3b](#) is the diagonal of [Fig. 2b](#).

As shown in [Fig. 3b](#), in this trellis CA, the input mode is parallel for the multiplicand (a_i is located on column i of the trellis) and sequential for the multiplicator (the b_i 's enter cell 0, one after the other). However, this is just a point of view as illustrated by [Fig. 4](#): the two factors can be given as parallel inputs distributed left and right of the “central” cell. Moreover, from that trellis automaton, one can construct a usual linear (non-trellis) CA implementing the same computation as shown in [Fig. 4](#) (retain one line of cells out of two). This construction is in fact quite a general result, as illustrated by [Fig. 5a, b](#).

The output is sequential in the trellis automata: all bits are successively obtained on the “central” cell.

Fig. 4

Parallel inputs on multiplier trellis CA. Dashed arrows illustrate the flow of the bits of the multiplicator, and full arrows the flow of the bits of the multiplicand.

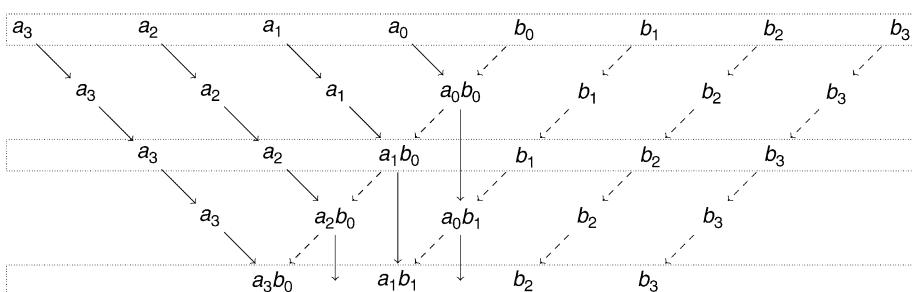
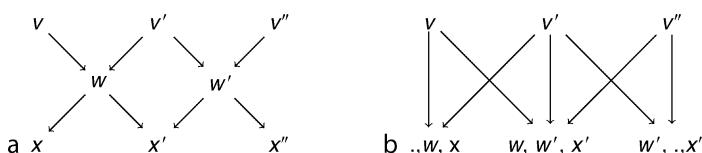


Fig. 5

(a) Trellis CA. (b) Corresponding non-trellis CA.



Thus, with this algorithm, there is a dissymmetry between input and output modes. This prevents any obvious composition of the algorithm with itself in order to iterate multiplications. Therefore, it cannot be considered as an elementary programming block.

2.2 About Time

The pertinent definition of a time step in a trellis CA is illustrated in [Fig. 5](#). What is important is not the definition of time one can superimpose on the graph of dependencies, but the dependencies themselves, so that a time step is the shortest path from a cell to itself. For a more general discussion on this topic, cf. Wolfram's book (see Wolfram 2002, Chap. 9).

This reflects two different views on the topic. First, one is only interested in causalities intrinsic to computation and tries to synthesize a machine. Second, one is interested in programming machines. As seen in the following, these two views are both interesting to consider.

3 Real-Time Multiplication

Atrubin (1965) designed a cellular automaton that was able to multiply two binary numbers in real time, answering McNaughton's problem about the existence of such a machine. Recall that real-time computation was defined by McNaughton (see McNaughton 1961) as the following:

- ▶ If a machine has input sequence i_0, i_1, i_2, \dots and output sequence o_0, o_1, o_2, \dots , where o_k is a function of inputs up to i_k , and if the machine computes o_k within a fixed time limit independent of n following the receipt of i_k , then the computation is said to be accomplished in real time.

Goyal (1976) modified Atrubin's machine so that all the cells were identical (the first cell of Atrubin's machine was slightly different than the other ones). And later on, in 1991, Even (1991) used that construction to compute modular multiplication in linear time.

Knuth (1997) also designed an iterative machine able to multiply in real time.

Of course, designing a fast multiplier circuit is not a new problem as it is one of the elementary building blocks of any computer. Previous references are mentioned just because they focus on models of machines not on circuit design: concerns are not exactly the same.

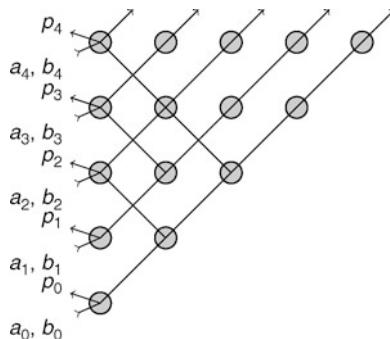
All those solutions use the same key idea: store two pairs of consecutive bits of the factors on each cell. Another way to understand it is to remark that to produce the n least significant bits of the product in real time, the number of elementary products, $a_i \cdot b_j$, that need to be computed is of the order of $n^2/2$. However, from [Fig. 6](#), it is seen that there are only about $n^2/4$ locations in the space–time diagram, so that some grouping is needed.

Here, such constructions are exhibited, which differ in many details of the former ones but fully respect the scheme. The machine has two layers on which different flows of information take place. The first layer is essentially devoted to the space–time layout of the digits of the factors, and the second is used to produce the bits of the product.

In the following, the two factors of the multiplication are denoted by A and B , P the product $A \times B$, and by extension, a_i (resp. b_i , p_i) denotes the i th bit of the binary writing of A (resp. B , P).

Fig. 6

How data flow through the graph. Inputs propagate from bottom-left to top-right. Bits of the product propagate from bottom-right to top-left.



3.1 Input/Output Mode

The machine has a sequential input mode, that is, pairs of corresponding bits of the two factors successively enter the machine. The computation generates two main flows of information as illustrated in [Fig. 6](#) and described below.

Output mode of the machine is also sequential, that is, bits of the results are successively produced on the first cell in real time.

Thus, this machine can be composed with itself.

3.2 Information Flow

There are two main flows of information during the process as illustrated in [Fig. 6](#). First, the input (a_i, b_i) at time i travels from the left to right (bottom-left to top-right arrows in the space-time diagram). Second, the output p_i is generated at time $i + 1$, and its computation is done bit by bit all along a right to left path in the space (bottom-right to top-left arrows in the space-time diagram).

3.3 Propagating Bits of Factors

As illustrated in [Fig. 7](#), there are two internal write-once storages in each cell, say S^0 (at the bottom) and S^1 (at the top). When a value, (a_i, b_i) , enters from the left to a cell, there are three cases:

- If S^0 is empty, then the input is pushed into, storing the value in S^0 .
- Else if S^1 is empty, then the input is pushed into, storing the value in S^1 .
- Else the input is pushed to the right, giving it as input to the next right cell at the next time step (this transient storage will be denoted by T).

Thus, (a_i, b_i) is stored in $S_{[i/2]}^{i \bmod 2}$.

Fig. 7

Layer 1: how factors' bits are stored. One can easily observe that a given input (a_i, b_j) propagates bottom-left to top-right until it finds its right location.

$t = 6$	(a_1, b_1) (a_0, b_0)	(a_3, b_3) (a_2, b_2)	(a_4, b_4)	
(a_6, b_6)	(a_5, b_5)	(a_4, b_4)		
$t = 5$	(a_1, b_1) (a_0, b_0)	(a_3, b_3) (a_2, b_2)		
(a_5, b_5)	(a_4, b_4)			
$t = 4$	(a_1, b_1) (a_0, b_0)	(a_3, b_3) (a_2, b_2)		
(a_4, b_4)	(a_3, b_3)			
$t = 3$	(a_1, b_1) (a_0, b_0)	(a_2, b_2)		
(a_3, b_3)	(a_2, b_2)			
$t = 2$	(a_1, b_1) (a_0, b_0)			
(a_2, b_2)				
$t = 1$	(a_1, b_1) (a_0, b_0)			
(a_1, b_1)				
$t = 0$	(a_0, b_0)			
(a_0, b_0)				

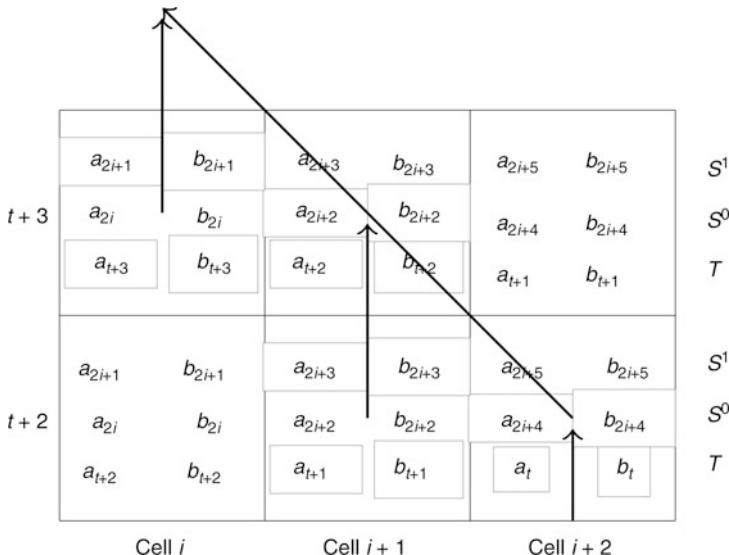
3.4 Accumulating Product Bits

The second layer is used to carry the effective computation of the multiplication. In that layer, the information moves backward to the first cells, accumulating at each time step some new partial computation of the product. In each cell, there are two transient storages, one used to compute the partial product and another to manage the carries.

At a given time, stored values in the first layer are used to compute products accumulated in the second layer. For example, if there is (a, b) in T_b , (a', b') in S_i^1 , and (a'', b'') in S_i^0 then it is possible to compute $a \cdot b' + a' \cdot b$ and $a \cdot b'' + a'' \cdot b$. As time goes, values (a, b) represent all digits with ranks greater than those of (a', b') , of (a'', b'') . Thus, every possible product of digits $a_i \cdot b_j + a_j \cdot b_i$ can be computed as shown below.

Fig. 8

Some cells at time $t + 2$ and $t + 3$ ($t > 2i + 5$). One can remark that all a, b , such that $i + j = k$ which contribute to p_k are located on two successive diagonals. Here, bits contributing to p_{t+2i+4} have been boxed. The way they are cumulated is illustrated by arrows: in each cell necessary elementary products are computed and their values are propagated in order to be added at next time step.



► *Figure 8* shows the contents of the three storages in the layer 1 of the machine at two successive time steps. Boxed contents exhibit the contribution of input bits to the computation of the bit $t + 2i + 4$ of the final product.

At some time $t + 2$ ($t > 2i + 5$), one can observe that one can compute:

$$\begin{cases} a_{t+1} \cdot b_{2i+3} + a_{2i+3} \cdot b_{t+1} & \text{on cell } i + 1 \\ a_t \cdot b_{2i+4} + a_{2i+4} \cdot b_t & \text{on cell } i + 2 \end{cases}$$

and, at time $t + 1$:

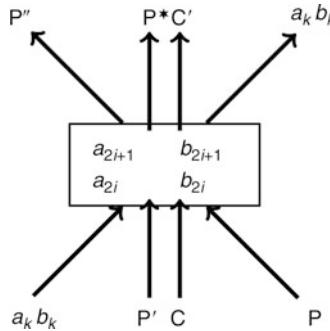
$$\begin{cases} a_{t+3} \cdot b_{2i+1} + a_{2i+1} \cdot b_{t+3} & \text{on cell } i \\ a_{t+2} \cdot b_{2i+2} + a_{2i+2} \cdot b_{t+2} & \text{on cell } i + 1 \end{cases}$$

From this, one can infer that there is a “natural” dependency between cells as illustrated in ► *Fig. 9*, where:

- P is a partial computation of some result bit p_n (with $n = k + 2i$) coming from the right (namely, $\sum_{j=0}^{k-1} a_j \cdot b_{n-j} + a_{n-j} \cdot b_j \bmod 2$).
- P' is a contribution to bit p_n computed at the previous time step (namely, $a_{k-1} \cdot b_{2i+1} + a_{2i+1} \cdot b_{k-1}$).
- P'' is a partial computation of bit p_n (namely, $\sum_{j=0}^k a_j \cdot b_{n-j} + a_{n-j} \cdot b_j \bmod 2$).
- P^* is a contribution to bit p_{n+1} (namely, $a_k \cdot b_{2i+1} + a_{2i+1} \cdot b_k$).

Fig. 9

The multiplier.



- C is the carry of the partial computation of bit p_{n-1} computed at a previous time step.
- C' is the carry of the partial computation of bit p_n which is sent to the computation of bit p_{n+1} .

This computation follows the relations (which more generally are written in base b)

$$\left(\sum_{i=0}^n d_i \right) \bmod b = \left(\left(\sum_{i=0}^{n-1} d_i \right) \bmod b + d_n \right) \bmod b \quad (1)$$

and

$$\left(\sum_{i=0}^n d_i \right) / b = \left(\sum_{i=0}^{n-1} d_i \right) / b + \left(\left(\sum_{i=0}^{n-1} d_i \right) \bmod b + d_n \right) / b \quad (2)$$

(where $/$ denotes integer division). Thus, bits of the product are produced in real time.

To be compliant with the definition of cellular automata and their intrinsic finiteness, one must verify that P , P' , P'' , P^* , C , and C' always carry a finite amount of information. The corresponding equations are

$$P'' = (a_k b_{2i} + a_{2i} b_k + C + P + P') \bmod 2 \quad (3)$$

$$P^* = (a_k b_{2i+1} + a_{2i+1} b_k) \quad (4)$$

$$C' = (a_k b_{2i} + a_{2i} b_k + C + P + P') / 2 \quad (5)$$

Observe that P'' is a single bit, and that P''' is two bits wide (three values). From [Eqs. 4](#) and [5](#) it is seen that there are at most four different possible values since $C = 4$ is the smallest solution of $(C + 5)/2 \leq C$. Thus, C and C' consist of 3 bits.

3.5 Boundary Conditions

Two special cases have to be considered when the storages of a_i , b_i occur in layer 1: this corresponds to the initiation of computations of the partial products.

First, when a cell is empty and a value (a_{2k}, b_{2k}) enters in (remember that if a cell number i is empty, the first value to store in is a pair of even indexed bits), the value $a_{2k} \cdot b_{2k}$ is produced as P' , initiating the production of bit $4k$ of the output.

Second, when a cell is half-empty and a value (a_{2k+1}, b_{2k+1}) enters in, the value $a_{2k+1} \cdot b_{2k+1}$ is produced as C' , initiating the production of bit $4k + 2$ of the output.

3.6 Analysis

It is shown that even if multiplying this way seems to be very tricky at first look, one can decompose the machine as the “superposition” of three different very simple machines. This is similar to modularization in programming. The three simpler machines are:

- A storing machine which places every bits of the factors at the right place
- A bit producing machine which computes every single elementary product
- A parallel additioner which sweeps the space–time to gather every bit and adds them appropriately.

4 Prime Numbers Sieve and CAs

Searching or enumerating prime numbers is a mathematical activity, at least as old as Eratosthenes’s algorithm which is a very elementary method to get, by exclusion, all prime numbers up to n :

- ▶ Write the sequence of every integer from 2 to n . Check every proper multiple of 2, then every proper multiple of 3, etc. The unchecked integers are the prime ones.

One sees that this very simple algorithm can be implemented with CAs. Such work was first done by Fischer (1965) and intensively studied later on, around 1990, by Korec (1997).

All these implementations heavily use the concept of signal. Signals are very important when programming CAs, it is commonly considered as an elementary brick to build a CA program. A signal can be roughly defined here as a *moving information* whose trace in the space–time diagram is generally a connected path. This is a very “natural” concept, and very simple signals are used in implementation of prime number sieves.

These implementations use clocks. If a clock has a period of length p steps, it is obvious to remark that it marks every multiple of p , if it has been set at time p . Thus combining multiple clocks, one is able to mark every multiple of every integer, as in Eratosthenes’s algorithm. Clocks are implemented with signals.

It is important to note that this computation is not a function, as commonly defined in mathematics. There is no input and no output, that is, no value is introduced nor produced. The result is a behavior, as one can observe that something happens during the computation. In the example, the computation will leave some cell in a given remarkable state at all prime times. Nevertheless, one can look at this behavior as the characteristic function of the set of prime numbers: identify all states associated with the production of a prime number to 1 and all other ones to 0. Otherwise said, the language $\{1^p | p \text{ prime}\}$. And this is very different of many computations, where more traditional mathematical functions are realized (see, e.g., [Sect. 2](#)).

Fig. 10

A naive prime number sieve. Dotted lines represent the newly launched clock. At time step i the new clock has a period of length i .

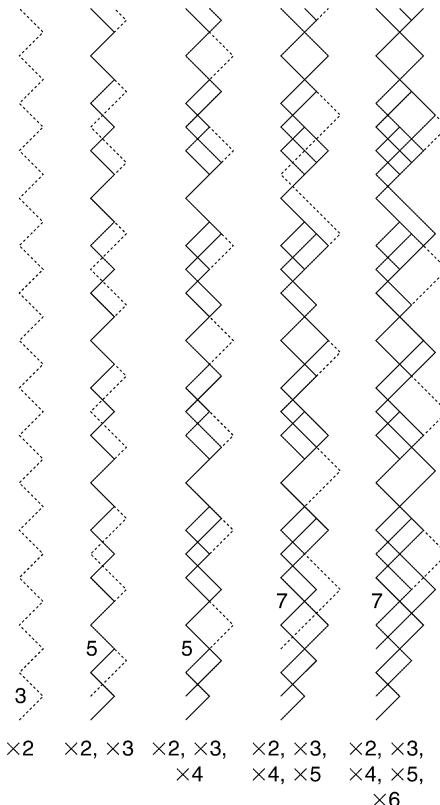


Figure 10 illustrates how Eratosthenes's idea leads to a naive implementation that could not be implemented in CA's. This construction is a simple superposition of all appropriate clocks.

First, at time 2, a clock of period 2 is launched such that every even tick is marked. Such a clock is implemented *via* a signal (an information) which goes to the right one unit of time and then goes back one unit of time, thus bouncing on the initial cell one tick over two.

At time 3, a clock of period 3 is started. At the same time, it can be observed that no signal has entered the first cell showing that this tick is a prime integer (indeed 3 is prime). The clock of period 3 is implemented with a signal going to the right one half more than the previous zigzag signal (of period 2), since the difference of half the periods of the two signals is 0.5. It may seem strange to move an information one half in the two directions since space–time is discrete in both dimensions, but that can be simply simulated going one time in the chosen direction, staying one time at the same place. Such a pause may be viewed as an internal simulation of two half opposite moves.

At time 4, a clock of period 4 is started, and so on.

But this cannot be computed by CA, as it is easy to remark that unbounded many signals need to be started as time grows (at times, in cases such as $\prod_{i=0}^k i$, k signals have to be differentiated), which is not possible with a finite automaton.

One would also like to simplify the process such that only necessary signals have to be launched (signals of periods p for any p prime). But at times such like $\prod_{i=0}^k p_i$, k signals should also be launched. Another construction should be necessary to construct right borders (as this will not correct the problem, how these walls can be constructed is not shown and this is left as an exercise to the curious reader).

To solve the problem, one simply needs to create disjoint corridors in which individual clocks are built. As one can remark, the problem is to synchronize all those separated clocks in such a way that ticks can be reported at the right time on the first cell. \bullet *Figure 11a* shows how a clock running into a corridor located far away from the first cell generates delayed ticks.

\bullet *Figure 12* illustrates the whole process involved in the “computation” of prime numbers. As in more traditional programming and mathematical reasoning, the whole problem is decomposed into a few simpler problems and then reassembled.

First, the construction of corridors is set up. Suppose height n has been built, then \bullet *Fig. 11b* shows how corridor with width n is built and how height $n + 1$ is built on the appropriate cell.

Second, in each corridor of width n , a clock of period $2n + 1$ will be set up to mark multiples of $2n + 1$ on the first cell. As distance from the left side of the corridor of width n to the first cell is $\sum_{i=1}^{n-1} i$ one needs to set up the clock of period $2n + 1$ such that it beats its corridor’s left side at times $k \times (2n + 1) - \sum_{i=1}^{n-1} i$.

\blacksquare **Fig. 11**

(a) Corridors and delays. (b) Next corridor.

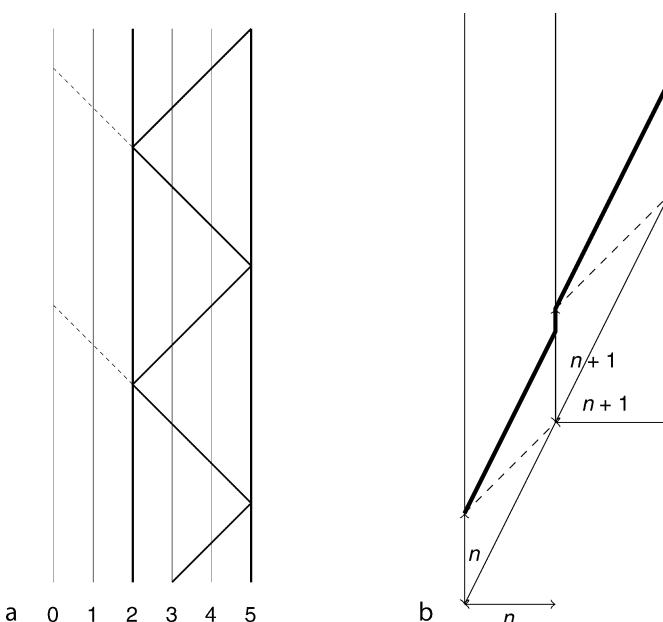
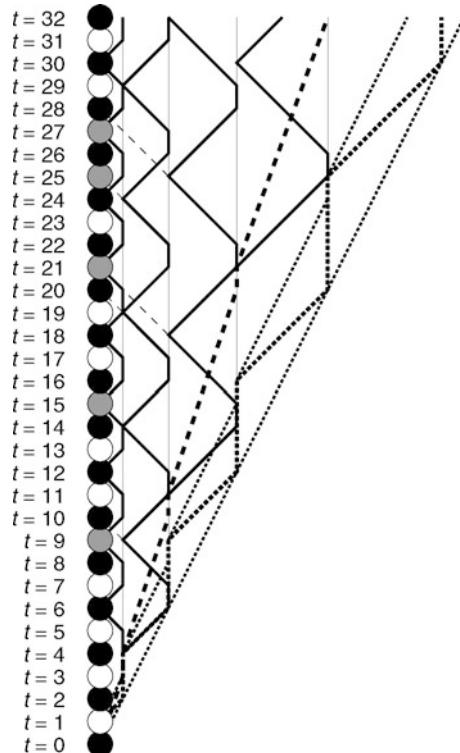


Fig. 12

Prime number sieve construction. Black disks represent “even ticks.” Gray disks represent “composite non-even ticks.” White disks correspond to “prime ticks.”



5 CA Computations with Grids

5.1 Dependencies

As seen in [Sects. 2](#) and [3](#) about multiplication on CA, to compute on CA is to construct and manage different flows of information. To multiply, two main flows are built so that, at their meeting, elementary computations are made. The computations are chained together following the “natural” dependency graph of the space–time diagram of a linear cellular automaton with neighborhood of nearest cells, $v = \{-1, 0, +1\}$. Obviously, the number of information flows is only bounded by imagination.

In this construction, the key idea is to manage the meeting of the first two flows so that a new flow is generated which, in its turn, meets the third flow. One can remind that these constructions are very similar to techniques used in systolic algorithms, up to one more constraint: the graph of data moves must be injected into the dependency graph of the cellular automaton.

Determining this dependency graph is a very important question to be solved. Historically, two approaches emerged. The first one is due to Cole ([1969](#)) who showed that every

one-dimensional CA with any arbitrary neighborhood can be simulated by a 1D-CA with neighborhood $v = \{-1, 0\}$ or $v = \{0, +1\}$. The second one has been introduced by Čulik (1982) and is now known as trellis automaton with neighborhood $v = \{-1, +1\}$. In fact, these two approaches are very similar and tightly related (see Choffrut and Čulik 1984).

The basic idea is as follows:

- ▶ It is sufficient to have two independent communication channels to be able to simulate any one-dimensional CA whatever be its neighborhood.

To understand how these approaches are intrinsically the same, the following monoid is introduced:

$$\langle e, a, a^{-1}, \alpha_{-\ell}, \dots, \alpha_0, \dots, \alpha_r | aa^{-1} = a^{-1}a = e; \alpha_i = \alpha_0 a^i = a^i \alpha_0 \rangle \quad (6)$$

where e is the neutral element, a and a^{-1} generate the initial line, and the influence of cell $c - i$ onto cell c is represented by generator α_i with $-\ell \leq i \leq r$. The dependency graph of a CA with a neighborhood $v = \{-\ell, \dots, -1, 0, +1, \dots, r\}$ is obtained from the Cayley graph of the monoid (❷ Eq. 6) by forgetting all arcs labeled a or a^{-1} .

In case of the usual neighborhood $v = \{-1, 0, +1\}$, the above monoid is

$$\langle e, a, a^{-1}, \alpha_{-1}, \alpha_0, \alpha_1 | aa^{-1} = a^{-1}a = e; \alpha_1 = \alpha_0 a = a \alpha_0; \alpha_{-1} = \alpha_0 a^{-1} = a^{-1} \alpha_0 \rangle \quad (7)$$

In the case of neighborhood $\{0, +1\}$ (resp. $\{-1, 0\}$, resp. $\{-1, +1\}$), the associated monoid is

$$\langle e, b, b^{-1}, \beta, \gamma | bb^{-1} = b^{-1}b = e; \gamma = b\beta = \beta b \rangle \quad (8)$$

where $b = a$, $\beta = \alpha_0$, and $\gamma = \alpha_1$ (resp. $b = a$, $\beta = \alpha_{-1}$, and $\gamma = \alpha_0$, resp. $b = a^2$, $\beta = \alpha_{-1}$, and $\gamma = \alpha_1$).

Now, the dependency graph associated with ❷ Eq. 7 can be embedded into the dependency graph associated with ❷ Eq. 8 using the relations $\alpha_0 = \beta\gamma = \gamma\beta$, $\alpha_1 = \beta^2$, and $\alpha_{-1} = \gamma^2$.

Consider that the inputs of the computation are not located on the initial line (cells $(x, 0)$), but on the two main diagonals issued from the initial cell $(0, 0)$ (i.e. cells $(x, |x|)$), then the dependency graph associated by ❷ Eq. 8, illustrated by ❷ Fig. 13a, is the one associated to the monoid:

$$\langle e, \alpha, \beta | \alpha\beta = \beta\alpha \rangle \quad (9)$$

The embedding, in the general case, is left to the reader.

In the following, we will sometimes say that α is *stepping to the right*, β is *stepping to the left*, and their product ($\alpha\beta$ or $\beta\alpha$) is *staying in place* (cf. ❷ Fig. 13b).

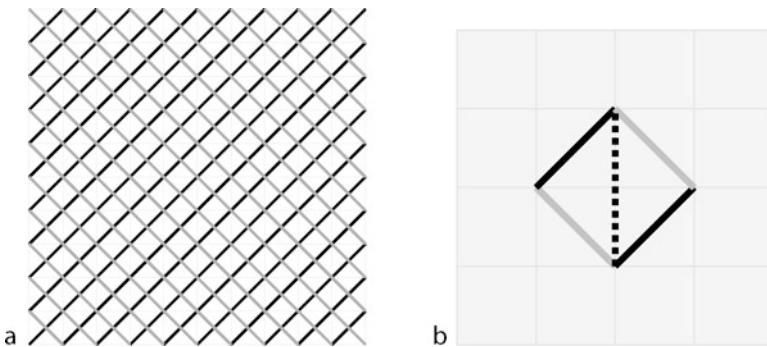
The proof of the main Theorem 1 shows how the simulation of an automaton with neighborhood $\{-1, 0, +1\}$ can be done on such a grid, with an appropriate distribution of the initial configuration.

5.2 First Approach to the Notion of Grids

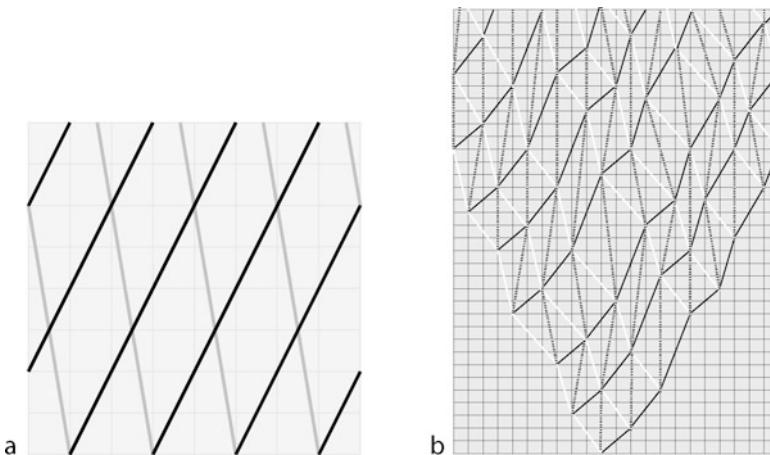
From an initial finite line of cellular automaton with neighborhood $\{-1, 0, +1\}$, one can construct many graphs associated to monoid of ❷ Eq. 9. Here, “construct” informally means drawing with signals in the space-time diagram (see the chapter ❷ Algorithmic Tools on Cellular Automata); where signals correspond to the generators of the semigroup (i.e., α, β) and their meeting corresponds to the elements of the semigroup.

Fig. 13

(a) Graph induced by the monoid $\{e, \alpha, \beta | \alpha\beta = \beta\alpha\}$ (point e not shown). (b) Relation $\alpha \cdot \beta = \beta \cdot \alpha$, otherwise said “stay in place is as the same as moving one step to the right then one step to the left (or the converse).”

**Fig. 14**

(a) A rational grid. (b) A nonrational grid (with point e).



Such a graph is called a **grid**. It will be more formally defined in [Sect. 5.3](#).

If the site corresponding to the element of the semigroup $\alpha^p\beta^t$ is denoted by (p, t) , then, one observes that:

- Given p_e, t_e, λ, μ , one can construct a **rational grid** so that $(p, t) = (p_e + p \cdot \lambda, t_e + t \cdot \mu)$, see [Fig. 14a](#).
- Even **nonrational grids** are constructible (see [Fig. 14b](#)).
- Not every recursive grid is constructible as the value of $(0, t)$ only depends on the value of sites $(-t, 0), \dots, (0, 0), \dots, (t, 0)$, and is then constructible in time by a Turing machine computing with a semi-infinite ribbon and simulating the cellular automaton from the following initial configuration:

$$(\langle 0, 0 \rangle, \star), (\langle 1, 0 \rangle, \langle -1, 0 \rangle) \dots (\langle t, 0 \rangle, \langle -t, 0 \rangle)$$

5.3 Definitions

Now, the definition of what is really meant by constructing or drawing a grid in the orbit of a particular configuration of a cellular automaton with neighborhood $\{-1, 0, +1\}$ is given. In the following, the concept of **signals**, introduced in the chapter [Algorithmic Tools on Cellular Automata](#), is used intensively (see Adamatzky 2002; Delorme and Mazoyer 1999; Poupet 2005; Mazoyer and Terrier 1999; [Definitions 6](#) and [8](#)), but, the main goal to compute, these definitions will be restricted a little bit.

Definition 1 (consequences) The *immediate consequence* $\mathbb{C}(\mathcal{X})$ of $\mathcal{X} \subset \mathbb{Z} \times \mathbb{N}$ is defined as the set of points $(x, y) \in \mathbb{Z} \times \mathbb{N}$, such that $(x, y) \notin \mathcal{X}$ but such that all three sites $(x - 1, y - 1)$, $(x, y - 1)$, and $(x + 1, y - 1)$ are in \mathcal{X} .

The *consequence* of \mathcal{X} is defined as $\mathbb{C}^*(\mathcal{X}) = \lim_{n \rightarrow \infty} U_n$, where $U_0 = \mathcal{X}$ and for all $n > 0$, $U_{n+1} = U_n \cup \mathbb{C}(U_n)$.

Definition 2 (influences) The *immediate influence* $\mathbb{I}(\mathcal{X})$ of $\mathcal{X} \subset \mathbb{Z} \times \mathbb{N}$ is defined as the set of points $(x, t) \in \mathbb{Z} \times \mathbb{N}$, such that at least one of $(x - 1, t - 1)$, $(x, t - 1)$, or $(x + 1, t - 1)$ is in \mathcal{X} .

The *influence* of \mathcal{X} is defined as $\mathbb{I}^*(\mathcal{X}) = \lim_{n \rightarrow \infty} U_n$, where $U_0 = \mathcal{X}$ and for all $n > 0$, $U_{n+1} = U_n \cup \mathbb{I}(U_n)$.

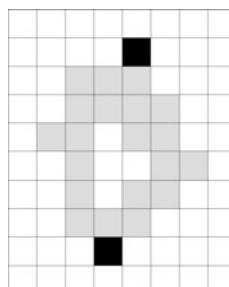
Definition 3 (pluggable set) $\mathcal{X} \subset \mathbb{Z} \times \mathbb{N}$ is an *input/output pluggable set* if, letting t_0 be the least t such that $\exists x (x, t) \in \mathcal{X}$,

- There exists an **input point** $(x_0, t_0) \in \mathcal{X}$ such that $\mathcal{X} \cap \{(x, t) | x \in \mathbb{Z}, t > t_0\} \subset \mathbb{I}^*(\{(x_0, t_0)\})$
- There exists an **output point** $(x_f, t_f) \in \mathcal{X}$ such that $\mathcal{X} \cap \{(x, t_f) | x \in \mathbb{Z}\} = \{(x_f, t_f)\}$
- $\mathcal{X} \cap \{(x, t) | x \in \mathbb{Z}\} \subset \mathbb{I}(\mathcal{X} \cap \{(x, t - 1) | x \in \mathbb{Z}\})$ for all t such that $t_0 < t \leq t_f$

[Figure 15](#) shows what a pluggable set looks like. One observes that, by definition, either there is a unique input point, or there are two input points (two direct neighbors or at distance 1) or there are three consecutive inputs points.

Fig. 15

A pluggable set. The two black points are the input and the output points of the set.



Definition 4 (signals) Given a cellular automaton $\mathcal{A} = (Q, \delta, \{-1, 0, +1\})$, $\mathcal{S} \subset \mathbb{Z} \times \mathbb{N}$ is a *signal* over $S \subset Q$, if

- \mathcal{S} is indexed by an initial segment of \mathbb{N} . $\mathcal{S} = (x_k, t_k)_{k \in \{0, \dots, \ell_S\}}$ where ℓ_S is either an integer or $+\infty$ and $t_{k+1} = t_k + 1$.
- $\langle x_k, t_k \rangle$, the state of the cell (x_k, t_k) , verifies $\langle x_k, t_k \rangle \in S$.
- \mathcal{S} is connected so that if $(x_k, t_k) \in \mathcal{S}$ either $x_{k+1} = x_k$, $x_{k+1} = x_k - 1$, or $x_{k+1} = x_k + 1$.
- If $(x, t) \notin \mathcal{S}$ and at least one of its eight neighbors $(x-1, t)$, $(x+1, t)$, $(x, t-1)$, $(x, t+1)$, $(x-1, t-1)$, $(x-1, t+1)$, $(x+1, t-1)$, or $(x+1, t+1)$ is in \mathcal{S} , then its state $\langle x, t \rangle$ is not in S .

The point (x_0, y_0) is the **initial point** of signal \mathcal{S} .

If ℓ_S is finite, (x_{ℓ_S}, y_{ℓ_S}) is the **final point** of signal \mathcal{S} , which will be conveniently denoted by (x_f, y_f) .

If ℓ_S is finite, a signal is called a **left signal** (resp. **right signal**) if $x_f - x_0 < 0$ (resp. $x_f - x_0 > 0$).

It is worth noticing that from a unique configuration, it is possible to construct many signals defined over the same set of states.

Definition 5 (interacting signals) Signals \mathcal{S} and \mathcal{T} *interact* if

- $(x_{f_S}, y_{f_S}) = (x_{f_T} + 2, y_{f_T})$ and the interacting site is $(x_{f_S} + 1, y_{f_S} + 1)$.
- Or $(x_{f_S}, y_{f_S}) = (x_{f_T} + 1, y_{f_T})$ and the interacting sites are $(x_{f_S}, y_{f_S} + 1)$ and $(x_{f_T}, y_{f_T} + 1)$.
- Or $(x_{f_S}, y_{f_S}) = (x_{f_T}, y_{f_T})$ and the interacting site is $(x_{f_S}, y_{f_S} + 1)$.

The three cases are illustrated by Fig. 16.

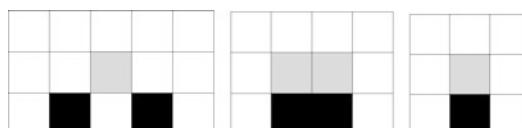
Definition 6 (signals creation) Given a cellular automaton $\mathcal{A} = (Q, \delta, \{-1, 0, +1\})$, a signal \mathcal{S} over S is created by a pluggable set $\mathcal{X} \subset \mathbb{Z} \times \mathbb{N}$ with respect to an output point (x_{f_S}, y_{f_S}) of \mathcal{X} , if (x_{0_S}, y_{0_S}) is in the immediate consequences of (x_{f_S}, y_{f_S}) , that is, the input point of \mathcal{S} is in the immediate consequence of the output point of \mathcal{X} .

Definition 7 (symbol encoding) A cellular automaton $\mathcal{A} = (Q, \delta, \{-1, 0, +1\})$ can encode symbols of alphabet A with μ channels if $Q = Q' \times (A \cap \{\varepsilon\})^\mu$ (with ε being a state used to denote the lack of encoding on the corresponding channel).

Definition 8 (grids) Given a cellular automaton $\mathcal{A} = (Q, \delta, \{-1, 0, +1\})$ and three disjoint subsets L, R , and W of Q , it can be said that \mathcal{A} constructs a grid $\mathcal{G}_{L,R,W}$ if there are three integers x_{max} , h_{max} and t_{max} such that:

Fig. 16

Possible signal interactions. The three different cases correspond to the three possible relative positions of the final points of the interacting signals.

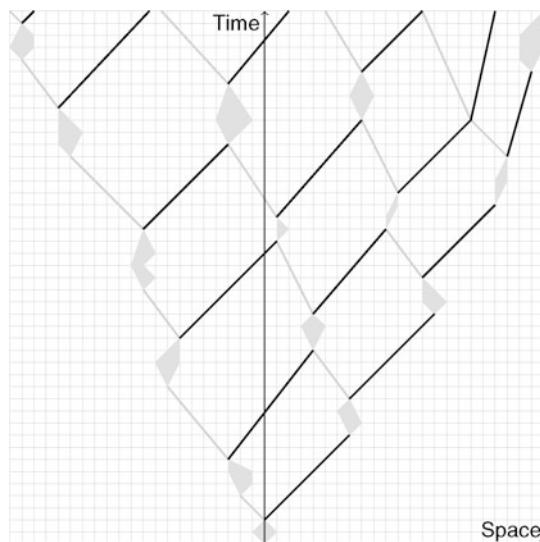


1. There exist pluggable sets $D_{(n,m)}$ of $\mathbb{Z} \times \mathbb{N}$ over W (with $n, m \in \mathbb{N}$) so that:
 - (a) The input point $(x_{0_{(n+1,m)}}, t_{0_{(n+1,m)}})$ of $D_{(n+1,m)}$ is such that $x_{0_{(n+1,m)}} - x_{0_{(n,m)}} \geq x_{max}$ and $t_{0_{(n+1,m)}} - t_{0_{(n,m)}} \geq t_{max}$.
 - (b) The input point $(x_{0_{(n,m+1)}}, t_{0_{(n,m+1)}})$ of $D_{(n,m+1)}$ is such that $x_{0_{(n,m)}} - x_{0_{(n,m+1)}} \geq x_{max}$ and $t_{0_{(n,m+1)}} - t_{0_{(n,m)}} \geq t_{max}$.
 - (c) The output point $(x_{f_{(n,m)}}, t_{f_{(n,m)}})$ of $D_{(n,m)}$ is such that $|x_{f_{(n,m)}} - x_{0_{(n,m)}}| \leq x_{max}$ and $t_{f_{(n,m)}} - t_{0_{(n,m)}} \leq t_{max}$.
2. There exists an infinite family of *left signals* $\mathcal{L}_{(n,m)}$ over L , for n, m in \mathbb{N} , so that $\mathcal{L}_{(n,m)}$ is created by $D_{(n,m)}$. These signals are called *left moves*.
3. There exists an infinite family of *right signals* $\mathcal{R}_{(n,m)}$ over R , for n, m in \mathbb{N} , so that $\mathcal{R}_{(n,m)}$ is created by $D_{(n,m)}$. These signals are called *right moves*.
4. For n, m in \mathbb{N}^* , $\mathcal{L}_{(n,m-1)}$ and $\mathcal{R}_{(n-1,m)}$ interact, and $(x_{0_{(n,m)}}, t_{0_{(n,m)}})$ the entry point of $D_{(n,m)}$ is one of the interacting sites (recall that there are one or two interacting sites).
5. (*Case of sites on the first right diagonal of the grid*) for $n \in \mathbb{N}^*$, $(x_{0_{(n,0)}}, t_{0_{(n,0)}})$ is in the immediate consequences of $(x_{f_{\mathcal{R}_{(n,0)}}, t_{f_{\mathcal{R}_{(n,0)}}}})$.
6. (*Case of sites on the first left diagonal of the grid*) for $m \in \mathbb{N}^*$, $(x_{0_{(0,m)}}, t_{0_{(0,m)}})$ is in the immediate consequences of $(x_{f_{\mathcal{L}_{(0,m)}}, t_{f_{\mathcal{L}_{(0,m)}}}})$.
7. Any left (resp. right) diagonal segment of a left signal $\mathcal{L}_{(0,i)}$ (resp. right signal $\mathcal{R}_{(i,0)}$) has length bounded by h_{max} .
8. The ending point of signal $\mathcal{L}_{(0,i)}$ (resp. $\mathcal{R}_{(i,0)}$) is on the left (resp. right) border of the left hull of $\mathcal{L}_{(0,i)}$ (resp. right hull of $\mathcal{R}_{(i,0)}$).

The initial point of $\mathcal{G}_{L,R,W}$ is $(x_{0_{(0,0)}}, t_{0_{(0,0)}})$ the initial point of $D_{(0,0)}$.

Fig. 17

A continuous geometric representation of a grid. Subsets $D_{(n,m)}$ are represented by polygons filled in gray, right moves are segments drawn in black and left ones in gray. $2x_{max}$ and t_{max} are, respectively, the width and height of enclosing rectangles in which $D_{(n,m)}$ are embedded.



➊ *Figure 17* illustrates the geometrical representation of a grid. In the space–time diagram, $D_{(n,m)}$ ’s (indexed by $\mathbb{N} \times \mathbb{N}$) can be seen as *rough points*. Note that the size of these points is bounded due to conditions ➋ 1c of ➋ Definition 8 (this can be relaxed in a more general case).

One can imagine that a grid is a grillage, turned clockwise by 90° and then deformed. Signals \mathcal{L} and \mathcal{R} are wires that connect two neighboring cells in the grid. \mathcal{L} connects cells with the same first coordinate, while \mathcal{R} connects cells with the same second coordinate. ➋ Figures 13a and ➋ 14a, b illustrate different grids with $x_{\max} = t_{\max} = 0$ (domains are reduced to their minimum: a simple interacting site).

Some intuition about the $D_{(n,m)}$ ’s is given. If such grids are considered as the support of some computation, the $D_{(n,m)}$ ’s are where local transition function computations occur. They are created by the interaction (as in ➋ Definition 5) of two signals \mathcal{L} and \mathcal{R} (item 4 of ➋ Definition 8), except those with a null coordinate that are created *ex-nihilo* from the initial configuration. If signals $\mathcal{L}_{(n,m-1)}$ and $\mathcal{R}_{(n-1,m)}$ carry values (see ➋ Definition 7) to the entry point of some $D_{(n,m)}$, then a computation can take place inside $D_{(n,m)}$, which produces a result that is finally encoded into its output point and transmitted to the neighbors through $\mathcal{L}_{(n,m+1)}$ and $\mathcal{R}_{(n+1,m)}$ that are created from the output point. The full computation is initiated on sites $D_{(n,0)}$ ($n \in \mathbb{N}$) and $D_{(0,m)}$ ($m \in \mathbb{N}$) and this initialization is not related to grids.

A grid can also be interpreted as a trellis cellular automaton where values $\cdots a_{-m} \cdots a_{-1} a_0 a_1 \cdots a_n \cdots$ are encoded (see ➋ Definition 7 below) as:

- a_i is encoded into states of signal $\mathcal{R}_{(i,0)}$, for $i \geq 0$.
- a_i is encoded into states of $\mathcal{L}_{(0,i-1)}$, for $i < 0$.

Thus, every $D_{(n,0)}$ (resp. $D_{(0,m)}$) gets a value encoded in $\mathcal{R}_{(n-1,0)}$ (resp. $\mathcal{L}_{(0,m-1)}$), and sends and encodes it in $\mathcal{L}_{(n,1)}$ (resp. in $\mathcal{R}_{(1,m)}$) but not in $\mathcal{R}_{(n,0)}$ (resp. $\mathcal{L}_{(0,n)}$).

Each $D_{(n,m)}$ ’s ($n, m \in \mathbb{N}^*$) receives two values sent by $\mathcal{L}_{(n,m-1)}$ and $\mathcal{R}_{(n-1,m)}$, computes a new value in $D_{(n,m)}$ using the states of W and sends the result through $\mathcal{L}_{(n,m)}$ and $\mathcal{R}_{(n,m)}$.

Now, given the preceding considerations, the main proposition about grids is introduced. In the following, without loss of generality, cellular automata is considered with a fixed neighborhood $\{-1, 0, +1\}$.

Theorem 1 (CA simulation on grids) *Suppose given \mathcal{B} and an initial configuration $c_{\mathcal{B}}$ which constructs a grid \mathcal{G} in the sense of ➋ Definition 8. For every cellular automaton \mathcal{A} there exists a CA \mathcal{C} and a delay $d_{\mathcal{A}} \in \mathbb{N}$ such that for every initial configuration $c_{\mathcal{A}}$, there exists a configuration $c_{\mathcal{C}}$ so that automaton \mathcal{C} on $c_{\mathcal{C}}$ constructs the grid \mathcal{G} vertically translated by $d_{\mathcal{A}}$ on which output points of $D_{(n,m)}$ encode state $\langle n - d_{\mathcal{A}}, \min\{n, m\} \rangle_{\mathcal{A}}$ of the space–time diagram of \mathcal{A} on $c_{\mathcal{A}}$.*

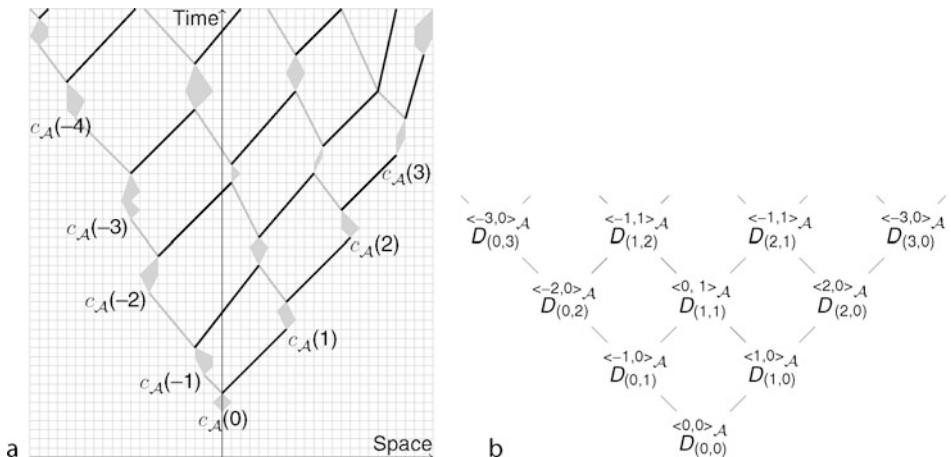
Moreover, \mathcal{C} depends on \mathcal{A} and \mathcal{B} (not on $c_{\mathcal{B}}$), and the maps $\mathcal{A} \mapsto \mathcal{C}$, and $c_{\mathcal{A}}, c_{\mathcal{B}} \mapsto c_{\mathcal{C}}$ are computable.

Proof The key ideas are as follows:

1. The initial configuration of \mathcal{A} is distributed on the initial configuration of the grid constructed by \mathcal{B} , so that the state of the output point of $D_{n,0}$ encodes state $\langle n, 0 \rangle_{\mathcal{A}}$, whereas that of $D_{0,n}$ encodes state $\langle -n, 0 \rangle_{\mathcal{A}}$ (see ➋ Fig. 18a). This necessitates to wedge $c_{\mathcal{A}}(0)$ with the entry point of $D_{0,0}$ (hence the delay $d_{\mathcal{A}}$), and then to appropriately distribute the remaining parts of $c_{\mathcal{A}}$ all along the borders of the grid.

Fig. 18

(a) Distribution of configuration $c_{\mathcal{A}}$ on the grid generated by automaton \mathcal{B} . (b) How sites of \mathcal{A} are located on domains D .



- The simulation of automaton \mathcal{A} with neighborhood $\{-1, 0, +1\}$ is done using an automaton computing on a grid generated by \mathcal{B} on configuration $c_{\mathcal{B}}$, which is exactly the grid generated by the monoid of [Eq. 9](#). In that simulation, the state $\langle n, t \rangle_{\mathcal{A}}$ (with $n \in \mathbb{N}$) is encoded into the state of the output point of $D_{(n+t,t)}$ whereas the state $\langle -n, t \rangle_{\mathcal{A}}$ is encoded into the state of the output point of $D_{(t,n+t)}$, see [Fig. 18b](#).

These ideas will now be developed.

5.3.1 Wedging the Initial Configuration of \mathcal{A}

First, one must remark that the entry point $(x_{0(0,0)}, y_{0(0,0)})$ of $D_{0,0}$ can be easily characterized as the first point in the orbit of \mathcal{B} that is in a state of W and that is not in the influence of any point in state in W, L , or R .

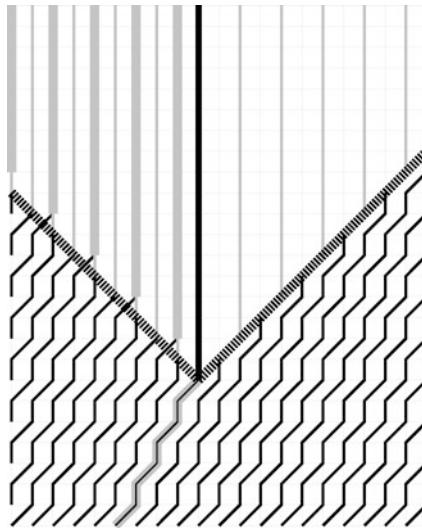
The value of $x_{0(0,0)}$ is not known: the sole way to determine is to let automaton \mathcal{B} run. This leads one to shift $c_{\mathcal{A}}$ in both left and right directions simultaneously. The speed of the shifts cannot be the maximum speed 1 as it would then be impossible to stop them when needed. Though any other speed <1 can be chosen, for the sake of simplicity, speed $\frac{1}{2}$ is used to shift the initial configuration of \mathcal{A} . Only the right shift construction is treated and assumed that it is the right shift of $c_{\mathcal{A}}(0)$ which eventually meets the vertical with abscissa $x_{0(0,0)}$ at site $(x_{0(0,0)}, y)$.

Consider the three possible cases:

- Case $y = y_{0(0,0)}$.* [Figure 19](#) illustrates this simple case and shows how the flow of data of the shift is stopped. When $c_{\mathcal{A}}(0)$ meets $(x_{0(0,0)}, y_{0(0,0)})$ two maximum speed opposite signals are launched. These signals then meet the flow of data and then stop their moves (this is why the shift cannot be done at maximum speed, otherwise it would be impossible to stop the moves). One can remark that this construction compresses the left part and sparses the right part of $c_{\mathcal{A}}$ (see comments of [Fig. 19](#)).

Fig. 19

Data shift and how this flow is stopped. For $i \in \mathbb{N}$ the $c_{\mathcal{A}}(i)$'s are distributed on one cell out of two: those with coordinates $x_{0(0,0)} + 2i$. Also, $c_{\mathcal{A}}(-3i)$ is distributed on $x_{0(0,0)} - 2i$ whereas $c_{\mathcal{A}}(-3i - 1)$ and $c_{\mathcal{A}}(-3i - 2)$ are both distributed on $x_{0(0,0)} - 1 - 2i$. The width of vertical gray lines illustrates this property (thick lines represent two data on the same cell).



- *Case $y > y_{0(0,0)}$.* In that case (see [Fig. 20a](#)), as cell $x_{0(0,0)}$ has not met $c_{\mathcal{A}}(0)$, automaton \mathcal{C} freezes the computation of \mathcal{B} from time $y_{0(0,0)}$ until $c_{\mathcal{A}}(0)$ is met. Then automaton \mathcal{C} thaws the computation of \mathcal{B} .
- *Case $y < y_{0(0,0)}$.* In that case (see [Fig. 20b](#)), cell $x_{0(0,0)}$ has already met $c_{\mathcal{A}}(0)$. This means that the shift has gone too far on the right so that one has to shift it to the left. Then \mathcal{C} freezes the computation of \mathcal{B} from time $y_{0(0,0)}$ until $c_{\mathcal{A}}(0)$ has gone back to cell $x_{0(0,0)}$ at which time \mathcal{C} thaws the computation of \mathcal{B} . In the additional left shift of $c_{\mathcal{A}}$, two cases occur:
 - if $i > 0$, the additional left shift of $c_{\mathcal{A}}(i)$ hits the right thawing signal, and the flow is simply straightened up.
 - if $i < 0$ there are two subcases. For $|i|$ big enough, when $c_{\mathcal{A}}(i)$ meets the left freezing signal, data are compressed as in case $y = y_{0(0,0)}$. For $|i|$ small enough, the additional left shift of $c_{\mathcal{A}}(i)$ hits the vertical $x_{0(0,0)}$. Then a new vertical is inserted which necessitates to push one step to the left all the already-built verticals.

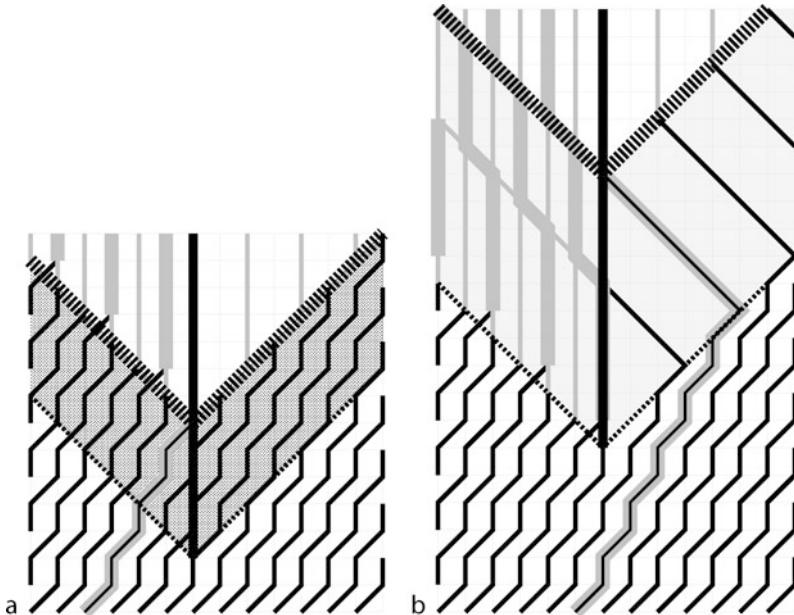
The delay $d_{\mathcal{A}}$ mentioned in the proposition is exactly the time during which \mathcal{C} freezes the computation of \mathcal{B} .

5.3.2 Distributing the Initial Configuration of \mathcal{A}

Now, the initial configuration, $c_{\mathcal{A}}$, has been lifted. And so one has to position the $c_{\mathcal{A}}(i)$'s and $c_{\mathcal{A}}(-i)$'s ($i \in \mathbb{N}$) in the appropriate input points of the $D_{(i,0)}$'s and $D_{(0,i)}$'s of the grid as illustrated in [Fig. 18a](#).

Fig. 20

- (a) $c_{\mathcal{A}}(0)$ passes after $(x_{0,0}, y_{0,0})$. In the gray part, the construction of the grid is frozen.
 (b) $c_{\mathcal{A}}(0)$ passes before $(x_{0,0}, y_{0,0})$. In the gray part, the construction of the grid is frozen. $c_{\mathcal{A}}$'s are sent back appropriately.



To do so, the idea is to make data follow the “hull” of the grid (cf. [Definition 9](#) below). Left (resp. right) data must follow the leftmost (resp. rightmost) border of signals $\mathcal{L}_{(0,i)}$ (resp. $\mathcal{R}_{(i,0)}$) and of domains $D_{(0,i)}$ (resp. $D_{(i,0)}$). Thus the data for the left (resp. right) border never moves to the right (resp. left).

For clarity, the focus is only on the right side.

The distribution is as illustrated in [Fig. 22](#). The distribution of the $c_{\mathcal{A}}(i+k)$'s ($k \geq 1$) on the right diagonal issued from the entry point of $D_{(i+1,0)}$ is the same as that on the diagonal issued from the entry point of $D_{(i,0)}$.

The idea is to push every $c_{\mathcal{A}}(i)$ to the right each time the right border of the right hull goes to the right. Thus, on a diagonal segment of length ℓ on the right border, up to 2ℓ distinct successive $c_{\mathcal{A}}(i)$'s have to be coded (recall that up to two values can be encoded into a single cell, cf. [Fig. 19](#)). To do so, a stack is necessary and the size of the stack is bounded by the longest diagonal segment of the border. This is a reason for the constraint in item 7 of [Definition 8](#).

Definition 9 (hull of a set of sites) The hull of a set of sites \mathcal{X} is defined as the set of sites:

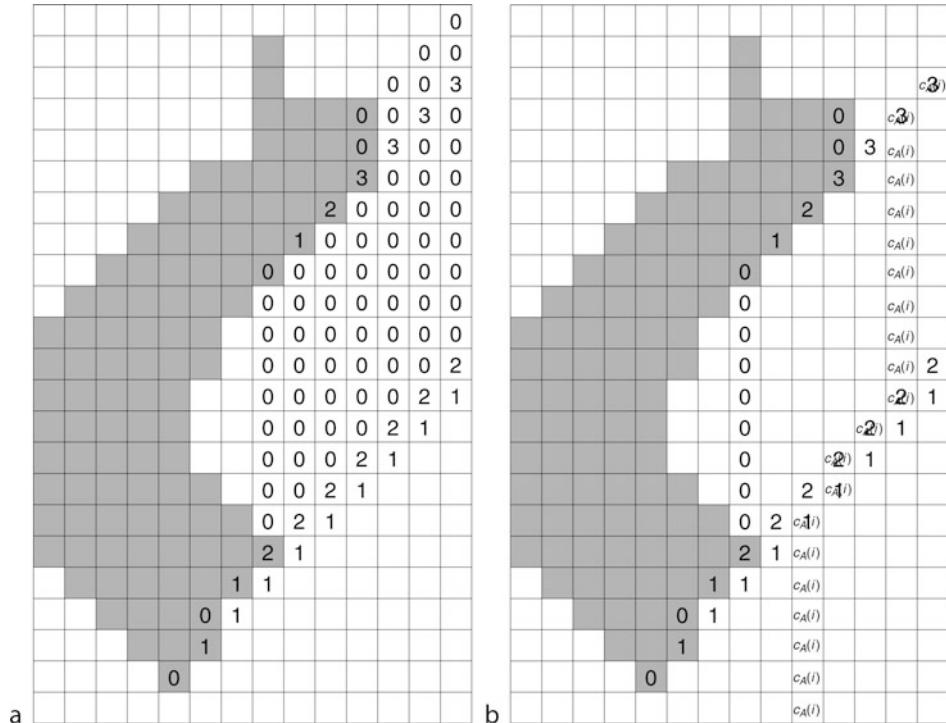
$$\{(x, t) | \exists t' \leq t, (x, t') \in \mathcal{X}\}$$

The hull can be characterized by a simple construction.

Now, for each diagonal segment of the right hull, one has to compute its length. This value is used all along the diagonal of the space–time diagram issued from the last point of the

Fig. 21

(a) Constructing the right hull of a set of sites and propagating the lengths of right diagonal segments. (b) Pushing $c_{\mathcal{A}}(i)$ to the right. For the sake of visibility, right 0 counters have been omitted.



considered segment to shift every value of $c_{\mathcal{A}}$ (see [Fig. 21a](#)). Starting from the initial point of the grid a counter is incremented and moved right each time the grid expands to the right. When the grid does not expand to the right, the counter stops its incrementation and its value is broadcasted to the right; the counter is then reset and moved straight up until it meets the grid again (see [Fig. 21a](#)).

[Figure 21b](#) illustrates how a value of $c_{\mathcal{A}}$ is shifted to follow the right border, the right hull of the grid.

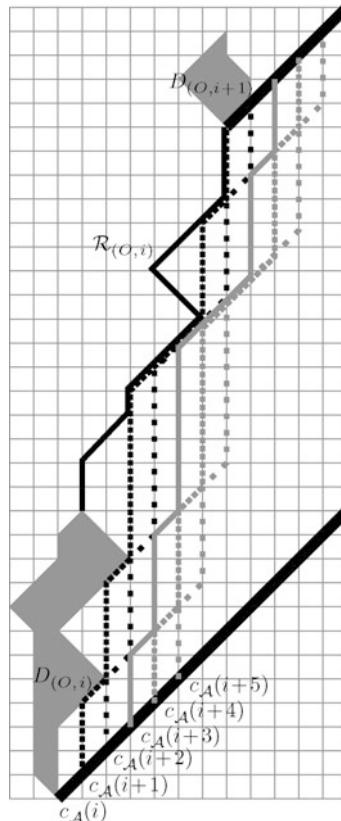
[Figure 22](#) shows how the full process takes place.

5.3.3 Simulation of \mathcal{A} on the Grid

Recall that one wants the state $\langle i, t \rangle_{\mathcal{A}}$ to be encoded in the output point of $D_{(i,t)}$ and the state $\langle -i, t \rangle_{\mathcal{A}}$ on the output point of $D_{(t,i)}$, for $i \in \mathbb{N}$. Since the value $\langle j, t \rangle_{\mathcal{A}}$ is necessary to compute the values $\langle j-1, t+1 \rangle_{\mathcal{A}}$, $\langle j, t+1 \rangle_{\mathcal{A}}$, and $\langle j+1, t+1 \rangle_{\mathcal{A}}$ ($j \in \mathbb{Z}$), one has to show that one can carry this value $\langle j, t \rangle_{\mathcal{A}}$ from the output point of the domain which encodes it

Fig. 22

How the right hull of a grid is built and how values of the initial configuration of \mathcal{A} are shifted to follow the shape of the hull as closely as possible and how they are injected into appropriate domains. Only the right part of the process is illustrated for convenient reasons. $c_{\mathcal{A}}(i)$ is injected into $D_{(0,i)}$. $c_{\mathcal{A}}(i+1)$ is used to build the right hull and is then injected into $D_{(0,i+1)}$. $c_{\mathcal{A}}(i+k)$ (for $k \geq 2$) are shifted to the right according to the path constructed by $c_{\mathcal{A}}(i+1)$. One can easily show that if the longest diagonal segment followed by $c_{\mathcal{A}}(i+1)$ is of length ℓ , at most ℓ values are stacked on the diagonals issued from the hull.

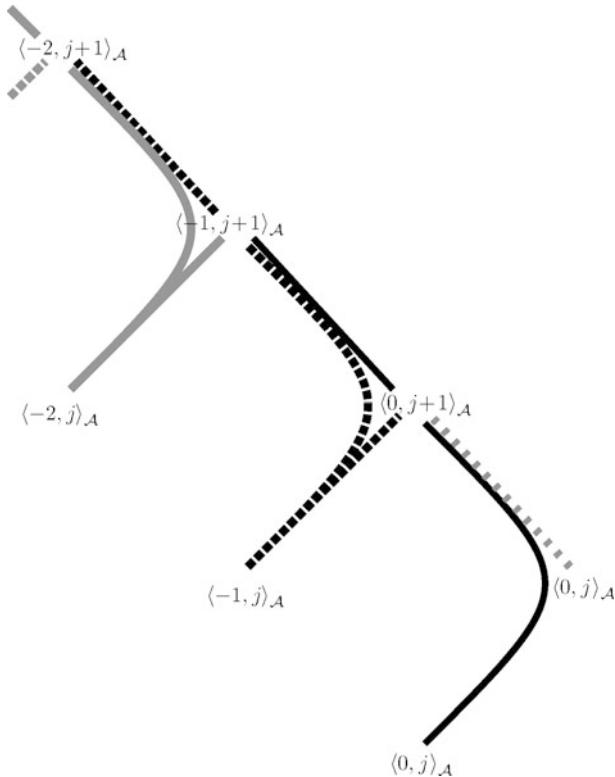


(namely, $D_{(j+t,t)}$ if $j > 0$, $D_{(t,|j|+t)}$ if $j < 0$, or $D_{(t,t)}$ if $j = 0$) to the input points of the appropriate domains. This leads to the following three cases:

- *Case $j > 0$.* Let $\langle j, t \rangle_{\mathcal{A}}$ follow signal $\mathcal{L}_{(j+t,t)}$, travel through $D_{(j+t,t+1)}$, then follow signal $\mathcal{R}_{(j+t,t+1)}$, travel through $D_{(j+t+1,t+1)}$, then follow $\mathcal{R}_{(j+t+1,t+1)}$. This allows us to carry $\langle j, t \rangle_{\mathcal{A}}$ onto the input points of $D_{(j+t,t+1)}$, $D_{(j+t+1,t+1)}$, and $D_{(j+t+2,t+1)}$. With the notations in the end of [Sect. 5.1](#), this is done following the three paths α , $\alpha\beta$, and $\alpha\beta\beta$ of the dependencies Cayley graph.
- *Case $j < 0$.* This case is illustrated by [Fig. 23](#). Let $\langle j, t \rangle_{\mathcal{A}}$ follow signal $\mathcal{R}_{(t,|j|+t)}$, travel through $D_{(t+1,|j|+t)}$, then follow signal $\mathcal{L}_{(t+1,|j|+t)}$, travel through $D_{(t+1,|j|+t+1)}$, then follow

Fig. 23

How the computation of the simulated CA is embedded into a grid. The figure only shows case $j < 0$. Except for the central cell, the state of a simulated cell is broadcasted through the right output wire, then to the two following left wires (see the black dashed line for example). This carries the state to the three corresponding “neighbors” at next time step.



$\mathcal{L}_{(t+1,|j|+t+1)}$. This allows us to carry $\langle j, t \rangle_{\mathcal{A}}$ onto the input points of $D_{(t+1,|j|+t)}$, $D_{(t+1,|j|+t+1)}$, and $D_{(t+1,|j|+t+2)}$. With the notations in the end of [Sect. 5.1](#), this is done following the three paths β , $\beta\alpha$, and $\beta\alpha\alpha$ of the dependencies Cayley graph.

- *Case $j = 0$.* This is a particular case since the left and right neighbors are on two different diagonals. Let $\langle 0, t \rangle_{\mathcal{A}}$ follow signal $\mathcal{R}_{(t,t)}$, travel through $D_{(t+1,t)}$, then follow signal $\mathcal{L}_{(t+1,t)}$, travel through $D_{(t+1,t+1)}$, then at this point, the value is distributed on the two opposite directions, namely: $\mathcal{L}_{(t+1,t+1)}$ and $\mathcal{R}_{(t+1,t+1)}$. This allows us to carry $\langle 0, t \rangle_{\mathcal{A}}$ onto the input points of $D_{(t+1,t+1)}$, $D_{(t+2,t+1)}$, and $D_{(t+1,t+2)}$. With the notations in the end of [Sect. 5.1](#), this is done following the three paths $\beta\alpha$, $\beta\alpha\alpha$, and $\beta\alpha\beta$ of the dependencies Cayley graph.

Now, every domain receives the three needed states at its input point and can compute the transition function of automaton \mathcal{A} and encode the resulting value at its output point.

6 Beyond Grids

As there exists many kinds of intrinsic simulations of CAs by CAs, one can ask what makes such simulation via grids especially interesting. The answer is that this construction is very flexible. The authors think that it is closer to an hypothetical, massively parallel programming tool. The theorem in itself does not give much more power to compute, but can be used as a basic tool to understand how recursive schemas can be implemented. Some arguments in favor of this thesis is given.

6.1 Going Back to the Multiplication: Composition

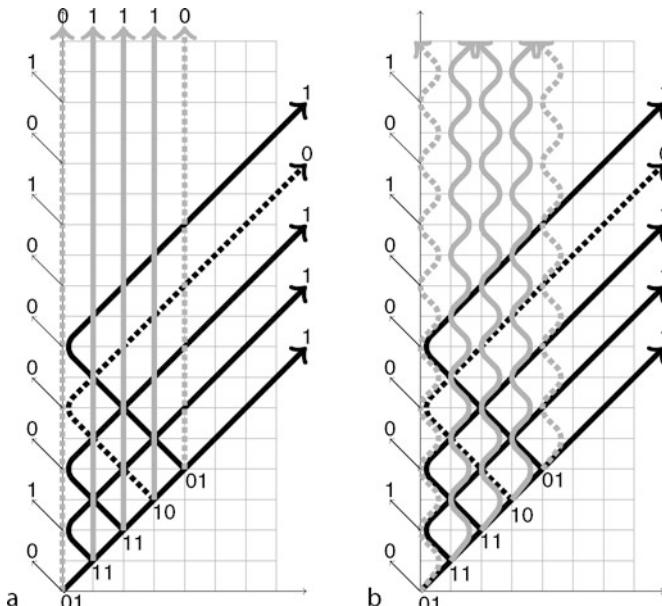
Go back and start again with a simple algorithm: the multiplication. Recall that at the beginning of this chapter, it was observed that a multiplication algorithm naturally leads to a cellular automaton synthesis, but that such a construction is not always (or at least not easily) composable with itself. Note that inputs and outputs are a recurrent problem in computer science.

Now, in the light of grid computation, a multiplication is simply a computation taking place on a grid. No matter what the real shape of that grid is, provided that the grid has some necessary properties, the computation can take place on it. In other words, it can be said that if the grid realizes the dependency graph, the computation is doable within it.

From Fig. 24a, b, one can show that it could be possible to view the multiplication as a process taking place on a grid, with inputs given onto the first diagonal and the

Fig. 24

(a) Simple multiplication algorithm of 14 by 23. This figure illustrates the main flows of bits of the multiplicand (gray flows) and the multiplier (black flows). Details are omitted. (b) The same algorithm implemented as a trellis. It is clear that every single vertical move has been replaced by a right move followed by a left move.



result produced onto the main vertical line of the grid (the left part of the grid is of no usage).

Now, one can bend the grid as wanted, provided that the bending leads to a constructible grid. [Figure 25a, b](#) show two different possible bendings.

It is easy to show that the two bendings are compatible, that is, that the two functions can be combined to provide a multiplication of three numbers. The two first are multiplied together, and the third is injected to the output location of the result to make them the input of the second multiplication. This implements the fact that to compute $n_1 \times n_2 \times n_3$, it is sufficient to have a binary multiplier and to combine it to compute $(n_1 \times n_2) \times n_3$.

Remark that the former construction is such that two grids linked together were constructed so that the right move of the second grid, $\beta_{\mathcal{G}_2}$, is exactly the vertical move of the first grid, $\beta_{\mathcal{G}_1} = \beta_{\mathcal{G}_2} \alpha_{\mathcal{G}_1}$. The exact values of these parameters relatively to the underlying space–time diagram are left to the implementation. In the provided example, they have been chosen so that the computation time is minimized, and so the result is produced on the first cell of the underlying space–time diagram.

This can be extended as far as required. [Figure 26a](#) illustrates the case where three grids are fitted together. The first, \mathcal{G}_1 , has $\alpha_{\mathcal{G}_1} = \alpha$ and $\beta_{\mathcal{G}_1} = \beta^3$ (so that the vertical move is $v_{\mathcal{G}_1} = \beta^3 \alpha$). The second, \mathcal{G}_2 , has $\alpha_{\mathcal{G}_2} = \alpha$ and $\beta_{\mathcal{G}_2} = v_{\mathcal{G}_1} = \beta^3 \alpha$ (so that the vertical move is $v_{\mathcal{G}_2} = \beta^3 \alpha^2$). The third, \mathcal{G}_3 , has $\alpha_{\mathcal{G}_3} = \alpha$ and $\beta_{\mathcal{G}_3} = v_{\mathcal{G}_2} = \beta^3 \alpha^2$ (so that the vertical move

Fig. 25

(a) Bended grid of [Fig. 24b](#). A right move is obtained by two right moves ($\beta_{\text{bended}} = \beta^2$). A left move is a simple left move ($\alpha_{\text{bended}} = \alpha$). (b) Bended grid of [Fig. 24b](#). A right move is obtained by two right moves and a left move ($\beta_{\text{bended}} = \beta^2 \alpha$). A left move is a simple left move ($\alpha_{\text{bended}} = \alpha$).

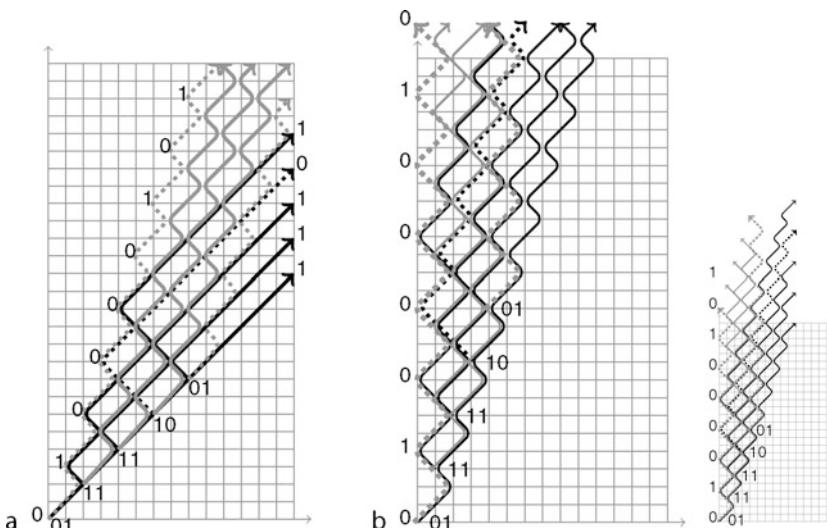
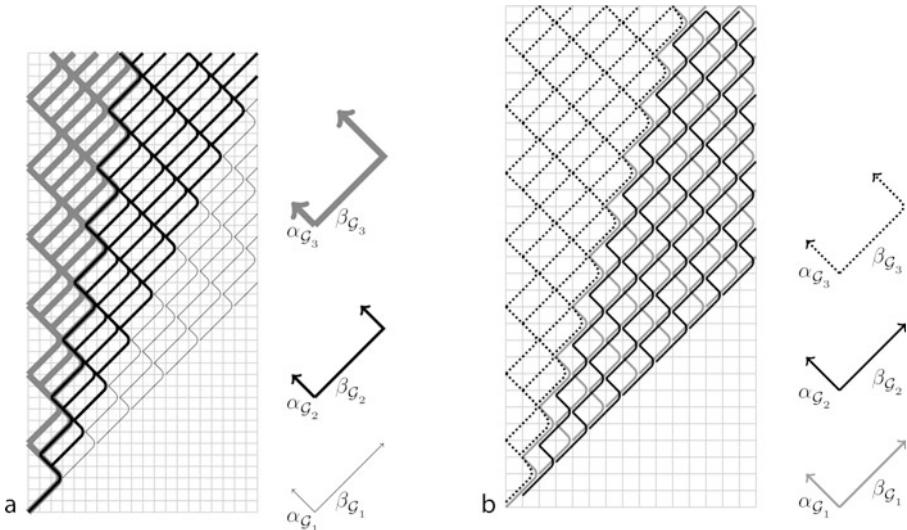


Fig. 26

(a) Three fitted grids to compute the composition of three functions. For example, one can compute the product of four numbers as $((n_1 \cdot n_2) \cdot n_3) \cdot n_4$. (b) Three grids to compute a tree of three functions: two interleaved grids are fitted to the third. For example, one can compute the product of four numbers as $((n_1 \cdot n_2) \cdot (n_3 \cdot n_4))$.



is $v_{G_3} = \beta^3 \alpha^3$). Details such as input locations are left to the reader. Observe that to multiply n numbers, it is sufficient to have $n - 1$ grids, and that the bit p of the result is produced at time $2(n - 1)p$.

➊ *Figure 26b* illustrates how parallelism can be really achieved to compute the product of 2^n numbers more efficiently than the previous method. The idea is to decompose the product as a tree of computations. Each level of the tree is implemented by interleaving the necessary grids (2^m numbers leads to 2^{m-1} products computed by 2^{m-1} grids).

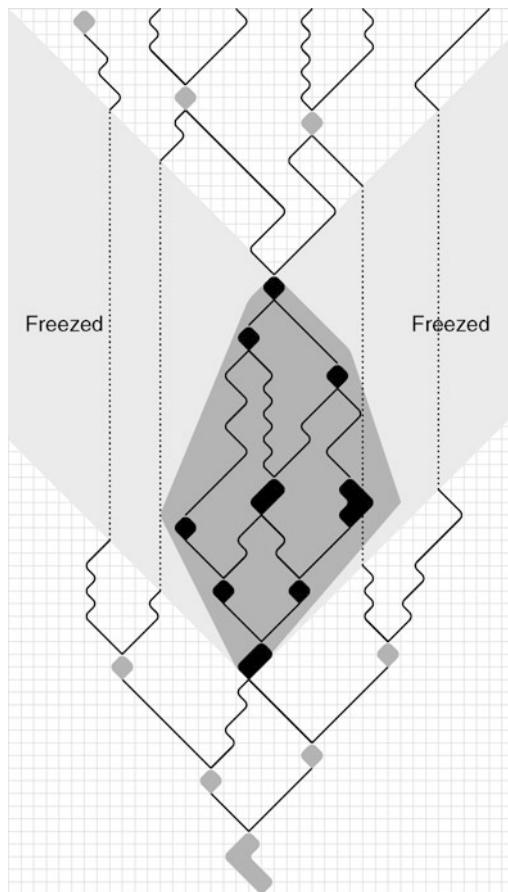
The reader is referred to Mazoyer (1987) to see how an infinite family of signals can be built and used to split the space–time diagram in a regular manner.

6.2 Achieving Recursion

Now one can ask how recursion can be achieved using such constructions. The answer is (at least theoretically) very simple: it is sufficient to view a domain as a grid. Since domains are defined as sets of sites, there can be room to do computation within them. For example, one can construct a grid inside a domain. Now to implement such a construction, one needs to be very careful, as there are many difficulties involved. ➋ *Figure 27* illustrates how a function call can be implemented.

Fig. 27

Implementing a function call. A domain is used to compute a function. As no one can know how long that *internal* computation will take, the calling process is frozen and then thawed appropriately. Black domains are *internal* call domains. The gray hull is here to represent the *external* view of the domain.



References

- Adamatzky A (2002) Collision-based computing. Springer, London. ISBN 978-1852335403
- Atrubin AJ (1965) A one-dimensional real-time iterative multiplier. IEEE Trans Electron Comput 14:394–399
- Choffrut C, Čulik K II (1984) On real-time cellular automata and trellis automata. Acta Inform 21:393–407
- Cole SN (1969) Real-time computation by n-dimensional iterative arrays of finite-states machines. IEEE Trans Comput C-18(4):349–365
- Čulik K II, Gruska J, Salomaa A (1982) Systolic trellis automata for VLSI on balanced trees. Acta Inform 18:335–344
- Delorme M, Mazoyer J (1999) Cellular automata: A parallel model. Mathematics and its applications, vol 460. Kluwer, Dordrecht. ISBN 0-7923-5493-1
- Even S (1991) Systolic modular multiplication. In: Advances in cryptology: CRYPT0'90. Lecture notes on computer science, vol 537. Springer, New York, pp 620–624

- Fischer PC (1965) Generation of primes by a one-dimensional real-time iterative array. *J ACM* 12(3):388–394
- Goyal LN (1976) A note on Atrubin's real-time iterative multiplier. *IEEE Trans Electron Comput* C25(5):546–548
- Knuth DE (1997) Seminumerical algorithms, the art of computer programming. vol 2, 2nd edn. Addison-Wesley, Reading, MA. ISBN 0-201-03822-6
- Korec I (1997) Real-time generation of primes by a one-dimensional cellular automaton with 9 States. Preprint Series of Mathematical Institute of Slovak Academy of Sciences, Bratislava, Preprint 13/1997
- Mazoyer J (1987) A six states minimal time solution to the firing squad synchronization problem. *Theor Comput Sci* 50:183–328
- Mazoyer J, Terrier V (1999) Signals in one-dimensional cellular automata. *Theor Comput Sci* 217(1):53–80
DOI 10.1016/S0304-3975(98)00150-9
- McNaughton R (1961) The theory of automata, a survey. *Adv Comput* 2:379–421
- Poupet V (2005) Cellular automata: Real-time equivalence between one-dimensional neighborhoods. In: STACS 2005. Proceedings of the 22nd annual symposium on theoretical aspects of computer science, Stuttgart, Germany, February 24–26, 2005 pp 133–144
- Wolfram S (2002) A new kind of science. Wolfram Media, Champaign, IL

6 Universalities in Cellular Automata*

Nicolas Ollinger

Laboratoire d'informatique fondamentale de Marseille (LIF),
Aix-Marseille Université, CNRS, Marseille, France
nicolas.ollinger@lif.univ-mrs.fr

1	<i>Introduction</i>	190
2	<i>Copper: A Simple Universal Cellular Automaton</i>	191
3	<i>Definitions</i>	199
4	<i>Chronology</i>	202
5	<i>Formalizing Universalities</i>	205
6	<i>High Dimensions: 2D and More</i>	207
7	<i>Turing Universality in 1D</i>	213
8	<i>Intrinsic Universality in 1D</i>	217
9	<i>Sub-universalities: The Reversible Case</i>	226
10	<i>Conclusion</i>	227

*A preliminary version of this work appeared under the title *Universalities in Cellular Automata: A (Short) Survey* in the proceedings of the first edition of the JAC conference (Ollinger 2008).

Abstract

This chapter is dedicated to computational universalities in cellular automata, essentially Turing universality, the ability to compute any recursive function, and intrinsic universality, the ability to simulate any other cellular automaton. Constructions of Boolean circuits simulation in the two-dimensional case are explained in detail to achieve both kinds of universality. A detailed chronology of seminal papers is given, followed by a brief discussion of the formalization of universalities. The more difficult one-dimensional case is then discussed. Seminal universal cellular automata and encoding techniques are presented in both dimensions.

1 Introduction

Universality is a key ingredient of the theory of computing, grounding its roots in the work of Gödel (1931), that describes the ability for some formal systems to manipulate themselves through proper encodings. By applying such an encoding to machines, Turing (1936) introduces a universal machine with the capacity to simulate every other machine of its class. Stated in an axiomatic way, an acceptable enumeration (Odifreddi 1989) of partial recursive functions, $\varphi_n : \mathbb{N} \rightarrow \mathbb{N}$, is endowed with a bijective and recursive bracket function, $\langle \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$, for which there exists u such that, for all pairs of integers $m, n \in \mathbb{N}$:

$$\varphi_u(\langle m, n \rangle) \simeq \varphi_m(n)$$

where $x \simeq y$ if either both x and y are undefined, or $x = y$. Notice that this does not define a property of being universal but rather provides one universal function. Given a machine in a well-defined family admitting a universal machine, a natural question is to identify the computational power of the machine: what can it compute, up to some admissible encoding? Is it universal, in the sense that it can compute every computable function? As such, universality is a first criterion to discriminate among machines. For Turing machines, a proper formal definition of universality that captures the intuition of what is universal and what is not and encompasses previous commonly admitted constructions remains to be found. That does not stop people from searching and designing very clever small universal machines.

As Turing machines, cellular automata are machines, in the sense that they are described as finite state machines interacting locally with an environment, receiving finite local information from the environment as inputs and interacting locally with the environment through finite outputs. Whereas a Turing machine consists of just one finite state machine behaving sequentially, a cellular automaton consists of an infinite regular grid of copies of a same finite state machine interacting synchronously, uniformly, and locally. Investigating the respective computational power of cellular automata leads to consideration of universality.

The idea and construction of a universal cellular automaton is as old as the formal study of the object itself, starting with the work of von Neumann (1966) on self-reproduction in the 1940s, using cellular automata under suggestions by Ulam. Following the work of Turing, a Turing universal cellular automaton is an automaton encompassing the whole computational power of the class of Turing machines, or by the so-called Church–Turing thesis the class of recursive functions. To encode complex behaviors in a cellular automaton's dynamics, one can describe how to encode any computing device of a universal class of machines (Turing machine, tag systems, etc.) and use classical tools of computability theory to shape wanted

behaviors of the object. This is basically what von Neumann did. He designed a cellular automaton able to encode any Turing machine, the machine being moreover equipped with a construction arm controlled by the machine's head.

But Turing universality is not the only reasonable kind of universality one might expect from cellular automata. It is quite unnatural to consider a universality of highly parallel potentially infinite devices as cellular automata by simulation of the dynamics of sequential finite machines – indeed, as we will discuss, to give a both widely acceptable yet precise definition of Turing universality is a very difficult and unfulfilled challenge. As the study of cellular automata shifted both to dimension 1 and to the study of its dynamics, another kind of universality emerged. An intrinsically universal cellular automaton is an automaton able to properly simulate the behavior of any other cellular automaton on any type of configuration (might it be infinite). It turns out that most of the historical constructions in dimension 2 and more, designed as Turing universality, are intrinsically universal by the simple fact that they are designed to encode any Boolean circuit.

A formal definition of universality might not seem so important. In fact, when building a precise cellular automaton from scratch, to be universal, a definition is often implicit: the obtained behavior is the one engineered by the designer. The definition turns out to be required more when proceeding by analysis: given a cellular automaton rule, is it universal?

The present chapter is constructed as follows. [Section 2](#) gently introduces universalities through the study of an original two-dimensional simple universal cellular automaton using Boolean circuits encoding. [Section 3](#) provides the formal definitions and notations used for cellular automata, configurations, and dynamics. [Section 4](#) is an annotated chronology of seminal papers concerning universality and universal cellular automata. [Section 5](#) discusses the right definition of universalities in cellular automata. [Section 6](#) discusses the construction and analysis of universal cellular automata in dimensions 2 and more, mostly using Boolean circuits simulation. [Section 7](#) discusses Turing universality, its links with universal Turing machines, and the main techniques of construction. [Section 8](#) discusses intrinsic universality and the main techniques of construction. [Section 9](#) discusses universality in the special restricted case of reversible cellular automata.

Remark Cellular automata are dynamical objects; the figures in this chapter are static. For a better experience, the reader is strongly encouraged to complete his reading by experimenting with some reasonable cellular automata simulator software. While writing this chapter, the author enjoyed using Golly ([2008](#)), both to verify ideas and produce pictures for the two-dimensional automata.

2 Copper: A Simple Universal Cellular Automaton

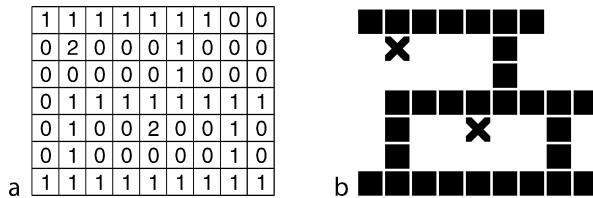
Before discussing formal definitions of different kinds of universalities, an example of a simple universal cellular automaton is considered. The automaton that is considered is called *Copper*, abbreviated as **Cu**. It is modeled after classical universal 2D rules, in the spirit of (albeit different to) the automata of Banks ([1970, 1971](#)).

2.1 Rule of the Game

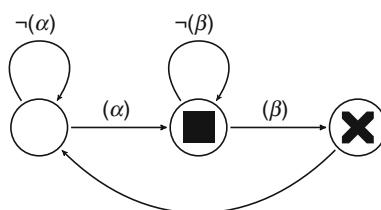
Configurations of **Cu** consist of an infinite two-dimensional grid of cells. Each cell is in one of three possible states: 0, 1, or 2. Thus, a configuration can be described as a matrix of states, as

Fig. 1

Cu configuration drawing. (a) Configuration. (b) Symbolic representation.

**Fig. 2**

Cu local rule: evolution of a cell state depending on its neighbors.



- (α) both north and south, or east and west, neighbors in state **X** or **■**;
- (β) at least two neighbors in state **X** or **■** and either exactly one neighbor in state **X** or exactly one neighbor in state **■**.

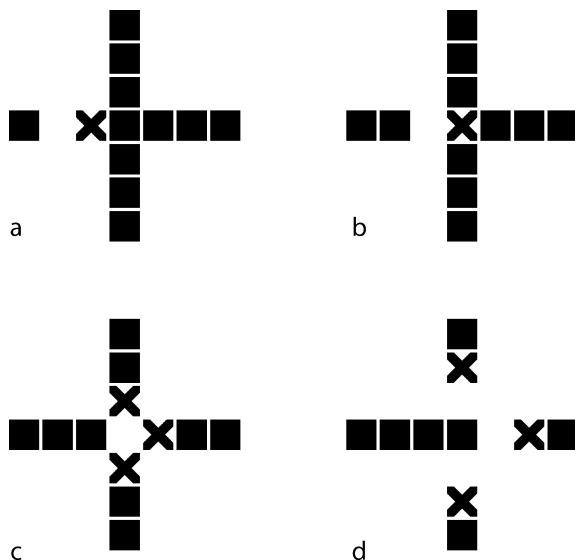
depicted in [Fig. 1a](#). For the sake of readability, a symbolic representation inspired by the usage of states is preferred, depicting state 0 as *void*, state 1 as *wire*, and state 2 as *particle*, as depicted in [Fig. 1b](#).

The world of **Cu** is equipped with a global clock controlling the evolution of configurations. At each time step, all cells change their states synchronously according to a same *local rule*. The local rule consists of looking at the states of the four cardinal neighbors (north, east, south, and west) to choose the new state of a cell. The local rule of **Cu**, invariant by rotation, is explained in [Fig. 2](#) and reads as follows. Cells in void state stay in a void state but enter a wire state if two opposite neighbors (north and south or west and east) are both not in void state. Cells in a wire state stay in the wire state but enter a particle state if the number of neighbors not in the void state is greater than or equal to two, and there is exactly one neighbor in the wire state or in the particle state. Cells in the particle state always enter the void state.

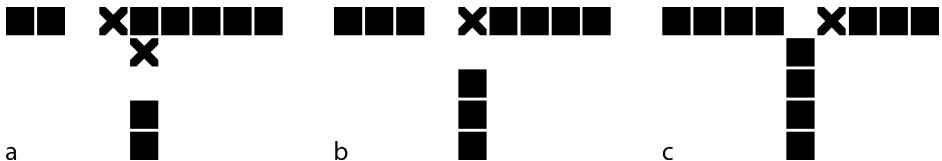
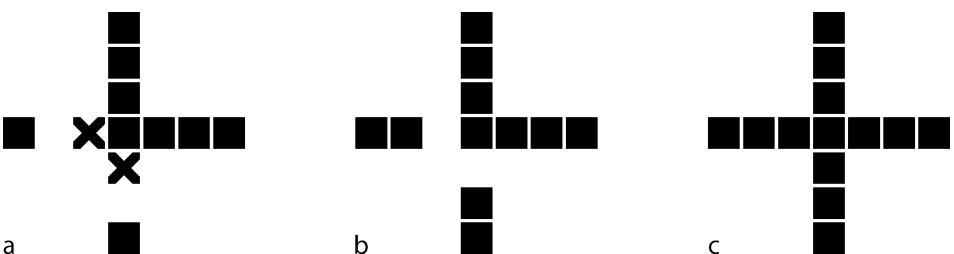
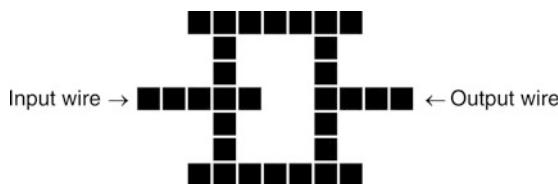
2.2 Particles on Wires

The rule of **Cu** is designed so that properly constructed configuration patches behave as networks of wires on which particles move, interact, and vanish. A *wire* consists of a straight line of wire cells of width 1 bordered by void states. For a particle to move on a wire, it should be oriented. Orientation is obtained by encoding *moving particles* on a wire as a pair of a particle state and a void state. The direction of the movement is given by the particle state as depicted in [Fig. 3](#).

When a particle arrives at the end of a wire, a *dead end*, the particle vanishes, as depicted in [Fig. 4](#). If two particles arrive from opposite directions on a same wire and *collide*, both particles vanish, as depicted in [Fig. 5](#). Notice that the collisions should happen with odd spacing, otherwise the wire would be damaged.

Fig. 3**Cu particle moving on a wire from left to right.****Fig. 4****Cu particle vanishing in a dead end.****Fig. 5****Cu particles vanishing on (odd) collision.****Fig. 6****Cu particle spreading at intersection.**

Wires can be connected at *intersection points* where three or four wires meet. The connection of two wires is possible but does not behave in an interesting way from the particle point of view (a particle destroys a two-wire connection when it moves through it), it can be replaced by a three-wire intersection plus a dead end. At an intersection, a single particle entering the intersection generates a leaving particle on every other end, as depicted in [Fig. 6](#). Symmetrically, when particles arrive synchronously from every other end at an intersection, they join as a single leaving particle on the empty end, as depicted in [Fig. 7](#). Every other scenarios of particles entering an intersection leads to no leaving particle, as depicted in [Fig. 8](#).

Fig. 7**Cu particles joining at intersection.****Fig. 8****Cu particles vanishing at intersection.****Fig. 9****Cu a diode.**

2.3 Computing Boolean Functions

Using particles and networks of wires makes it possible to encode bits and gates to encode every Boolean function. This is done by constructing and combining widgets, exploiting properties of intersections. Each widget connects input wires to output wires. By entering synchronized particles on the input wires, after some *traversal delay*, synchronized leaving particles appear on output wires. Moreover, each widget has a proper *safety delay* ensuring that if inputs are spaced by at least this delay, the widget behaves according to its specification (the safety delay can be smaller or bigger than the traversal delay). The synchronization of particles is obtained by adding delays on the path by introducing turns on wires using intersections.

A first widget is the *diode*. A diode is a widget computing the identity function from an input wire to an output wire, with the special property that if a particle improperly enters the output wire (respecting the safety delay), it is guaranteed that no particle will leave the input wire. Combining a four-wire intersection and a three-wire intersection, one obtains the diode depicted in [Fig. 9](#).

Combining two diodes with an intersection provides two elementary widgets computing an OR gate or a XOR gate, as depicted in [Fig. 10](#). The diodes ensure that no particles leave the input wires and the intersection combines the input particles into an output particle.

Planar acyclic circuits consisting of fan-outs, OR, and XOR gates are known to be able to compute any Boolean function, using a proper encoding of Boolean values. For the sake of completeness, a description is given on how this can be used in Cu to construct wire crossing, NOT, AND, and OR gated. Planar crossing is obtained using two fan-outs and three XOR gates, as depicted in [Fig. 11](#) and implemented more compactly in Cu as depicted in [Fig. 12](#). All monotone Boolean functions are computable using OR and XOR gates because $A \wedge B = (A \vee B) \oplus (A \oplus B)$. To encode all Boolean functions with monotone functions, a classical trick is to encode each Boolean value A as the pair of values (A, \bar{A}) . Using this encoding, Boolean values are represented on two wires: no particle means no value, values are encoded by exactly one particle on the corresponding wire. A Boolean NOT gate is encoded as a wire crossing, implemented in Cu as depicted in [Fig. 12](#). A Boolean AND gate with inputs (A, \bar{A}) and (B, \bar{B}) outputs $(A \wedge B, \bar{A} \vee \bar{B})$ is implemented more compactly in Cu as depicted in [Fig. 13](#). A Boolean OR gate is encoded symmetrically, outputting $(A \vee B, \bar{A} \wedge \bar{B})$. Notice that, using this construction scheme, one can encode every Boolean function into a widget with a traversal delay that depends on the size of the associated circuit but with a safety delay that can be uniformly bounded by the maximal safety delay of the elementary widgets composing the circuit. The safety delay does not depend on the computed function.

Remark Different encodings are possible. For example, a more compact encoding of a Boolean function $f: \{0, 1\}^k \rightarrow \{0, 1\}^l$ adds one input and one output with constant values 1. Using this encoding, NOT maps $(1, A)$ into $(1, 1 \oplus A)$, AND maps $(1, A, B)$ into $(1, A \wedge B)$, and OR maps $(1, A, B)$ into $(1, A \vee B)$. The constant value 1 acts like a global constant, an asynchronous clock signal indicating that a computation is occurring. This solution provides simpler gates at

Fig. 10

Cu elementary particle gates. (a) OR gate. (b) XOR gate.

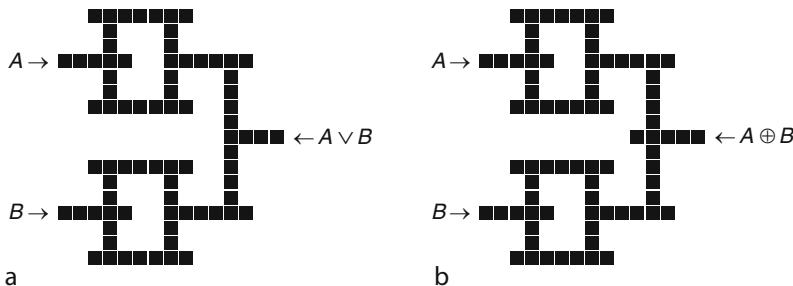
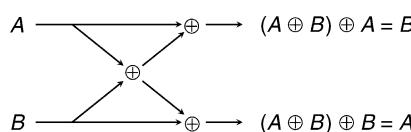
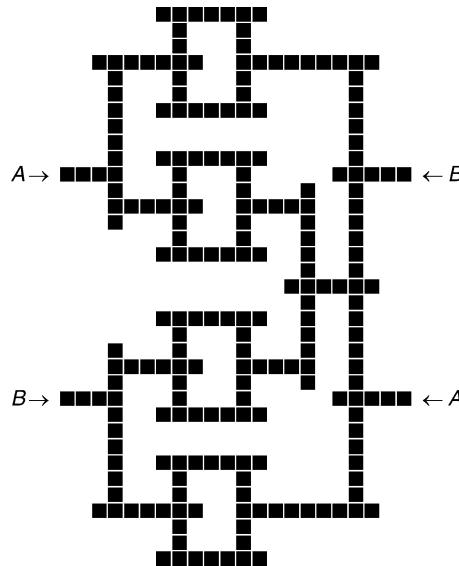


Fig. 11

Planar crossing with fan-outs and XOR gates.



■ Fig. 12
Cu particles crossing.



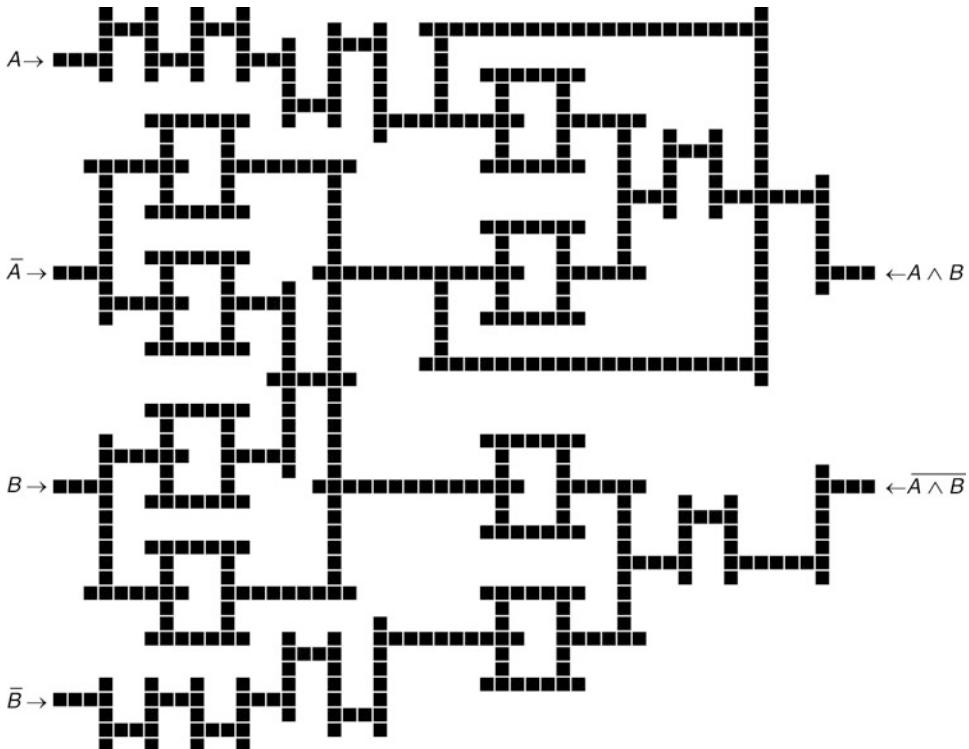
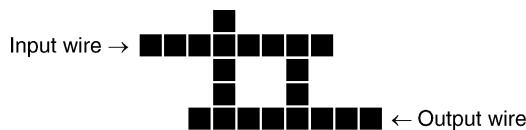
the cost of more complicated routing problems to route and synchronize the constant to each NOT gate.

Remark Different gates are possible. For example, a more compact logic can be derived from the compact diode depicted in ▷ Fig. 14. Sample OR and XOR gates are depicted in ▷ Fig. 15a, b. In this case, better delays are obtained at the cost of more complicated routing and synchronization issues.

2.4 Encoding Finite State Machines

Using Boolean functions make it possible to encode finite state machines. A finite state machine can be seen as a map $f: S \times I \rightarrow S \times O$ where S is a finite set of states, I is a finite set of input values, and O is a finite set of output values. At each time step, reading input i , the machine evolves from states s to state s' and outputs o where $f(s, i) = (s', o)$. Once a binary encoding is chosen for S , I , and O , f can be seen as a Boolean function. The finite state machine is encoded by looping the S bits from the output to the input of the corresponding encoding of the Boolean function, as depicted in ▷ Fig. 16.

Some details should be considered carefully. Each finite state machine constructed this way has a proper clocking: it is the traversal time of the Boolean function plus the time for the state to go from the output back to the input of the circuit. By construction, the circuit is synchronized on this clocking: inputs and outputs are considered synchronized on the circuits and no garbage signal is produced in between two heartbeats on any outside wire (state, input, and output). For this to work, the clocking has to be larger than the safety delay of the Boolean function (this can be fixed by adding extra delays if required). Notice that the input and output signals are synchronized with the circuit, so a direct binary encoding on wires is possible.

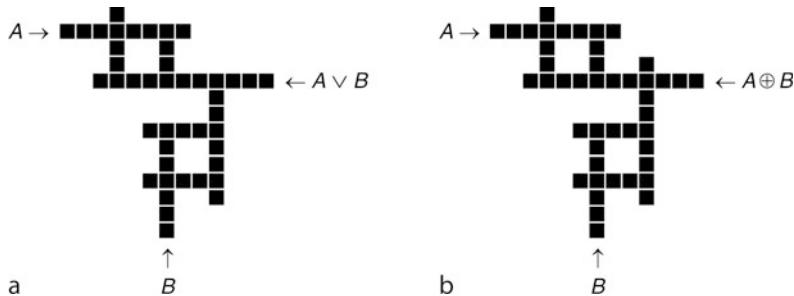
Fig. 13**Cu Boolean AND gate.****Fig. 14****Cu compact diode.**

To handle complex computations, finite state machines have to be connected via input and output wires to their environment, encoded as other dedicated finite state machines. The easiest way to connect two finite state machines is to have them share the same clocking. When considering infinite sequences of finite state machines, this might lead to unbounded clocking. To avoid such pitfalls, one might insert in between the two computing finite state machines a simple memory finite state machine with a small fixed clocking, τ , independent of the number of wires. By ensuring that the memory does not send signals on output wires without being ordered to do so on input wires, and by fixing the clocking of the two computing finite state machine to be a multiple of τ , as depicted in Fig. 17, one obtains better clockings.

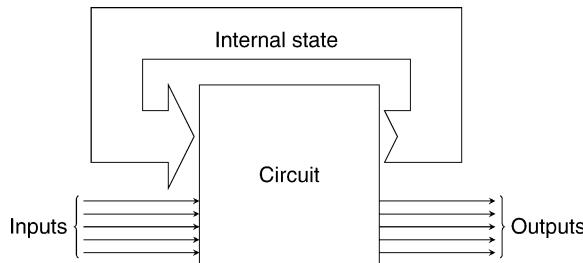
Remark As the interest here is only in the computational potentiality of cellular automata, such a naive encoding is sufficient. For efficient implementations, one might consider efficient Cu

Fig. 15

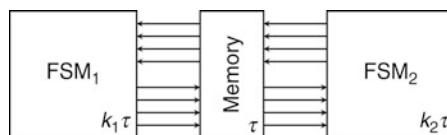
Cu compact elementary gates. (a) Compact OR gate. (b) Compact XOR gate.

**Fig. 16**

Encoding a finite state machine with a Boolean circuit.

**Fig. 17**

Synchronization using memory.



encodings of multiplexers, demultiplexers, memory lattices, busses, etc. – mimicking classical circuit design.

2.5 Turing Universality

Using finite state machines makes it possible to achieve Turing universality by encoding classical computational models. One possibility is to consider Turing machines (a finite state machine plus a tape, represented as a biinfinite sequence of finite state machines encoding tape cells). Another possibility is to encode a Minsky machine: a finite state machine equipped with two counters. On each counter, the machine can increment the counter, decrement the counter if it is not zero, and test if the counter value is zero or positive. Such

a counter can be encoded as an infinite sequence of identical finite state machines. The complete encoding is depicted symbolically in [Fig. 18](#).

Given a Minsky machine, one can recursively construct an ultimately periodic configuration of **Cu** that simulates the behavior of the machine. Thus, **Cu** is Turing universal, it can compute every recursive function.

Remark In this chapter, the use of infinite configurations is allowed, provided that the configurations are ultimately periodic. After all, what is void? Is it really empty or is it somehow regular? It is possible to consider universality in the restricted case of finite configurations, the universal cellular automata have a few more states. The notion of intrinsic universality requires ultimately periodic configurations.

2.6 Intrinsic Universality

Using finite state machines makes it possible to achieve a simpler yet stronger form of universality: intrinsic universality, that is the ability for **Cu** to simulate every two-dimensional cellular automaton. The idea is simple: encode the local rule of a given cellular automaton as a finite state machine and put infinitely many copies of that same machine on a regular grid, using wires to synchronously connect outputs to the proper neighbor inputs, as depicted in [Fig. 19](#).

Given a cellular automaton, one can recursively find a delay T (the clocking of the finite state machine) and associate to each state s of the automaton a square patch of configuration $\tau(s)$ of **Cu** (the finite state machine with s encoded on its wires) so that for each configuration x of the automaton, the configuration $\tau(x)$ of **Cu** evolves in T steps into $\tau(y)$ where y is the image configuration of x by the encoded automaton.

3 Definitions

It is now time to associate formal definitions to the objects.

A *cellular automaton* \mathcal{A} is a tuple (d, S, N, f) where d is the *dimension* of space; S is a finite set of *states*; N , a finite subset of \mathbb{Z}^d , is the *neighborhood*; and $f: S^N \rightarrow S$ is the *local rule*, or *transition function*, of the automaton. A *configuration* of a cellular automaton is a coloring of

[Fig. 18](#)

Two-counters machine encoding.

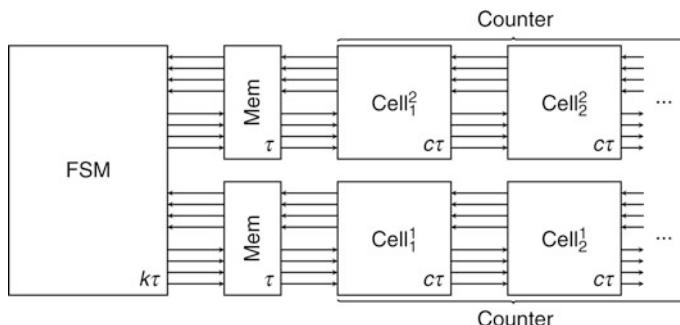
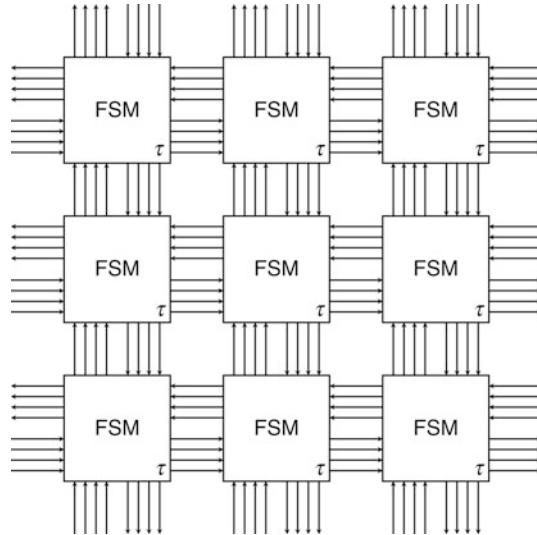
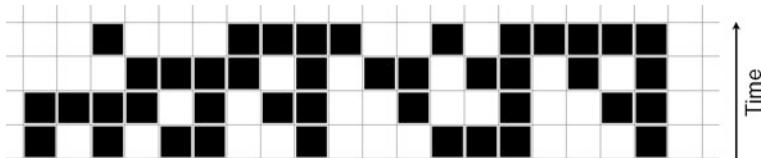


Fig. 19

Intrinsic universality encoding.

**Fig. 20**Partial space–time diagram of the $(\mathbb{Z}_2, +)$ rule.

the space by S , an element of $S^{\mathbb{Z}^d}$. The *global rule* $G : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ of a cellular automaton maps a configuration $c \in S^{\mathbb{Z}^d}$ to the configuration $G(c)$ obtained by applying f uniformly in each cell: for all position $z \in \mathbb{Z}^d$, $G(c)(z) = f(c(z + v_1), \dots, c(z + v_k))$ where $N = \{v_1, \dots, v_k\}$. A *space–time diagram* of a given cellular automaton is a mapping $\Delta \in S^{\mathbb{N} \times \mathbb{Z}^d}$ such that for all time step $t \in \mathbb{N}$, $\Delta(t+1) = G(\Delta(t))$.

Example 1 [Figure 20](#) is a partial representation of a space–time diagram of the cellular automaton $(1, \mathbb{Z}_2, \{0, 1\}, f)$, where $f(x, y) = x + y$. State 0 is represented by white, state 1 by black. Time goes from bottom to top.

This chapter, considers, for the most part, cellular automata of dimensions 1 and 2 with the typical neighborhoods depicted in [Fig. 21](#): von Neumann $\{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ and Moore $\{-1, 0, 1\}^2$ in dimension 2, first neighbors $\{-1, 0, 1\}$ and one way $\{-1, 0\}$ in dimension 1.

Several subsets of the space of configurations are considered. Given a *quiescent state* q satisfying $f(q, \dots, q) = q$, a *q-finite configuration* c is a configuration equal to q in all but finitely

Fig. 21

Typical neighborhoods. (a) von Neumann. (b) Moore. (c) First neighbors. (d) One way.



many cells: there exists α such that for all position $z \in \mathbb{Z}^d$, $\|z\|_\infty > \alpha \rightarrow c(z) = q$. A configuration c admits p as a *periodicity vector* if for all position $z \in \mathbb{Z}^d$, $c(z + p) = c(z)$. A configuration c in dimension d is *periodic* if it admits a family of d non-colinear periodicity vectors: there exists $p \in \mathbb{N}^d$ such that $(p_1, 0, \dots, 0)$, $(0, p_2, 0, \dots, 0)$, \dots , and $(0, \dots, 0, p_d)$ are periodicity vectors of c . A configuration c in dimension d is *ultimately periodic* if there exists α and d non-colinear vectors v_i such that for all position $z \in \mathbb{Z}^d$ and all vector v_i , $\|z\|_\infty > \alpha \rightarrow c(z + v_i) = c(z)$. Notice that in dimension 1, an ultimately periodic configuration can have two different ultimately periodic patterns on each side.

Constraints can also be added to the local rule. Symmetries are usually considered to obtain more natural rules mimicking physical systems. A symmetry rule can be seen as a one-to-one mapping $\rho : \mathbb{Z}^d \rightarrow \mathbb{Z}^d$: the image of a configuration $c \in S^{\mathbb{Z}^d}$ by the symmetry rule ρ is the configuration $\rho(c)$ satisfying for all position $z \in \mathbb{Z}^d$, $\rho(c)(z) = c(\rho(z))$. A cellular automaton \mathcal{A} respects a symmetry rule ρ if ρ and G commute, that is, $\rho(G(c)) = G(\rho(c))$. Typical symmetries include reflections around point ($\rho_0(x, y) = (-x, -y)$), around axes ($\rho_x(x, y) = (-x, y)$) and rotations ($\theta(x, y) = (-y, x)$). A cellular automaton is *totalistic* if its set of states is a subset of \mathbb{N} and the local rule f can be written as $f(s_1, \dots, s_k) = g(\sum_{i=1}^k s_i)$. Totalistic rules respect all symmetries that preserve the neighborhood (*i.e.*, such that the image of the neighborhood by the symmetry rule is equal to the neighborhood): totalistic cellular automata with the von Neumann or Moore neighborhood are reflection and rotation invariants.

A cellular automaton is injective (resp. surjective, one-to-one) if its global rule is injective (resp. surjective, one-to-one). A cellular automaton \mathcal{A} is reversible if there exists a cellular automaton \mathcal{B} that reverts it, that is such that $G_{\mathcal{B}} \circ G_{\mathcal{A}}$ is the identity map.

Theorem 1 (Hedlund 1969; Richardson 1972) *A cellular automaton is reversible if and only if it is injective.*

Theorem 2 (Amoroso and Patt 1972) *It is decidable given a one-dimensional cellular automaton to decide whether it is reversible.*

Theorem 3 (Kari 1990, 1994) *It is undecidable given a two-dimensional cellular automaton to decide whether it is reversible.*

Whereas reversibility is an undecidable question, the construction of reversible cellular automata is possible, provided that the backward rule is constructed at the same time as the forward rule. Partitioned cellular automata provide a convenient way to construct reversible cellular automata. A *partitioned cellular automaton* is a cellular automaton with state

set $S_1 \times S_2 \times \dots \times S_k$ whose local rule can be rewritten as $f((s_1^1, \dots, s_1^k), \dots, (s_k^1, \dots, s_k^k)) = \varphi(s_1^1, s_2^2, \dots, s_k^k)$ where $\varphi : \prod S_i \rightarrow \prod S_i$ is the partitioned rule. As it is straightforward to verify, a partitioned cellular automaton is reversible iff partitioned rule is one-to-one. As the partitioned rule is a mapping from a finite set to itself, any partially defined injective rule can be completed to a reversible cellular automaton.

For a better and more complete introduction to the theory of cellular automata, see the chapter [❶ Basic Concepts of Cellular Automata](#), and/or Delorme (1999), and/or Kari (2005).

4 Chronology

It is a difficult task to give a fair and complete chronology of a research topic. This section proposes an exploration of the history of the field in three main eras:

1. The *computation and machines* era describes seminal papers outside the realm of cellular automata that lead to the main tools necessary to consider computation in the context of abstract machines.
2. The *universality and cellular automata* era is the core part of the chronology: it describes seminal papers along the path of universality study in the realm of cellular automata, from the early work of von Neumann in the 1950s to the end of the 1990s.
3. The *recent trends* era is a more subjective choice of some papers in the field in the twenty-first century.

4.1 Computation and Machines

Gödel (1931) in his now classical paper describing incompleteness theorems, introduces the so-called Gödel numberings: the ability to encode and manipulate a formal system inside itself if the system is complex enough. The concept of universality directly depends on such an encoding: a universal machine simulates a machine through its encoding. For a precise analysis from a logic and computer science point of view of Gödel's paper, see Lafitte (2008). **Turing (1936)**, while introducing Turing machines and proving the undecidability of the halting problem by a diagonal argument, also introduced his universal machine. Fixing an enumeration of Turing machines and a recursive bijective pairing function $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$, he describes a machine U that, on input $\langle m, n \rangle$, computes the same value as the machine encoded m on input n . Universality as a property is not discussed: a unique universal Turing machine U is given. For a discussion of the development of ideas from Leibniz to Turing results, see Davis (2000).

Post (1943) At that time, many different models of computation were proposed and proved equivalent, leading to the so-called Church–Turing thesis. Post introduces tag systems, a combinatorial word-based system successfully used since to construct size-efficient universal Turing machines. For a modern definition and discussion of tag systems, see Minsky (1967).

Kleene (1956) Finite state machines are at the heart of many models of computation. Kleene's paper proves the equivalence between three different families of objects: regular languages, finite automata, and Boolean circuits. Boolean circuits are modeled after the formal study of abstract neurons by McCulloch and Pitts (1943). This equivalence is fundamental both to concrete computer design and discrete models of computation like cellular automata. For a modern discussion on this equivalence and its consequences from the point of view of

computation and machines, see Minsky (1967). Perrin (1995) gives a history of this period and important achievements with respect to the field of finite automata and formal languages.

Minsky (1967) In the spirit of the question from Shannon (1956) about the size of a smallest Turing machine, Minsky explains how to efficiently encode tag systems computations into Turing machines and describe a universal Turing machine with four symbols and seven states. This marks the real start of a (still running) competition.

Lecerf (1963), Bennett (1973) Reversible computation is concerned with computing devices that can unroll their computation, going back in time. In their independent papers, Lecerf and Bennett prove that reversible Turing machines are able to simulate just any Turing machine. Thus, there exists reversible universal Turing machines.

Fredkin and Toffoli (1982) To encode classical computations into discrete models, Kleene's (1956) theorem permits to go freely from circuits to finite state machine, which is an essential ingredient for computation. Fredkin and Toffoli discuss an analogous for reversible computation: elementary building blocks to encode any reversible finite state machine as a reversible circuit. This paper also introduces the so-called billiard ball model of computation: a discrete cellular automata model to encode reversible computations. The encoding of reversible finite state machines into circuits was later improved by Morita (1990).

4.2 Universality and Cellular Automata

von Neumann (1966) Introducing cellular automata in order to construct a self-reproducing machine, reflecting the nature of life, von Neumann takes a fixed-point approach. His two-dimensional, 29 states, von Neumann neighborhood cellular automaton is able to simulate a particular kind of Turing machine that can also control a construction arm. The power of the construction arm is rich enough to construct with finitely many instructions a copy of the Turing machine itself. Whereas the machine is constructed with a form of Turing universality in mind, the simulation of the Turing machine is done with very simple components wiring down a particular family of Boolean circuits. As a consequence, the original cellular automaton is also, *a posteriori*, intrinsically universal. The construction of von Neumann leads to various improvements and discussions on the encoding of Boolean circuits, the different organs that compose the machine and the transmission of signals. A non-exhaustive list of interesting papers might be Arbib (1966), Burks (1970a, b, c), Moore (1962, 1970), and Thatcher (1970a, b).

Codd (1968) Following the principle of von Neumann's idea on self-reproduction, Codd drastically reduces the complexity of the automaton. Codd's two-dimensional rule uses eight states with the von Neumann neighborhood. Signals are conveyed by pairs of states (an oriented particle) moving between walls and reacting upon collision. This cellular automaton is also universal for Boolean circuits and so intrinsically universal. A later construction by Langton (1984), based on Codd ideas, has fewer states and a very simple family of self-reproducing loops but loses its computation universal capabilities.

Banks (1970) The work of Banks is noticeable with respect to several aspects and also because of its relatively small diffusion in the cellular automata community. Banks constructs a family of very small cellular automata (two-dimensional, von Neumann neighborhood, very symmetric, four to two states) simulating Boolean circuits in a very simple and modern way (signals moving in wires, Boolean gates on collisions), he identified and used explicitly the property of intrinsic universality and gave a transformation to construct relatively small

universal one-dimensional cellular automata with large neighborhoods starting from two-dimensional ones (re-encoding it into a one-dimensional first-neighbors automaton with 18 states). The construction of a two-dimensional four state universal cellular automaton in the spirit of Banks is provided by Noural and Kashef (1975).

Conway (1970) (Gardner, 1970; Berlekamp 1982) The Game of Life introduced by Conway is certainly among the most famous cellular automata and the first rule to be proven universal by analysis of a given rule rather than on purpose construction. A modern exposition of the Game of Life universality and a proof of its intrinsic universality was later proposed by Durand and Róka (1999).

Smith III (1971) The simulation of Turing machine by cellular automata to construct one-dimensional Turing universal cellular automata is studied by Smith III. Among several results, he explains how to construct a one-dimensional Turing universal cellular automaton with first neighbors and 18 states.

Toffoli (1977) Any cellular automaton of dimension d can be simulated, in a certain sense, by a cellular automaton of dimension $d + 1$. Using this assertion, Toffoli shows that two-dimensional reversible cellular automata can be Turing universal. The result was later improved by Hertling (1998).

Margolus (1984) While Toffoli transforms any Turing machine into a two-dimensional cellular automaton by using a new spatial dimension to store computational choices, Margolus constructs a Turing universal two-dimensional reversible cellular automaton by simulation of a bouncing billiard ball, complex enough to compute any reversible Boolean function of conservative logic. The billiard ball model cellular automaton has 16 states defined as two-by-two blocks of binary cells and von Neumann neighborhood.

Albert and Čulik (1987) Each cellular automaton can be simulated by a totalistic cellular automaton with one-way neighborhood. With the help of the last proposition, Albert and Čulik construct the first universal cellular automaton obtained by simulation of any cellular automaton of the same dimension. The automaton works along the following principle: each macro-cell copies the state of its left neighbor and adds it to its state obtaining some n , then by copying the n th element of a reference table, it selects its new state. Whereas the spirit of intrinsic universality is definitely there, the technical implementation is less clear. The one-dimensional first-neighbors automaton obtained has 14 states. The construction was later improved by Martin (1993, 1994) with better transition time complexity and using the smn theorem.

Morita and Harao (1989) Introducing partitioned cellular automata, Morita and Harao explicitly simulate any reversible Turing machine on a one-dimensional reversible cellular automaton, proving that one-dimensional reversible cellular automata can be Turing universal. The construction was later improved by Dubacq (1995), simulating any Turing machine in real time (without loss of time).

Lindgren and Nordahl (1990) The direct simulation of Turing machine on one-dimensional cellular automata proposed by Smith III can be improved and any m states n symbols machine can be simulated by a $(m + n + 2)$ -states cellular automaton following Lindgren and Nordhal. Applying this to Minsky's seven states and four symbols machine and then transforming the simple simulation into a macro-state signal-based simulation, Lindgren and Nordhal obtain a one-dimensional first neighbors seven-state Turing universal cellular automaton. The intrinsic universality status of this automaton is unknown.

Durand and Róka (1996) Revisiting the Game of Life and filling holes in the universality proof, Durand and Róka publish the first discussion on the different kinds of universality for cellular automata and the problem of formal definition.

Durand-Lose (1997) Using a modern definition of intrinsic universality, Durand-Lose goes one step further than Morita and Harao by constructing a reversible one-dimensional cellular automata intrinsically simulating any reversible cellular automaton.

4.3 Recent Trends

Imai and Morita (2000) The improvement in the construction of small and simple two-dimensional reversible cellular automata continues. Imai and Morita use partitioned cellular automata to define an eight-state universal automaton.

Ollinger (2002b) Using simulation techniques between cellular automata, strong intrinsically universal cellular automata with few states can be constructed, here six states.

Cook (2004) Very small universal cellular automata cannot be constructed, they have to be obtained by analysis. Realizing a real tour de force, Cook was able to prove the Turing universality of the two-states first-neighbors so-called rule 110 by analyzing signals generated by the rule and their collisions. The intrinsic universality of this automaton remains open. The original construction, simulation of a variant of tag system, was exponentially slow. For a proof of Cook's result using signals, see Richard (2008).

Neary and Woods (2006) Recently, the prediction problem of rule 110 was proven *P*-complete by Neary and Woods by a careful analysis and modification of Turing machines simulation techniques by tag systems. As *P*-completeness is required for intrinsic universality, this is another hint of the potential strong universality of rule 110.

Richard (2008) The limits of constructed small intrinsically universal cellular automata are converging toward analyzed cellular automata. Using particles and collisions, Richard was recently able to construct a four-state intrinsically universal first-neighbors one-dimensional cellular automaton.

5 Formalizing Universality

What is a universal cellular automaton? At first, the question might seem simple and superficial: a universal cellular automaton is an automaton able to compute anything recursive. In fact, a formal definition is both required and difficult to obtain. The requirement for such a definition is needed to define a frontier between cellular automata of maximal complexity and the others: in particular when considering the simplest cellular automata, to be able to identify the most complex cellular automata. The difficulty arises from the fact that a definition broad enough to encapsulate all constructions of the literature and all *fair enough* future constructions is required. For more details concerning this philosophical question, see Durand and Róka's (1999) attempt to give formal definitions.

5.1 Turing Universality

Turing universality is the easiest form of universality one might think about, that is, with a computability culture: let the cellular automaton *simulate* a well-known universal model of computation, either simulating one universal object of the family or any object of the family.

The first approach pushes back the problem to the following one: what is a universal Turing machine? What is a universal tag system? In its original work, Turing did not define

universal machines but *a unique* universal machine. The definition of universality for Turing machines was later discussed by Davis (1956) who proposed to rely on recursive degrees, defining universal machines as machines with maximal recursive degree. This definition, while formal, lacks precise practical view of encoding problems: the issue continues to be discussed in the world of Turing machines becoming more important, as smaller and smaller universal machines are proposed. For a view on the universality of Turing machines and pointers to literature related to the topic, see Woods (2007).

The second approach leads to the problem of heterogeneous simulation: classical models of computation have inputs, step function, halting condition, and output. Cellular automata have no halting condition and no output. As pointed out by Durand and Róka (1999), this leads to very tricky encoding problems: their own attempt at a Turing universality based on this criterion as encoding flaw permitting us counterintuitively to consider very simple cellular automata as universal.

Turing universality of dynamical systems in general and cellular automata in particular has been further discussed by Delvenne et al. (2006) and Sutner (2004). None of the proposed definitions is completely convincing so far, so it has been chosen on purpose not to provide the reader with yet another weak formal definition.

5.2 Intrinsic Universality

Intrinsic universality, on the other hand, is easier to formalize, yet a more robust notion (in the sense that variations along the lines of the definition lead to the same set of universal automata). Consider a homogenous type of simulation: cellular automata simulated by cellular automata in a shift invariant, time invariant way. A natural type of universal object exists in this context: cellular automata that are able to simulate each cellular automaton. Following the ideas of grouping and bulking (Rapaport 1998; Mazoyer and Rapaport 1999; Ollinger 2002a), a general notion of simulation broad enough to scope all reasonable constructions of the literature is introduced.

Direct simulation between two cellular automata can be formalized as follows. A cellular automaton \mathcal{B} *directly simulates* a cellular automaton \mathcal{A} , denoted $G_{\mathcal{A}} \prec G_{\mathcal{B}}$, of the same dimension according to a mapping $\varphi : S_{\mathcal{A}} \rightarrow 2^{S_{\mathcal{B}}}$ if for any pair of states $a, b \in S_{\mathcal{A}}$, $\varphi(a) \cap \varphi(b) = \emptyset$ and for any configuration $c \in S_{\mathcal{A}}^{\mathbb{Z}^d}$, $G_{\mathcal{B}}(\varphi(c)) \subseteq \varphi(G_{\mathcal{A}}(c))$.

For any state set S , let (m_1, \dots, m_d) be a tuple positive integers, the *unpacking bijective map* $o_{(m_1, \dots, m_d)} : (S^{\prod m_i})^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ is defined for any configuration $c \in (S^{\prod m_i})^{\mathbb{Z}^d}$ and any position $z \in \mathbb{Z}^d$ and $r \in \prod_i \mathbb{Z}_{m_i}$ as $o_{(m_1, \dots, m_d)}(c)(m_1 z_1 + r_1, \dots, m_d z_d + r_d) = c(z)(r)$. The *translation* of vector $v \in \mathbb{Z}^d$ is defined for any configuration $c \in S^{\mathbb{Z}^d}$ and position $z \in \mathbb{Z}^d$ as $\sigma_v(c)(z) = c(z - v)$.

Simulation between two cellular automata is extended by considering packing, cutting, and shifting of the two cellular automata such that direct simulation occur between both transformed objects. Universal objects are then maximum in the induced preorder. In fact, it can be proved that simulation on one side is sufficient for universal objects.

Definition 1 (intrinsic universality) A cellular automaton \mathcal{U} is intrinsically universal if for each cellular automaton \mathcal{A} of the same dimension there exists an unpacking map o_m , a positive integer $n \in \mathbb{N}$, and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Theorem 4 (Rapaport 1998; Mazoyer and Rapaport 1999) *No cellular automaton is intrinsically universal in real time (i.e., when constraining cutting constant n to be equal to $\max(m)$): simulation cannot perform both information displacement and transition computation at the same time.*

Theorem 5 (Ollinger 2003) *Given a cellular automaton, it is undecidable to determine whether it is intrinsically universal.*

Turing universality and intrinsic universality notions are really different notions. Some erroneous claims by Wolfram (1984, 2002) affirm, for example, that rule 110 is intrinsically universal. In fact, the question is yet open, Turing universality is the only proven thing.

Theorem 6 (Ollinger 2002a; Theyssier 2005a) *There exists Turing universal cellular automata which are not intrinsically universal. Moreover, some of them are at the bottom of an infinite increasing chain of equivalences classes of the preorder.*

Universality can also be discussed when considering language recognition or computation on grids. This topic is out of scope of the present paper. For more on this topic, see Mazoyer (1996, 1999).

6 High Dimensions: 2D and More

In two and more dimensions, an easy way to construct both intrinsically and Turing universal cellular automata is to go through Boolean circuit simulation. Boolean circuits can encode any finite state machine, and a cell of a cellular automaton or the control and tape of a Turing machine can be described as finite state machines.

6.1 Boolean Circuits Simulation

The topic of Boolean circuits simulation with cellular automata is quite popular and a lot has been written on it, see for example, recreations around the wireworld cellular automaton designed by Silverman and discussed in Dewdney (1990). This was already discussed with Cu in [Sect. 2](#). Technical hints and possible exotic extensions without entering details are given here.

To simulate Boolean circuits, one typically needs to mix the following ingredients.

Wires Boolean signals travel in piecewise straight line in space, their paths are the wires. Several encoding of Boolean signals with or without explicit wires are possible: moving particles encoded as states with a direction vector, bouncing on walls to turn as in game of life (Gardner 1970); wire path encoded as wire cells with explicit direction vector on each wire cell as in von Neumann (1966); undirected wire cells on which directed signals travel; undirected wire cells on which pairs of two different signal cells travel, the direction being given by the orientation of the pair as in wireworld (Dewdney 1990); pairs of undirected wire paths in between which pairs of two different signal cells travel as in Codd (1968) or Banks (1971).

Turn and delay Boolean signals should be able to turn in space and delay their arrival to permit signal synchronization.

Signal crossing In order to encode all Boolean circuits, crossing of signals has to be encoded either explicitly (adding crossing states) or implicitly using delaying techniques (as in von Neumann 1966) or Boolean logic tricks.

Gates Signals must be combined using Boolean gates at least taken in a Boolean universal family of gates. AND, OR, NOT are the classical ones but NAND or NOR is sufficient alone.

Fan-out Signals must be duplicated in some way either with an explicit fan-out state or using specific wire split rules.

Remarks and encoding tricks regarding Boolean circuit simulation:

- Universal Boolean function families and their expressive power are described in Post (1941). But, in cellular automata encoding, it is easy to use constants and multiple wires encoding, thus the number of Boolean classes depending on the implemented gates is finite and small.
- Clocks are only needed when dealing with some form of synchronized logic simulation. It is often used because Boolean signals are encoded with two values: empty wire or signal on wire. With such an encoding, NOT gate has to generate new signal on wire, and clock signal is used to determine at which time steps to do so. However, a classical coding trick to avoid the use of clocks and diodes is to only implement OR and AND gates and use the two wires trick to gain Boolean universality: a signal is encoded as one signal on one wire, the second being empty (thus no signal is encoded as no signal on both wires), then the NOT gate is just the wire crossing $(x, y) \mapsto (y, x)$, the AND gate can be encoded as $(x, y) \mapsto (x \wedge y, x \vee y)$ and the OR gate as $(x, y) \mapsto (x \vee y, x \wedge y)$. As both OR and AND produce signal only if there is at least one signal in input, the need for clock vanishes.
- Wire crossing can be gained for free by using the XOR gate as planar crossing can be implemented with XORs.
- Delays come for free if the wires can turn in all directions.
- In dimension 3, wire crossing is not needed, use the third dimension to route wires.
- Signal encoding can be done using signal constructions, in the spirit of Mazoyer and Terrier (1999), in order to reduce the number of states, see the chapter [Algorithmic Tools on Cellular Automata](#).

Remark Boolean circuit simulation is not restricted to square grids. As an example of a more exotic lattice, Gajardo and Goles (2001) encoded a Boolean circuit simulator on a hexagonal lattice (with proper cellular automata definition).

6.2 The Game of Life

[Section 2](#) provides an original example of a universal 2D cellular automaton. One of the most famous and celebrated cellular automata, the Game of Life, introduced by Conway (Gardner 1970; Berlekamp et al. 1982) is now discussed. For a hint of the richness of the game, the reader is referred to the Life lexicon (Silver S, http://www.argentum.freeserve.co.uk/lex_home.htm).

The Game of life is a two-dimensional cellular automaton with two states (dead or alive) and a Moore neighborhood. Each cell evolves according to the state of its eight neighbors. When a dead cell has exactly three alive neighbors, it becomes alive. Survival occurs when an alive cell has exactly two or three alive neighbors. Death occurs either by overcrowding when

an alive cell has five or more alive neighbors, or by exposure when an alive cell has one or no alive neighbor. The rule is completely rotation and mirror image invariant.

The Turing universality of the Game of Life was sketched by Conway et al. (1982). A more detailed proof including a simplification considering intrinsic universality was provided by Durand and Róka (1999). An explanation on how to use almost the same techniques as for Cu to improve their construction and avoid the clocking problems is given here.

6.2.1 Objects

To encode Boolean circuit's components, one has to choose proper elementary gate encoding elements.

Glider In the Game of Life, a particle on a wire is represented by a glider moving in straight line, as depicted in [Fig. 22](#). There is no physical representation of the wire itself. The glider slides diagonally one cell every four steps.

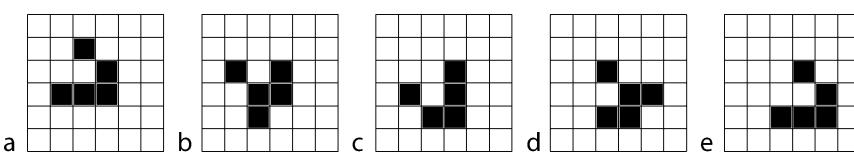
Crash When two orthogonal gliders meet, under proper synchronization conditions, a crash occurs, destroying both gliders without leaving garbage, as depicted in [Fig. 23](#).

Eater A special static object called an eater is used to destroy, under proper synchronization conditions, a colliding glider, leaving the eater in place, as depicted in [Fig. 24](#).

Duplicator A more elaborated object is the duplicator. Its static part consists only of a two-by-two block. When SE and NE gliders meet, under proper synchronization conditions, near the block, both gliders are destroyed, the block remains in place, and a new NE glider, with a different synchronization, is created, as depicted in [Fig. 25](#). If the input SE glider is not present, then the NE glider continues its way without being perturbed by the block. If the SE glider is considered an input x , one might see the duplicator as a widget with two outputs, an NE output with value $\neg x$ and an SE output with value x .

Gosper p46 gun A gun is a device emitting gliders at a given rate. The construction here uses the Gosper p46 gun, one of the first discovered guns, emitting gliders with period 46, as depicted in [Fig. 26](#). The construction would work with any gun with a period large enough to permit glider streams crossing.

 **Fig. 22**
GoL glider.



 **Fig. 23**
GoL crash (C).

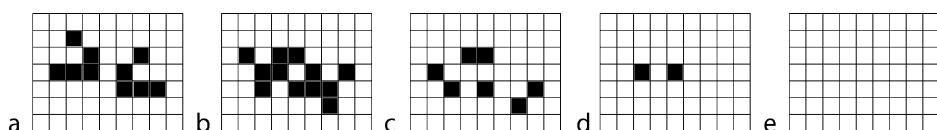


Fig. 24
GoL eater (E).

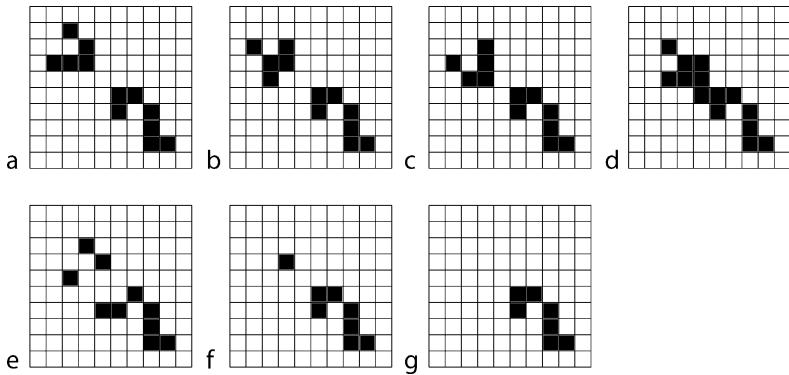
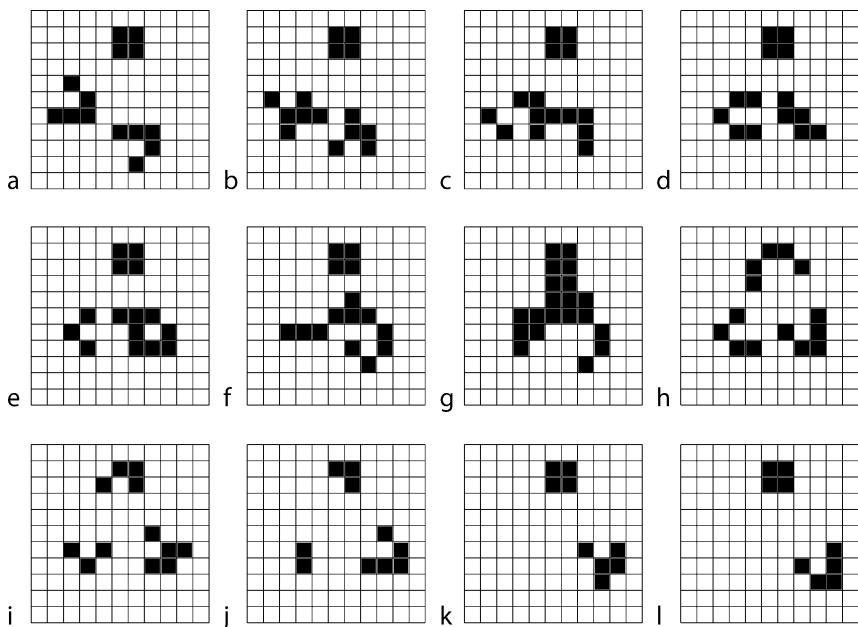


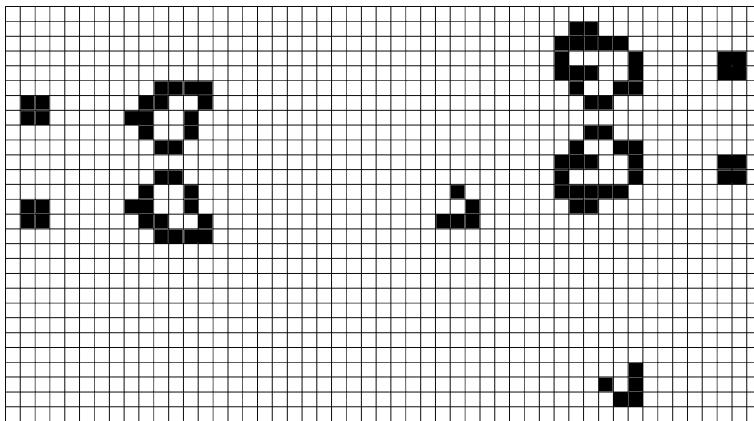
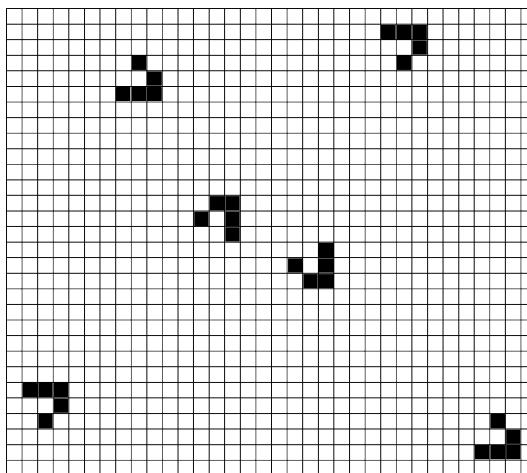
Fig. 25
GoL duplicator (D).



Crossing The use of guns will force one to synchronize gliders: on a wire, particles are encoded every period as a glider or no glider. The chosen period 46 permits that two properly synchronized streams cross, as depicted in [Fig. 27](#).

6.2.2 Gates

Using these objects, one can construct elementary gates. Wires are straight empty paths along which glider streams synchronized with period 46 can move straight forward.

Fig. 26**GoL Gosper p46 gun (G).****Fig. 27****GoL p46 crossing (X).**

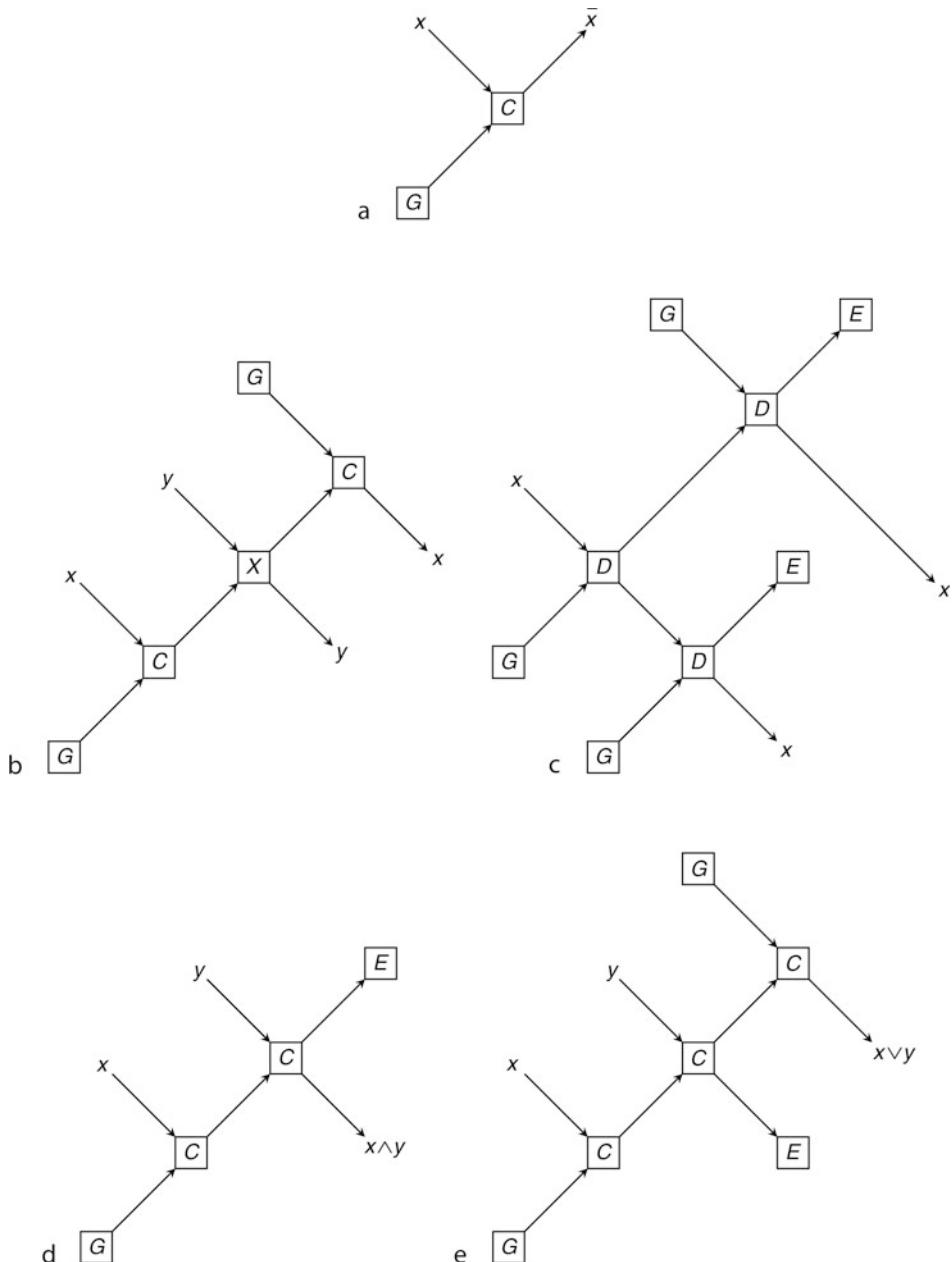
A first widget is the NOT turn widget, the principle of which is depicted in [Fig. 28a](#). Using a gun and crashes, it transforms an incoming stream of gliders into its negation, turning the stream by 90°. Several such widgets can be combined to achieve delays, side translation, and synchronization.

Using the objects and proper synchronization, eventually adding delays and translations, one can encode crossing, fan-out, AND gate, and OR gate as symbolically depicted in [Fig. 28](#).

Once these elementary gates are obtained, using the same encoding tricks as in [Sect. 2](#), every Boolean circuit can be encoded (e.g., a turn of a Boolean value encoded as a pair of glider streams is obtained by combining NOT turns and crossing).

Fig. 28

GoL (symbolic) elementary gates. (a) NOT turn. (b) Crossing. (c) Fan-out. (d) AND gate.
(e) OR gate.



6.2.3 Universalities

Using exactly the same techniques as in [Sect. 2](#), encoding finite state machines, one concludes that the Game of Life is intrinsically universal. To obtain the Turing universality of the Game of Life on finite configurations, in the style of Conway et al. (Berlekamp et al. 1982), one further needs to encode a counter on a finite configuration. As the main interest here is in intrinsic universality, universality on finite configurations is not discussed.

6.3 Banks' Two-State Universal CA

Small intrinsically universal cellular automata are quite simple to construct in dimension 2 with few states: Banks (1970, 1971) does it with two states and von Neumann neighborhood with reflection and rotation symmetry. Universality is yet again obtained using Boolean circuits simulation. The construction is briefly sketched.

The local rule is rotation and mirror image invariant. An alive cell with exactly two alive neighbors dies if the two alive neighbors are not aligned. A dead cell becomes alive if exactly three or four of its neighbors are alive. In all the remaining cases, the cell keeps its current state. **Particles on wires** Wires are encoded by thick bars of alive cells of thickness three, bordered by dead cells. Dead ends are encoded by extending the extremity column of the wire to be of thickness five. A particle on one side of a wire is encoded as two diagonal holes, as depicted in [Fig. 29](#).

Widgets To achieve all possible paths and fan-out, two widgets are provided, as depicted in [Fig. 30a, b](#). Moreover, one can construct an AND-NOT gate, as depicted in [Fig. 30c](#). Such a gate is not universal by itself. Fortunately, it can be combined by a clock to construct NOT gate and then a universal NOR gate. A clock is depicted in [Fig. 30d](#). Using NOR gates, one can construct an almost-crossing gate (the almost-crossing requires one of the inputs to have value 0). Combining almost-crossing and delays, one obtains a crossing.

Gates Combining fan-out, crossing and NOR gates, one can encode every Boolean circuit, as per [Sect. 2](#). This cellular automaton is intrinsically universal.

7 Turing Universality in 1D

In dimension one, Boolean circuit encoding is more puzzling as wire-crossing capabilities are bounded by the local rule. Thus, historically, computation universality is achieved by direct simulation of universal models of computations.

Fig. 29

Banks particle moving on a wire from left to right.

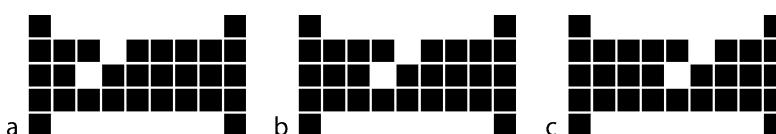
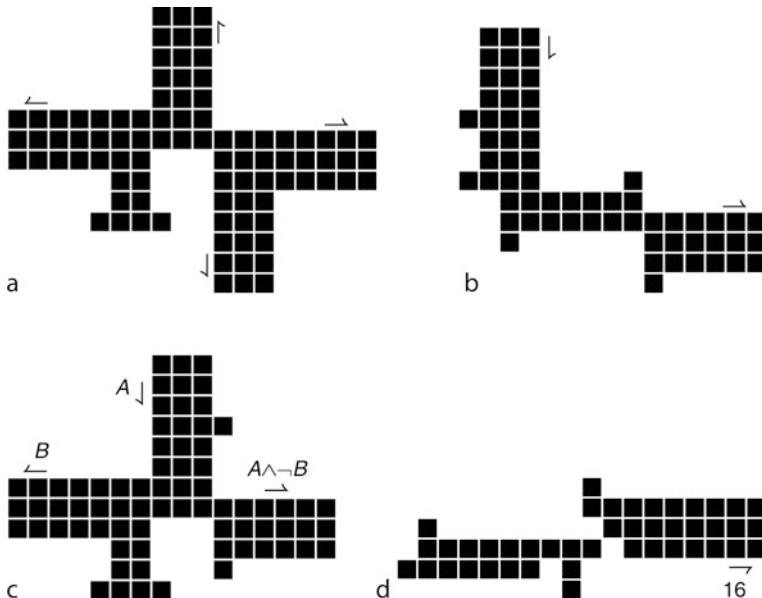


Fig. 30

Banks elementary widgets. (a) Fan-out. (b) Turn. (c) AND-NOT gate. (d) 16-Clock.



7.1 Universal Models of Computation

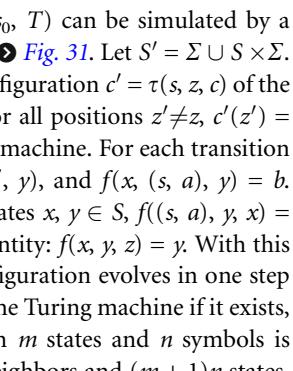
Turing machines Turing machines are easy to encode on cellular automata (see below) as an infinite tape really looks like a configuration of a cellular automaton. In fact, several variants of Turing machines exist and an important literature on universality in the Turing world provide useful objects to build small universal automaton based on this model. The question of existence of small universal Turing machines was first raised by Shannon (1956), different variants of Turing machines are discussed by Fischer (1965). For a survey on small Turing machine construction, see Woods and Neary (2007).

Tag systems Tag systems provide a better model to design very small universal objects. In fact, very small universal Turing machines are constructed by simulation of tag systems and their variants as originally proposed by Minsky (1967) and Cocke and Minsky (1964). The original drawback of tag system was its exponential slowdown when simulating Turing machines. This drawback was removed recently by Woods and Neary (2006, 2007) by achieving polynomial time simulation. The Turing universality of rule 110 is obtained by Cook (2004) by direct simulation of a proper variant of tag systems.

The variant of Turing machine used is the following. A *Turing machine* is a tuple (S, Σ, B, s_0, T) where S is a finite set of states, Σ is a finite alphabet with a special blank symbol $B \in \Sigma$, $s_0 \in S$ is the initial state, and $T : S \times \Sigma \rightarrow S \times \Sigma \times \{\leftarrow, \rightarrow\}$ is a partial transition map. A transition rule $T(s, a) = (s', b, d)$ reads as follows: when reading a from state s , write b on the tape, move in direction d , and enter state s' . A configuration of the machine is a triple (s, z, c) where $s \in S$ is the current state of the machine, $z \in \mathbb{Z}$ is the position of the head, and $c \in S^{\mathbb{Z}}$ is the content of the tape. The machine goes in one step, from a configuration (s, z, c) to a

configuration (s', z', c') , if the transition rule $T(s, c(z)) = (s'', d, b)$ is defined and verifies $s' = s''$, $z' - z = d$, $c'(z) = b$ and for all position $z'' \neq z$, $c'(z') = c(z)$. Starting from a configuration \mathbf{c} , a halting computation of the machine in time t consists of a sequence of configurations $(\mathbf{c}_i)_{i=0}^t$ such that $\mathbf{c}_0 = \mathbf{c}$, the machine cannot reach any configuration from \mathbf{c}_t and for all i , the machine goes in one step from \mathbf{c}_i to \mathbf{c}_{i+1} . The configuration \mathbf{c}_t is the output of the computation.

7.2 à la Smith III

Following Smith III (1971), a given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows, as depicted in  Fig. 31. Let $S' = \Sigma \cup S \times \Sigma$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: $c'(z) = (s, c(z))$ and for all positions $z' \neq z$, $c'(z') = c(z)$. The local rule encodes the transition function of the Turing machine. For each transition $T(s, a) = (s', b, \leftarrow)$, for all states $x, y \in S$, $f(x, y, (s, a)) = (s', y)$, and $f(x, (s, a), y) = b$. Symmetrically, for each transition $T(s, a) = (s', b, \rightarrow)$, for all states $x, y \in S$, $f((s, a), y, x) = (s', y)$, and $f(x, (s, a), y) = b$. All undefined transitions apply identity: $f(x, y, z) = y$. With this encoding, starting from an encoded configuration $\tau(\mathbf{c})$, the configuration evolves in one step to a configuration $\tau(\mathbf{c}')$ where \mathbf{c}' is the next computation step of the Turing machine if it exists, $\mathbf{c}' = \mathbf{c}$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $(m + 1)n$ states.

7.3 à la Lindgren and Nordahl

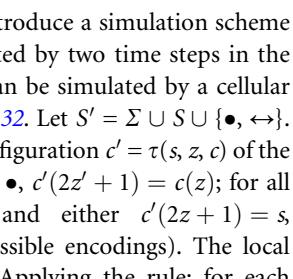
To lower the number of states, Lindgren and Nordahl (1990) introduce a simulation scheme where each step of the Turing machine computation is emulated by two time steps in the cellular automaton. A given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows, as depicted in  Fig. 32. Let $S' = \Sigma \cup S \cup \{\bullet, \leftrightarrow\}$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: for all $z' < z$, $c'(2z') = \bullet$, $c'(2z' + 1) = c(z)$; for all $z' > z$, $c'(2z' + 1) = \bullet$, $c'(2z' + 2) = c(z)$; $c'(2z) = \bullet$ and either $c'(2z + 1) = s$, $c'(2z + 2) = c(z)$ or $c'(2z + 1) = c(z)$, $c'(2z + 2) = s$ (two possible encodings). The local rule encodes the transition function of the Turing machine. Applying the rule: for each

 Fig. 31

Turing machine simulation à la Smith III.

B	B	(q_0, a)	a	B
B	B	a	(q_1, B)	B
B	B	(q_1, b)	B	B
B	(q_0, B)	b	B	B
B	B	(q_0, a)	B	B

Fig. 32

Turing machine simulation à la Lindgren and Nordahl.

B	\bullet	B	\bullet	a	q_0	\bullet	a	\bullet	B
B	\leftrightarrow	B	\leftrightarrow	a	\leftrightarrow	q_0	a	\leftrightarrow	B
B	\bullet	B	\bullet	a	\bullet	q_1	B	\bullet	B
B	\leftrightarrow	B	\leftrightarrow	a	q_1	\leftrightarrow	B	\leftrightarrow	B
B	\bullet	B	\bullet	q_1	b	\bullet	B	\bullet	B
B	\leftrightarrow	B	q_1	\leftrightarrow	b	\leftrightarrow	B	\leftrightarrow	B
B	\bullet	B	q_0	\bullet	b	\bullet	B	\bullet	B
B	\leftrightarrow	B	\leftrightarrow	q_0	b	\leftrightarrow	B	\leftrightarrow	B
B	\bullet	B	\bullet	q_0	a	\bullet	B	\bullet	B

transition $T(s, a) = (s', b, \leftarrow)$, $f(\bullet, s, a) = s'$, $f(s, a, \bullet) = b$, $f(\bullet, a, s) = s'$, $f(a, s, \bullet) = b$, for each transition $T(s, a) = (s', b, \rightarrow)$, $f(\bullet, s, a) = b$, $f(s, a, \bullet) = s'$, $f(\bullet, a, s) = b$, $f(a, s, \bullet) = s'$. Moving: for all $s \in S$, $a, b \in \Sigma$, $f(\leftrightarrow, s, a) = \bullet$, $f(a, \leftrightarrow, s) = sf(a, s, \leftrightarrow) = \bullet$, $f(s, \leftrightarrow, a) = s$. All undefined transitions apply the identity rule but for \bullet and \leftrightarrow that alternates: for all states $x, y \in S$, $f(x, \bullet, y) = \leftrightarrow$ and $f(x, \leftrightarrow, y) = \bullet$. With this encoding, starting from an encoded configuration $\tau(c)$, the configuration evolves in two steps to a configuration $\tau(c')$ where c' is the next computation step of the Turing machine if it exists, $c' = c$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $m + n + 2$ states.

7.4 à la Cook

Simulation of tag systems is more tricky due to the non-locality of one computation step of the system. Following Cook (2004), one can consider cyclic tag systems. A cyclic tag system is given as a finite set of words (w_0, \dots, w_{N-1}) on the alphabet $\{\circ, \bullet\}$. A configuration of the system is a word $u \in \{\circ, \bullet\}^*$. At time step t , the configuration u evolves to a configuration v according to either $u_0 = \circ$ and $u_0 v = u$, or $u_0 = \bullet$ and $u_0 v = uw_{t \bmod N}$. Cyclic tag systems can encode any recursive function. To encode all cyclic tag systems in a same cellular automaton $(1, S, \{-1, 0, 1\}, f)$, one can follow the following principle. Encode each configuration u of a cyclic tag system (w_0, \dots, w_{N-1}) as a configuration of the kind ${}^\omega(T_-^k) \cdot u \cdot \blacksquare (w_0 \blacktriangle \dots \blacktriangle w_{N-1})^\omega$ where intuitively T is a clock signal, \blacksquare is the frontier between u and the rule and the rule is repeated on the right, each word separated by a \blacktriangle . Giving the complete local rule is tedious but its principle can be sketched: each time a clock signal hits the first letter of u , it erases it and sends a signal to the right transporting the value of that letter; when the signal meets the \blacksquare it removes it and begins to treat the w word on the right, either erasing it or just crossing it; when the signal meets a \blacktriangle , it changes it into a \blacksquare and the signal disappears. This principle is used by Cook to simulate cyclic tag systems with rule 110 particles and collisions.

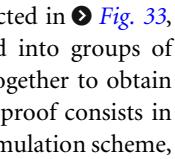
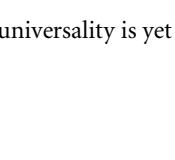
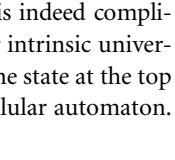
7.5 Rule 110

The universality of rule 110 is certainly one of the most complicated construction and a real tour de force. It is yet the smallest known Turing universal cellular automaton. Due to its size, universality was obtained by analysis. There is not enough room here to completely describe

the construction, so the reader will only be given an idea of its taste. For a complete exposure of the proof, read the original paper by Cook (2004) or the modern particle-based exposure by Richard (2008).

Rule 110 is a two state one-dimensional cellular automaton with a first-neighbors neighborhood. Its local rule can be given as follows:

$$f(x, y, z) = \lfloor 110/2^{4x+2y+z} \rfloor \bmod 2$$

To encode every cyclic tag system (to be precise, a subfamily of cyclic tag systems rich enough to encode any recursive function) into space–time diagrams of rule 110 following the technique described above, Cook selected, by careful analysis, 18 particles, depicted in  Fig. 33, and 23 collisions, depicted in  Figs. 34–36. These particles are arranged into groups of parallel particles to encode bits and other signals. Collisions are grouped together to obtain the proper interaction in between particle groups. The tedious part of the proof consists in proving that, with proper synchronization, particles combine to follow the simulation scheme, as depicted in  Fig. 37.

Cook's construction proved the Turing universality of rule 110. Intrinsic universality is yet still open.

Open Problem *Is rule 110 intrinsically universal?*

Another step was obtained by Neary and Woods, showing that rule 110 is indeed complicated. P -completeness of the prediction problem is a necessary condition for intrinsic universality. The prediction problem consists, given a segment of states, to decide the state at the top of the computation triangle obtained by iterating the local rule of a fixed cellular automaton.

Theorem 7 (Neary and Woods 2006) *The prediction problem of rule 110 is P -complete.*

7.6 Encoding Troubles

The main point of discussion here is to decide which kind of configurations are acceptable for encoding Turing universal computation. Finite configurations are certainly not a problem and using any potentially non-recursive configuration would permit trivial cellular automata to be misleadingly called universal. The previous constructions involving Turing machines use finite or ultimately periodic configurations with a same period on both sides, the same one for all simulated machines, whereas the tag system encoding uses ultimately periodic configurations with different periods, moreover these periodic parts depend on the simulated tag system. The tag system really needs this ultimate information, as transforming the simulating cellular automaton into one working on finite configurations would have a constant but large impact on the number of states. As pointed in Durand and Róka (1999), this configuration encoding problem adds difficulties to the formal definition of Turing universality.

8 Intrinsic Universality in 1D

Even if the concept of intrinsically universal cellular automata took some time to emerge, intrinsic universality does not require more complex constructions to be achieved.

Fig. 33

110 particles used in the construction (with highlighted finite coloring).

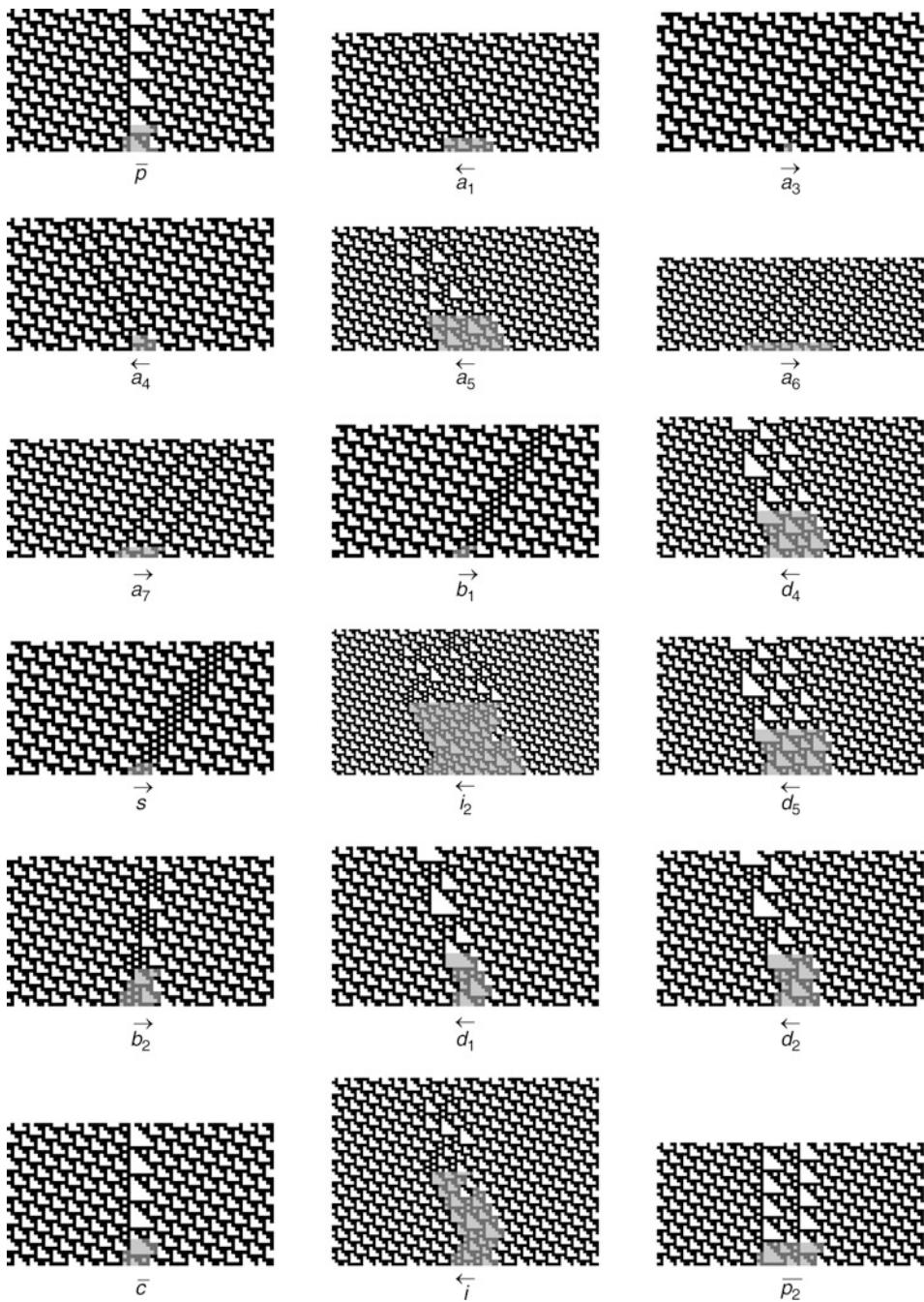


Fig. 34

110 collisions used in the construction.

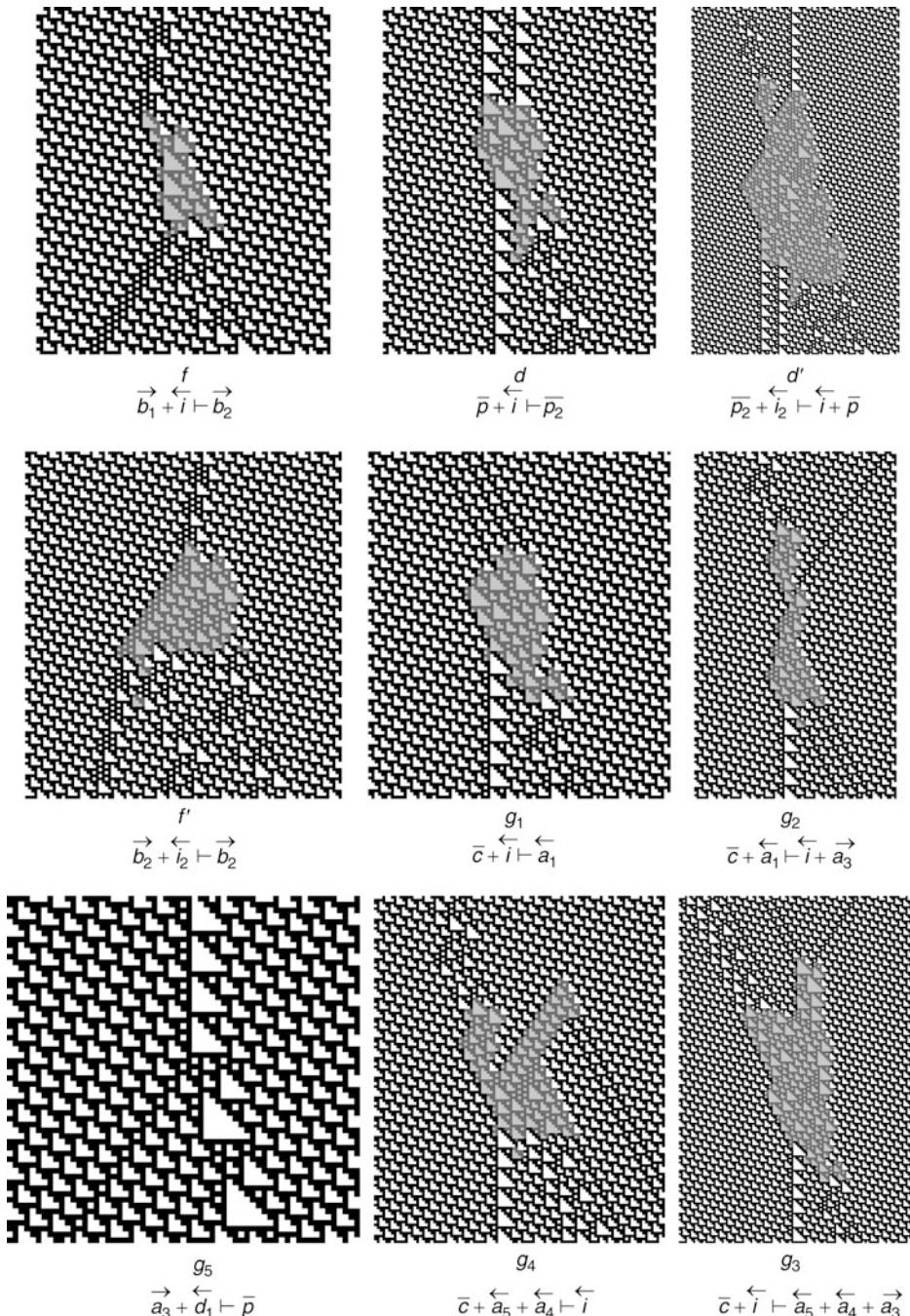
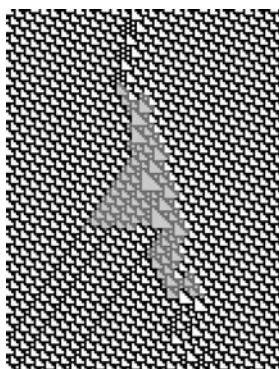
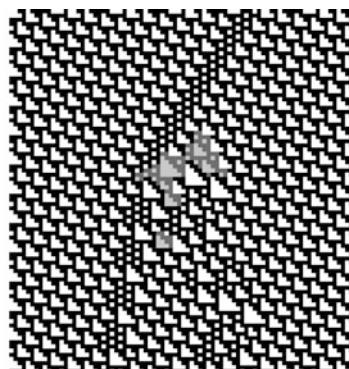


Fig. 35

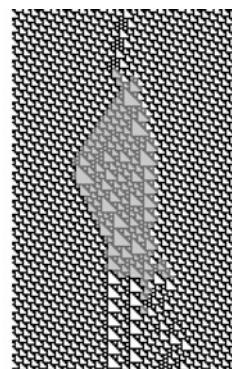
110 collisions used in the construction (cont.).



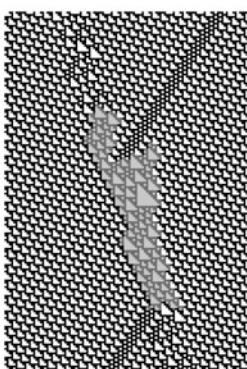
$$g'_7 \quad \overrightarrow{a}_6 + i \vdash \overrightarrow{b}_2$$



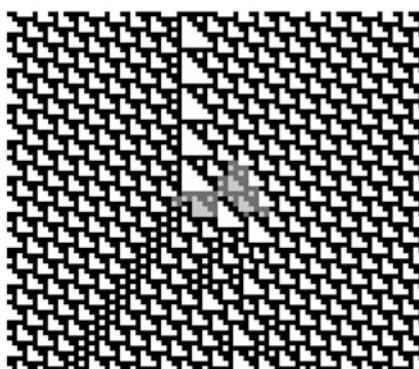
$$g'_8 \quad \overrightarrow{b}_2 + i \vdash \overrightarrow{b}_1$$



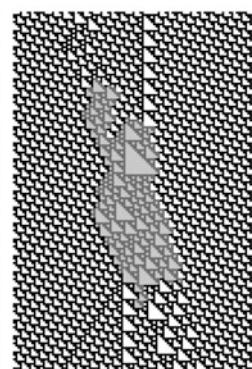
$$g_8 \quad \overleftarrow{p}_2 + i \vdash \overrightarrow{b}_2$$



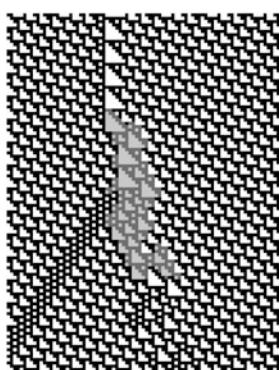
$$h \quad \overrightarrow{s} + i \vdash \overleftarrow{i} + \overrightarrow{s}$$



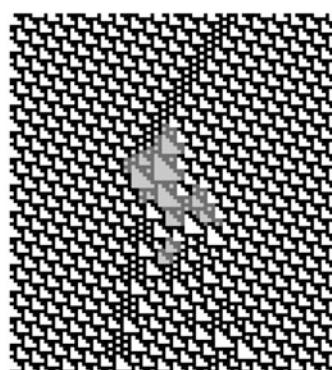
$$g'_9 \quad \overrightarrow{b}_1 + i \vdash \overleftarrow{p}$$



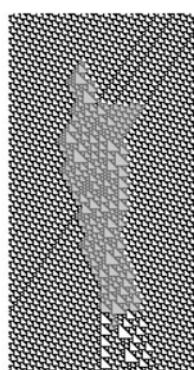
$$g_6 \quad \overrightarrow{a}_3 + \overleftarrow{p} + \overleftarrow{d}_4 \vdash \overleftarrow{i} + \overleftarrow{p}$$



$$k \quad \overrightarrow{s} + i \vdash \overleftarrow{c}$$



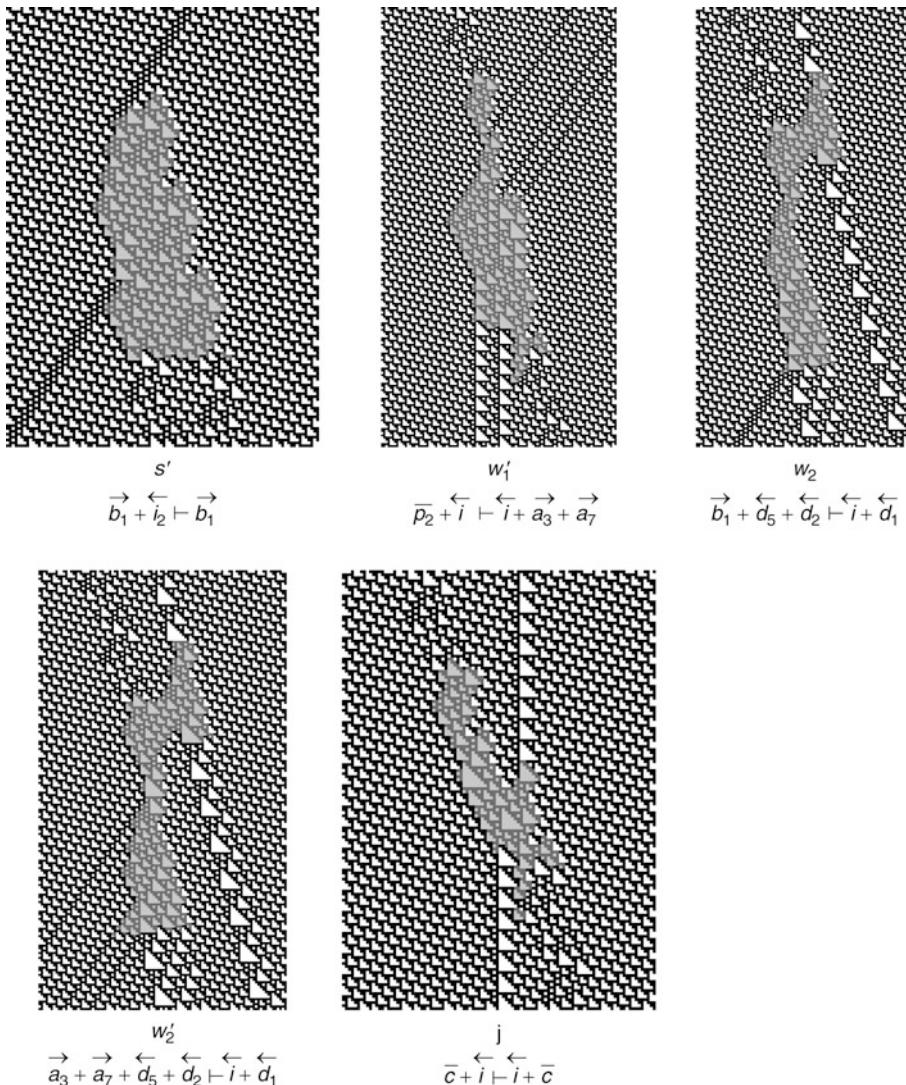
$$s \quad \overrightarrow{b}_2 + i \vdash \overrightarrow{b}_1$$



$$g'_6 \quad \overrightarrow{a}_3 + \overleftarrow{p} + \overleftarrow{d}_4 \vdash \overrightarrow{a}_6$$

Fig. 36

110 collisions used in the construction (end).



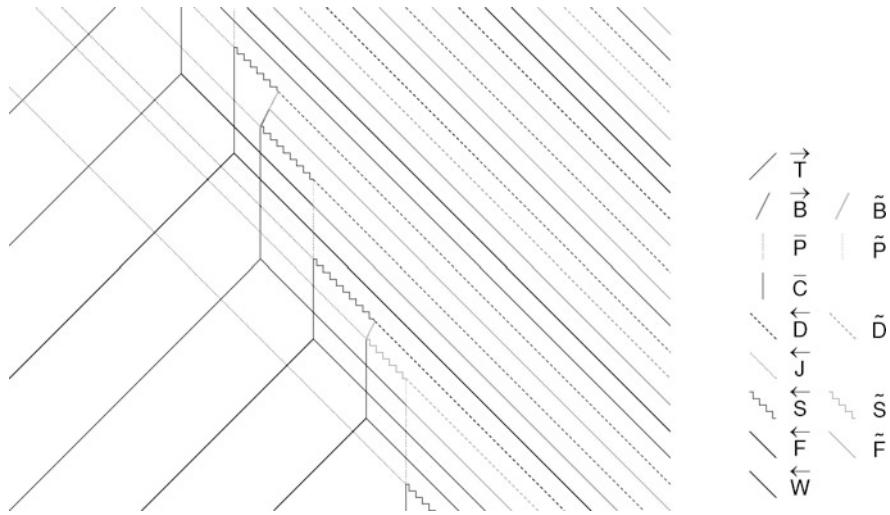
8.1 Techniques

Several techniques are used to construct them:

Parallel Turing machines table lookup A simple way to achieve intrinsic universality is to use synchronized parallel Turing heads (one copy of the same Turing machine per encoded cell) to lookup in the transition table (one copy in each encoded cell) of the encoded cellular automaton. Notice that the Turing machines used for this are not the same ones that are

Fig. 37

110 symbolic description of the simulation.



Turing universal. In fact, their computational power is very small but they can carry precise information movement tasks.

One-way totalistic lookup Another more cellular automata-centric way to achieve intrinsic universality is, following Albert and Čulik (1987), to simplify the task of the previous machine by simulating only one-way totalistic cellular automata that are sufficient to simulate all cellular automata.

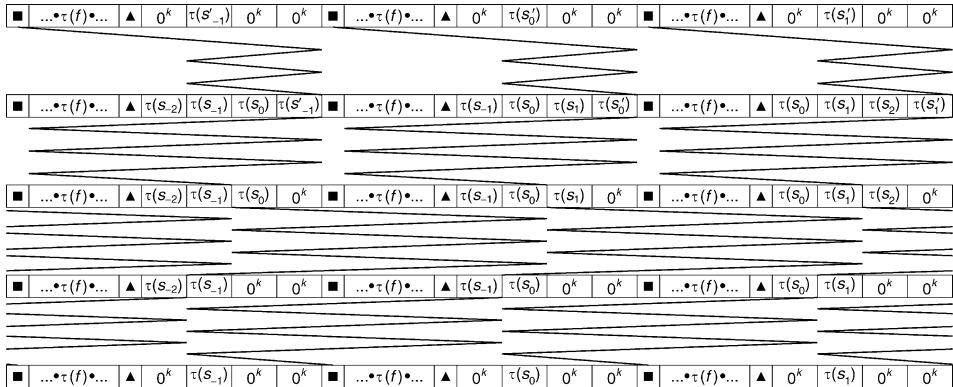
Signals The previous models are complex because the information displacement involved is still complex due to the sequential behavior of head-based machines. Following Ollinger (2002b) and Ollinger and Richard (2008), particles and collisions, that is, signals in the style of Mazoyer and Terrier (1999), can be used to encode the information and perform the lookup task with parallel information displacement.

8.2 Parallel Turing Machines Table Lookup

The parallel Turing machines table lookup technique is explained here, the other ones being refinements based on it. The one-dimensional first-neighbors universal cellular automaton \mathcal{U} simulates a cellular automaton $(1, S, \{-1, 0, 1\}, f)$ the following way, as depicted in Fig. 38. Each configuration c is encoded as the concatenation of $\psi_{\mathcal{A}}(c(z))$ for all z . For each state $s \in S$, $\psi_{\mathcal{A}}(s)$ is a word of the kind $\blacksquare \tau(f(1, 1, 1)) \bullet \tau(f(1, 1, 2)) \bullet \dots \bullet \tau(f(N, N, N)) \blacktriangle 0^k \tau(s) 0^k 0^k$ where N is the size of S , k is the number of bits needed to encode numbers from 1 to N , and $\tau(s)$ is a binary encoding of the state s . The simulation proceeds as follows so that the movement of each head is the same, up to translation, on each well encoded configuration. First the \blacksquare letter is activated as a Turing head in initial state. The Turing head then moves to the left and copies the state $\tau(s_L)$ of the left neighbor in place of the first 0^k block. Then it symmetrically copies the state $\tau(s_R)$ of the right neighbor in place of the second 0^k block.

Fig. 38

Principle of parallel Turing machine table lookup.



This being done, the head scans the entire transition table, incrementing a counter (e.g., stored on top of the encoded states) at each step: if at some point the counter is equal to the triple of states, the result is copied from the transition table to the third 0^k block. At the end of the scan, the counter information is cleared, the result of the transition is copied in the $\tau(s)$ place and all three 0^k blocks are restored. The head then goes back to the ■. Using this simulation, one step of the simulated cellular automaton is simulated in a constant number of steps for each cell by each Turing head. The universal automaton does not depend on the simulated automaton and is so intrinsically universal. A careful design can lead to less than 20 states. If the simulation uses the one-way totalistic technique, encoding states in unary, then it is easy to go under 10 states.

Notice that general Turing universal cellular automata construction schemes from previous section concerning Turing machines can be adapted to produce small intrinsically universal cellular automata: apply the encoding schemes on machines performing the cell simulation task. However, the tag system simulations do not provide direct way to obtain intrinsic universality. Moreover, it is possible to design cellular automata Turing universal by tag system simulation and not intrinsically universal.

8.3 Four States

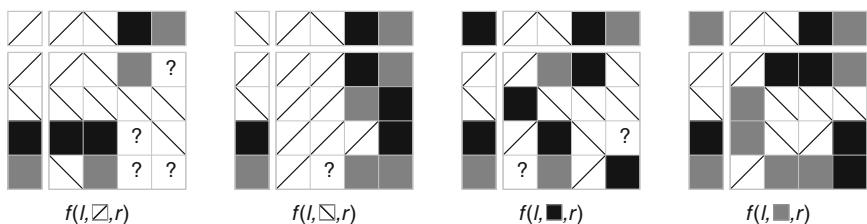
The smallest known intrinsically universal one-dimensional cellular automaton with first neighbors has four states. Due to its size, universality was obtained by a combination of design and analysis. There is not enough room here to completely describe the construction, so the reader will only be given an idea of its taste. For a complete description of the proof, read Ollinger and Richard (2008).

The local rule is given in Fig. 39 with five particles and several families of signals (groups of particles encoding integers and integer sequences). The principle of the construction (well, in fact it is a bit more complicated due to encoding problems) is to simulate a one-way cellular automaton encoding states as an integer encoded into a signal. The simulation sums the signals from the cell and its neighbors and reads the result of the transition in a transition table

Fig. 39

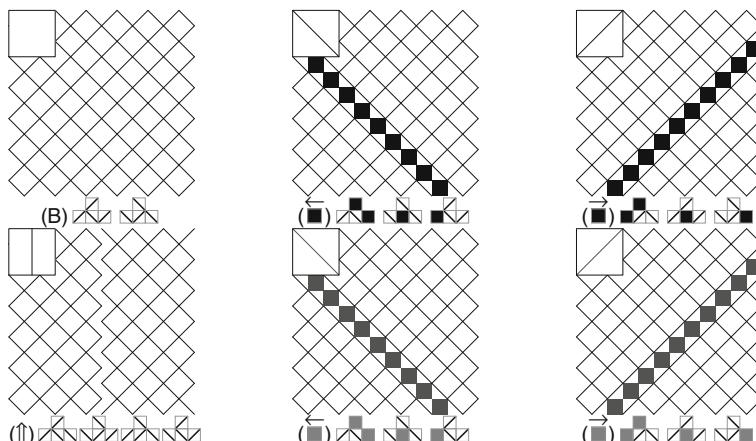
4st local rule, background, particles, and signals.

Local rule

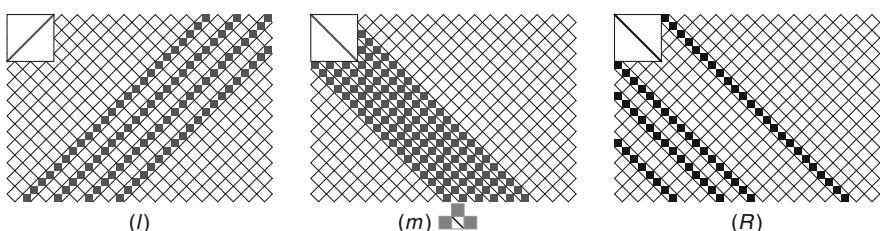


Question marks denote transitions that are not used in the construction

Background and particles



Signals



Collisions

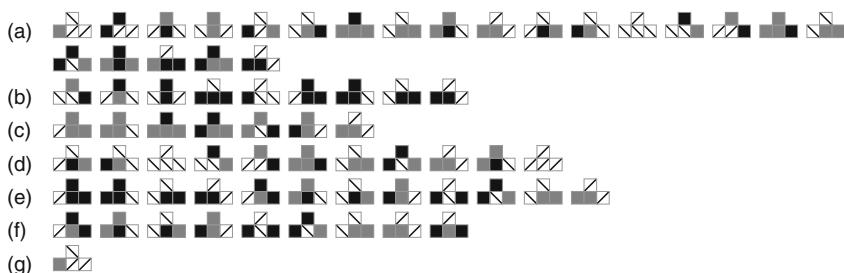
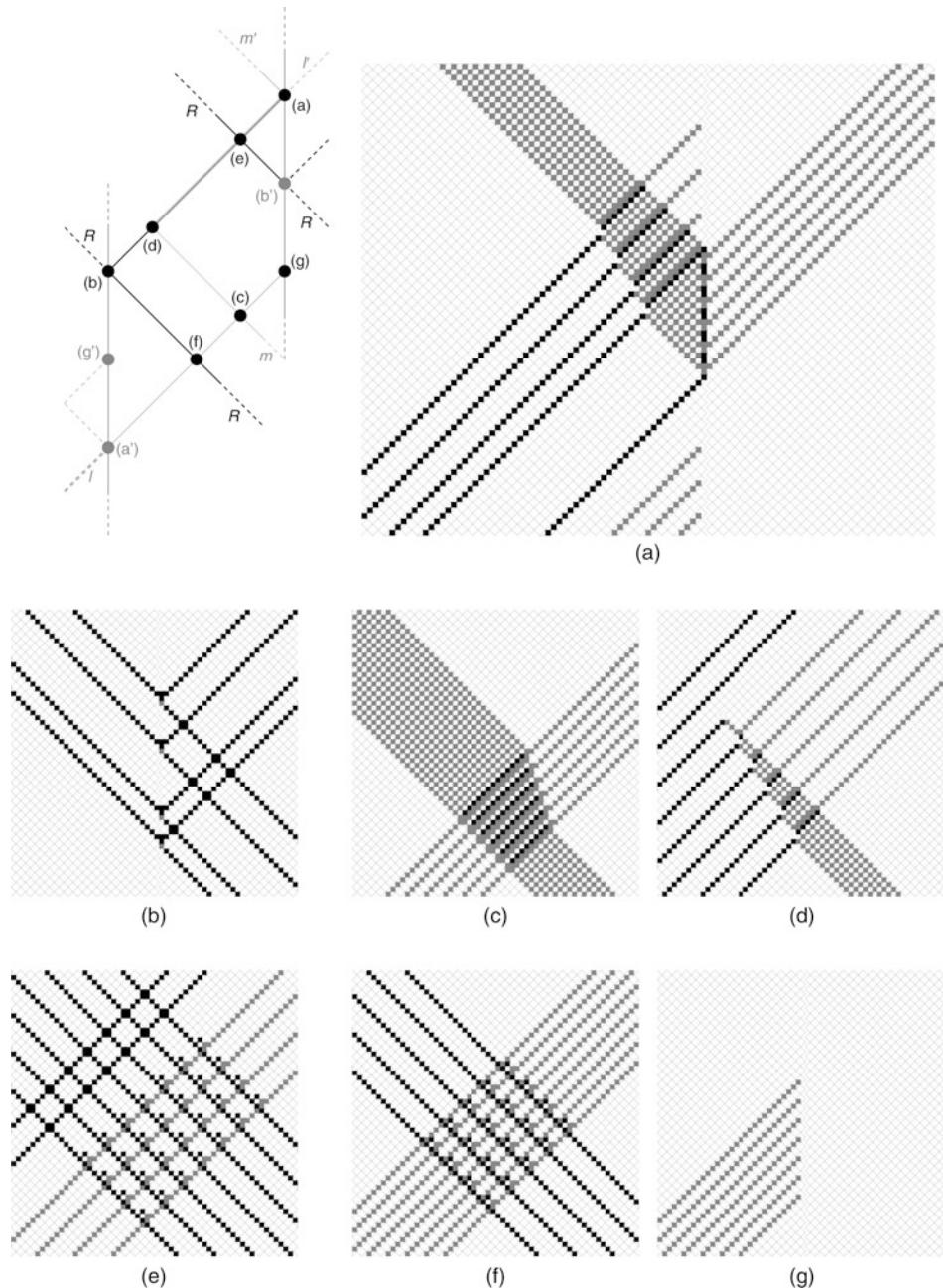


Fig. 40**4st simulation scheme.**

encoded in a signal as a sequence of integers. Finally, it duplicates the result. The simulation is depicted in [Fig. 40](#) including the different collisions.

8.4 Computing Under Constraints

More exotic intrinsically universal cellular automata have been studied on constrained rules. As an example, Moreira ([2003](#)) constructed number-conserving intrinsically universal automata, Bartlett and Garzon ([1993, 1995](#)) and Ollinger ([2001](#)) did the same for bilinear cellular automata, and Theyssier ([2005b](#)) for captive cellular automata for which he proves that almost all captive automata are intrinsically universal.

9 Sub-universalities: The Reversible Case

Universality might also be considered a proper subfamily of cellular automata, giving birth to some notions of sub-universality. Reversible cellular automata are special in the sense that they can achieve Turing universality as any Turing machine can be simulated by a reversible Turing machine but they cannot achieve intrinsic universality: reversible cellular automata only simulate reversible cellular automata. However, there exists reversible cellular automata which are universal with respect to the class of reversible cellular automata.

The chapter [Reversible Cellular Automata](#) of the present volume, authored by Morita, is dedicated to reversible cellular automata, the reader is referred to that chapter for a proper discussion on reversible computing and reversible Boolean circuits.

Definition 2 (reversible intrinsic universality) A reversible cellular automaton \mathcal{U} is intrinsically universal for reversible cellular automata if for each reversible cellular automaton \mathcal{A} of the same dimension there exists an unpacking map o_m , a positive integer $n \in \mathbb{N}$, and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Turing universality and a weak form of intrinsic universality have been proposed by Morita ([1995](#)), Morita and Imai ([1996, 2001](#)), Durand-Lose ([1995, 1996](#)), and Miller and Fredkin ([2005](#)). As for classical cellular automata in higher dimensions, the simulation of reversible Boolean circuits automatically gives reversible intrinsic universality.

For one-dimensional cellular automata, reversible intrinsic universality can be achieved by simulating any one-way-reversible partitioned-reversible cellular automaton with a first-neighbor-reversible partitioned-reversible cellular automaton. How to use a scheme similar to parallel Turing machine table lookup with reversible Turing machines to achieve this goal is briefly sketched. The first adaptation is to remark that the local partition rule of a reversible automaton is a permutation, thus it can be encoded as a finite sequence of permutation pairs. So, the table of transition is encoded as a finite sequence of pairs of states. The reversible Turing-machine task is to scan the transition table and for each pair it contains to replace the actual state by the second element of the pair if the state appears in the current pair. It is technical but straightforward to see that a reversible Turing machine can achieve this. The information movement is reversible, as in partitioned automata each cell gives half its state to a neighbor and takes half a state from the other without erasing any information.

Developing this simulation scheme, one constructs a reversible intrinsically universal cellular automaton.

10 Conclusion

This chapter has given the reader keys both to further explore the literature and to construct by himself conceptually simple Turing universal and/or intrinsically universal cellular automata in one and two dimensions. The following broad bibliography can be explored with the help of the main text, all references being cited.

Acknowledgment

The author would like to thank G. Richard for providing and preparing figures to illustrate both rule 110 and the 4-state intrinsically universal cellular automaton.

References

- Albert J, Čulík II K (1987) A simple universal cellular automaton and its one-way and totalistic version. *Complex Syst* 1(1):1–16
- Amoroso S, Patt YN (1972) Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *J Comput Syst Sci* 6:448–464
- Arbib MA (1966) Simple self-reproducing universal automata. *Info Control* 9(2):177–189
- Banks ER (1970) Universality in cellular automata. In: Symposium on switching and automata theory, Santa Monica, 1970. IEEE, New York, pp 194–215
- Banks ER (1971) Information processing and transmission in cellular automata. Ph.D. thesis, Massachusetts Institute of Technology
- Bartlett R, Garzon M (1993) Monomial cellular automata. *Complex Syst* 7(5):367–388
- Bartlett R, Garzon M (1995) Bilinear cellular automata. *Complex Syst* 9(6):455–476
- Bennett CH (1973) Logical reversibility of computation. *IBM J Res Dev* 17(6):525–532
- Berlekamp ER, Conway JH, Guy RK (1982) Winning ways for your mathematical plays, vol 2, Games in particular. Academic Press [Harcourt Brace Jovanovich Publishers], London
- Burks AW (1970a) Von Neumann's self-reproducing automata. In: Burks AW (ed) Essays on cellular automata. University of Illinois Press, Urbana, IL, pp 3–64 (Essay one)
- Burks AW (1970b) Programming and the theory of automata. In: Burks AW (ed) Essays on cellular automata. University of Illinois Press, Urbana, IL, pp 65–83 (Essay two)
- Burks AW (1970c) Towards a theory of automata based on more realistic primitive elements. In: Burks AW (ed) Essays on cellular automata. University of Illinois Press, Urbana, IL, pp 84–102 (Essay three)
- Cocke J, Minsky M (1964) Universality of tag systems with $p=2$. *J ACM* 11(1):15–20
- Codd EF (1968) Cellular automata. Academic Press, New York
- Cook M (2004) Universality in elementary cellular automata. *Complex Syst* 15:1–40
- Davis MD (1956) A note on universal turing machines. In: Shannon CE, McCarthy J (eds) Automata studies. Princeton University Press, Princeton, NJ, pp 167–175
- Davis MD (2000) The universal computer: The road from Leibniz to Turing. W.W. Norton, New York
- Delorme M (1999) An introduction to cellular automata: some basic definitions and concepts. In: Delorme M, Mazoyer J (eds) Cellular automata (Saissac, 1996). Kluwer, Dordrecht, pp 5–49
- Delvenne J, Kurka P, Blondel VD (2006) Decidability and universality in symbolic dynamical systems. *Fundam Info* 74(4):463–490
- Dewdney AK (1990) The cellular automata programs that create Wireworld, Rugworld and other diversions. *Sci Am* 262:146–149
- Dubacq JC (1995) How to simulate Turing machines by invertible one-dimensional cellular automata. *Int J Found Comput Sci* 6(4):395–402
- Durand B, Róka Z (1999) The Game of Life: universality revisited. In: Delorme M, Mazoyer J (eds) Cellular automata (Saissac, 1996). Kluwer, Dordrecht, pp 51–74

- Durand-Lose J (1995) Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In: Baeza-Yates RA, Goles E, Poblete PB (eds) LATIN '95, Valparaiso, 1995. Lecture notes in computer science, vol 911. Springer, Berlin/New York, pp 230–244
- Durand-Lose J (1996) Automates cellulaires, automates partitions et tas de sable. Ph.D. thesis, Université Bordeaux I
- Durand-Lose J (1997) Intrinsic universality of a 1-dimensional reversible cellular automaton. In: STACS 97 (Lübeck), Lecture notes in computer science, vol 1200. Springer, Berlin, pp 439–450
- Fischer PC (1965) On formalisms for Turing machines. J ACM 12(4):570–580
- Fredkin E, Toffoli T (1982) Conservative logic. Int J Theor Phys 21:219–253
- Gajardo A, Ch EG (2001) Universal cellular automaton over a hexagonal tiling with 3 states. IJAC 11(3):335–354
- Gardner M (1970) Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'. Sci Am 223(4):120–123
- Gödel K (1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik 38:173–198. Reprinted in Feferman S, Dawson W, Kleene SC, Moore G, Solovay RM, van Heijendorf J (eds) (1986) Kurt Gödel: Collected works, vol 1. Oxford University Press, Oxford, pp 144–195
- Golly Gang: Golly version 2.0 (2008) An open-source, cross-platform application for exploring the Game of Life and other cellular automata. <http://golly.sf.net/>
- Hedlund GA (1969) Endomorphisms and automorphisms of the shift dynamical system. Math Syst Theory 3:320–375
- Hertling P (1998) Embedding cellular automata into reversible ones. In: Calude CS, Casti J, Dinneen MJ (eds) Unconventional models of computation. Springer
- Imai K, Morita K (2000) A computation-universal two-dimensional 8-state triangular reversible cellular automaton. Theor Comput Sci 231(2):181–191
- Kari J (1990) Reversibility of 2D cellular automata is undecidable. Phys D. Nonlinear Phenomena 45(1–3): 379–385. Cellular automata: theory and experiment, Los Alamos, NM, 1989
- Kari J (1994) Reversibility and surjectivity problems of cellular automata. J Comput Syst Sci 48(1):149–182
- Kari J (2005) Theory of cellular automata: a survey. Theor Comput Sci 334:3–33
- Kleene SC (1956) Representation of events in nerve nets and finite automata. In: Shannon CE, McCarthy J (eds) Automata studies. Princeton University Press, Princeton, NJ, pp 3–41
- Lafitte G (2008) Gödel incompleteness revisited. In: Durand B (ed) Symposium on cellular automata journées automates cellulaires, JAC'2008, Uzès, 2008. MCCME, Moscow, pp 74–89
- Langton C (1984) Self-reproduction in cellular automata. Phys D 10(1–2):135–144
- Lecerf Y (1963) Machines de Turing réversibles. C R Acad Sci Paris 257:2597–2600
- Lindgren K, Nordahl MG (1990) Universal computation in simple one-dimensional cellular automata. Complex Syst 4(3):299–318
- Margolus N (1984) Physics-like models of computation. Phys D 10:81–95
- Martin B (1993) Construction modulaire d'automates cellulaires. Ph.D. thesis, École Normale Supérieure de Lyon
- Martin B (1994) A universal cellular automaton in quasi-linear time and its $S\text{-}m\text{-}n$ form. Theor Comput Sci 123(2):199–237
- Mazoyer J (1996) Computations on one dimensional cellular automata. Ann Math Artif Intell 16: 285–309
- Mazoyer J (1999) Computations on grids. In: Cellular automata (Saissac, 1996). Kluwer, Dordrecht, pp 119–149
- Mazoyer J, Rapaport I (1999) Inducing an order on cellular automata by a grouping operation. Discrete Appl Math 91(1–3):177–196
- Mazoyer J, Terrier V (1999) Signals in one-dimensional cellular automata. Theor Comput Sci 217(1):53–80. Cellular automata, Milan, 1996
- McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133
- Miller DB, Fredkin E (2005) Two-state, reversible, universal cellular automata in three dimensions. In: Bagherzadeh N, Valero M, Ramírez A (eds) Proceedings of the conference on computing frontiers, Ischia, pp 45–51. ACM
- Minsky M (1967) Computation: finite and infinite machines. Prentice Hall, Englewood Cliffs, NJ
- Moore EF (1962) Machine models of self-reproduction. In: Proceedings of symposia in applied mathematics, vol 14, 1962. American Mathematical Society, Providence, pp 17–33
- Moore EF (1970) Machine models of self-reproduction. In: Burks AW (ed) Essays on cellular automata. University of Illinois Press, Urbana, IL, pp 187–203 (Essay six)
- Moreira A (2003) Universality and decidability of number-conserving cellular automata. Theor Comput Sci 292(3):711–721
- Morita K (1990) A simple construction method of a reversible finite automaton out of Fredkin gates, and its related problem. IEICE Trans Info Syst E73(6):978–984

- Morita K (1995) Reversible simulation of one-dimensional irreversible cellular automata. *Theor Comput Sci* 148(1):157–163
- Morita K, Harao M (1989) Computation universality of one-dimensional reversible (injective) cellular automata. *IEICE Trans Info Syst* E72:758–762
- Morita K, Imai K (1996) Self-reproduction in a reversible cellular space. *Theor Comput Sci* 168(2):337–366
- Morita K, Imai K (2001) Number-conserving reversible cellular automata and their computation-universality. *Theor Inf Appl* 35(3):239–258
- Neary T, Woods D (2006) P-completeness of cellular automaton rule 110. In: *Proceedings of ICALP 2006*, Venice. Lecture notes in Computer science, vol 4051. Springer, Berlin, pp 132–143
- von Neumann J (1966) In: Burks AW (ed) *Theory of self-reproducing automata*. University of Illinois Press, Urbana, IL
- Noural F, Kashef R (1975) A universal four-state cellular computer. *IEEE Trans Comput* 24(8):766–776
- Odifreddi PG (1989) Classical recursion theory. Elsevier, Amsterdam
- Ollinger N (2001) Two-states bilinear intrinsically universal cellular automata. In: Freivalds R (ed) *Fundamentals of computation theory*, Riga, 2001. Lecture notes in computer science, vol 2138. Springer, Berlin, pp 396–399
- Ollinger N (2002a) Automates cellulaires: structures. Ph.D. thesis, École Normale Supérieure de Lyon
- Ollinger N (2002b) The quest for small universal cellular automata. In: Widmayer P, Triguero F, Morales R, Hennessy M, Eidenbenz S, Conejo R (eds) *International colloquium on automata, languages and programming*, (Málaga, 2002). Lecture notes in computer science, vol 2380. Springer, Berlin, pp 318–329
- Ollinger N (2003) The intrinsic universality problem of one-dimensional cellular automata. In: *Symposium on theoretical aspects of computer science*, Berlin, 2003. Lecture notes in computer science. Springer, Berlin, doi:10.1007/3-540-36494-3_55
- Ollinger N (2008) Universalities in cellular automata: a (short) survey. In: Durand B (ed) *Symposium on cellular automata journées automates cellulaires*, JAC'2008, Uzès. MCCME, Moscow, pp 102–118
- Ollinger N, Richard G (2008) A particular universal cellular automaton. In: Neary T, Woods D, Seda AK, Murphy N (eds) *CSP*. Cork University Press, Cork, Ireland
- Perrin D (1995) Les débuts de la théorie des automates. *Tech Sci Info* 14:409–433
- Post E (1941) The two-valued iterative systems of mathematical logic. Princeton University Press, Princeton, NJ
- Post E (1943) Formal reductions of the general combinatorial decision problem. *Am J Math* 65(2):197–215
- Rapaport I (1998) Inducing an order on cellular automata by a grouping operation. Ph.D. thesis, École Normale Supérieure de Lyon
- Richard G (2008) Rule 110: Universality and catenations. In: Durand B (ed) *Symposium on cellular automata journées automates cellulaires*, JAC'2008, Uzès. MCCME, Moscow, pp 141–160
- Richardson D (1972) Tessellations with local transformations. *J Comput Syst Sci* 6:373–388
- Shannon CE (1956) A universal Turing machine with two internal states. In: Shannon CE, McCarthy J (eds) *Automata studies*. Princeton University Press, Princeton, NJ, pp 157–165
- Smith AR III (1971) Simple computation-universal cellular spaces. *J ACM* 18:339–353
- Sutner K (2004) Universality and cellular automata. In: Margenstern M (ed) *MCU 2004*, Saint-Petersburg, 2004. Lecture notes in computer science, vol 3354. Springer, Berlin/Heidelberg, pp 50–59
- Thatcher JW (1970a) Self-describing Turing machines and self-reproducing cellular automata. In: Burks AW (ed) *Essays on cellular automata*. University of Illinois Press, Urbana, IL, pp 103–131 (Essay four)
- Thatcher JW (1970b) Universality in the von Neumann cellular model. In: Burks AW (ed) *Essays on cellular automata*. University of Illinois Press, Urbana, IL, pp 132–186 (Essay five)
- Theysier G (2005a) Automates cellulaires: un modèle de complexités. Ph.D. thesis, École Normale Supérieure de Lyon
- Theysier G (2005b) How common can be universality for cellular automata? In: STACS 2005, Stuttgart, 2005. Lecture notes in computer science, vol 3404. Springer, Berlin/Heidelberg, pp 121–132
- Toffoli T (1977) Computation and construction universality of reversible cellular automata. *J Comput Syst Sci* 15(2):213–231
- Turing AM (1936) On computable numbers with an application to the Entscheidungsproblem. *Proc Lond Math Soc* 42(2):230–265
- Wolfram S (1984) Universality and complexity in cellular automata. *Phys D Nonlinear Phenomena* 10(1–2): 1–35. Cellular automata: Los Alamos, NM, 1983
- Wolfram S (2002) *A new kind of science*. Wolfram Media, Champaign, IL
- Woods D, Neary T (2006) On the time complexity of 2-tag systems and small universal Turing machines. In: FOCS 2006, Berkeley, 2006. IEEE Computer Society, Washington, DC, pp 439–448
- Woods D, Neary T (2007) The complexity of small universal Turing machines. In: Cooper SB, Löwe B, Sorbi A (eds) *Computability in Europe*, CiE 2007, Siena, 2007. Lecture notes in computer science, vol 4497. Springer, Berlin/Heidelberg, pp 791–799

7 Reversible Cellular Automata

Kenichi Morita

Department of Information Engineering, Graduate School of
Engineering, Hiroshima University, Japan
morita@iec.hiroshima-u.ac.jp

1	<i>Introduction</i>	232
2	<i>Definitions and Basic Properties</i>	233
3	<i>Finding Reversible Cellular Automata</i>	236
4	<i>Computing Ability of Reversible Cellular Automata</i>	240
5	<i>Concluding Remarks</i>	254

Abstract

A reversible cellular automaton (RCA) is a special type of cellular automaton (CA) such that every configuration of it has only one previous configuration, and hence its evolution process can be traced backward uniquely. Here, we discuss how RCAs are defined, their properties, how one can find and design them, and their computing abilities. After describing definitions on RCAs, a survey is given on basic properties on injectivity and surjectivity of their global functions. Three design methods of RCAs are then given: using CAs with block rules, partitioned CAs, and second-order CAs. Next, the computing ability of RCAs is discussed. In particular, we present simulation methods for irreversible CAs, reversible Turing machines, and some other universal systems by RCAs, in order to clarify the universality of RCAs. In spite of the strong constraint of reversibility, it can be seen that RCAs have rich abilities in computing and information processing, and even very simple RCAs have universal computing ability.

1 Introduction

An RCA is a CA whose global transition function (i.e., a mapping from configurations to configurations) is injective. The study of RCAs has a relatively long history. In the early 1960s, Moore (1962) and Myhill (1963) showed the so-called Garden-of-Eden theorem, which gives a relation between injectivity and surjectivity of a global function. Note that a Garden-of-Eden configuration is one that can exist only at time 0, and hence if there is such a configuration in a CA, then its global function is not surjective. Since then, injectivity and surjectivity of global functions were studied extensively and more generally. But the term “reversible” CA was not used in the early stage of the history.

The notion of reversibility comes from physics. It is one of the fundamental microscopic physical laws of nature. Landauer (1961) argued how physical reversibility is related to “logical reversibility” (roughly speaking, logical reversibility is a property of “backward determinism” in a computing system). He posed Landauer’s principle that states that an irreversible logical operation inevitably causes heat generation in a physical computing system. After that, various models of reversible computing, such as reversible Turing machines (Bennett 1973; Lecerf 1963) and reversible logic circuits (Fredkin and Toffoli 1982; Toffoli 1980), appeared, and the research field of reversible computing was formed.

Toffoli (1977) first studied RCAs from such a viewpoint. He showed that every k -dimensional (irreversible) CA can be embedded in a $k + 1$ -dimensional RCA, and hence a two-dimensional RCA is computation-universal. An RCA can be thought of as an abstract model of physically reversible space. Therefore, it is a useful framework to formalize and study the problem of how computation and information processing are performed efficiently in physically reversible spaces. Since then, much research on RCAs was performed, and it gradually turned out that RCAs have rich abilities in computing, though an RCA is a very special subclass of CA.

An outline of this chapter is as follows. In [Sect. 2](#), it is first shown how RCAs are defined, and then there is a discussion on their basic properties of injectivity and surjectivity. In [Sect. 3](#), useful frameworks for designing RCAs are discussed. This is because it is in general hard to find RCAs if the standard framework of CAs is used. Here, three frameworks are explained: CAs with block rules, partitioned CAs, and second-order CAs. In [Sect. 4](#), simulation

methods of other (irreversible) CAs, reversible Turing machines, cyclic tag systems, and reversible logic circuits, by one- and two-dimensional RCAs, are given. By this, one can see computation universality of such RCAs, and some of them can be very simple.

2 Definitions and Basic Properties

In this section, formal definitions on cellular automata (CAs), and RCAs are given. To define reversibility of a CA, the notions of injectivity and invertibility are needed, which later turn out to be equivalent. Basic properties related to RCAs are also discussed.

2.1 Definitions

A CA is a system consisting of infinitely many finite automata (called cells) placed and interconnected uniformly in a space. Each cell changes its state depending on the current states of the neighbor cells, according to a given local transition function. It is assumed that all the cells change their states synchronously at every discrete time step. Hence, the whole state (called configuration) of the cellular space also changes at every discrete time, and this gives a global transition function. Until now many interesting variants of CAs, such as asynchronous CAs, CAs in a non-Euclidian space, and so on, have been proposed and studied, but here we restrict them only to standard ones, that is, deterministic CAs in the k -dimensional Euclidian space. A formal definition of them is given as follows.

Definition 1 A k -dimensional m -neighbor cellular automaton (CA) is a system defined by

$$A = (\mathbb{Z}^k, Q, (n_1, \dots, n_m), f)$$

Here, \mathbb{Z} is the set of all integers, and thus \mathbb{Z}^k is the set of all k -dimensional points with integer coordinates at which cells are placed. Q is a nonempty finite set of states of each cell, and (n_1, \dots, n_m) is an element of $(\mathbb{Z}^k)^m$ called a *neighborhood* ($m = 1, 2, \dots$). $f: Q^m \rightarrow Q$ is a *local function*, which determines how each cell changes its state depending on the states of m neighboring cells.

A k -dimensional configuration over Q is a mapping $\alpha: \mathbb{Z}^k \rightarrow Q$. Let $\text{Conf}(Q)$ denote the set of all k -dimensional configurations over Q , that is, $\text{Conf}(Q) = \{\alpha \mid \alpha: \mathbb{Z}^k \rightarrow Q\}$. If some state $\# \in Q$ that satisfy $f(\#, \dots, \#) = \#$ is specified as a *quiescent state*, then the notion of finiteness of a configuration can be defined as follows. A configuration α is called *finite* iff the set $\{x \mid x \in \mathbb{Z}^k \wedge \alpha(x) \neq \#\}$ is finite. Otherwise it is *infinite*. Let $\text{Conf}_{\text{fin}}(Q)$ denote the set of all k -dimensional finite configurations over Q , that is, $\text{Conf}_{\text{fin}}(Q) = \{\alpha \mid \alpha \in \text{Conf}(Q) \wedge \alpha \text{ is finite}\}$.

The *global function* $F: \text{Conf}(Q) \rightarrow \text{Conf}(Q)$ of A determines how a configuration changes to other configuration. It is defined by the following formula.

$$\forall \alpha \in \text{Conf}(Q), x \in \mathbb{Z}^k : F(\alpha)(x) = f(\alpha(x + n_1), \dots, \alpha(x + n_m))$$

Let F_{fin} denote the restriction of F to $\text{Conf}_{\text{fin}}(Q)$ provided that a quiescent state is specified. One can see every finite configuration is mapped to a finite configuration by F . Therefore, F_{fin} defines a mapping $\text{Conf}_{\text{fin}}(Q) \rightarrow \text{Conf}_{\text{fin}}(Q)$.

A reversible CA will be defined as a special subclass of a (standard) CA. To do so, two notions related to reversibility of CAs are first introduced. They are injectivity and invertibility.

Definition 2 Let $A = (\mathbb{Z}^k, Q, (n_1, \dots, n_m), f)$ be a CA.

1. A is called an *injective CA* iff its global function F is one-to-one.
2. A is called an *invertible CA* iff there exists a CA $A' = (\mathbb{Z}^k, Q, N', f')$ that satisfies the following condition.

$$\forall \alpha, \beta \in \text{Conf}(Q) : F(\alpha) = \beta \text{ iff } F'(\beta) = \alpha$$

where F and F' are the global functions of A and A' , respectively. Here, A' is called the *inverse CA* of A .

It is clear from the above definition that if a CA is invertible then it is injective. One can see that its converse also holds from the results independently proved by Hedlund (1969) and Richardson (1972). This is stated by the following theorem.

Theorem 1 (Hedlund 1969 and Richardson 1972) *Let A be a CA. A is injective iff it is invertible.*

Since the notions of injectivity and invertibility in CAs are thus proved to be equivalent, the terminology *reversible CA* (RCA) for such a CA will be used hereafter. Note that, in the case of reversible Turing machines (Bennett 1973), or reversible logic circuits (Fredkin and Toffoli 1982), the situation is somewhat simpler. In these models, injectivity is defined on their “local” operations. For example, a reversible logic circuit is a one consisting of logic elements with injective logical functions. Hence, the operation (or the state transition) of the whole circuit is also injective. Furthermore, by replacing each logic element by the one with the inverse logical function an inverse logic circuit that “undoes” the original one can be easily obtained. Thus, injectivity of a circuit is trivially equivalent to its invertibility. Therefore, in these models, reversibility can be directly defined without introducing the two notions of injectivity and invertibility.

2.2 Injectivity and Surjectivity of CAs

Historically, injectivity and surjectivity of a global function of CAs were first studied in the Garden-of-Eden problem (Moore 1962; Myhill 1963). A configuration α is called a *Garden-of-Eden configuration* of a CA A iff it has no preimage under the global function F , that is, $\forall \alpha' \in \text{Conf}(Q) : F(\alpha') \neq \alpha$. Hence, existence of a Garden-of-Eden configuration is equivalent to non-surjectivity of the global function F . Moore (1962) first showed the following proposition: if there are “erasable configurations” (which are mapped to the same configuration by the global function) then there is a Garden-of-Eden configuration (it is sometimes called Moore’s Garden-of-Eden theorem). Later, Myhill (1963) proved its converse. Their results are now combined and stated in the following theorem called the Garden-of-Eden theorem.

Theorem 2 (Moore 1962 and Myhill 1963) *Let A be a CA, and F be its global function. F is surjective iff F_{fin} is injective.*

Since it is clear if F is injective then F_{fin} is also so, the following corollary is derived.

Corollary 1 *If F is injective, then it is surjective.*

Note that there is a CA that is surjective but not injective (Amoroso and Cooper 1970). Hence the converse of this corollary does not hold. After the works of Moore and Myhill, relations among injectivity and surjectivity in unrestricted, finite, and periodic configurations have been extensively studied (Amoroso and Cooper 1970; Maruoka and Kimura 1976, 1979; Richardson 1972).

Next, decision problems on injectivity and surjectivity on CAs are considered. For the case of one-dimensional CAs, injectivity and surjectivity of the global function are both decidable.

Theorem 3 (Amoroso and Patt 1972) *There is an algorithm to determine whether the global function of a given one-dimensional CA is injective (surjective) or not.*

Later, a quadratic time algorithm to determine the injectivity (surjectivity) of a global function using de Bruijn graphs was given by Sutner (1991).

For two-dimensional CAs, injectivity (i.e., equivalently reversibility) and surjectivity are both undecidable. Thus, they are also undecidable for the higher dimensional case. The following theorem was proved by a reduction from the tiling problem, which is a well-known undecidable problem.

Theorem 4 (Kari 1994) *The problem whether the global function of a given two-dimensional CA is injective (surjective) is undecidable.*

2.3 Inverse CAs and Their Neighborhood Size

For each reversible CA, there is an inverse CA, which undoes the operations of the former. In the two-dimensional case, the following fact on the inverse CAs (Kari 1994) can be observed. Let A be an arbitrary two-dimensional reversible CA, and A' be the inverse CA of A . One cannot, in general, compute an upper-bound of the neighborhood size $|N'|$ from A . If one can assume, on the contrary, such an upper-bound is computable, then the total number of candidates of the inverse CA of A is also bounded. Since it is easy to test if a CA \tilde{A} is the inverse of the CA A for any given \tilde{A} , the inverse of A can be found among the above candidates in finite steps. From this, one can design an algorithm to decide whether a given CA is reversible or not, which contradicts \blacktriangleright Theorem 4. It is stated in the following theorem.

Theorem 5 (Kari 1994) *There is no computable function g that satisfies $|N'| \leq g(|Q|, |N|)$ for any two-dimensional reversible CA $A = (\mathbb{Z}^2, Q, N, f)$ and its inverse $A' = (\mathbb{Z}^2, Q, N', f')$.*

On the other hand, for the case of one-dimensional CAs, the following result has been shown.

Theorem 6 (Czeizler and Kari 2007) *Let $A = (\mathbb{Z}, Q, N, f)$ be an arbitrary one-dimensional reversible CA, and $A' = (\mathbb{Z}, Q, N', f')$ be its inverse, where $|Q| = n$ and $|N| = m$. If N consists of m consecutive cells, then $|N'| \leq n^{m-1} - (m - 1)$. Furthermore, when $m = 2$, this upper-bound is tight.*

3 Finding Reversible Cellular Automata

As seen in [Sect. 2.2](#), it is in general hard to find RCAs. From [Theorem 4](#), it is not possible to test if a given two-dimensional CA is reversible or not. It causes a difficulty to design RCAs with specific properties like computation-universality. In the case of one-dimensional CAs, there is a decision algorithm for reversibility ([Theorem 3](#)), and there are also several studies on enumerating all one-dimensional RCAs (e.g., Boykett 2004 and Mora et al. 2005). But, since the decision algorithm is not so simple, it is still difficult to design RCAs with specific properties by using the standard framework of CAs even in the one-dimensional case.

So far several methods have been proposed to make it feasible to design RCAs. They are CAs with block rules (Margolus 1984; Toffoli and Margolus 1990), partitioned CAs (Morita and Harao 1989), CAs with second-order rules (Margolus 1984; Toffoli and Margolus 1990; Toffoli et al. 2004), and others.

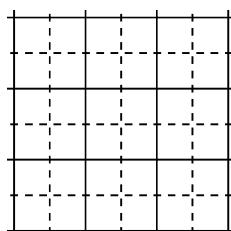
3.1 CAs with Block Rules

A local function of a standard CA is a mapping $f: Q^m \rightarrow Q$, and therefore it is impossible to make f itself be one-to-one except the trivial case where $|Q| = 1$ or $m = 1$. Instead, if a local function is of the form $f: Q^m \rightarrow Q^m$, then it can be one-to-one. Assume all the cells are grouped into blocks each of which consists of m cells. Such a local function maps a state of each block to a new state of the same block. If f is one-to-one, and is applied to all the blocks in a given configuration in parallel, then the induced global function will be also one-to-one. Of course, at the next time step, the grouping into blocks should be changed. Otherwise, interaction between blocks is impossible. Such a CA is often called a block CA.

Margolus (1984) first proposed this type of CA, by which he composed a specific model of a computation-universal two-dimensional two-state RCA. In his model, all the cells are grouped into blocks of size 2×2 as in [Fig. 1](#). Its local function is specified by “block rules,” and is shown in [Fig. 2](#). This CA evolves as follows: At time 0, the local function is applied to every solid line block, then at time 1 to every dotted line block, and so on, alternately. This kind of neighborhood is called Margolus neighborhood. One can see that the local function defined by the block rules in [Fig. 2](#) is one-to-one (such a one-to-one local

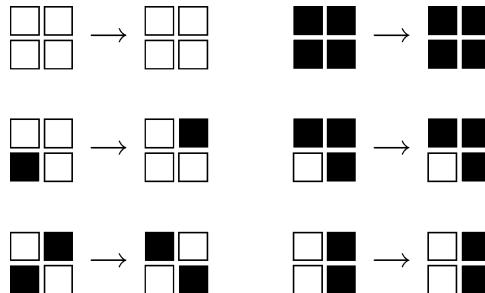
 **Fig. 1**

Grouping cells into blocks of size 2×2 by solid or dotted lines, which form the Margolus neighborhood.



■ Fig. 2

An example of block rules for the Margolus CA (Margolus 1984). (Since rotation-symmetry is assumed here, rules obtained by rotating both sides of a rule are also included.)



function is sometimes called a block permutation). Therefore the Margolus CA is reversible. Note that CAs with Margolus neighborhood are not conventional CAs, because each cell should know the relative position in a block and the parity of time besides its own state.

A k -dimensional block RCA can be also defined for any k in a similar manner. ◉ [Figure 3](#) gives a one-dimensional case of block size 2. It shows how a one-to-one local function f is applied to blocks at each time step.

3.2 Partitioned CAs

A partitioned cellular automaton (PCA) (Morita and Harao 1989) is also a framework where the notion of “local reversibility” can be defined, and hence has some similarity to a CA with block rules. But, it is noted that PCAs are in the framework of conventional CAs (i.e., a PCA is a special case of a CA). In addition, flexibility of neighborhood is rather high. So, it is convenient to use PCAs, for example, for showing results on computation-universality of RCAs. Therefore, in the following sections, this framework is mainly used to give specific examples of RCAs. A shortcoming of PCA is, in general, the number of states per cell becomes large.

Definition 3 A deterministic k -dimensional m -neighbor partitioned cellular automaton (PCA) is a system defined by

$$P = (\mathbb{Z}^k, (Q_1, \dots, Q_m), (n_1, \dots, n_m), f)$$

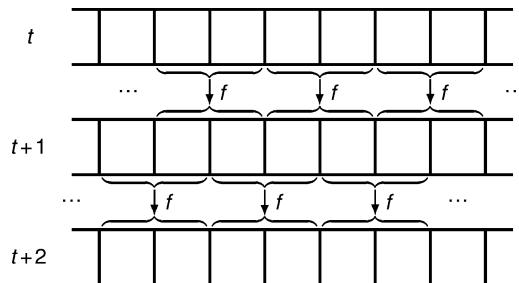
Here, Q_i ($i = 1, \dots, m$) is a nonempty finite set of states of the i th part of each cell (thus the state set of each cell is $Q = Q_1 \times \dots \times Q_m$), the m -tuple $(n_1, \dots, n_m) \in (\mathbb{Z}^k)^m$ is a neighborhood, and $f: Q \rightarrow Q$ is a local function.

Let $p_i: Q \rightarrow Q_i$ be the projection function such that $p_i(q_1, \dots, q_m) = q_i$ for all $(q_1, \dots, q_m) \in Q$. The global function $F: \text{Conf}(Q) \rightarrow \text{Conf}(Q)$ of P is defined as the one that satisfies the following formula.

$$\forall \alpha \in \text{Conf}(Q), x \in \mathbb{Z}^k : F(\alpha)(x) = f(p_1(\alpha(x + n_1)), \dots, p_m(\alpha(x + n_m)))$$

Fig. 3

A one-dimensional RCA with block rules defined by a one-to-one local function f . Here the block size is 2.



If some state in Q is specified as a quiescent state, a finite (and infinite) configuration can be defined in a similar manner as in CAs. A quiescent state $(\#, \dots, \#_m) \in Q$ is a one such that $f(\#, \dots, \#_m) = (\#, \dots, \#_m)$. (In general, the states $\#, \dots, \#_m$ may be different from each other. However, by renaming the states in each part appropriately, the states $\#, \dots, \#_m$ can be identified as representing the same state $\#$. In what follows, it is often assumed so, and the quiescent state is written by $(\#, \dots, \#)$.)

A one-dimensional three-neighbor (i.e., radius 1) PCA P_{1D} is therefore defined as follows.

$$P_{1D} = (\mathbb{Z}, (L, C, R), (1, 0, -1), f)$$

Each cell has three parts, that is, left, center, and right, and their state sets are L , C , and R . The next state of a cell is determined by the present states of the left part of the right-neighboring cell, the center part of this cell, and the right part of the left-neighboring cell (not depending on all three parts of the three cells). \circlearrowleft Figure 4 shows its cellular space, and how the local function f is applied.

Let $(l, c, r), (l', c', r') \in L \times C \times R$. If $f(l, c, r) = (l', c', r')$, then this equation is called a local rule (or simply a rule) of the PCA P_{1D} . It can be written in a pictorial form as shown in \circlearrowleft Fig. 5. Note that, in the pictorial representation, the arguments of the left-hand side of $f(l, c, r) = (l', c', r')$ appear in a reverse order.

Similarly, a two-dimensional PCA P_{2D} with Neumann-like neighborhood is defined as follows.

$$P_{2D} = (\mathbb{Z}^2, (C, U, R, D, L), ((0, 0), (0, -1), (-1, 0), (0, 1), (1, 0)), f)$$

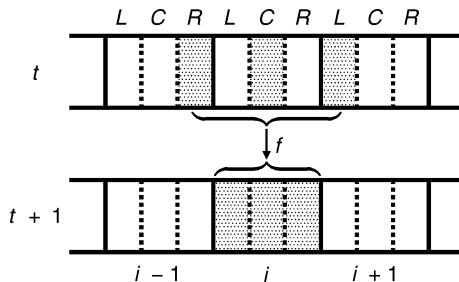
\circlearrowleft Figure 6 shows the cellular space of P_{2D} , and a pictorial representation of a rule $f(c, u, r, d, l) = (c', u', r', d', l')$.

It is easy to show the following proposition stating the equivalence of local and global injectivity (a proof for the one-dimensional case given in Morita and Harao (1989) can be extended to higher dimensions).

Proposition 1 Let $P = (\mathbb{Z}^k, (Q_1, \dots, Q_m), (n_1, \dots, n_m), f)$ be a k -dimensional PCA, and F be its global function. Then the local function f is one-to-one, iff the global function F is one-to-one.

Fig. 4

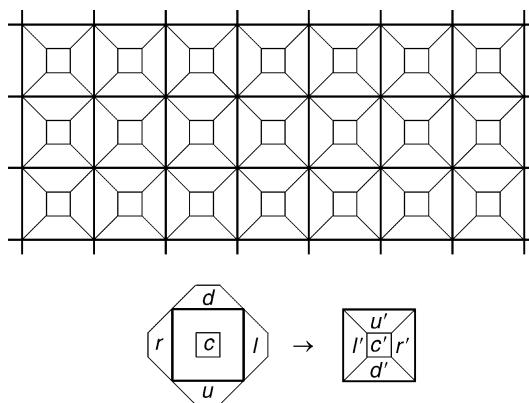
Cellular space of a one-dimensional 3-neighbor PCA P_{1D} , and its local function f .

**Fig. 5**

A pictorial representation of a local rule $f(l, c, r) = (l', c', r')$ of a one-dimensional 3-neighbor PCA P_{1D} .

**Fig. 6**

Cellular space of a two-dimensional 5-neighbor PCA P_{2D} and its local rule. Each cell is divided into five parts. The next state of each cell is determined by the present states of the center part of the cell, the downward part of the upper neighbor, the left part of the right neighbor, the upper part of the downward neighbor, and the right part of the left neighbor.



It is also easy to see that the class of PCAs is regarded as a subclass of CAs (Morita and Harao 1989).

Proposition 2 *For any k -dimensional m -neighbor PCA P , a k -dimensional m -neighbor CA A whose global function is identical with that of P can be obtained.*

From the above, if one wants to construct an RCA it is sufficient to give a PCA whose local function f is one-to-one. This makes a design of an RCA feasible.

3.3 Second-Order CAs

A second-order CA is one where the state of a cell at time $t + 1$ is determined not only by the states of neighbor cells at time t but also by the state of the cell at $t - 1$ (Margolus 1984; Toffoli and Margolus 1990; Toffoli et al. 2004). Margolus first proposed a method of using second-order CAs to design a reversible one, and gave a specific model of a computation-universal two-dimensional three-state reversible second-order CA (Margolus 1984).

It is assumed that a binary operator \ominus is defined on the state set Q , which satisfies $a \ominus b = c$ iff $a \ominus c = b$ for all $a, b, c \in Q$. A typical example is $Q = \{0, 1, \dots, r - 1\}$ and the operation \ominus is the mod- r subtraction. Let $N = (n_1, \dots, n_m) \in (\mathbb{Z}^k)^m$ be a neighbor in the k -dimensional cellular space, and let $f: Q^m \rightarrow Q$ be an arbitrary mapping. Consider the local transition of the state of each cell defined by the following equation, where $\alpha^t \in \text{Conf}(Q)$ denotes a configuration at time t , and $x \in \mathbb{Z}^k$.

$$\alpha^{t+1}(x) = f(\alpha^t(x + n_1), \dots, \alpha^t(x + n_m)) \ominus \alpha^{t-1}(x)$$

It naturally induces a global function that determines the next configuration from the present and the previous configurations. From the above equation, the following one is easily derived, which elucidates that the state of each cell at time $t - 1$ is determined by the states of neighbor cells at t and the state of the cell at $t + 1$ by the *same* formula as above.

$$\alpha^{t-1}(x) = f(\alpha^t(x + n_1), \dots, \alpha^t(x + n_m)) \ominus \alpha^{t+1}(x)$$

Hence, a second-order CA defined in such a method is reversible in this sense for *any* mapping $f: Q^m \rightarrow Q$. Though second-order CAs are not in the standard framework of CAs, they are also useful for giving RCAs. The relation between second-order CAs and the lattice gas model is discussed in Toffoli et al. (2004).

4 Computing Ability of Reversible Cellular Automata

A CA is called *computation-universal* (or Turing-universal) if it can simulate any Turing machine. From the beginning of the history of CA studies, it has been one of the main research topics. In the 1950s, von Neumann (1966) designed a 29-state two-dimensional CA, and showed that any Turing machine can be embedded in its cellular space, and it can reproduce itself. In this section, the problem of how reversible CAs can have universality is investigated. In what follows, there is a description on how various universal systems can be simulated by RCAs.

Though a precise definition of “simulation” is not given here, it is explained as follows. When we consider the case that a system A simulates another system B , we should first give an encoding method of computational configurations of B into those of A . Assume A starts from a configuration α that contains an encoding of an initial configuration β of B . We say A simulates B , if for every configuration β' that appears in the computing process of B starting from β , a configuration α' containing an encoding of β' appears at some time step in the computing process of A .

See also the chapter [• Universality in Cellular Automata](#) on computing ability and universality of CAs.

4.1 Simulating Irreversible CAs by RCAs

Toffoli (1977) first showed the following theorem stating that every irreversible CA can be simulated by an RCA by increasing the dimension by one.

Theorem 7 (Toffoli 1977) *For any k -dimensional (irreversible) CA A , a $k + 1$ -dimensional RCA A' that simulates A in real time can be constructed.*

The intuitive idea for proving it is as follows. A' simulates A by using a k -dimensional subspace, say $\{(x_1, \dots, x_k, 0) | x_i \in \mathbb{Z}, i = 1, \dots, k\}$, in real time. Since this subspace always keeps the present configuration of A , A' can compute the new configuration of A using only this subspace. At the same time, A' sends the information of the old configuration to the subspace $\{(x_1, \dots, x_k, 1) | x_i \in \mathbb{Z}, i = 1, \dots, k\}$. The previous information stored in $\{(x_1, \dots, x_k, 1) | x_i \in \mathbb{Z}, i = 1, \dots, k\}$ is also sent to the subspace $\{(x_1, \dots, x_k, 2) | x_i \in \mathbb{Z}, i = 1, \dots, k\}$, and so on. In other words, the additional dimension is used to record *all* the past history of the evolution of A from $t = 0$ up to now, and thus A' can be an RCA. (If the framework of PCA is used, then to construct an RCA A' is rather simple (Morita 2001a).)

From this result, it can be seen that a computation-universal two-dimensional RCA exists, since there is a (irreversible) one-dimensional CA that can simulate a universal Turing machine.

As for one-dimensional CA with finite configurations, reversible simulation is possible without increasing the dimension. Hence, a computation-universal one-dimensional RCA also exists.

Theorem 8 (Morita 1995) *For any one-dimensional (irreversible) CA A with finite configurations, a one-dimensional reversible CA A' that simulates A (but not in real time) can be constructed.*

4.2 Simulating Reversible Turing Machines by One-Dimensional RCAs

It is also possible to design an RCA that can simulate a reversible Turing machine directly. Bennett (1973) proved the universality of a reversible Turing machine by showing a conversion method of a given irreversible Turing machine to an equivalent reversible one, which never

leaves garbage symbols except an input and an output on its tape at the end of computation. Bennett (1973) used a quadruple formalism to define a reversible Turing machine, because it is useful to define an *inverse* Turing machine for a given reversible Turing machine. But here, a quintuple formalism is employed, because it is commonly used in the theory of irreversible Turing machines, and these two formalisms are convertible to each other (Morita and Yamaguchi 2007).

Definition 4 A *1-tape Turing machine in the quintuple form* (TM) is defined by

$$T = (Q, S, q_0, q_f, s_0, \delta)$$

where Q is a nonempty finite set of states, S is a nonempty finite set of symbols, q_0 is an initial state ($q_0 \in Q$), q_f is a final (halting) state ($q_f \in Q$), s_0 is a special blank symbol ($s_0 \in S$). δ is a move relation, which is a subset of $(Q \times S \times S \times \{-, 0, +\} \times Q)$. Each element of δ is a quintuple of the form $[p, s, s', d, q]$. It means if T reads the symbol s in the state p , then write s' , shift the head to the direction d ($-$, 0 , and $+$ means left-, zero-, right-shift, respectively), and go to the state q .

T is called a *deterministic TM* iff the following condition holds for any pair of distinct quintuples $[p_1, s_1, s'_1, d_1, q_1]$ and $[p_2, s_2, s'_2, d_2, q_2]$ in δ .

$$\text{If } p_1 = p_2, \text{ then } s_1 \neq s_2$$

T is called a *reversible TM* iff the following condition holds for any pair of distinct quintuples $[p_1, s_1, s'_1, d_1, q_1]$ and $[p_2, s_2, s'_2, d_2, q_2]$ in δ .

$$\text{If } q_1 = q_2, \text{ then } s'_1 \neq s'_2 \wedge d_1 = d_2$$

The next theorem shows computation-universality of a reversible three-tape Turing machine.

Theorem 9 (Bennett 1973) *For any deterministic (irreversible) one-tape TM, there is a deterministic reversible three-tape TM which simulates the former.*

It is also shown in Morita et al. (1989) that for any irreversible one-tape TM, there is a reversible one-tape two-symbol TM, which simulates the former. In fact, to prove computation-universality of a one-dimensional reversible PCA, it is convenient to simulate a reversible one-tape TM.

Theorem 10 (Morita and Harao 1989) *For any deterministic reversible one-tape TM T , there is a one-dimensional reversible PCA P that simulates the former.*

It is now explained how the reversible PCA P is constructed (here, the method in Morita and Harao (1989) is modified to adopt the quintuple formalism, and to reduce the number of states of P). Let $T = (Q, S, q_0, q_f, s_0, \delta)$. It is assumed that q_0 does not appear as the fifth element in any quintuple in δ (such a reversible TM can always be constructed from an irreversible one (Morita et al. 1989)). It can also be assumed that for any quintuple $[p, s, t, d, q]$ in δ ,

$d \in \{-, +\}$ iff q is a non-halting state, and $d \in \{0\}$ iff q is a halting state. A reversible PCA $P = (\mathbb{Z}, (L, C, R), (1, 0, -1), f)$ that simulates T is as follows. The state sets L , C , and R are

$$L = R = Q \cup \{\#\}, C = S$$

Let Q_+ , Q_- , and Q_H be as follows.

$$Q_+ = \{q \mid \exists p \in Q \ \exists s, t \in S ([p, s, t, +, q] \in \delta)\}$$

$$Q_- = \{q \mid \exists p \in Q \ \exists s, t \in S ([p, s, t, -, q] \in \delta)\}$$

$$Q_H = \{p \mid \neg(\exists q \in Q \ \exists s, t \in S \ \exists d \in \{-, 0, +\} ([p, s, t, d, q] \in \delta))\}$$

Since T is a reversible TM, Q_+ , Q_- , and Q_H are mutually disjoint. The local function f is as below.

1. For every $s, t \in S$, and $q \in Q - (Q_H \cup \{q_0\})$

$$f(\#, s, \#) = (\#, s, \#)$$

$$f(\#, s, q_0) = (\#, s, q_0)$$

$$f(q_0, s, \#) = (q_0, s, \#)$$

$$f(q_0, s, q_0) = (\#, t, q) \text{ if } [q_0, s, t, +, q] \in \delta$$

$$f(q_0, s, q_0) = (q, t, \#) \text{ if } [q_0, s, t, -, q] \in \delta$$

2. For every $p, q \in Q - (Q_H \cup \{q_0\})$, and $s, t \in S$

$$f(\#, s, p) = (\#, t, q) \text{ if } p \in Q_+ \text{ and } [p, s, t, +, q] \in \delta$$

$$f(p, s, \#) = (\#, t, q) \text{ if } p \in Q_- \text{ and } [p, s, t, +, q] \in \delta$$

$$f(\#, s, p) = (q, t, \#) \text{ if } p \in Q_+ \text{ and } [p, s, t, -, q] \in \delta$$

$$f(p, s, \#) = (q, t, \#) \text{ if } p \in Q_- \text{ and } [p, s, t, -, q] \in \delta$$

3. For every $p \in Q - (Q_H \cup \{q_0\})$, $q \in Q_H$, and $s, t \in S$

$$f(\#, s, p) = (q, t, q) \text{ if } p \in Q_+ \text{ and } [p, s, t, 0, q] \in \delta$$

$$f(p, s, \#) = (q, t, q) \text{ if } p \in Q_- \text{ and } [p, s, t, 0, q] \in \delta$$

4. For every $q \in Q_H$ and $s \in S$

$$f(\#, s, q) = (\#, s, q)$$

$$f(q, s, \#) = (q, s, \#)$$

One can verify that the right-hand side of each rule differs from that of any other rule, because T is deterministic and reversible. If the initial computational configuration of T is

$$\cdots s_0 t_1 \cdots q_0 t_i \cdots t_n s_0 \cdots$$

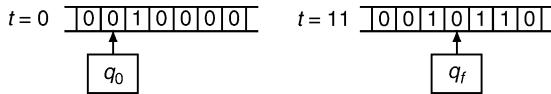
then set P to the following configuration.

$$\dots, (\#, s_0, \#), (\#, t_1, \#), \dots, (\#, t_{i-1}, q_0), (\#, t_i, \#), (q_0, t_{i+1}, \#), \dots, (\#, t_n, \#), (\#, s_0, \#), \dots$$

The simulation process starts when left- and right-moving signals q_0 's meet at the cell containing t_i . It is easily seen that, from this configuration, P can correctly simulate T by the rules in (2) step by step. If T becomes a halting state q ($\in Q_H$), then the two signals q 's are created, and travel leftward and rightward indefinitely. Note that P itself cannot halt, because P is reversible, but the final result is kept unchanged.

Fig. 7

The initial and the final computational configuration of T_{2n} for a given unary input 1.



Example 1 Consider a reversible TM $T_{2n} = (Q, \{0, 1\}, q_0, q_f, 0, \delta)$, where $Q = \{q_0, \dots, q_7, q_f\}$, and δ is as below.

$$\begin{aligned}\delta = \{ &[q_0, 0, 0, +, q_1], \\ &[q_1, 0, 0, 0, q_f], [q_1, 1, 0, +, q_2], \\ &[q_2, 0, 0, +, q_3], [q_2, 1, 1, +, q_2], \\ &[q_3, 0, 1, +, q_4], [q_3, 1, 1, +, q_3], \\ &[q_4, 0, 1, +, q_5], \\ &[q_5, 0, 0, -, q_6], \\ &[q_6, 0, 0, -, q_7], [q_6, 1, 1, -, q_6], \\ &[q_7, 0, 1, +, q_1], [q_7, 1, 1, -, q_7]\}. \end{aligned}$$

For a given unary number n on the tape, T_{2n} computes the function $2n$ and writes it on the tape as in [Fig. 7](#). A simulation process of T_{2n} by a reversible PCA P_{2n} , which is constructed by the method described above, is in [Fig. 8](#).

Each cell of the reversible PCA P constructed in [Theorem 10](#) has $(m+1)^2 n$ states, if T has m states and n symbols. Since there is a 15-state 6-symbol universal reversible TM (Morita 2008a), a 1536-state universal RCA is obtained by this method.

4.3 Simulating Cyclic Tag Systems by One-Dimensional RCAs

A cyclic tag system (CTS) is a kind of string rewriting system proposed by Cook (2004) to show universality of the elementary CA of rule 110. This system is also useful for constructing computation-universal one-dimensional RCAs with a small number of states.

Definition 5 A CTS is defined by $C = (k, \{Y, N\}, (p_0, \dots, p_{k-1}))$, where k ($k = 1, 2, \dots$) is the length of a cycle (i.e., period), $\{Y, N\}$ is the alphabet used in this system, and $(p_0, \dots, p_{k-1}) \in (\{Y, N\}^*)^k$ is a k -tuple of production rules. A pair (v, m) is called an *instantaneous description* (ID) of C , where $v \in \{Y, N\}^*$ and $m \in \{0, \dots, k-1\}$. m is called a *phase* of the ID. A transition relation \Rightarrow on the set of IDs is defined as follows. For any $(v, m), (v', m') \in \{Y, N\}^* \times \{0, \dots, k-1\}$

$$\begin{aligned}(Yv, m) \Rightarrow (v', m') &\text{ iff } [m' = m + 1 \bmod k] \wedge [v' = vp_m] \\ (Nv, m) \Rightarrow (v', m') &\text{ iff } [m' = m + 1 \bmod k] \wedge [v' = v]\end{aligned}$$

A sequence of IDs $(v_0, m_0), (v_1, m_1), \dots$ is called a *computation starting from* $v \in \{Y, N\}^*$ iff $(v_0, m_0) = (v, 0)$ and $(v_i, m_i) \Rightarrow (v_{i+1}, m_{i+1})$ ($i = 0, 1, \dots$). We also write a computation by $(v_0, m_0) \Rightarrow (v_1, m_1) \Rightarrow \dots$.

Fig. 8

Simulating T_{2n} by a one-dimensional reversible PCA P_{2n} . The state # is indicated by a blank.

$t = 0$	0	q_0	0	q_0	1		0	0	0	0	0
$t = 1$	0		0	q_1	1		0	0	0	0	0
$t = 2$	0		0		q_2	0	0	0	0	0	0
$t = 3$	0		0			0	q_3	0	0	0	0
$t = 4$	0		0			0		1	q_4	0	0
$t = 5$	0		0			0		1		1	q_5
$t = 6$	0		0			0		1		1	q_6
$t = 7$	0		0			0		1		q_6	1
$t = 8$	0		0			0		q_6	1		1
$t = 9$	0		0		q_7	0		1		1	0
$t = 10$	0		0		1	q_1	0		1		1
$t = 11$	0		0		1	q_f	0	q_f	1		0
$t = 12$	0		0	q_f	1		0		1	q_f	1
$t = 13$	0	q_f	0		1		0		1	q_f	0
$t = 14$	q_f	0	0		1		0		1		0

In a CTS, production rules are applied cyclically to rewrite a given string. If the first symbol of the host string is Y , then it is removed and a specified string at that phase is attached to the end of the host string. If it is N , then it is simply removed and no string is attached.

Example 2 Consider the following CTS.

$$C_0 = (3, \{Y, N\}, (Y, NN, YN))$$

If NYY is given as an initial string, then

$$(NYY, 0) \Rightarrow (YY, 1) \Rightarrow (YNN, 2) \Rightarrow (NNYN, 0) \Rightarrow (NYN, 1) \Rightarrow (YN, 2)$$

is an initial segment of a computation starting from NYY .

A 2-tag system is a special class of classical tag systems, and Minsky (1967) proved that for any TM, there is a 2-tag system that simulates any computing process of the TM. Hence a 2-tag system is computation-universal. Cook (2004) proved the following theorem, and thus it can be seen that a CTS is also computation-universal.

Theorem 11 (Cook 2004) *For any 2-tag system, there is a CTS that simulates the former.*

In Morita (2007), two models of one-dimensional reversible PCAs that can simulate any CTS are given. The first one is a 36-state model that works on infinite configurations, and the second one is a 98-state model that works on finite configurations. As shown in the following theorem, the former result was improved in Morita (2008b).

Theorem 12 (Morita 2008b) *There is a 24-state one-dimensional two-neighbor reversible PCA P_{24} that can simulate any CTS on infinite (but ultimately periodic) configurations.*

The reversible PCA P_{24} in [Theorem 12](#) is given below.

$$P_{24} = (\mathbb{Z}, (\{Y, N, +, -\}, \{y, n, +, -, *, /\}), (0, -1), f)$$

where f is defined as follows.

$$\begin{aligned} f(\mathbf{c}, \mathbf{r}) &= (\mathbf{c}, \mathbf{r}) \quad \text{for } \mathbf{c} \in \{Y, N\}, \text{ and } \mathbf{r} \in \{y, n, +, -, /\} \\ f(Y, *) &= (+, /) \\ f(N, *) &= (-, /) \\ f(-, \mathbf{r}) &= (-, \mathbf{r}) \quad \text{for } \mathbf{r} \in \{y, n, *\} \\ f(\mathbf{c}, \mathbf{r}) &= (\mathbf{r}, \mathbf{c}) \quad \text{for } \mathbf{c}, \mathbf{r} \in \{+, -\} \\ f(+, y) &= (Y, *) \\ f(+, n) &= (N, *) \\ f(+, /) &= (+, y) \\ f(-, /) &= (+, n) \\ f(+, *) &= (+, *) \end{aligned}$$

Note that a cell of P_{24} has only the center and the right parts: $C = \{Y, N, +, -\}$, and $R = \{y, n, +, -, *, /\}$. It is easily seen that $f: C \times R \rightarrow C \times R$ is one-to-one.

In the following, there is an explanation as to how P_{24} can simulate a CTS by using the previous example C_0 with an initial string NYY . The initial configuration of P_{24} is set as shown in the first row of [Fig. 9](#). The string NYY is given in the center parts of three consecutive cells. The right-part states of these three cells are set to $-$. The states of the cells right to the three cells are set to $(-, -)$, $(Y, -)$, $(Y, -)$, \dots . The production rules (Y, NN, YN) is given by a sequence of the right-part states $y, n, *,$ and $-$ in a reverse order, where the sequence $- *$ is used as a delimiter indicating the beginning of a rule. Thus, one cycle of the rules (Y, NN, YN) is represented by the sequence $ny - * nn - * y - *$. Since these rules are applied cyclically, infinite copies of the sequence $ny - * nn - * y - *$ must be given. These right-part states $y, n, *,$ and $-$ act as right-moving signals until they reach the first symbol of a rewritten string.

P_{24} can simulate a rewriting process of C_0 as shown below. If the signal $*$ meets the state Y (N , respectively), which is the head symbol of a rewritten string, then the signal changes Y (N) to the state $+ (-)$, and the signal itself becomes $/$ that is sent rightward as a used (garbage) signal. At the next time step, the center-part state $+ (-)$ meets the signal $-$, and the former becomes a right-moving signal $+ (-)$, and the latter (i.e., $-$) is fixed as a center-part state at this position. The right-moving signal $+ (-)$ travels through the rewritten string consisting of Y 's and N 's, and when it meets the center-part state $+$ or $-$, then it is fixed as a new center-part state, which indicates the last head symbol is Y (N). Note that the old center-part state $+ (-)$ is sent rightward as a used signal.

Fig. 9

Simulating the computing process $(NYY, 0) \rightarrow^* (YN, 2)$ of the CTS C_0 in [Example 2](#) by the reversible PCA P_{24} (Morita 2008b).

-n -y	- - * -n -n	- - * -y	- - * N	Y Y	- - Y	Y Y	Y Y	Y Y
- * -n -y	- - * -n -n	- - * -y	- - / Y	Y Y	- - Y	Y Y	Y Y	Y Y
- - * -n -y	- - * -n -n	- - * -y	- Y / Y	Y Y	- - Y	Y Y	Y Y	Y Y
-y	- - * -n -y	- - * -n -n	- * -y	Y Y /	- - Y	Y Y	Y Y	Y Y
- * -y	- - * -n -y	- - * -n -n	- * -Y y	Y + n Y	- Y	Y Y	Y Y	Y Y
- - * -y	- - * -n -y	- - * -n -n	- + / Y y	- + Y n Y	- Y	Y Y	Y Y	Y Y
-n - - * -y	- - * -n -y	- - * -n -n	- + Y /	- y Y + Y n	Y Y	Y Y	Y Y	Y Y
-n -n - - * -y	- - * -n -y	- - * -n -n	- + n Y + Y y	Y + Y n	Y Y	Y Y	Y Y	Y Y
- * -n -n - - * -y	- - * -n -y	- - * -n -n	- + Y n Y + Y n	Y Y + Y n	Y Y	Y Y	Y Y	Y Y
- - * -n -n - - * -y	- - * -n -y	- - * -n -n	- + Y n N * Y + Y n	Y Y + Y n	Y Y	Y Y	Y Y	Y Y
-y	- - * -n -n - - * -y	- - * -n -y	- + / N n + / Y	Y n Y Y	Y Y	Y Y	Y Y	Y Y
-n -y	- - * -n -n - - * -y	- - * -n -y	- + N / N * Y /	Y + Y n Y Y	Y Y	Y Y	Y Y	Y Y
- * -n -y	- - * -n -n - - * -y	- - * -n -y	- + N / N / + / Y	Y / Y + Y n	Y Y	Y Y	Y Y	Y Y
- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - * -n N y N + + y	Y / Y / Y	Y Y	Y Y	Y Y	Y Y
-y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - * N n N y + + Y y	Y Y / Y / Y	Y Y	Y Y	Y Y
- * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - / N n Y * Y + Y y	Y Y / Y Y	Y Y	Y Y	Y Y
- - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - N / Y n + / Y + Y y	Y Y / Y Y	Y Y	Y Y	Y Y
-n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - N / Y / N * Y / Y +	Y Y / Y Y	Y Y	Y Y	Y Y
-n -n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - N y Y / N / + / Y	Y Y / Y Y	Y Y	Y Y	Y Y
- * -n -n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -y	- - / Y y N - + y	Y Y / Y Y	Y Y	Y Y	Y Y
-y	- - * -n -n - - * -n -n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -n N y N / - y	Y Y / Y + Y	Y Y	Y Y	Y Y
-n -y	- - * -n -n - - * -n -n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -n N n N / + Y n	Y Y / Y Y	Y Y	Y Y	Y Y
- * -n -y	- - * -n -n - - * -n -n - - * -y	- - * -n -y	- - * -n -n - - * -n -n	- - * -n N n N / + Y n	Y Y / Y Y	Y Y	Y Y	Y Y

Signals y and n go rightward through the rewritten string consisting Y 's and N 's until it meets $+$ or $-$. If y (n , respectively) meets $+$, then the signal becomes $Y(N)$ and is fixed at this position (since the last head symbol is Y), and $+$ is shifted to the right by one cell. If y or n meets $-$, then the signal simply continues to travel rightward without being fixed (since the last head symbol is N). Note that all the used and unused information are sent rightward as garbage signals, because they cannot be deleted by the constraint of reversibility. [Figure 9](#) shows how a computation of C_0 is performed in P_{24} . In this way, any CTS can be simulated.

4.4 Simulating Reversible Logic Elements by Two-Dimensional RCAs

The universality of a CA can also be proved by showing any logic circuit is simulated in its cellular space. In the case of two-dimensional CAs, such a method is often used. A set of logic elements is called *logically universal*, if any logic function can be realized by using only these elements. For example, {AND, NOT} is a well-known logically universal set. Furthermore, if delay elements can be used (or memory elements) besides these elements, then any sequential machine (i.e., finite automaton with outputs) can be composed from them. Since a finite-state control and tape cells of a Turing machine are regarded as sequential machines, any Turing machine can be constructed by using these elements.

In the following, several computation-universal two-dimensional RCAs are shown in which universal reversible logic elements are embedded. In particular, the Fredkin gate, and the rotary element (RE) are taken into consideration.

4.4.1 Simulating the Fredkin Gate

A typical example of a universal reversible logic gate is the Fredkin gate (Fredkin and Toffoli 1982). It is a reversible (i.e., its logical function is one-to-one) and bit-conserving (i.e., the number of 1's is conserved between inputs and outputs) logic gate shown in Fig. 10. It is known that any combinational logic circuit is composed only of Fredkin gates. Fig. 11 shows that AND, NOT, and Fan-out elements can be implemented only with Fredkin gates (Fredkin and Toffoli 1982) by allowing constant inputs and garbage outputs (e.g., in the implementation of AND, constant 0 should be supplied to the third input line, and the first and the third output lines produce garbage signals x and $\bar{x}y$). Hence, {Fredkin gate} is logically universal, and any sequential machine can be constructed with Fredkin gate and delay elements.

Though the Fredkin gate is a simple element, it can be decomposed into a much simpler gate called a switch gate (Fig. 12) and its inverse gate (Fredkin and Toffoli 1982). A switch gate is also a reversible and bit-conserving logic gate. In addition, it is known to be realized by the billiard ball model (BBM) of computation (Fredkin and Toffoli 1982). The BBM is a kind

Fig. 10
A Fredkin gate.

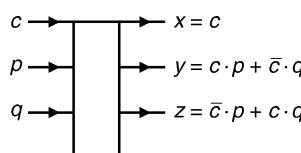


Fig. 11
AND, NOT, and Fan-out made of Fredkin gates (Fredkin and Toffoli 1982).

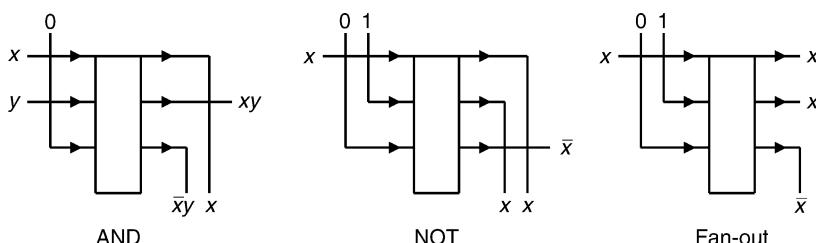


Fig. 12
A switch gate (left), and a Fredkin gate implemented by two switch gates and two inverse switch gates (right) (Fredkin and Toffoli 1982).

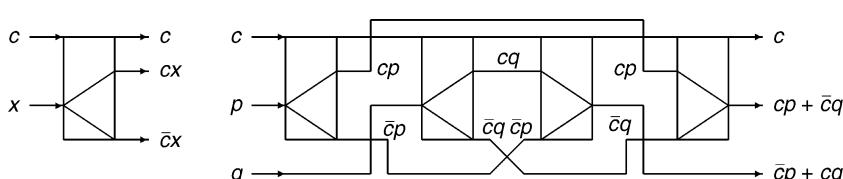
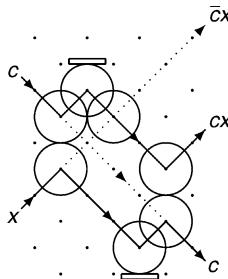


Fig. 13

A switch gate realized in the Billiard Ball Model (Fredkin and Toffoli 1982).



of physical model of computation where a signal “1” is represented by an ideal ball, and logical operations and routing can be performed by their elastic collisions and reflections by reflectors. [Figure 13](#) shows a BBM realization of a switch gate.

Now, consider the two-dimensional 2-state RCA with Margolus neighborhood, which has block rules shown in [Fig. 2](#) (Margolus 1984). Margolus showed that the BBM can be realized in this cellular space. [Figure 14](#) shows a reflection of a ball by a mirror in the Margolus CA. Hence, the following theorem is arrived at.

Theorem 13 (Margolus 1984) *There is a computation-universal two-dimensional 2-state RCA with Margolus neighborhood.*

Margolus (1984) also constructed a two-dimensional 3-state second-order RCA in which BBM can be realized.

Theorem 14 (Margolus 1984) *There is a computation-universal two-dimensional 3-state second-order RCA.*

There are also other RCAs that can simulate the Fredkin gate in their cellular spaces. Here, an 8-state reversible PCA model T_1 on a triangular grid (Imai and Morita 2000) is considered. Its local function is extremely simple as shown in [Fig. 15](#). In this PCA, each of the switch gate and the inverse switch gate is simply simulated by using only one cell (hence, it does not simulate the BBM). On the other hand, signal routing, crossing, and delay are very complex to realize, because a kind of “wall” is necessary to make a signal go straight ahead. [Figure 16](#) shows the configuration that simulates the Fredkin gate. Thus, we have the following theorem.

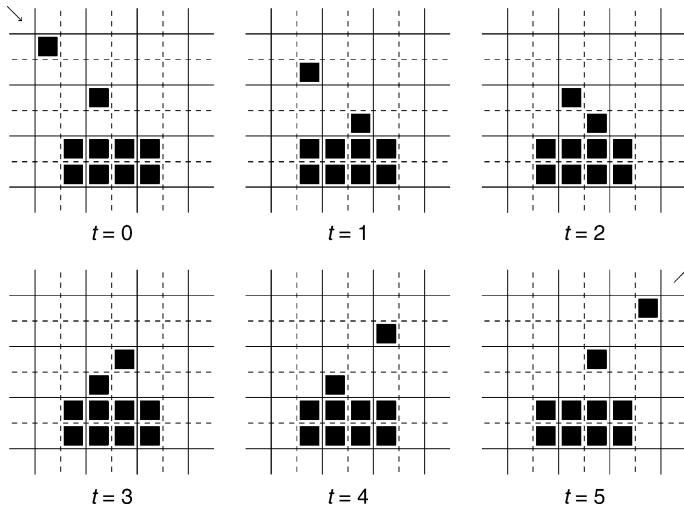
Theorem 15 (Imai and Morita 2000) *There is a computation-universal two-dimensional 8-state reversible triangular PCA.*

4.4.2 Simulating a Reversible Logic Element with Memory

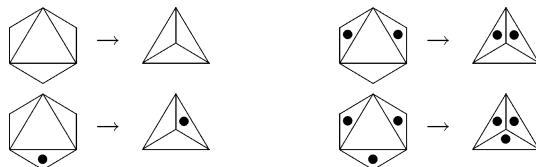
Besides reversible logic gates, there are also universal reversible logic elements with memory, with which reversible TMs are constructed rather concisely. A RE (Morita 2001b) is one such element. Conceptually, it has four input lines $\{n, e, s, w\}$ and four output lines $\{n', e', s', w'\}$,

Fig. 14

Reflection of a ball by a reflector in the Margolus RCA (Margolus 1984).

**Fig. 15**

The local function of the 2^3 -state rotation-symmetric reversible triangular PCA T_1 .



and two states called H-state and V-state as in [Fig. 17](#) (hence it has a 1-bit memory). The values of inputs and outputs are either 0 or 1, but the input (and the output) are restricted as follows: at most one “1” appears as an input (output) at a time. The operation of an RE is undefined for the cases where signal 1’s are given to two or more input lines. Signals 1 and 0 are interpreted as existence and nonexistence of a particle. An RE has a “rotating bar” to control the moving direction of a particle. When no particle exists, nothing happens. If a particle comes from a direction parallel to the rotating bar, then it goes out from the output line of the opposite side without affecting the direction of the bar ([Fig. 18a](#)). If a particle comes from a direction orthogonal to the bar, then it makes a right turn, and rotates the bar by 90° ([Fig. 18b](#)). It is clear its operation is reversible.

It is known that a Fredkin gate can be simulated by a circuit composed only of REs (Morita 2001b). Hence, $\{\text{RE}\}$ is logically universal. It is also shown that any reversible TM can be constructed by using only REs (Morita 2001b).

Consider the following two-dimensional 4-neighbor reversible PCA P_3 (Morita et al. 2002).

$$P_3 = (\mathbb{Z}^2, (\{0, 1, 2\}^4), ((0, -1), (-1, 0), (0, 1), (1, 0)), f)$$

Fig. 16

A Fredkin gate realized in the 2^3 -state reversible triangular PCA T_1 (Imai and Morita 2000).

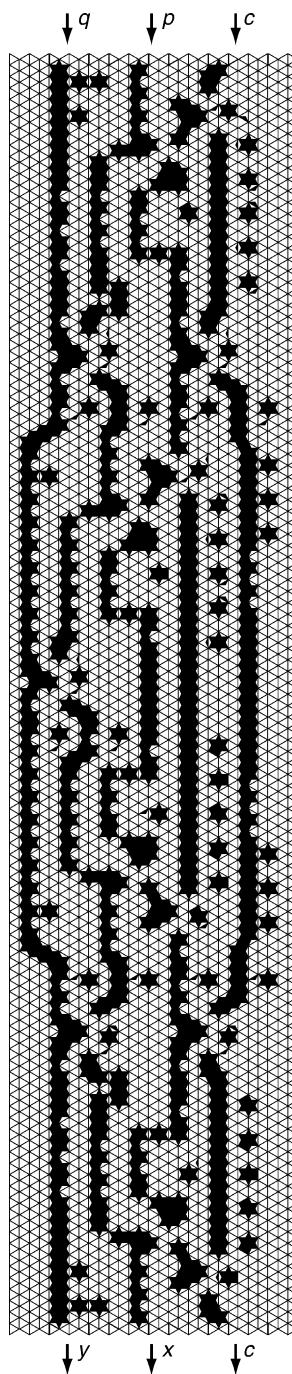
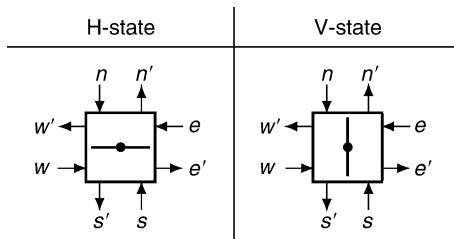
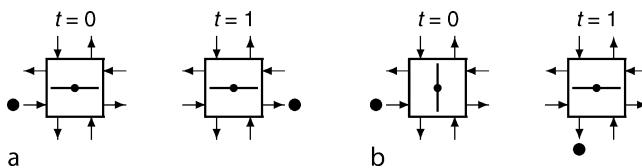


Fig. 17

Two states of a rotary element (RE).

**Fig. 18**

Operations of an RE: (a) the parallel case, and (b) the orthogonal case.



where f is given in [Fig. 19](#). In the cellular space of P_3 , an RE is realized as shown in [Fig. 20](#). It is also possible to design subconfigurations for signal routing. By using them, any circuit composed of REs can be simulated in the P_3 space, and thus it is computation-universal.

Furthermore, in P_3 , any reversible counter machine, which is also known to be computation-universal (Morita 1996), can be embedded as stated in the following theorem. Here, we only show an example of a configuration realizing a counter machine in [Fig. 21](#) (its detail is in Morita et al. (2002)).

Theorem 16 (Morita et al. 2002) *There is a computation-universal two-dimensional 81-state reversible PCA in which any reversible counter machine is simulated in finite configurations.*

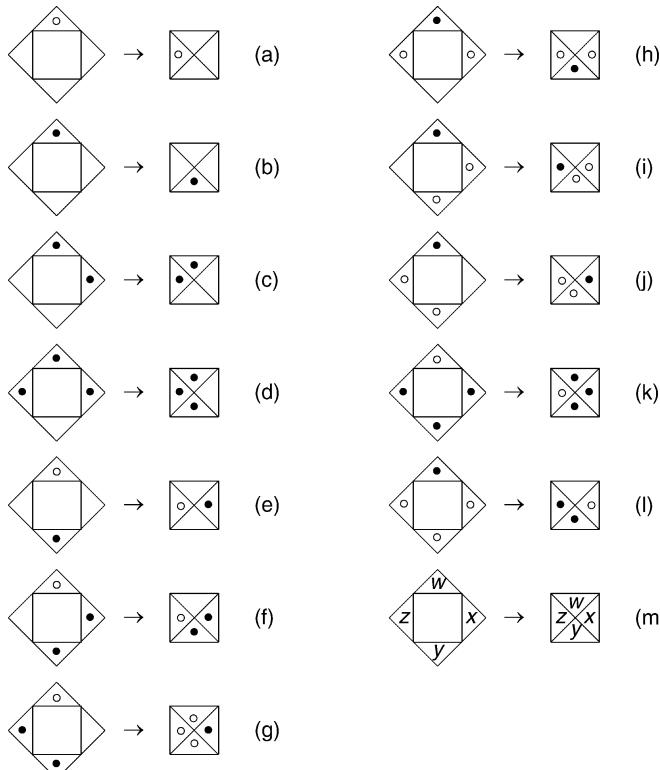
4.5 Intrinsically Universal RCAs

There is another notion of universality for CAs called intrinsic universality. A CA is *intrinsically universal* if it can simulate any CA. Von Neumann's 29-state CA (von Neumann 1966) is an example of an intrinsically universal irreversible CA, because any logic circuit composed of AND, OR, NOT, and memory elements is simulated in its cellular space, and every CA can be implemented by such a logic circuit. Studies on intrinsically universal one-dimensional CAs are found in Ollinger (2002) and Ollinger and Richard (2006).

In the case of RCAs, intrinsic universality can be defined as follows: An RCA is *intrinsically universal* if it can simulate any RCA. One such two-dimensional RCA is given in Durand-Lose (1995). Related to this topic, there is a problem whether every RCA is structurally reversible, that is, whether it can be expressed by locally reversible transformations or not (Toffoli and

Fig. 19

The local function of the 3^4 -state rotation-symmetric reversible PCA P_3 (Morita et al. 2002). The states 0, 1, and 2 are represented by blank, white dot, and black dot, respectively. The rule scheme (m) represents 33 rules not specified by (a)–(l), where $w, x, y, z \in \{\text{blank}, \circ, \cdot\}$.

**Fig. 20**

Operations of an RE in the reversible PCA P_3 : (a) the parallel case, and (b) the orthogonal case.

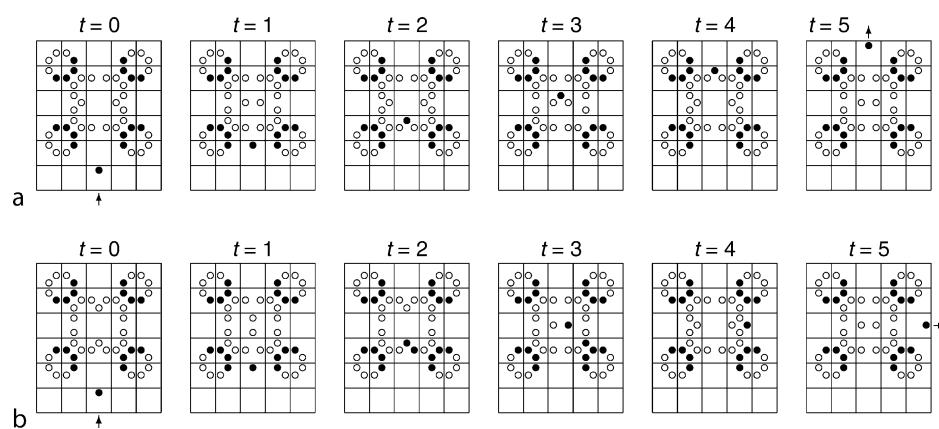
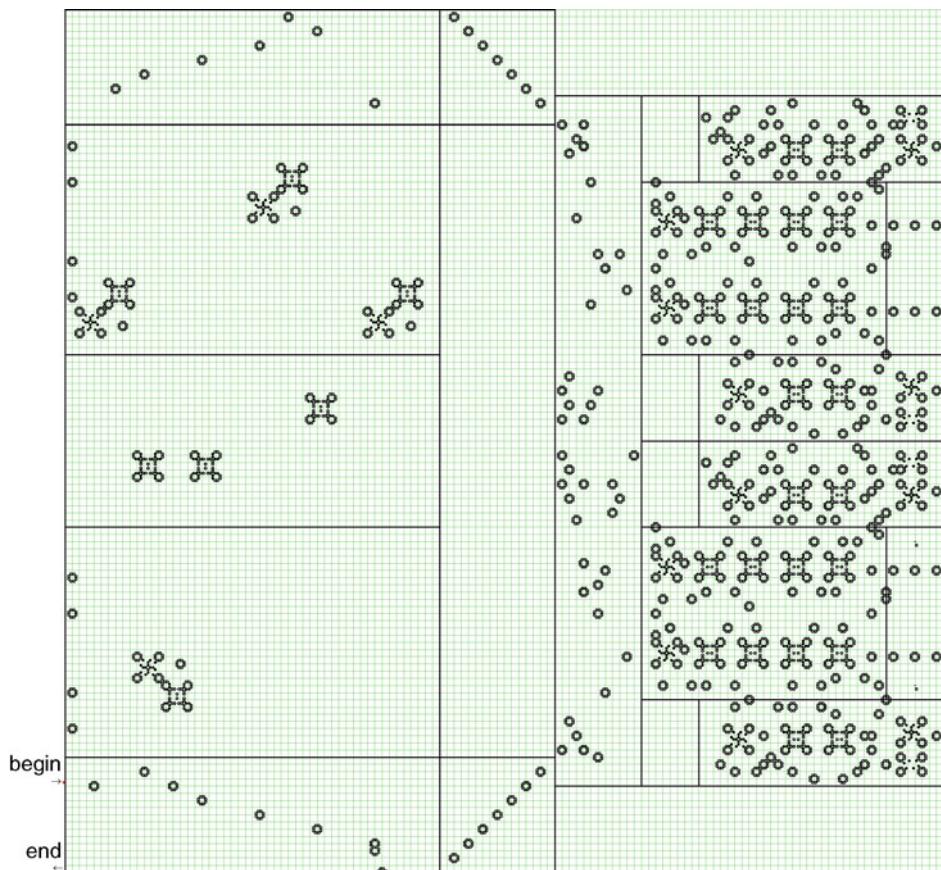


Fig. 21

An example of a reversible counter machine, which computes the function $2x+2$, embedded in the reversible PCA P_3 (Morita et al. 2002).



Margolus 1990). Kari (1996) solved this problem affirmatively by showing that every one- and two-dimensional RCA can be represented by block permutations and translations. Hence, an RCA in which any operation of block permutation and translation can be realized (e.g., by implementing a reversible logic circuit made of Fredkin gates) is intrinsically universal.

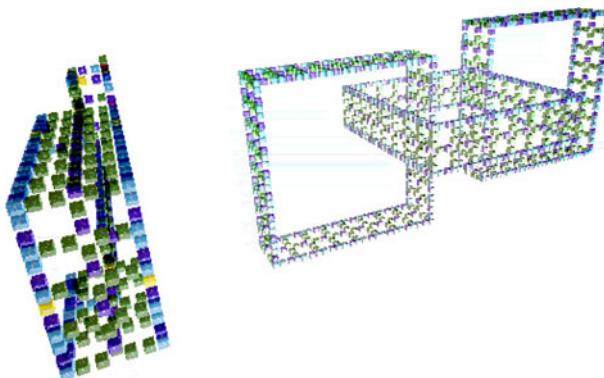
5 Concluding Remarks

In this chapter, several aspects and properties of RCAs, especially their computing ability, are discussed. In spite of the strong constraint of reversibility, RCAs have a rich ability in computing and information processing, and even very simple RCAs have computation-universality.

There are many open problems left for future studies. For example, consider computation-universality of RCAs. All the universal RCAs obtained so far use the framework of

Fig. 22

Self-replication of objects in a three-dimensional RCA (Imai et al. 2002).



either CAs with block rules, partitioned CAs (PCAs), or second-order CAs, as seen in the previous sections. In the one-dimensional case, a 24-state computation-universal RCA exists (☞ [Theorem 12](#)). But the number of states seems to be reduced much more, and the conjecture is that there is a universal one-dimensional RCA with fewer than ten states. To obtain such a result and develop the theory of RCAs, new techniques of designing RCAs will be necessary.

There are many topics on RCAs that are not stated in the previous sections. Some of them are briefly discussed. An early work on quantum CAs, which is also a kind of RCA, is found in the paper by Watrous (1995). Complexity of RCAs is studied by Sutner (2004). Kutrib and Malcher (2008) deal with RCAs as a model of formal language recognition systems. Imai and Morita (1996) showed a $3n$ -step solution of the firing squad synchronization problem on an RCA. Self-replication of a pattern is also possible in two- and three-dimensional RCAs (Imai et al. 2002; Morita and Imai 1996) (☞ [Fig. 22](#)).

There are several survey papers and books related to RCAs that are written from different viewpoints. Toffoli and Margolus (1990), and Kari (2005b) give good survey papers on RCAs. The paper by Kari (2005a), the books by Toffoli and Margolus (1987), and by Wolfram (2001) are references on (general) CAs, which also describe RCAs to some extent. On universality of general CAs, see the chapter ☞ [Universality in Cellular Automata](#) by Ollinger.

References

- Amoroso S, Cooper G (1970) The Garden of Eden theorem for finite configurations. *Proc Am Math Soc* 26:158–164
- Amoroso S, Patt YN (1972) Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *J Comput Syst Sci* 6:448–464
- Bennett CH (1973) Logical reversibility of computation. *IBM J Res Dev* 17:525–532
- Boykett T (2004) Efficient exhaustive listings of reversible one dimensional cellular automata. *Theor Comput Sci* 325:215–247
- Cook M (2004) Universality in elementary cellular automata. *Complex Syst* 15:1–40
- Czeizler E, Kari J (2007) A tight linear bound for the synchronization delay of bijective automata. *Theor Comput Sci* 380:23–36

- Durand-Lose J (1995) Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In: Proceedings of LATIN 95, Valparaíso, Chile, LNCS 911. Springer, pp 230–244
- Fredkin E, Toffoli T (1982) Conservative logic. *Int J Theor Phys* 21:219–253
- Hedlund GA (1969) Endomorphisms and automorphisms of the shift dynamical system. *Math Syst Theory* 3:320–375
- Imai K, Hori T, Morita K (2002) Self-reproduction in three-dimensional reversible cellular space. *Artif Life* 8:155–174
- Imai K, Morita K (1996) Firing squad synchronization problem in reversible cellular automata. *Theor Comput Sci* 165:475–482
- Imai K, Morita K (2000) A computation-universal two-dimensional 8-state triangular reversible cellular automaton. *Theor Comput Sci* 231:181–191
- Kari J (1994) Reversibility and surjectivity problems of cellular automata. *J Comput Syst Sci* 48:149–182
- Kari J (1996) Representation of reversible cellular automata with block permutations. *Math Syst Theory* 29:47–61
- Kari J (2005a) Theory of cellular automata: a survey. *Theor Comput Sci* 334:3–33
- Kari J (2005b) Reversible cellular automata. In: Proceedings of the DLT 2005, Palermo, Italy, LNCS 3572. Springer, pp 57–68
- Kutrib M, Malcher A (2008) Fast reversible language recognition using cellular automata. *Inf Comput* 206:1142–1151
- Landauer R (1961) Irreversibility and heat generation in the computing process. *IBM J Res Dev* 5:183–191
- Lecerf Y (1963) Machines de Turing réversibles — Reursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences* 257:2597–2600
- Margolus N (1984) Physics-like model of computation. *Physica* 10D:81–95
- Maruoka A, Kimura M (1976) Condition for injectivity of global maps for tessellation automata. *Inf Control* 32:158–162
- Maruoka A, Kimura M (1979) Injectivity and surjectivity of parallel maps for cellular automata. *J Comput Syst Sci* 18:47–64
- Minsky ML (1967) Computation: finite and infinite machines. Prentice-Hall, Englewood Cliffs, NJ
- Moore EF (1962) Machine models of self-reproduction. In: Proceedings of the symposia in applied mathematics, vol 14. American Mathematical Society, New York, pp 17–33
- Mora JCST, Vergara SVC, Martinez GJ, McIntosh HV (2005) Procedures for calculating reversible one-dimensional cellular automata. *Physica D* 202:134–141
- Morita K (1995) Reversible simulation of one-dimensional irreversible cellular automata. *Theor Comput Sci* 148:157–163
- Morita K (1996) Universality of a reversible two-counter machine. *Theor Comput Sci* 168:303–320
- Morita K (2001a) Cellular automata and artificial life — computation and life in reversible cellular automata. In: Goles E, Martínez S (eds) Complex systems. Kluwer, Dordrecht, pp 151–200
- Morita K (2001b) A simple reversible logic element and cellular automata for reversible computing. In: Proceedings of the 3rd international conference on machines, computations, and universality, LNCS 2055. Springer, Heidelberg, Germany, pp 102–113
- Morita K (2007) Simple universal one-dimensional reversible cellular automata. *J Cell Autom* 2:159–165
- Morita K (2008a) Reversible computing and cellular automata — a survey. *Theor Comput Sci* 395: 101–131
- Morita K (2008b) A 24-state universal one-dimensional reversible cellular automaton. In: Adamatzky A et al. (eds) Proceedings of AUTOMATA-2008. Luniver Press, Bristol, UK, pp 106–112
- Morita K, Harao M (1989) Computation universality of one-dimensional reversible (injective) cellular automata. *Trans IEICE Jpn E-72:758–762*
- Morita K, Imai K (1996) Self-reproduction in a reversible cellular space. *Theor Comput Sci* 168:337–366
- Morita K, Yamaguchi Y (2007) A universal reversible Turing machine. In: Proceedings of the 5th international conference on machines, computations, and universality, LNCS 4664. Springer, pp 90–98
- Morita K, Shirasaki A, Gono Y (1989) A 1-tape 2-symbol reversible Turing machine. *Trans IEICE Jpn E-72:223–228*
- Morita K, Tojima Y, Imai K, Oguro T (2002) Universal computing in reversible and number-conserving two-dimensional cellular spaces. In: Adamatzky A (ed) Collision-based computing. Springer, London, UK, pp 161–199
- Myhill J (1963) The converse of Moore's Garden-of-Eden theorem. *Proc Am Math Soc* 14:658–686
- von Neumann J (1966) In: Burks AW (ed) Theory of self-reproducing automata. The University of Illinois Press, Urbana, IL
- Ollinger N (2002) The quest for small universal cellular automata. In: Proceedings of ICALP, LNCS 2380. Lyon, France, pp 318–329
- Ollinger N, Richard G (2006) A particular universal cellular automaton, oai:hal.archives-ouvertes.fr:hal-00095821_v2
- Richardson D (1972) Tessellations with local transformations. *J Comput Syst Sci* 6:373–388
- Sutner K (1991) De Bruijn graphs and linear cellular automata. *Complex Syst* 5:19–31

- Sutner K (2004) The complexity of reversible cellular automata. *Theor Comput Sci* 325:317–328
- Toffoli T (1977) Computation and construction universality of reversible cellular automata. *J Comput Syst Sci* 15:213–231
- Toffoli T (1980) Reversible computing. In: de Bakker JW, van Leeuwen J (eds) *Automata, languages and programming, LNCS 85*. Springer, Berlin, Germany, pp 632–644
- Toffoli T, Margolus N (1987) Cellular automata machines. The MIT Press, Cambridge, MA
- Toffoli T, Margolus N (1990) Invertible cellular automata: a review. *Physica D* 45:229–253
- Toffoli T, Capobianco S, Mentrasti P (2004) How to turn a second-order cellular automaton into lattice gas: a new inversion scheme. *Theor Comput Sci* 325:329–344
- Watrous J (1995) On one-dimensional quantum cellular automata. In: *Proceedings of the 36th symposium on foundations of computer science*. Las Vegas, NV. IEEE, pp 528–537
- Wolfram S (2001) *A new kind of science*. Wolfram Media, Champaign, IL

8 Conservation Laws in Cellular Automata

Siamak Taati

Department of Mathematics, University of Turku, Finland

siamak.taati@gmail.com

1	<i>Introduction</i>	260
2	<i>Mathematical Formulation</i>	267
3	<i>Algorithmics</i>	272
4	<i>Flows and Particles</i>	278
5	<i>Further Topics</i>	284

Abstract

A conservation law in a cellular automaton (CA) is the statement of the invariance of a local and additive energy-like quantity. This chapter reviews the basic theory of conservation laws in cellular automata. A general mathematical framework for formulating conservation laws in cellular automata is presented and several characterizations of them are summarized. Computational problems regarding conservation laws (verification and existence problems) are discussed. Microscopic explanations of the dynamics of the conserved quantities in terms of flows and particle flows are explored. The related concept of dissipating energy-like quantities is also addressed.

1 Introduction

A cellular automaton (CA) is an abstract structure, consisting of a d -dimensional checkerboard ($d = 1, 2, 3, \dots$). Each cell of the board has a state chosen from a finite set of states. The state of each cell changes with time, according to a uniform, deterministic rule, which takes into account the previous state of the cell itself and those in its neighborhood. The changes, however, happen synchronously, and in discrete time steps.

In mathematics and computer science, cellular automata are studied as abstract models of computation, in the same way that Turing machines are (see the chapters [Algorithmic Tools on Cellular Automata](#), [Language Recognition by Cellular Automata](#), [Computations on Cellular Automata](#), and [Universality in Cellular Automata](#)). They are also treated as paradigms of symmetric dynamical systems on the Cantor space (see the chapter [Cellular Automata Dynamical Systems](#)). However, the chief reason for the interest in cellular automata comes from their characteristic similarities with nature: they are spatially extended dynamical systems; they are uniform (the same laws are applied everywhere in the space); the interactions are local (no action-at-a-distance); the amount of information in a finite region of space is finite (cf. Fredkin et al. 1982). Further characteristics of nature, such as the microscopic reversibility and conservation laws also arise in CA in a natural way (see the chapter [Reversible Cellular Automata](#)). This makes cellular automata exceptionally suitable for modeling physical and biological phenomena on the one hand (see the chapter [Cellular Automata and Lattice Boltzmann Modeling of Physical Systems](#)), and as a design framework for natural computing, on the other hand (see, e.g., Margolus 1984).

This chapter is a survey of the basic known results about local and additive conservation laws in cellular automata. Studying conservation laws in cellular automata may be beneficial from different aspects. When modeling a physical system, conservation laws of the system serve as design constraints that one would like to program in the model (see the chapter [Cellular Automata and Lattice Boltzmann Modeling of Physical Systems](#)). In physically realistic models of computation, conservation laws should naturally be taken into account (see Fredkin and Toffoli 1982 and Margolus 1984). Moreover, conservation laws in a CA may provide the same kind of “physical” insight about its dynamics as the insight that conservation laws in physics provide about the physical world.

In the rest of this section, the concept of conservation laws in cellular automata is illustrated by a number of examples. [Section 2](#) provides a precise mathematical formulation of conservation laws in cellular automata. [Section 3](#) is dedicated to the algorithmic problems that arise from studying conservation laws: how to discover them, and how to verify their validity. [Section 4](#) discusses local explanations of conservation laws in terms of the

flow of the conserved quantity. Finally, [Sect. 5](#) comments on some related issues. Throughout the exposition, the closely related concept of nonincreasing energy-like quantities is also discussed.

1.1 Few Examples

Conservation laws in cellular automata are obtained in a similar fashion as in physics. A real value is associated to each local pattern of cell states, interpreted as the “energy” (or “electric charge”, or ...) of that particular arrangement of states. The total energy of a configuration is the sum of the energy values of the patterns seen in different places on it. Intuitively, a conservation law asserts that the total energy of each configuration remains unchanged under the iterations of the CA.

As an example, consider the well-known *Traffic CA*, which resembles cars moving on a highway. The Traffic CA is a one-dimensional CA, consisting of an infinite number of cells arranged next to each other on a line. Each cell has two possible states: \blacksquare (interpreted as a “car”) or \square (“empty space”). At each step, a car moves one cell forward if and only if its front cell is empty. [Figure 1](#) shows a typical space–time diagram of the evolution of the Traffic CA. Not surprisingly, the number of cars on the highway remains constant along the evolution of the CA. To state this more precisely, the various positions on the line are indexed with integers $i \in \mathbb{Z}$. A configuration of the model is an assignment of values \blacksquare or \square to every position on the line. For each configuration $i \mapsto x[i]$, Fx is written for the configuration after one step. The car conservation law can now be stated, by saying that, for any configuration x , the following equality holds:

$$\sum_{i=-\infty}^{+\infty} \theta(x[i]) = \sum_{i=-\infty}^{+\infty} \theta((Fx)[i]) \quad (1)$$

where $\theta(\blacksquare) = 1$ and $\theta(\square) = 0$. Note that if the number of cars on a configuration x is infinite, the sum $\sum_i \theta(x[i])$ becomes $+\infty$. However, in this case, the configuration Fx has also infinitely many cars on it, and the equality still holds. (To learn more about cellular automata models of car traffic, see for example Nagel and Schreckenberg (1992) and Nagel and Herrmann (1993), and the relevant discussion in the chapter [Cellular Automata and Lattice Boltzmann Modeling of Physical Systems](#).)

\blacksquare Fig. 1

A typical space–time diagram of the Traffic CA. Time evolves downward. The highway is directed toward the left.

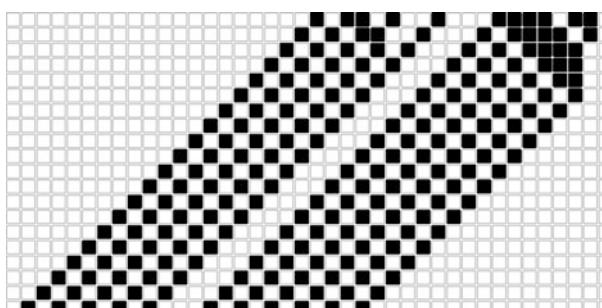
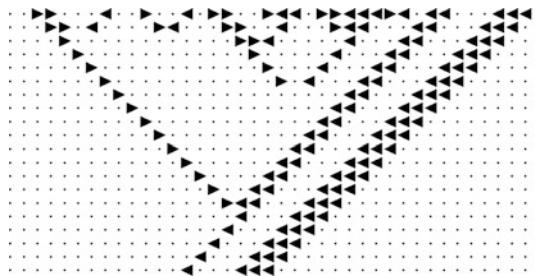


Fig. 2

A typical space–time diagram of the Just Gliders CA. The moving objects annihilate on encounter.



Another example is the *Just Gliders* CA. This is also one-dimensional. Each cell can be in either state \blacktriangleleft (a “particle” moving to the left), or \triangleright (a “particle” moving to the right), or \cdot (“empty space”). At each step, each particle moves one cell ahead. Particles moving in opposite directions annihilate when they meet. See [Fig. 2](#) for a typical space–time diagram. If the momentum of a right-moving particle \triangleright is defined to be 1, and the momentum of a left-moving particle \blacktriangleleft to be -1 , one can easily see that the total momentum of a configuration remains constant with time. More precisely, setting

$$\theta(a) \stackrel{\Delta}{=} \begin{cases} 1 & \text{if } a = \triangleright \\ -1 & \text{if } a = \blacktriangleleft \\ 0 & \text{if } a = \cdot \end{cases} \quad (2)$$

[Equation 1](#) is valid for any configuration x with a finite number of particles. For an infinite configuration (i.e., a configuration with infinitely many particles on it), the sum $\sum_i \theta(x[i])$ is not necessarily meaningful anymore. Therefore, the conservation law is expressed in terms of finite configurations only.

Alternatively, the above conservation law can be formulated in terms of the average momentum per cell of spatially periodic configurations. Namely, if a configuration x has period $p > 0$ (i.e., if $x[i+p] = x[i]$ for every i), one can say that

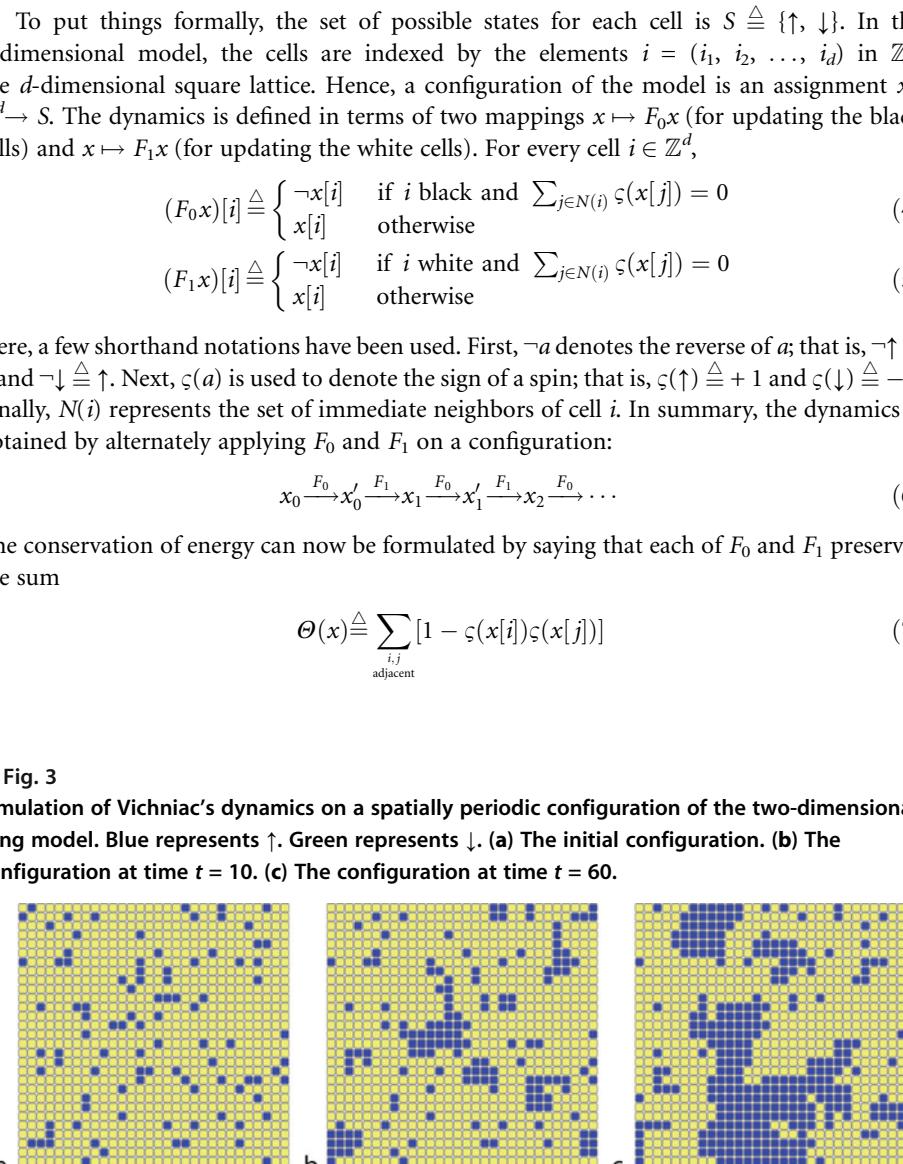
$$\frac{\sum_{i=1}^p \theta(x[i])}{p} = \frac{\sum_{i=1}^p \theta((Fx)[i])}{p} \quad (3)$$

As discussed later, these two formulations are equivalent in general; a CA conserves the energy of every finite configuration if and only if it preserves the average energy per cell of each spatially periodic configuration.

Next, a few physically interesting examples will be discussed. The *Ising* model was introduced by Wilhelm Lenz (1888–1957) and Ernst Ising (1900–1998) to explain the phenomenon of phase transition in ferromagnetic materials. It is a stochastic model and is extensively studied in statistical mechanics (see, e.g., Kindermann and Snell (1980); Sinai (1982); Georgii et al. (2000)). Gérard Vichniac has introduced a deterministic CA-like dynamics on it (Vichniac 1984) (see also Toffoli and Margolus 1987; Chopard and Droz 1998).

In the Ising model, each cell represents a tiny piece of ferromagnetic material having a spin (i.e., a magnetic moment resulting from the angular momentum of the electrons). For simplicity, each spin is approximated by either of two values: \uparrow (spin-up) or \downarrow (spin-down).

Adjacent spins tend to align. This tendency is depicted by assigning an energy 1 to each pair of adjacent spins that are anti-aligned, and energy -1 to those that are aligned.

Vichniac's dynamics is specially designed in such a way as to conserve this energy, hence emulating the regime where there is no heat transfer in and out of the material. The states of the cells are updated in two stages. The cells are colored black and white as on the chess board. At the first stage, all the black cells are updated in the following way: a spin on a black cell is flipped (from \uparrow to \downarrow , or from \downarrow to \uparrow) if and only if the change does not affect the total energy of the bonds with its adjacent spins. At the second stage, the white cells are updated in a similar fashion.  **Figure 3** shows few snapshots from a simulation of this CA-like dynamics.

To put things formally, the set of possible states for each cell is $S \triangleq \{\uparrow, \downarrow\}$. In the d -dimensional model, the cells are indexed by the elements $i = (i_1, i_2, \dots, i_d)$ in \mathbb{Z}^d , the d -dimensional square lattice. Hence, a configuration of the model is an assignment $x : \mathbb{Z}^d \rightarrow S$. The dynamics is defined in terms of two mappings $x \mapsto F_0 x$ (for updating the black cells) and $x \mapsto F_1 x$ (for updating the white cells). For every cell $i \in \mathbb{Z}^d$,

$$(F_0 x)[i] \triangleq \begin{cases} \neg x[i] & \text{if } i \text{ black and } \sum_{j \in N(i)} \varsigma(x[j]) = 0 \\ x[i] & \text{otherwise} \end{cases} \quad (4)$$

$$(F_1 x)[i] \triangleq \begin{cases} \neg x[i] & \text{if } i \text{ white and } \sum_{j \in N(i)} \varsigma(x[j]) = 0 \\ x[i] & \text{otherwise} \end{cases} \quad (5)$$

Here, a few shorthand notations have been used. First, $\neg a$ denotes the reverse of a ; that is, $\neg \uparrow \triangleq \downarrow$ and $\neg \downarrow \triangleq \uparrow$. Next, $\varsigma(a)$ is used to denote the sign of a spin; that is, $\varsigma(\uparrow) \triangleq +1$ and $\varsigma(\downarrow) \triangleq -1$. Finally, $N(i)$ represents the set of immediate neighbors of cell i . In summary, the dynamics is obtained by alternately applying F_0 and F_1 on a configuration:

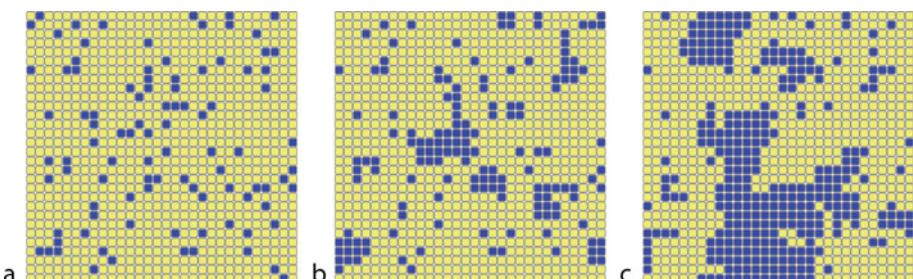
$$x_0 \xrightarrow{F_0} x'_0 \xrightarrow{F_1} x_1 \xrightarrow{F_0} x'_1 \xrightarrow{F_1} x_2 \xrightarrow{F_0} \dots \quad (6)$$

The conservation of energy can now be formulated by saying that each of F_0 and F_1 preserves the sum

$$\Theta(x) \triangleq \sum_{\substack{i,j \\ \text{adjacent}}} [1 - \varsigma(x[i])\varsigma(x[j])] \quad (7)$$

Fig. 3

Simulation of Vichniac's dynamics on a spatially periodic configuration of the two-dimensional Ising model. Blue represents \uparrow . Green represents \downarrow . (a) The initial configuration. (b) The configuration at time $t = 10$. (c) The configuration at time $t = 60$.



for any configuration x with only finitely many downward spins \downarrow , or finitely many upward spins \uparrow . A similar statement can be formulated for spatially periodic configurations.

Notice that Vichniac's dynamics is reversible; one can completely regenerate the configuration at time t , knowing the configuration at time $t + 1$. In fact, all one needs to do is to apply F_0 and F_1 in the reverse order. Reversibility is apparently a fundamental feature of nature. Even though the macroscopic world as one perceives it looks irreversible, every known physical process behaves reversibly in the ultimate microscopic scale (cf. Feynman 1965).

The next example, due to Pomeau (1984) and Margolus (1987), identifies an interesting energy-like invariant in a large class of reversible models. Each cell of the lattice has a state from the finite ring $\mathbb{Z}_m = \{0, 1, 2, \dots, m - 1\}$. The dynamics is of second order; that is, the configuration at time $t + 1$ depends not only on the configuration at time t , but also on the configuration at time $t - 1$; that is, $c_{t+1} = F(c_t, c_{t-1})$. The state of a cell $i \in \mathbb{Z}^d$ at time $t + 1$ is obtained by a rule of the form

$$c_{t+1}[i] = c_{t-1}[i] + f(c_t[N(i)]) \quad (8)$$

Here $N(i)$ is a finite set of cells that is called the neighborhood of cell i . The neighborhood is assumed to be uniform; that is, there is a finite set $N \subseteq \mathbb{Z}^d$ such that $N(i) \triangleq \{i + k : k \in N\}$ for every cell i . $c_t[N(i)]$ means the pattern of the states seen on the neighborhood of cell i in configuration c_t . Mathematically, this can be seen as an element of \mathbb{Z}_m^N , the set of all possible assignments $p : N \rightarrow \mathbb{Z}_m$. The function $f : \mathbb{Z}_m^N \rightarrow \mathbb{Z}_m$ assigns a value $f(p) \in \mathbb{Z}_m$ to each neighborhood pattern $p : N \rightarrow \mathbb{Z}_m$. Intuitively, in order to update its state, a cell i applies a function f on the current state of its neighbors, and depending on the result, permutes the state it used to have one step before.

Notice that any automaton that is defined this way is automatically reversible; one can retrace an orbit $\dots, c_{t-1}, c_t, c_{t+1}, \dots$ backward using the rule

$$c_{t-1}[i] = c_{t+1}[i] - f(c_t[N(i)]) \quad (9)$$

Now, suppose that the neighborhood N is symmetric, meaning that $k \in N$ if and only if $-k \in N$. Suppose further that one finds a function $g : \mathbb{Z}_m^N \rightarrow \mathbb{Z}_m$ of the form

$$g(p) \triangleq \sum_{k \in N} \beta_k p[k] \quad (10)$$

($\beta_k \in \mathbb{Z}_m$) that has the following two properties:

- (i) It is symmetric; that is, $\beta_{-k} = \beta_k$ for every $k \in N$.
- (ii) It is orthogonal to f , in the sense that $f(p)g(p) = 0$ for every $p \in \mathbb{Z}_m^N$.

Denote the configuration in which every cell is in state 0 by $\mathbf{0}$. For simplicity, assume that $\dots, \mathbf{0}, \mathbf{0}, \mathbf{0}, \dots$ is a valid orbit of the automaton. Equivalently, this means that f maps the uniformly-0 pattern into 0. (One could avoid this requirement by formulating the conservation law in terms of spatially periodic configurations.) A configuration is called finite if only a finite number of cells have nonzero states in it.

Let $\dots, c_{t-1}, c_t, c_{t+1}, \dots$ be an arbitrary orbit consisting of finite configurations. The value of the sum

$$\Theta(c_{t-1}, c_t) \triangleq \sum_{i \in \mathbb{Z}^d} c_{t-1}[i] g(c_t[N(i)]) \quad (11)$$

is claimed to be independent of the time t .

From [Eq. 8](#) and property (ii) one can write

$$(c_{t+1}[i] - c_{t-1}[i])g(c_t[N(i)]) = 0 \quad (12)$$

Summing over all cells i , one obtains

$$\sum_{i \in \mathbb{Z}^d} c_{t-1}[i]g(c_t[N(i)]) = \sum_{i \in \mathbb{Z}^d} c_{t+1}[i]g(c_t[N(i)]) \quad (13)$$

By the symmetry of g (property (i)) one can rewrite the right-hand side as follows:

$$\sum_{i \in \mathbb{Z}^d} c_{t+1}[i]g(c_t[N(i)]) = \sum_{i \in \mathbb{Z}^d} \sum_{k \in N} \beta_k c_{t+1}[i]c_t[i+k] \quad (14)$$

$$= \sum_{i' \in \mathbb{Z}^d} \sum_{k' \in N} \beta_{-k'} c_{t+1}[i' + k']c_t[i'] \quad (15)$$

$$= \sum_{i' \in \mathbb{Z}^d} \sum_{k' \in N} \beta_{k'} c_{t+1}[i' + k']c_t[i'] \quad (16)$$

$$= \sum_{i' \in \mathbb{Z}^d} c_t[i']g(c_{t+1}[N(i')]) \quad (17)$$

Therefore, one obtains

$$\Theta(c_{t-1}, c_t) = \sum_{i \in \mathbb{Z}^d} c_{t-1}[i]g(c_t[N(i)]) \quad (18)$$

$$= \sum_{i \in \mathbb{Z}^d} c_t[i]g(c_{t+1}[N(i)]) \quad (19)$$

$$= \Theta(c_t, c_{t+1}) \quad (20)$$

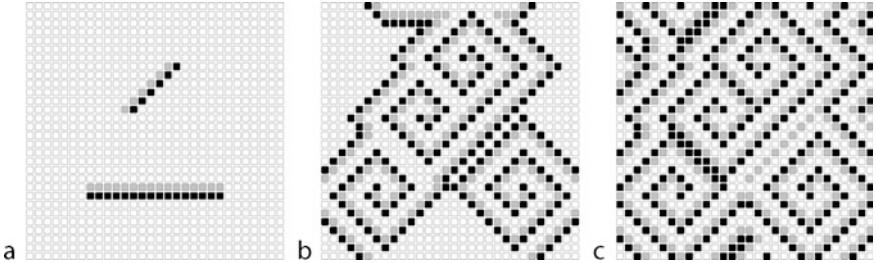
proving the claim.

Yet another beautiful example is the following discrete model of an excitable medium, due to Greenberg and Hastings (1978b) (see the relevant part in the chapter [Cellular Automata and Lattice Boltzmann Modeling of Physical Systems](#), and also Greenberg et al. (1978a, 1980)). The CA runs on a two-dimensional board. Each cell is either “at rest” (state \square), “excited” (state \blacksquare), or in a “refractory phase” (state $\blacksquare\square$). A cell that is at rest remains so unless it is “stimulated” by one or more of its four neighbors (i.e., if at least one of its neighbors is excited). An excited cell undergoes a 1-step refractory phase before going back to rest and starting to respond to stimulations again. Typically, a configuration of the infinite board contains a number of “singularities” with waves continuously swirling around them ([Fig. 4](#)). The singularities are never created or destroyed. Therefore, the number of such singularities remain constant throughout time. To be more precise, the singularities are the 2×2 blocks of cells with states $\blacksquare\square\square\square$, $\blacksquare\square\square\square$, or $\blacksquare\square\square\square$, or the rotations or mirror images of these blocks. One can easily verify that a singular 2×2 block remains singular after one step, and a non-singular block remains non-singular.

Sometimes, one can find an energy-like function that is not perfectly conserved by the evolution of a CA, yet whose total value is never increased (or never decreased) with time. Physically, such a situation is comparable with a system that is isolated from its environment, except that it may dissipate heat, resulting in a decrease of its total energy. Mathematically, a nonincreasing energy function may be helpful in studying stability properties of a CA.

Fig. 4

Simulation of Greenberg–Hastings model on a spatially periodic configuration. (a) The initial configuration. (b) The configuration at time $t = 10$. (c) The configuration at time $t = 60$.



As an example, in the Just Gliders CA that was discussed before, the number of left-moving particles is never increased, though it may decrease. Formally, one has

$$\sum_{i=-\infty}^{+\infty} \theta_L(x[i]) \geq \sum_{i=-\infty}^{+\infty} \theta_L((Fx)[i]) \quad (21)$$

where

$$\theta_L(a) \triangleq \begin{cases} 1 & \text{if } a = \blacktriangleleft \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

for any configuration x with a finite number of particles on it. Equivalently, one can write

$$\frac{\sum_{i=1}^p \theta_L(x[i])}{p} \geq \frac{\sum_{i=1}^p \theta_L((Fx)[i])}{p} \quad (23)$$

for every spatially periodic configuration x with period $p > 0$. In the Traffic CA, it is easy to verify that the number of blocks $\blacksquare\blacksquare$ of two consecutive cars is never increased; that is,

$$\sum_{i=-\infty}^{+\infty} \theta'(x[i, i+1]) \geq \sum_{i=-\infty}^{+\infty} \theta'((Fx)[i, i+1]) \quad (24)$$

where

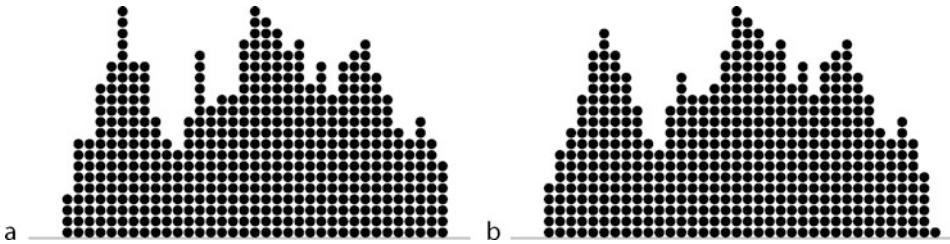
$$\theta'(ab) \triangleq \begin{cases} 1 & \text{if } ab = \blacksquare\blacksquare \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

A more interesting example is the *Sand Pile* model due to Bak et al. (1987, 1988). On each cell of the board, there is a stack of sand grains. One can consider the height of this stack as the state of the cell. So, each cell $i \in \mathbb{Z}^d$ has a state $h[i] \in \{0, 1, \dots, N\}$. (To keep the state set finite, it has been assumed that each cell may contain no more than $N > 0$ grains.) If the difference between the height of the stacks in two adjacent cells is more than a threshold $K \geq 4d$, one grain of sand from the higher cell tumbles down onto the lower cell. More precisely, if $h : \mathbb{Z}^d \rightarrow \{0, 1, \dots, N\}$ is a configuration of the automaton, its configuration after one step would be $Fh : \mathbb{Z}^d \rightarrow \{0, 1, \dots, N\}$, where

$$(Fh)[i] = h[i] - |\{j \in N(i) : h[i] - h[j] \geq K\}| + |\{j \in N(i) : h[j] - h[i] \geq K\}| \quad (26)$$

Fig. 5

Two consecutive configurations of the one-dimensional Sand Pile model with parameter $K = 5$.
(a) The initial configuration. (b) The configuration after one step.



Here $N(i)$ denotes the set of immediate neighbors of cell i (☞ Fig. 5). Clearly, the total number of sands on a finite configuration is never changed; that is,

$$\sum_{i \in \mathbb{Z}^d} (Fh)[i] = \sum_{i \in \mathbb{Z}^d} h[i] \quad (27)$$

A bit less trivial is the fact that the sum of the squares of the number of sands on the cells is nonincreasing:

$$\sum_{i \in \mathbb{Z}^d} ((Fh)[i])^2 \leq \sum_{i \in \mathbb{Z}^d} (h[i])^2 \quad (28)$$

To see this, note that the height of a single grain of sand does never increase with time. The sum of the heights of all grains on configuration h is

$$\sum_{i \in \mathbb{Z}^d} \sum_{k=1}^{h[i]} k = \sum_{i \in \mathbb{Z}^d} \frac{1}{2} h[i] (h[i] + 1) \quad (29)$$

$$= \frac{1}{2} \sum_{i \in \mathbb{Z}^d} (h[i])^2 + \frac{1}{2} \sum_{i \in \mathbb{Z}^d} h[i] \quad (30)$$

and does never increase with time. From ☞ Eq. 27 one knows that the second term in the above sum remains constant with time. Therefore, the first term cannot increase with time. (The original Sand Pile automaton of Bak et al. (1987, 1988) is more elegant in that its cell states store not the height $h[i]$, but the difference $h[i] - h[i+1]$ between the heights of two consecutive cells. This way, one can represent sand piles of arbitrary height using a bounded number of cell states. As remarked by Goles (1992), a similar nonincreasing energy can be found for such CA, provided the energy is allowed to depend on the previous configurations, too. Nonuniform nonincreasing energies are used in Anderson et al. (1989) and Goles and Kiwi (1993) to analyze Sand Pile and similar automata.)

2 Mathematical Formulation

In this section, what is exactly meant by a conservation law in a CA is formulated in a more precise language.

2.1 Cellular Automata

The discussion is restricted to cellular automata on the infinite d -dimensional square lattice. Thus the cells of the *lattice* are indexed by the elements of \mathbb{Z}^d . The *states* of the cells are chosen from a finite set S that contains at least two elements. A *configuration* of the lattice is a mapping $x : \mathbb{Z}^d \rightarrow S$ which assigns a state to each cell on the lattice. A *pattern* means an assignment $p : D \rightarrow S$ of states to a subset $D \subseteq \mathbb{Z}^d$ of cells. A *finite pattern* is a pattern that has a finite domain. If $p : D \rightarrow S$ is a pattern and $E \subseteq D$, $p|E$ is written to denote the restriction of p to E ; that is, $p|E$ stands for the pattern $q : E \rightarrow S$ seen over the set E under p .

For each $k \in \mathbb{Z}^d$, σ^k denotes the *translation* by k . That is, for each pattern $p : D \rightarrow S$, $\sigma^k p$ is the pattern with $(\sigma^k p)[i] = p[k + i]$ whenever $k + i \in D$.

To specify a *cellular automaton* (CA), one further needs to specify a *neighborhood* and a *local rule*. The neighborhood is presented by a finite set $0 \in N \subseteq \mathbb{Z}^d$. The neighborhood of a cell i is the set $N(i) \triangleq \{i + k : k \in N\}$. The local rule is a function $f : S^N \rightarrow S$ that provides a new state for each cell i by looking at its neighborhood pattern. Hence, for every configuration $x : \mathbb{Z}^d \rightarrow S$, one obtains a new configuration $Fx : \mathbb{Z}^d \rightarrow S$ where

$$(Fx)[i] = f((\sigma^i x)[N]) \quad (31)$$

$$= f(\sigma^i(x[N(i)])) \quad (32)$$

for each cell i . The dynamics of the CA is realized by iterating the *global mapping* F on an initial configuration x (This definition does not cover some of the examples [Sect. 1](#) (namely, Vichniac's dynamics on the Ising model, and the second order model). However, those models can be easily transformed into a standard CA as defined here (cf. Toffoli and Margolus [1987](#)).)

$$x \xrightarrow{F} Fx \xrightarrow{F} F^2x \xrightarrow{F} F^3x \xrightarrow{F} \dots . \quad (33)$$

One often identifies a CA with its global mapping.

It is useful to see the configuration space $S^{\mathbb{Z}^d}$ as a topological space. The product topology on $S^{\mathbb{Z}^d}$ is the smallest topology with respect to which all the projections $x \mapsto x[i]$ are continuous. In this topology, a sequence $\{x_t\}_{t=1}^\infty$ converges to a configuration x if and only if for each cell i , $x_t[i] = x[i]$ for all sufficiently large t . The space $S^{\mathbb{Z}^d}$ is compact and metrizable, and is homeomorphic to the Cantor set. A *cylinder* is a set of the form

$$[p] \triangleq \{x \in S^{\mathbb{Z}^d} : x[D] = p\} \quad (34)$$

where $p : D \rightarrow S$ is a finite pattern. Cylinders are both open and closed, and form a basis for the product topology. The global mapping of a cellular automaton is continuous with respect to the product topology. Moreover, every translation-invariant continuous mapping on $S^{\mathbb{Z}^d}$ is the global mapping of a CA (Hedlund [1969](#)). The topological aspects of cellular automata are discussed in the chapter [Cellular Automata Dynamical Systems](#).

The Borel σ -algebra on the space $S^{\mathbb{Z}^d}$ of configurations is the σ -algebra generated by the cylinders. A Borel probability measure π is completely determined by assigning a probability $0 \leq \pi([p]) \leq 1$ to each cylinder $[p]$ in a consistent way (cf. Parthasarathy [1967](#)). The space \mathcal{M} of all Borel probability measures on $S^{\mathbb{Z}^d}$ can be topologized by the vague topology (aka the weak* topology) (see, e.g., Walters [1982](#)). This space is also compact and metrizable. A sequence $\{\pi_t\}_{t=0}^\infty$ converges to a measure π if and only if for each cylinder $[p]$, the sequence

$\{\pi_t([p])\}_{t=0}^\infty$ of real numbers converges to $\pi([p])$. A CA F induces a continuous mapping on \mathcal{M} via $F\pi \triangleq \pi \circ F^{-1}$. A translation-invariant measure is a measure $\pi \in \mathcal{M}$ such that $\sigma^k \pi = \pi$ for every translation σ^k . The set of all translation-invariant Borel probability measures on $S^{\mathbb{Z}^d}$ is a closed and convex subspace of \mathcal{M} and is denoted by \mathcal{M}_σ .

2.2 Energy

Let S^* denote the set of all finite patterns *modulo* translations (i.e., forgetting their exact positions). To be strict, an element of S^* is a class $\langle p \rangle$ of patterns that can be obtained by translating p . However, unless there is a risk of confusion, one often abuses the notations and identifies a class $\langle p \rangle$ with any of its elements. The elements of S^* are seen as generalized words. In particular, when $d = 1$ (i.e., on the one-dimensional lattice \mathbb{Z}), one has $S^* \subseteq S^{\#}$, where S^* stands for the set of finite words on the alphabet S . Let \emptyset be the (unique) pattern with an empty domain.

A (local and additive) *energy* is specified by assigning an *interaction potential* $\theta(p) \in \mathbb{R}$ to each finite pattern $p \in S^*$. One requires that $\theta(\emptyset) = 0$, and that the set $\{p : \theta(p) \neq 0\} \subseteq S^* \setminus \{\emptyset\}$ is finite. The latter set is called the *support* of θ and is denoted by $\text{supp}(\theta)$. For a configuration $x : \mathbb{Z}^d \rightarrow S$ and a finite collection A of cells, the value $\theta(x[A])$ is interpreted as the energy resulting from the interaction of the cells in A . The *total energy* of x (whenever meaningful) is simply the sum

$$\Theta(x) \triangleq \sum_{\substack{A \subseteq \mathbb{Z}^d \\ \text{finite}}} \theta(x[A]) \quad (35)$$

However, the sum (Eq. 35) typically does not have a well-defined value or is infinite.

There are essentially three ways around this problem. The first approach is to consider only the difference between the energy of two configurations that are only slightly perturbed from each other (more precisely, differ on only a finite number of cells). The second approach is to work with the average or expected energy per cell in a configuration. The third approach is to avoid any global notion of energy and instead describe a conservation law in terms of the local redistribution of energy at each step. Fortunately, all these lead to equivalent concepts of a conservation law. Now, the first two approaches and their equivalence are discussed. The local approach is discussed later in Sect. 4.

It is said that two configurations $x, y : \mathbb{Z}^d \rightarrow S$ are *asymptotic* if they disagree on no more than a finite number of cells. The *difference* between the energy of two asymptotic configurations x and y is defined to be

$$\delta\Theta(x, y) \triangleq \sum_{\substack{A \subseteq \mathbb{Z}^d \\ \text{finite}}} [\theta(y[A]) - \theta(x[A])] \quad (36)$$

It is worth noting how the *locality* and *additivity* of energy translate in this framework. The *interaction range* of an energy θ can be identified by a minimal neighborhood $0 \in M \subseteq \mathbb{Z}^d$ such that, for every pattern $p : D \rightarrow S$ in the support of θ and every $i \in D$, one has $D \subseteq M(i)$. Since $\text{supp}(\theta)$ is finite, the interaction range of θ is also finite.

If two patterns $p : D \rightarrow S$ and $q : E \rightarrow S$ agree on the intersection $D \cap E$ of their domains (in particular, if $D \cap E = \emptyset$), one can merge them together and obtain a pattern $p \vee q : D \cup E \rightarrow S$ that agrees with p and q on their domains. The *boundary* of a set A (with respect to the neighborhood M) is the set $\partial M(A) \triangleq M(A) \setminus A$.

Proposition 1 (locality) Let $p, q : D \rightarrow S$, $r : \partial M(D) \rightarrow S$, and $u, v : \mathbb{Z}^d \setminus M(D) \rightarrow S$ be arbitrary patterns. Then,

$$\delta\Theta(p \vee r \vee u, q \vee r \vee u) = \delta\Theta(p \vee r \vee v, q \vee r \vee v) \quad (37)$$

Proposition 2 (additivity) Let $D, E \subseteq \mathbb{Z}^d$ be two finite nonempty sets such that $M(D) \cap E = D \cap M(E) = \emptyset$. Let $p_0, p_1 : D \rightarrow S$, $q_0, q_1 : E \rightarrow S$ and $w : \mathbb{Z}^d \setminus D \setminus E \rightarrow S$ be arbitrary patterns. Then,

$$\begin{aligned} \delta\Theta(p_0 \vee q_0 \vee w, p_1 \vee q_1 \vee w) &= \delta\Theta(p_0 \vee q_0 \vee w, p_1 \vee q_0 \vee w) \\ &\quad + \delta\Theta(p_0 \vee q_0 \vee w, p_0 \vee q_1 \vee w) \end{aligned} \quad (38)$$

A *local observable* (or a locally observable property) is a mapping $\mu : S^{\mathbb{Z}^d} \rightarrow \Gamma$ (Γ being an arbitrary set) that depends only on the states of a finite number of cells. That is, μ is a local observable if there is a finite neighborhood $W \subseteq \mathbb{Z}^d$ (the observation window) and a local rule $g : S^W \rightarrow \Gamma$ such that $\mu(x) = g(x[W])$. Observing a configuration x around a cell i one gets the value $\mu(\sigma^i x) = g(x[W(i)])$.

Every real-valued local observable μ with local rule $g : S^W \rightarrow \mathbb{R}$ defines an interaction potential θ via

$$\theta(p) \triangleq \begin{cases} g(p) & \text{if } p \in S^W \\ 0 & \text{otherwise} \end{cases} \quad (39)$$

The energy difference $\delta\Theta(x, y)$ can then be calculated using

$$\delta\Theta(x, y) = \sum_{i \in \mathbb{Z}^d} [\mu(\sigma^i y) - \mu(\sigma^i x)] \quad (40)$$

Conversely, given an interaction potential θ , one can construct a real-valued local observable μ that generates $\delta\Theta$ via [Eq. 40](#). For example, one can choose the interaction range M of θ as the observation window and define

$$\mu_\theta(x) \triangleq \sum_{0 \in A \subseteq M} \frac{1}{|A|} \theta(x[A]) \quad (41)$$

Hence, one can equivalently specify an energy using a real-valued local observable.

Consider an energy which is formalized using a local observable μ . For every $n \geq 0$, let $I_n \triangleq [-n, n]^d$ be the centered hypercube of size $(2n+1)^d$ on the lattice. The *average* energy per cell in a configuration x is obtained by taking the limit of the finite averages

$$\frac{\sum_{i \in I_n} \mu(\sigma^i x)}{|I_n|} \quad (42)$$

when $n \rightarrow \infty$. Since the limit does not always exist, one uses the *upper* or *lower* average energy per cell

$$\bar{\mu}(x) \triangleq \limsup_{n \rightarrow \infty} \frac{\sum_{i \in I_n} \mu(\sigma^i x)}{|I_n|} \quad (43)$$

$$\underline{\mu}(x) \triangleq \liminf_{n \rightarrow \infty} \frac{\sum_{i \in I_n} \mu(\sigma^i x)}{|I_n|} \quad (44)$$

For every translation-invariant Borel probability measure $\pi \in \mathcal{M}_\sigma$ on $S^{\mathbb{Z}^d}$, one can also define the *expected* energy per cell

$$\pi(\mu) \triangleq \int \mu d\pi \triangleq \sum_{p: W \rightarrow S} g(p)\pi([p]) \quad (45)$$

The mapping $\pi \mapsto \pi(\mu)$ is uniformly continuous. According to the pointwise ergodic theorem (see, e.g., Walters 1982), for every ergodic measure $\pi \in \mathcal{M}_\sigma$ (i.e., ergodic with respect to σ) and π -almost every configuration $x \in S^{\mathbb{Z}^d}$, one has $\bar{\mu}(x) = \underline{\mu}(x) = \pi(\mu)$. Ergodic measures are the extremal points of the convex set \mathcal{M}_σ .

2.3 Conservation of Energy

One can now formulate conservation laws in several different, but equivalent ways. One says that a CA F *conserves* an energy defined in terms of an interaction potential θ or an observable μ if any of the following equivalent conditions hold.

Theorem 1 (Hattori and Takesue 1991; Boykett and Moore 1998; Boccara and Fuks 1998; 2002; Pivato 2002; Kürka 2003a; Durand et al. 2003; Moreira et al. 2004) *Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a cellular automaton and $\theta : S^\# \rightarrow \mathbb{R}$ an interaction potential. Let $\mu : S^{\mathbb{Z}^d} \rightarrow \mathbb{R}$ be a local observable that generates the same energy as θ . Let $\diamond : \mathbb{Z}^d \rightarrow S$ be an arbitrary configuration. The following conditions are equivalent.*

- C-1 $\delta\theta(Fx, Fy) = \delta\theta(x, y)$ for every two asymptotic configurations x and y .
- C-2 $\delta\theta(F\diamond, Fx) = \delta\theta(\diamond, x)$ for every configuration x asymptotic to \diamond .
- C-3 $\bar{\mu}(Fx) = \bar{\mu}(x)$ for every configuration x .
- C-4 $\bar{\mu}(Fx) = \bar{\mu}(x)$ for every periodic configuration x .
- C-5 $(F\pi)(\mu) = \pi(\mu)$ for every translation-invariant probability measure π .
- C-6 $(F\pi)(\mu) = \pi(\mu)$ for every ergodic translation-invariant probability measure π .

Condition (C-1) states that the CA preserves the difference between the energy of every two asymptotic configurations, while condition (C-2) requires that the energy of a configuration relative to a fixed orbit remains constant. Condition (C-3) means that the (upper) average energy per cell of each configuration remains unchanged with time. Condition (C-3) only requires the CA to preserve the average energy per cell of the periodic configurations. Conditions (C-5) and (C-6) express a conservation law in terms of the expected energy per cell.

The equivalence of the conditions (C-1) and (C-2) is an immediate consequence of the locality of energy. To prove that (C-1) implies (C-3), one first shows that $\bar{\mu}(Fz) = \bar{\mu}(z)$ in the special case that z is a uniform configuration. An arbitrary configuration x can then be approximated by configurations x_n that agree with x on larger and larger centered hypercubes I_n and have a fixed state s on all the other cells. Similarly, the implication (C-4) \Rightarrow (C-1) follows from the locality of energy, by considering periodic configurations \hat{x} and \hat{y} that agree with x and y on a sufficiently large hypercube I_n . That (C-3) implies (C-6) follows from the ergodic theorem. The implication (C-6) \Rightarrow (C-5) is a consequence of the fact that every translation-invariant measure π is a limit of convex combinations of ergodic measures (cf. Walters 1982). To see that (C-5) implies (C-4), for every periodic configuration x , consider

a measure π_x whose probability mass is concentrated on the σ -orbit of x . More details can be found, for example, in Taati (2009). Yet other characterizations of conservation laws can be found in Pivato (2002) and Durand et al. (2003).

A uniform configuration \diamond for which $F\diamond = \diamond$ is called a *quiescent* configuration. For example, in the Traffic CA, the configuration with no car is quiescent; so is the configuration with cars on every cell. There is often a distinguished quiescent configuration \diamond , which is seen as the “background” or the “vacuum,” and one is interested in the evolution of the configurations that are asymptotic to \diamond . A configuration that is asymptotic to \diamond is then called a *finite* configuration. The image of a finite configuration under F is obviously finite.

Choosing the interaction potential θ properly, one can ensure that this distinguished quiescent configuration \diamond has zero total energy. Then every finite configuration x would have a finite well-defined total energy $\Theta(x) = \delta\Theta(\diamond, x)$. Hence, one can express the conservation law in terms of the total energy of the finite configurations: the CA F conserves θ if and only if

$$\text{C-7 } \Theta(Fx) = \Theta(x) \text{ for every finite configuration } x.$$

2.4 Dissipation of Energy

Recall, from the previous section, the Sand Pile example, in which an energy was presented that was not conserved, but its value was never increased. Such *dissipation laws* are now formalized. An energy θ is said to be *dissipative* (or nonincreasing) under a CA F if any of the following equivalent conditions hold.

Theorem 2 (Kůrka 2003a; Moreira et al. 2004; Bernardi et al. 2005) *Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a cellular automaton and $\theta : S^{\#} \rightarrow \mathbb{R}$ an interaction potential. Let $\mu : S^{\mathbb{Z}^d} \rightarrow \mathbb{R}$ be a local observable that generates the same energy as θ . The following conditions are equivalent.*

- D-1 $\bar{\mu}(Fx) \leq \bar{\mu}(x)$ for every configuration x .
- D-2 $\bar{\mu}(Fx) \leq \bar{\mu}(x)$ for every periodic configuration x .
- D-3 $(F\pi)(\mu) \leq \pi(\mu)$ for every translation-invariant probability measure π .
- D-4 $(F\pi)(\mu) \leq \pi(\mu)$ for every ergodic translation-invariant probability measure π .

Furthermore, if $\diamond : \mathbb{Z}^d \rightarrow S$ is a quiescent configuration, the following condition is also equivalent to the above.

$$\text{D-5 } \delta\Theta(\diamond, Fx) \leq \delta\Theta(\diamond, x) \text{ for every finite configuration } x.$$

The proofs are similar to those for conservation laws. In case the quiescent configuration \diamond has total energy zero, the condition (C-5) takes the following concise form:

$$\text{D-6 } \Theta(Fx) \leq \Theta(x) \text{ for every finite configuration } x.$$

3 Algorithmics

In this section, two algorithmic questions related to conservation laws in cellular automata are discussed. It is shown how one can verify the validity of a conservation law algorithmically. Next, the more difficult question of finding conservation laws in a CA is discussed. Though

one can enumerate all the possible candidates and verify their validity one by one, there is no algorithm to tell beforehand whether a CA has any nontrivial conservation law or not, even when restricted to one-dimensional CA.

Dissipative energies are more complicated to recognize. Fortunately, in one dimension, one can still decide whether a given energy is dissipative or not. The question is, however, undecidable in the higher-dimensional case.

3.1 Conservation Laws

Suppose that one is given a CA F and an interaction potential θ , and asked whether F conserves the energy defined by θ . Verifying either of the characterizations (C-1)–(C-6) requires establishing infinitely many equalities. However, as was first noticed by Hatori and Takesue (1991), it suffices to verify the equality

$$\delta\Theta(Fx, Fy) = \delta\Theta(x, y) \quad (46)$$

for all x and y that differ on exactly one cell.

Indeed, suppose that x and y are two configurations that disagree only on a finite set D of cells. One can then find a sequence

$$x = x_0, x_1, x_2, \dots, x_n = y \quad (47)$$

of configurations, where $n = |D|$, such that x_i and x_{i+1} disagree on exactly one cell. If \bullet Eq. 46 holds whenever two configurations differ on a single cell, one can write

$$\delta\Theta(x, y) = \sum_{i=0}^{n-1} \delta\Theta(x_i, x_{i+1}) \quad (48)$$

$$= \sum_{i=0}^{n-1} \delta\Theta(Fx_i, Fx_{i+1}) \quad (49)$$

$$= \delta\Theta(Fx, Fy) \quad (50)$$

(Note that $\delta\Theta(a, c) = \delta\Theta(a, b) + \delta\Theta(b, c)$ for every a, b, c .)

Proposition 3 (Hatori and Takesue 1991) *Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a cellular automaton and $\theta : S^\# \rightarrow \mathbb{R}$ an interaction potential. The following conditions are equivalent.*

C-1 $\delta\Theta(Fx, Fy) = \delta\Theta(x, y)$ for every two asymptotic configurations x and y .

C-8 $\delta\Theta(Fx, Fy) = \delta\Theta(x, y)$ for every two configurations x and y that disagree on exactly one cell.

Since θ (and hence $\delta\Theta$) is translation-invariant, one can, in fact, consider only configurations x and y that disagree on cell 0.

Now let $0 \in M \subseteq \mathbb{Z}^d$ be the interaction range of θ , and let $0 \in N \subseteq \mathbb{Z}^d$ be the neighborhood of F . For every two configurations x and y that disagree only on cell 0, the value $\delta\Theta(x, y)$ depends only on the state of the cells in x and y that are in the neighborhood $M(0)$. Similarly, since Fx and Fy may disagree only on the set $N^{-1}(0)$ (i.e., the set of cells that have cell 0 as neighbor) the value $\delta\Theta(Fx, Fy)$ depends on the state of the cells in Fx and Fy that are in the set $M(N^{-1}(0))$. Therefore, condition (C-8) reduces to a finite number of equalities, which can each be verified in a finite time.

The following algorithm exploits the above discussion to answer whether F conserves θ . Let $f: S^N \rightarrow S$ be the local rule of F . There is a natural way to extend the application of f to any finite pattern $p: N(A) \rightarrow S$. Namely, the image of p under f is a pattern $f(p): A \rightarrow S$, where $f(p)[i] \triangleq f((\sigma^i p)[N])$ for every $i \in A$. For every finite pattern $p: D \rightarrow S$, let one use the shorthand

$$\Theta(p) \triangleq \sum_{A \subseteq D} \theta(p[A]) \quad (51)$$

```

let  $W = M(N^{-1})$ ;
for every pattern  $p: N(W) \rightarrow S$  and every state  $s \in S$ ,
    let  $q: N(W) \rightarrow S$  and
        set  $q[0] = s$  and  $q[i] = p[i]$  for  $i \neq 0$ ;
    let  $p', q': W \rightarrow S$  and
        set  $p' = f(p)$  and  $q' = f(q)$ ;
        set  $\delta\Theta = \Theta(q) - \Theta(p)$ ;
        set  $\delta\Theta' = \Theta(q') - \Theta(p')$ ;
        if  $\delta\Theta' \neq \delta\Theta$ ,
            return "no" and halt;
    return "yes";

```

There are few ways to improve the efficiency of the above algorithm. One can order the cells on the lattice lexicographically. Namely, in this ordering, a cell $i = (i_1, i_2, \dots, i_d)$ precedes a cell $j = (j_1, j_2, \dots, j_d)$, written $i \prec j$, if there is a $1 \leq k \leq d$ such that $i_k < j_k$ and $i_l = j_l$ for all $1 \leq l < k$. Let $\diamond \in S$ be a fixed state. Using a more clever argument, one can restrict the main loop of the algorithm to all patterns $p: N(W) \rightarrow S$ for which $p[i] = \diamond$ for all cells $i \succeq 0$.

The set of all interaction potentials θ that are conserved by F is a linear space \mathcal{W}_F . It is interesting to note that, given a finite set $P \subseteq S^d$, one can use a variant of the above algorithm to construct the space $\mathcal{W}_F[P]$ of all those interaction potentials conserved by F that have support P . To do so, one considers $\theta: P \rightarrow \mathbb{R}$ as a vector of unknowns, and collects all the equations $\delta\Theta' = \delta\Theta$ given by the algorithm. The space $\mathcal{W}_F[P]$ is simply the solution space of this system of equations.

In Durand et al. (2003), a different approach has led to another efficient algorithm for verifying the validity of conservation laws.

As already mentioned at the beginning of this section, given a CA F , one can enumerate all the candidate energies θ and verify whether they are conserved by F . (Strictly speaking, one cannot enumerate all the energies, since the set of real-valued energies is uncountable. However, note that for each energy θ , the set of possible values that $\delta\Theta$ can take is a finitely generated subgroup of \mathbb{R} , and hence isomorphic to \mathbb{Z}^m for some $m > 0$. The discussion merely uses the group structure of the energy values. Therefore, one can equivalently work with energies whose values are in \mathbb{Z}^m . Such energies can indeed be enumerated.) Though every conservation law for F is eventually discovered by this algorithm, the algorithm never ends, and there is no way to predict whether, continuing to run the algorithm, one is going to find any new conservation law. In fact, it is undecidable whether a given CA has any nontrivial conservation law or not.

Proposition 4 *There is no algorithm that, given a cellular automaton, decides whether it has any nontrivial conservation law or not.*

This is an immediate consequence of the undecidability of the finite tiling problem. The finite tiling problem is a variant of the tiling problem, which asks whether one can tile the entire plane using decorated tiles chosen from a finite number of given types (see the chapter [Basic Concepts of Cellular Automata](#)).

A *Wang tile*, called after the Chinese–American mathematician Hao Wang (1921–1995), is a unit square with colored edges. Two Wang tiles can be placed next to each other if their abutting edges have the same color. A set of Wang tiles is given by a finite set T , and four mappings $n, w, s, e : T \rightarrow C$ that identify the colors of the northern, western, southern, and eastern edges of the tiles. A *valid tiling* of the plane is a configuration $c : \mathbb{Z}^2 \rightarrow T$ that respects the tiling rule; that is, $n(c[i, j]) = s(c[i, j + 1])$ and $w(c[i, j]) = e(c[i - 1, j])$ for every $i, j \in \mathbb{Z}$.

The finite tiling problem asks, given a set T of Wang tiles and a designated *blank* tile $\diamond \in T$, whether there is a nontrivial *finite* valid tiling of the plane; that is, a valid tiling in which all but a finite number of tiles are blank. Here a *nontrivial* tiling means a tiling that uses at least one non-blank tile. Reducing the halting problem of Turing machines to this problem, one can show that the finite tiling problem is undecidable.

Given a tile set T and a designated blank tile $\diamond \in T$, a two-dimensional CA F is constructed as follows: the state set of F is T . On every configuration $c : \mathbb{Z}^2 \rightarrow T$, the CA looks at each cell i and its four immediate neighbors. If there is a tiling error on cell i (i.e., if the tile on position i does not match with at least one of its adjacent tiles), the CA changes the state of i to blank. Otherwise, the state of cell i is kept unchanged.

Let \diamond be the uniformly blank configuration. If the tile set T admits no nontrivial finite valid tiling, every finite configuration of F is eventually turned into \diamond . Therefore, following the characterization (C-2), in this case F has no nontrivial conservation law. On the other hand, suppose that T admits a nontrivial finite valid tiling c . Let p be the finite pattern consisting of all the non-blank cells of c , as well as a margin of blank cells around them. Clearly, F preserves the number of the occurrences of p on any configuration c . In summary, T admits a nontrivial finite valid tiling if and only if F has a nontrivial conservation law. Since no algorithm can decide whether T admits a nontrivial finite valid tiling, no algorithm can either tell whether F has a nontrivial conservation law.

Note that the above argument does not rule out the existence of an algorithm that solves the problem only for one-dimensional cellular automata. Unfortunately, even when restricted to one-dimensional CA, the question remains undecidable.

Theorem 3 (Formenti et al. 2008) *There is no algorithm that, given a one-dimensional cellular automaton, decides whether it has any nontrivial conservation law or not.*

The proof of this fact relies on the undecidability of the existence of periodic orbits in 2-counter machines (Blondel et al. 2002). A 2-counter machine is a finite automaton equipped with two unbounded counters. At each step, the automaton can increase or decrease either of the counters and check whether its value is zero. It is well-known that 2-counter machines are equivalent, in power, with Turing machines (see, e.g., Minsky 1967); any algorithm can be implemented by a suitable 2-counter machine. As it is proven in Blondel (2002), there is no algorithm that decides whether a given 2-counter machine has a configuration whose orbit is periodic.

To prove [Theorem 3](#), one (algorithmically) transforms the problem of deciding whether a given 2-counter machine has a periodic configuration to the problem of deciding whether

a given CA has a nontrivial conservation law. Since any algorithm solving the latter problem would lead to an algorithm solving the former, one can conclude that no algorithm can solve the latter problem.

The idea of the transformation is simple: given a 2-counter machine M , one constructs a one-dimensional CA F with a distinguished quiescent configuration \diamond with the property that:

- (i) If M has no periodic orbit, F eventually transforms each finite configuration to the quiescent configuration \diamond .
- (ii) If M has a periodic orbit, F has a nontrivial conservation law.

Let \diamond be the state of the cells in \diamond . Every finite configuration in F is uniquely partitioned into disjoint segments. Each segment is either syntactically valid, in which case it simulates M on some configuration, or it is not syntactically valid, in which case the CA gradually turns all its cells into \diamond . The size of a valid segment, however, remains unchanged. So if at some point the simulation requires more cells than available in the segment, the segment overflows and becomes syntactically invalid.

Now, if the machine M does not have any periodic orbit, every valid segment eventually overflows. Hence, every finite configuration eventually changes into \diamond . On the other hand, if M does have a periodic configuration, a segment simulating M on such a configuration does never overflow. Therefore, one can construct a nontrivial energy that counts the number of such simulation segments, and that is conserved by F . See Formenti et al. (2008) and Taati (2009) for the details.

3.2 Dissipation Laws

Next, the problem of verifying whether a given energy is non-increasing under the iteration of a given CA have discussed. While there is an algorithm that solves the problem for one-dimensional CA (Kůrka 2003a) (see also Moreira et al. 2004, Bernardi et al. 2005, and Bernardi 2007), the problem is undecidable in higher dimensions (Bernardi et al. 2005).

One-dimensional CA have a convenient representation (up to composition with translations) using edge-labeled De Bruijn graphs. The *De Bruijn* graph of order k ($k > 0$) over an alphabet S is a graph $B_k[S]$ with vertex set $V \triangleq S^k$ and edge set $E \triangleq S^{k+1}$, where for every $a, b \in S$ and $u \in S^{k-1}$, there is an edge aub from au to ub .

Let $F: S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$ be a one-dimensional CA with neighborhood $[-l, r] = \{-l, -l+1, \dots, r\}$, and local rule $f: S^{[-l, r]} \rightarrow S$. For every $k \geq l+r$, the CA can be represented on the De Bruijn graph $B_k[S]$ with labeling $\lambda: E \rightarrow S^{k-(l+r)}$ that is defined as follows: for every edge $u_0 u_1 \dots u_k \in S^{k+1}$, let $\lambda(u_0 u_1 \dots u_k) = v_l v_{l+1} \dots v_{k-r}$, where $v_i = f(u_{i-l} u_{i-l+1} \dots u_{i+r})$.

The edge sequence $p = \{p[i]\}_{i \in \mathbb{Z}}$ of a bi-infinite walk on $B_k[S]$ represents a unique (up to translation) configuration $c: \mathbb{Z} \rightarrow S$, while its label sequence $\lambda(p) \triangleq \{\lambda(p[i])\}_{i \in \mathbb{Z}}$ represents Fc . Conversely, every configuration $c: \mathbb{Z} \rightarrow S$ corresponds to a unique bi-infinite walk on $B_k[S]$.

Consider now an energy defined in terms of a local observable $\mu: S^{\mathbb{Z}} \rightarrow \mathbb{R}$. Without loss of generality, one can assume that μ has observation window $[0, m] = \{0, 1, \dots, m\}$, for some $m \geq 0$, and local rule $g: S^{[0, m]} \rightarrow \mathbb{R}$. If $k \geq l+m+r$, one can also represent the energy μ on $B_k[S]$ in a suitable way. To each edge $u_0 u_1 \dots u_k \in S^{k+1}$, one can assign two real numbers

$$\alpha(u_0 u_1 \dots u_k) \triangleq g(u_0 u_1 \dots u_m) \quad (52)$$

and

$$\beta(u_0 u_1 \cdots u_k) \stackrel{\Delta}{=} g(v_l v_{l+1} \cdots v_{l+m}) \quad (53)$$

where $v_l v_{l+1} \cdots v_{k-r} = \lambda(u_0 u_1 \cdots u_k)$ is the label of $u_0 u_1 \cdots u_k$.

The average energy per cell of a periodic configuration c can be calculated by averaging the value of α over the edges of the closed walk p corresponding to c on the graph. The average of β over the same closed walk is the average energy per cell of the configuration Fc . Therefore, the energy μ is nonincreasing under F if and only if

$$\frac{\beta(p_1) + \beta(p_2) + \cdots + \beta(p_n)}{n} \leq \frac{\alpha(p_1) + \alpha(p_2) + \cdots + \alpha(p_n)}{n} \quad (54)$$

for every closed walk $p_1 p_2 \dots p_n$. It is not difficult to verify that in order to test the above property, one only needs to test [Eq. 54](#) for every p that is a cycle (i.e., a closed walk with no repeating vertices). Since the number of cycles on $B_k[S]$ is finite, this gives rise to the following simple algorithm for testing whether the energy μ is nonincreasing or not.

```
for every cycle  $p_1 p_2 \dots p_n$  on  $B_k[S]$ ,  
if  $\beta(p_1) + \beta(p_2) + \cdots + \beta(p_n) > \alpha(p_1) + \alpha(p_2) + \cdots + \alpha(p_n)$ ,  
    return "no" and halt;  
return "yes";
```

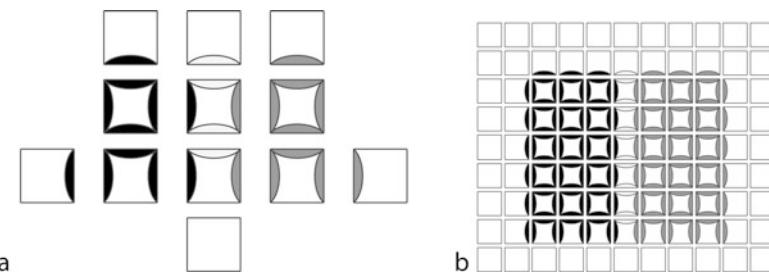
For higher-dimensional CA, there is no algorithmic way to verify whether a given energy is dissipative. Intuitively, this means that, for an energy to be non-increasing, cells that are very far from each other may need to collaborate. Any local increase in energy is compensated by a local decrease somewhere on the lattice, but there is no computable upper bound on the distance one may need to look to see such a decrease.

Theorem 4 (Bernardi et al. 2005) *There is no algorithm that, given a (two-dimensional) cellular automaton F and an energy θ , decides whether θ is nonincreasing under F .*

The proof is via a reduction from the finite tiling problem, which was introduced after [Proposition 4](#). Let (T, \diamond) be an instance of the finite tiling problem, where T is a set of Wang tiles and $\diamond \in T$ is the distinguished blank tile. First, the tile set P in [Fig. 6](#) is used to

Fig. 6

(a) Tile set P . **(b)** A typical finite valid tiling using tile set P .



construct a tile set $S \subseteq P \times T$. The tile set S consists of all tiles (p, t) satisfying the following two conditions:

- If $p \in \{\square, \square, \square, \square, \square\}$, then $t = \diamond$.
- If $p = \blacksquare$, then $t \neq \diamond$.

The color of an edge of tile (p, t) is simply the combination of the colors of that edge in the components p and t ; that is, $\alpha(p, t) \triangleq (\alpha(p), \alpha(t))$ for $\alpha = n, e, w, s$. The tile (\square, \diamond) is designated as the blank tile for S .

It is easy to see that S admits a nontrivial finite valid tiling if and only if T admits a nontrivial finite valid tiling. Also, note that every nontrivial finite valid tiling using the tiles in S contains at least one occurrence of \blacksquare . On the other hand, one can verify that if T (and hence S) does not admit a nontrivial finite valid tiling, on every finite configuration $c : \mathbb{Z}^2 \rightarrow S$, the number of occurrences of \blacksquare cannot exceed the number of tiling errors (i.e., those positions $i \in \mathbb{Z}^2$ where the tile on i does not match with at least one of its adjacent tiles).

Next, let a two-dimensional CA F and an interaction potential θ be constructed. The CA F has S as state set. On each configuration c , the CA looks at each cell i and its four immediate neighbors. If there is a tiling error on cell i , the CA changes the state of i to blank. On the other hand, if there is no tiling error on cell i , the CA keeps the state of i unchanged unless i has a state of the form (\blacksquare, t) with $t \in T$. In the latter case, the state of i is turned into an arbitrary value (p', t') with $p' \notin \{\square, \blacksquare\}$. The energy $\theta : S^\# \rightarrow \mathbb{R}$ is, in fact, context free; that is, it assigns nonzero potentials only to the single-cell patterns. For every state $s = (p, t)$ one defines

$$\theta(s) \triangleq \begin{cases} 0 & \text{if } (p, t) = (\square, \diamond) \\ 1 & \text{if } p = \blacksquare \\ 2 & \text{otherwise} \end{cases} \quad (55)$$

For an arbitrary finite configuration c , let $\gamma(c) \triangleq \Theta(Fc) - \Theta(c)$ denote the change in the total energy of c after one step. By construction, every tiling error contributes either -1 or -2 to γ , while every correctly tiled occurrence of \blacksquare contributes $+1$ to γ . If T does not admit any nontrivial finite valid tiling, the number of tiling errors on any configuration c is greater than or equal to the number of occurrences of \blacksquare on c . Therefore, for every configuration c , $\gamma(c) \leq 0$. On the other hand, suppose that T admits a nontrivial finite valid tiling. Then, S also admits a nontrivial finite valid tiling c . While there is no tiling error on c , there is at least one occurrence of \blacksquare . Hence, $\gamma(c) > 0$.

Since there is no general algorithm for deciding whether T admits a nontrivial finite valid tiling, it is concluded that no algorithm can decide whether θ is nonincreasing under F . See Bernardi et al. (2005) and Bernardi (2007).

4 Flows and Particles

A conservation law, as discussed in the previous sections, is a global property of a CA. It asserts that certain local additive quantity, which is called energy, is globally preserved. It does not, however, provide any microscopic mechanism behind this. Namely, it does not elaborate how the energy is manipulated locally so that its global quantity remains intact. Microscopic explanations of conservation laws can be given in terms of “flow” of energy from one cell to another. In fact, every conservation law in cellular automata has such flow explanations, as it was first noted in Hattori and Takesue (1991). However, these flows are not unique.

4.1 Local Conservation Laws

In physics, every local and additive conserved quantity (such as energy, momentum, electric charge, etc.) is *locally* conserved. In fact, any conservation law in physics must inevitably be expressible in a local form, in order to be compatible with the principle of relativity (cf. Feynman 1965).

In cellular automata, local conservation laws are formalized in terms of the concept of flow. The amount of *flow* from cell i to cell j on a configuration x is specified by a real number $\Phi_{i \rightarrow j}(x)$. The mapping $x, i, j \mapsto \Phi_{i \rightarrow j}(x) \in \mathbb{R}$ is required to satisfy the following natural conditions:

- (i) For every $i, j \in \mathbb{Z}^d$, the mapping $x \mapsto \Phi_{i \rightarrow j}(x)$ is a local observable.
- (ii) For every configuration x , all cells $i, j \in \mathbb{Z}^d$, and any displacement $a \in \mathbb{Z}^d$,

$$\Phi_{a+i \rightarrow a+j}(x) = \Phi_{i \rightarrow j}(\sigma^a x) \quad (56)$$

- (iii) There is a finite set $I \subseteq \mathbb{Z}^d$ such that $\Phi_{i \rightarrow j} = 0$ unless $i - j \in I$.

Equivalently, a mapping $x, i, j \mapsto \Phi_{i \rightarrow j}(x)$ is called a flow if there exist finite sets $K, I \subseteq \mathbb{Z}^d$ and a rule $\varphi : S^K \times I \rightarrow \mathbb{R}$ such that

$$\Phi_{i \rightarrow j}(x) = \begin{cases} \varphi(x[K(j)], i - j) & \text{if } i - j \in I \\ 0 & \text{otherwise} \end{cases} \quad (57)$$

for every $x : \mathbb{Z}^d \rightarrow S$ and $i, j \in \mathbb{Z}^d$. In summary, the amount of flow to each cell is decided locally, by looking at a finite neighborhood K of that cell. The set I is the set of directions from which energy flows to a cell.

Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a CA and $\mu : S^{\mathbb{Z}^d} \rightarrow \mathbb{R}$ a local observable defining an energy. A flow Φ is said to be *compatible* with μ and F (or, Φ is a flow for μ under the dynamics of F) if the following *continuity equations* hold (see Fig. 7a):

- (a) For every configuration x and every cell a ,

$$\mu(\sigma^a x) = \sum_{j \in \mathbb{Z}^d} \Phi_{a \rightarrow j}(x) \quad (58)$$

- (b) For every configuration x and every cell a ,

$$\sum_{i \in \mathbb{Z}^d} \Phi_{i \rightarrow a}(x) = \mu(\sigma^a F x) \quad (59)$$

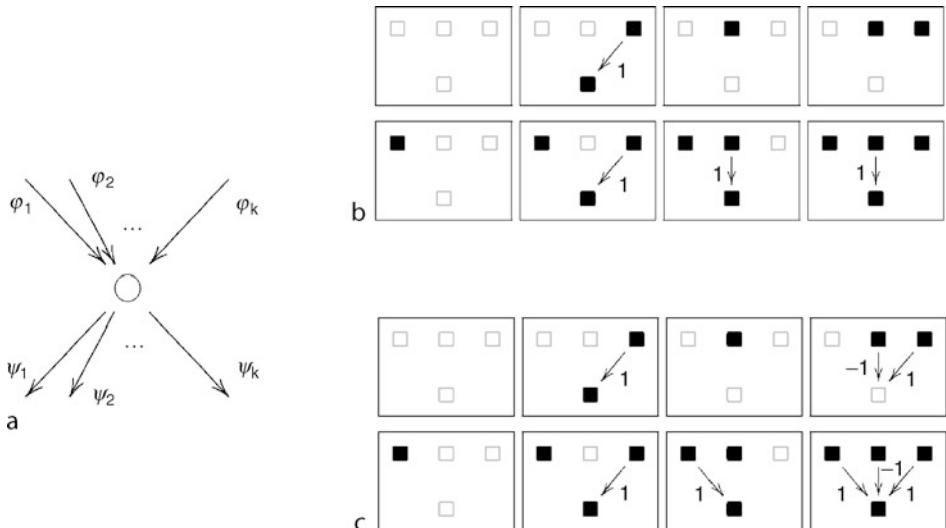
An energy μ is *locally conserved* by F if it has a flow under F . Conservation laws and local conservation laws are equivalent concepts in cellular automata (Hattori and Takesue 1991):

Theorem 5 *Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a cellular automaton and $\mu : S^{\mathbb{Z}^d} \rightarrow \mathbb{R}$ a local observable. There is a flow Φ compatible with μ and F if and only if F conserves the energy generated by μ .*

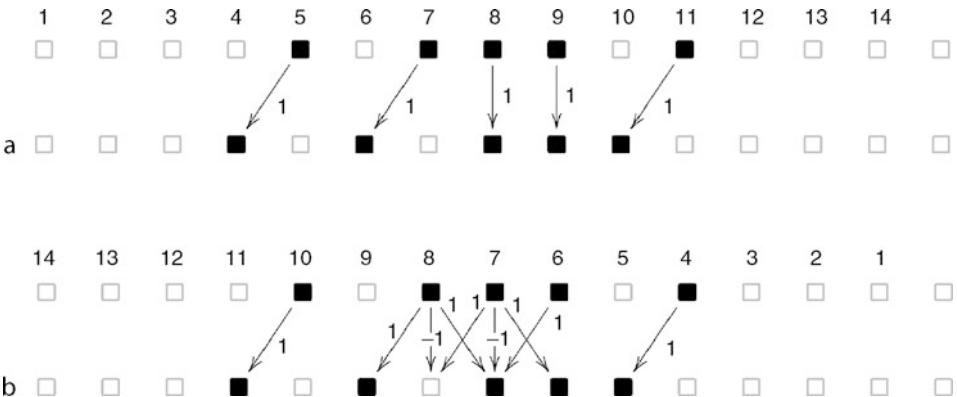
That the existence of a flow compatible with μ and F implies that F conserves μ is easy to see. To prove the converse, one needs to construct, for any CA F and any energy μ conserved by F , a flow Φ compatible with μ and F . One such construction can be found, for example, in Taati (2009). The idea is now illustrated by an example, namely, the car conservation law in the Traffic CA.

Fig. 7

(a) Continuity of the flow: $\sum_i \varphi_i = \mu = \sum_j \psi_j$. (b, c) Two different flows for the car conservation law in the Traffic CA.

**Fig. 8**

The identified effect of each car on the configuration after one step. (a) Effects identified from left to right. (b) Effects identified from right to left.



Let x be an arbitrary configuration consisting of a finite number of cars on the highway. Start from the empty highway and place the cars, one by one from left to right, on the highway to obtain x . At each step, identify the effect of placing one new car on position i on the configuration of the highway after one time step. This effect is expressed by flows from cell i to the neighboring cells (see Fig. 8a). Note that placing a new car on the cell i may only change the state of the cells i and $i - 1$ in the following configuration. Furthermore, these changes depend only on whether there has already been a car on the cell $i - 1$ or not. Therefore, the

effect of each car can be identified locally as depicted in [Fig. 7b](#). However, note that this is not the only way to identify the effect of each car. For example, if the cars are placed from right to left, a different flow is obtained as illustrated in [Figs. 8b](#) and [7c](#).

Even though flows provide intuitive ways to think about conservation laws, their nonuniqueness is not plausible. Hence, one may want to identify a flow that is the most natural in some sense.

4.2 Particle Flows

Consider the following game played with pebbles on configurations of a CA. Let $x : \mathbb{Z}^d \rightarrow S$ be an arbitrary configuration. On each cell of the lattice, a number of pebbles have been placed. The number of pebbles on cell i depends only on the state of cell i and is denoted by $\eta(x[i])$. At each iteration of the CA, the state of each cell i changes to its new value $(Fx)[i]$. The goal is to redistribute the pebbles on the lattice so that the number of pebbles on each cell matches with its new state; that is, each cell i obtains $\eta((Fx)[i])$ pebbles. Is there a local and uniform strategy to do this? “Local” means that each pebble can be moved within a bounded distance of its original position, and one is only allowed to look at the states of a bounded number of cells around it in order to decide where a pebble should be moved. “Uniform” means that the same strategy should be used for redistributing the pebbles on every cell and over every configuration.

The assignment $\eta : S \rightarrow \mathbb{N}$ ($\mathbb{N} \triangleq \{0, 1, 2, \dots\}$) defines an energy in its usual sense. A pebble can be interpreted as the “quantum” of energy; that is, the tiniest bit of energy, which is indecomposable. The desired strategy for the game is simply a flow Φ that takes its values in the set \mathbb{N} of nonnegative integers, and that is compatible with η and F . For example, the flow depicted in [Figs. 7b](#) and [8a](#) defines a valid strategy, while the flow in [Figs. 7c](#) and [8b](#) does not.

Let a flow Φ be called a *particle flow* if its values are from the set of nonnegative integers. Any function $\eta : S \rightarrow \mathbb{N}$ is called a *particle assignment*. A necessary condition for the existence of particle flow compatible with a particle assignment η and a CA F is, of course, that F conserves the energy generated by η . For one-dimensional CA, this condition is also known to be sufficient; Fukś (2000) and Pivato (2002) have shown that any particle assignment η conserved by a one-dimensional CA F has a particle flow (see also Moreira et al. 2004). For higher-dimensional CA, however, the question is open. See Kari and Taati (2008) for a partial solution in two dimensions.

Theorem 6 (Fukś 2000, Pivato 2002) *Let $F : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$ be a one-dimensional cellular automaton and $\eta : S \rightarrow \mathbb{N}$ a particle assignment. There is a particle flow Φ compatible with η and F if and only if F conserves η .*

Theorem 7 (Kari and Taati 2008) *Let $F : S^{\mathbb{Z}^2} \rightarrow S^{\mathbb{Z}^2}$ be a two-dimensional cellular automaton with radius- $\frac{1}{2}$ neighborhood, and let $\eta : S \rightarrow \mathbb{N}$ be a particle assignment. There is a particle flow Φ compatible with η and F if and only if F conserves η .*

The radius- $\frac{1}{2}$ neighborhood refers to the four-element neighborhood

$$\{(0, 0), (0, 1), (1, 0), (1, 1)\} \quad (60)$$

Particle flows, if exist, are not unique; a particle flow compatible with a particle assignment and a CA can be modified in infinitely many ways to obtain new particle flows compatible with the same assignment and CA. In one dimension, however, there are natural criteria that ensure the uniqueness. For example, it is shown in Moreira et al. (2004) that for each particle assignment conserved by a one-dimensional CA, there is a unique particle flow that preserves the order of the particles.

An argument, due to Pivato (2002), is now mentioned that strongly suggests that **Theorem 6** should be valid also for higher-dimensional CA.

Theorem 8 (Pivato 2002) *Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a cellular automaton with neighborhood $0 \in N \subseteq \mathbb{Z}^d$. A particle assignment $\eta : S \rightarrow \mathbb{N}$ is conserved by F if and only if*

$$\sum_{i \in A} \eta(x[i]) \leq \sum_{i \in N^{-1}(A)} \eta((Fx)[i]) \quad (61)$$

and

$$\sum_{i \in A} \eta((Fx)[i]) \leq \sum_{i \in N(A)} \eta(x[i]) \quad (62)$$

for every configuration x and every finite set $A \subseteq \mathbb{Z}^d$.

For every two consecutive configurations, x and $y = Fx$, a bipartite graph $G_N[\eta, x, y] = (U, V, E)$ is constructed as follows. For every particle on x , the graph has a vertex in U , and for every particle on y , there is a vertex in V . There is an edge between a particle $u \in U$ coming from a cell i and a particle $v \in V$ coming from a cell j if and only if i is a neighbor of j ; that is, if and only if $i - j \in N$.

A perfect matching in graph $G_N[\eta, x, y]$ is a way of identifying particles on x with particles on y in such a way that the position of each particle on x is a neighbor of its position on y . A necessary and sufficient condition for a (possibly infinite, but locally finite) bipartite graph to have a perfect matching is given by Hall's Marriage Theorem (see, e.g., van Lint and Wilson 1992): a bipartite graph $G = (U, V, E)$ has a matching that covers U if and only if for every finite set $A \subseteq U$, the number of vertices in V that are adjacent to A is at least $|A|$. If G has a matching that covers U and a matching that covers V , G also has a perfect matching.

It follows from **Theorem 8** that if a CA F with neighborhood N conserves a particle assignment η , then for every configuration x , the graph $G_N[\eta, x, Fx]$ must have a perfect matching. Therefore, if F conserves η , the particles on any configuration x can be identified with those on Fx in such a way that the two positions of each particle are within bounded distance from each other. It remains open whether, in general, such identification can be done in a local and uniform fashion.

4.3 Flows for Dissipative Energies

In the light of **Theorem 5**, it is natural to ask whether nonincreasing energies can also be explained locally. A flow explanation of a nonincreasing energy may be possible by relaxing the

continuity equations (❷ Eqs. 58 and ❷ 59). In one dimension, this is always possible. In higher dimensions, however, the undecidability result of ❷ Theorem 4 imposes an obstacle to the general existence of such flows.

Let $F : S^{\mathbb{Z}^d} \rightarrow S^{\mathbb{Z}^d}$ be a CA and $\mu : S^{\mathbb{Z}^d} \rightarrow \mathbb{R}$ a local observable defining an energy. A flow Φ is said to be *semi-compatible* with μ and F if

(a) For every configuration x and every cell a ,

$$\mu(\sigma^a x) \geq \sum_{j \in \mathbb{Z}^d} \Phi_{a \rightarrow j}(x) \quad (63)$$

(b) For every configuration x and every cell a ,

$$\sum_{i \in \mathbb{Z}^d} \Phi_{i \rightarrow a}(x) \geq \mu(\sigma^a Fx) \quad (64)$$

Proposition 5 *There exists a two-dimensional cellular automaton $F : S^{\mathbb{Z}^2} \rightarrow S^{\mathbb{Z}^2}$ and a local observable $\mu : S^{\mathbb{Z}^2} \rightarrow \mathbb{R}$ such that the energy generated by μ is non-increasing under F , but there is no flow semi-compatible with μ and F .*

As mentioned above, this is a consequence of ❷ Theorem 4. First, note that there is a semi-algorithm (a semi-algorithm means an algorithmic process that does not necessarily halt on every input) that recognizes those energies that are not nonincreasing: given an energy μ and a CA F , such a semi-algorithm tests the inequality $\bar{\mu}(Fx) \leq \bar{\mu}(x)$ for periodic configurations x with larger and larger periods. According to the characterization (D-2), if μ is not nonincreasing, the inequality fails on some periodic configuration, which will eventually be found. Now, suppose that every nonincreasing energy μ has a semi-compatible flow. Then, one can construct a semi-algorithm that recognizes those energies that are nonincreasing: given an energy μ and a CA F , one simply enumerates all the possible flows and verifies ❷ Eqs. 63 and ❷ 64 for them one by one. If μ is nonincreasing, the process will eventually find a semi-compatible flow. Running these two semi-algorithms in parallel one obtains an algorithm for deciding whether a given energy is nonincreasing; hence contradicting ❷ Theorem 4.

The above argument does not provide any explicit example of a nonincreasing energy with no semi-compatible flow. An explicit construction has been obtained by A. Rumyantsev (see Bernardi 2007).

In one-dimensional case, every nonincreasing energy has a flow explanation.

Theorem 9 *Let $F : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$ be a one-dimensional cellular automaton and $\mu : S^{\mathbb{Z}} \rightarrow \mathbb{R}$ a local observable. There is a flow Φ semi-compatible with μ and F if and only if the energy generated by μ is nonincreasing under F .*

A proof in a rather different setting is given in the chapter ❷ Cellular Automata Dynamical Systems. In one dimension, in addition, any nonincreasing energy defined using a particle assignment has a semi-compatible particle flow.

Theorem 10 (Moreira et al. 2004) *Let $F : S^{\mathbb{Z}} \rightarrow S^{\mathbb{Z}}$ be a one-dimensional cellular automaton and $\eta : S \rightarrow \mathbb{N}$ a particle assignment. If η is nonincreasing under F , there is a particle flow Φ semi-compatible with η and F .*

5 Further Topics

This chapter concludes with some miscellaneous remarks.

A large body of research done on conservation laws is concentrated on number-conserving automata (see, e.g., Boccara and Fukš 1998, 2002, 2006; Fukš 2000; Durand et al. 2003; Formenti and Grange 2003; Moreira et al. 2004; Bernardi et al. 2005; Fukš and Sullivan 2007). In a *number-conserving* CA, the state of a cell represents the number of particles in that cell, and the dynamics is such that the total number of particles is preserved. Most results on number-conserving CA have counterparts in the general setting of conservation laws, which can be proven in more or less similar manners.

There are, however, results that are specific to the framework of number-conserving CA. For example, Moreira (2003) has shown that every one-dimensional CA can be simulated by a number-conserving one. In particular, this implies that there are number-conserving one-dimensional CA that are intrinsically universal (cf. the chapter [Reversible Cellular Automata](#)). Morita and Imai have constructed computationally universal reversible CA that conserve a natural quantity associated to each cell (Morita and Imai 2001).

An interesting issue, which needs further investigation, is the possible connections between the conservation laws of a CA and its dynamical properties. Specifically, it is interesting to know what kind of restrictions a conservation law may impose on the dynamics of a CA? Dynamical properties of number-conserving CA are studied in Formenti and Grange (2003). There, among other things, it has been proved that in every surjective number-conserving CA, the set of temporally periodic configurations is dense. The same holds for any surjective one-dimensional CA that has a conserved energy with a unique ground configuration (Formenti et al.; Taati 2009). A ground configuration for an energy θ is a configuration x such that $\delta\theta(x, y) \geq 0$ for any configuration y asymptotic to x . There is a long-standing open question, asking whether every surjective CA has a dense set of temporally periodic configurations (see, e.g., Kari 2005 and Boyle 2008). It is also shown that positively expansive CA cannot have any nontrivial conservation laws (Formenti et al.; Taati 2009). See the chapter [Cellular Automata Dynamical Systems](#) or Kůrka (2001, 2003b) for information on cellular automata as dynamical systems.

Nonincreasing energies have a more established connection with the dynamics, as they define Lyapunov functions for the measure dynamical system (\mathcal{M}_σ, F) . See the chapter [Cellular Automata Dynamical Systems](#) for more on this point of view.

Another trend of research has been to find counterparts of Noether's theorem for cellular automata (e.g., Margolus 1984; Baez and Gilliam 1994; Boykett 2005; Boykett et al. 2008; personal communication with N. Ollinger). Noether's theorem – after the German mathematician Emmy Noether (1882–1935) – establishes a correspondence between the conservation laws of a dynamical system defined in terms of a Hamiltonian or a Lagrangian and its symmetries (see, e.g., Landau and Lifshitz 1976 and Arnold 1989).

Finally, the interesting connection between conservation laws and invariant Gibbs measures in reversible cellular automata is worth mentioning. Gibbs measures are especially important in statistical mechanics, as they characterize the equilibrium state of lattice Hamiltonian systems (see, e.g., Georgii 1988 and Ruelle 2004). In fact, early study of conservation laws in cellular automata was motivated by the issue of ergodicity in statistical mechanical systems (Takesue 1987).

In reversible cellular automata, there is a one-to-one correspondence between conservation laws and invariant Gibbsian specifications (Kari and Taati; Taati 2009). As a special case, a reversible CA conserves a context-free energy $\eta : S \rightarrow \mathbb{R}$ if and only if it preserves the Bernoulli

measure with probabilities proportional to 2^{-n} . This is at least partially valid also for surjective CA. Similar results, though in different settings, are obtained in Helvik et al. (2007) (see also Toffoli 1988). It is an interesting open issue to investigate the questions of statistical mechanics in the framework of (deterministic) reversible cellular automata.

References

- Anderson R, Lovász L, Shor P, Spencer J, Tardos E, Winograd S (1989) Disks, balls, and walls: analysis of a combinatorial game. *Am Math Mon* 96(6): 481–493
- Arnold VI (1989) Mathematical methods of classical mechanics, 2nd edn. Springer, New York. English Translation
- Baez JC, Gilliam J (1994) An algebraic approach to discrete mechanics. *Lett Math Phys* 31:205–212
- Bak P, Tang C, Wiesenfeld K (1987) Self-organized criticality: an explanation of 1/f noise. *Phys Rev Lett* 59(4):381–384
- Bak P, Tang C, Wiesenfeld K (1998) Self-organized criticality. *Phys Rev A* 38(1):364–374
- Bernardi V (2007) Lois de conservation sur automates cellulaires. Ph.D. thesis, Université de Provence
- Bernardi V, Durand B, Formenti E, Kari J (2005) A new dimension sensitive property for cellular automata. *Theor Comp Sci* 345:235–247
- Blondel VD, Cassaigne J, Nichitiu C (2002) On the presence of periodic configurations in Turing machines and in counter machines. *Theor Comput Sci* 289:573–590
- Boccara N, Fukáš H (1998) Cellular automaton rules conserving the number of active sites. *J Phys A Math Gen* 31(28):6007–6018
- Boccara N, Fukáš H (2002) Number-conserving cellular automaton rules. *Fundam Info* 52:1–13
- Boccara N, Fukáš H (2006) Motion representation of one-dimensional cellular automata rules. *Int J Mod Phys C* 17:1605–1611
- Boykett T (2005) Towards a Noether-like conservation law theorem for one dimensional reversible cellular automata. arXiv:nlin.CG/0312003
- Boykett T, Moore C (1998) Conserved quantities in one-dimensional cellular automata. Unpublished manuscript
- Boykett T, Kari J, Taati S (2008) Conservation laws in rectangular CA. *J Cell Automata* 3(2):115–122
- Boyle M (2008) Open problems in symbolic dynamics. In: Burns K, Dolgopyat D, Pesin Y (eds) Geometric and probabilistic structures in dynamics, contemporary mathematics. American Mathematical Society, Providence, RI
- Chopard B, Droz M (1998) Cellular automata modeling of physical systems. Cambridge University Press, Cambridge
- Durand B, Formenti E, Róka Z (2003) Number conserving cellular automata I: decidability. *Theor Comput Sci* 299:523–535
- Feynman R (1965) The character of physical law. MIT Press, Cambridge, MA
- Formenti E, Grange A (2003) Number conserving cellular automata II: dynamics. *Theor Comput Sci* 304:269–290
- Formenti E, Kari J, Taati S (2008) The most general conservation law for a cellular automaton. In: Hirsch EA, Razborov AA, Semenov A, Slissenko A (eds) Proceedings of CSR 2008, Moscow, June 7–12, 2008. LNCS, vol 5010. Springer, Berlin/Heidelberg, pp 194–203, doi: 10.1007/978-3-540-79709-8
- Formenti E, Kari J, Taati S Preprint
- Fredkin E, Toffoli T (1982) Conservative logic. *Int J Theor Phys* 21:219–253
- Fukáš H (2000) A class of cellular automata equivalent to deterministic particle systems. In: Feng S, Lawniczak AT, Varadhan SRS (eds) Hydrodynamic limits and related topics. Fields Institute communications, vol 27. American Mathematical Society, Providence, RI, pp 57–69
- Fukáš H, Sullivan K (2007) Enumeration of number-conserving cellular automata rules with two inputs. *J Cell Automata* 2:141–148
- Georgii HO (1988) Gibbs measures and phase transitions. Walter de Gruyter, Berlin
- Georgii HO, Häggström O, Maes C (2000) The random geometry of equilibrium phases. In: Domb C, Lebowitz J (eds) Phase transitions and critical phenomena, vol 18. Academic Press, London, pp 1–42
- Goles E (1992) Sand pile automata. *Ann Institut Henri Poincaré (A) Phys théor* 59(1):75–90
- Goles E, Kiwi MA (1993) Games on line graphs and sand piles. *Theor Comput Sci* 115:321–349
- Greenberg JM, Hassard BD, Hastings SP (1978a) Pattern formation and periodic structures in systems modeled by reaction-diffusion equations. *Bull Am Math Soc* 84(6):1296–1327
- Greenberg JM, Hastings SP (1978b) Spatial patterns for discrete models of diffusion in excitable media. *SIAM J Appl Math* 34(3):515–523
- Greenberg J, Greene C, Hastings S (1980) A combinatorial problem arising in the study of reaction-diffusion equations. *SIAM J Algebraic Discrete Meth* 1(1):34–42

- Hattori T, Takesue S (1991) Additive conserved quantities in discrete-time lattice dynamical systems. *Phys D* 49:295–322
- Hedlund GA (1969) Endomorphisms and automorphisms of the shift dynamical system. *Math Syst Theory* 3:320–375
- Helvik T, Lindgren K, Nordahl MG (2007) Continuity of information transport in surjective cellular automata. *Commun Math Phys* 272:53–74
- Kari J (2005) Theory of cellular automata: a survey. *Theor Comput Sci* 334:3–33
- Kari J, Taati S In preparation
- Kari J, Taati S (2008) A particle displacement representation for conservation laws in two-dimensional cellular automata. In: Durand B (ed) *Proceedings of JAC 2008*. MCCME Publishing House, Moscow, pp 65–73
- Kindermann R, Snell JL (1980) Markov random fields and their applications. American Mathematical Society, Providence, RI
- Kůrka P (2001) Topological dynamics of cellular automata. In: *Codes, systems and graphical models*. Springer, Berlin
- Kůrka P (2003a) Cellular automata with vanishing particles. *Fundam Info* 58:1–19
- Kůrka P (2003b) Topological and symbolic dynamics. Cours spécialisés, vol. 11. Société Mathématique de France, Paris
- Landau LD, Lifshitz EM (1976) Course of theoretical physics: mechanics, 3rd edn. Butterworth-Heinemann, Oxford. English Translation
- van Lint JH, Wilson RM (1992) A course in combinatorics. Cambridge University Press, Cambridge
- Margolus N (1984) Physics-like models of computation. *Phys D* 10:81–95
- Margolus N (1987) Physics and computation. Ph.D. thesis, Massachusetts Institute of Technology
- Minsky M (1967) Computation: finite and infinite machines. Prentice-Hall, Englewood Cliffs, NJ
- Moreira A (2003) Universality and decidability of number-conserving cellular automata. *Theor Comput Sci* 292:711–723
- Moreira A, Boccara N, Goles E (2004) On conservative and monotone one-dimensional cellular automata and their particle representation. *Theor Comput Sci* 325:285–316
- Morita K, Imai K (2001) Number-conserving reversible cellular automata and their computation-universality. *Theor Inf Appl* 35:239–258
- Nagel K, Herrmann HJ (1993) Deterministic models for traffic jams. *Phys A* 199:254–269
- Nagel K, Schreckenberg M (1992) A cellular automaton model for freeway traffic. *J Phys I* 2(12):2221–2229
- Partha Sarathy KR (1967) Probability measures on metric spaces. Academic Press, New York
- Pivato M (2002) Conservation laws in cellular automata. *Nonlinearity* 15:1781–1793
- Pomeau Y (1984) Invariant in cellular automata. *J Phys A Math Gen* 17(8):L415–L418
- Ruelle D (2004) Thermodynamic formalism, 2nd edn. Cambridge University Press, Cambridge
- Sinai YG (1982) Theory of phase transitions: rigorous results. Pergamon Press, Oxford
- Taati S (2009) Conservation laws in cellular automata. Ph.D. thesis, University of Turku
- Takesue S (1987) Reversible cellular automata and statistical mechanics. *Phys Rev Lett* 59(22):2499–2502
- Toffoli T (1988) Information transport obeying the continuity equation. *IBM J Res Dev* 32(1):29–36
- Toffoli T, Margolus N (1987) Cellular automata machines: a new environment for modeling. MIT Press, Cambridge
- Toffoli T, Margolus N (1990) Invertible cellular automata: a review. *Phys D* 45:229–253
- Vichniac GY (1984) Simulating physics with cellular automata. *Phys D* 10:96–116
- Walters P (1982) An introduction to ergodic theory. Springer, New York

9 Cellular Automata and Lattice Boltzmann Modeling of Physical Systems

Bastien Chopard

Scientific and Parallel Computing Group,
University of Geneva,
Switzerland

bastien.chopard@unige.ch

1	<i>Introduction</i>	288
2	<i>Definition of a Cellular Automata</i>	289
3	<i>Cellular Automata and Complex Systems</i>	293
4	<i>A Simple Model for a Gas of Particles</i>	307
5	<i>Lattice Boltzmann Models</i>	311
6	<i>Conclusions</i>	328

Abstract

Cellular automata (CA) and lattice Boltzmann (LB) methods provide a natural modeling framework to describe and study many physical systems composed of interacting components. The reason for this success is the close relation between these methods and a mesoscopic abstraction of many natural phenomena. The theoretical basis of the CA and LB approaches are introduced and their potential is illustrated for several applications in physics, biophysics, environmental sciences, traffic models, and multiscale modeling.

1 Introduction

As one can observe everyday, nature is made up of a large number of interacting parts, distributed over space and evolving in time. In many cases, one is interested in describing a natural system at a scale which is much larger than its elementary constituents. Then, often, the behavior of these parts can be reduced to a set of rather simple rules, without affecting the behavior of the whole. For instance, when modeling a fluid flow at a macroscopic scale, one does not have to account for the detailed microscopic interactions between the atoms of that fluid. Instead, one can assume the existence of abstract “fluid elements” interacting in such a way as to conserve mass and momentum.

Cellular automata (CA) can be thought of as a mathematical abstraction of the physical world, an abstraction in which time is discrete and space is made of little blocks, or cells. These cells are organized as a regular lattice, in such a way as to fully cover the spatial domain of interest. In such an approach, spatiotemporal physical quantities are introduced as numerical values associated with each cell. These quantities are called the *states* of the cells. Formally, the definition of a CA also assumes that these states can only take a finite number of discrete values. On the other hand, lattice Boltzmann (LB) methods are more flexible and allow cells to have real-valued quantities.

CA were proposed in the late 1940s by von Neumann and Ulam (Burks 1970) as an abstraction of a biological system, in order to study the algorithmic mechanisms leading to the self-reproduction of living organisms. Since then, the CA approach has been applied to a wide range of problems and is an acknowledged modeling technique (Toffoli and Margolus 1987; Wolfram 1994; Rothman and Zaleski 1997; Chopard and Droz 1998; Deutsch and Dormann 2005; Ilachinski 2001; Gaylord and Nishidate 1996; Weimar 1998; Wolfram 2002). Moreover, it is still a quite active field of research: international conferences (e.g., ACRI) and dedicated journals (e.g., Journal of Cellular Automata) describe current developments.

LB methods developed later, in the 1990s, are an extension of the concept of lattice gas automata (LGA), a family of CA designed to describe hydrodynamic processes in terms of a discrete kinetic model. LB is now recognized as an alternative way to simulate on a computer complex fluid flows or other complex systems such as reaction–diffusion and advection–diffusion phenomena, as well as wave propagation in complicated geometries (Chopard and Droz 1998; Wolf-Gladrow 2000; Succi 2001; Sukop and Thorne 2005). LB methods are also an important research topic and two international conferences (DSFD and ICMMES) are places to disseminate new results in this field.

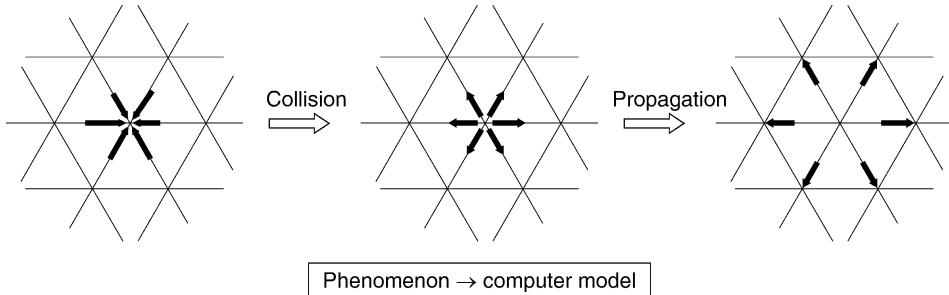
A key conceptual ingredient of CA is that they are not meant to be a space–time discretization of the partial differential equation (PDE) representing a given physical process. Instead, CA implement a mesoscopic model of this process, in terms of behavioral rules

Fig. 1

The solution process, from problem to numerical results for either a PDE (top) or a CA (bottom) approach. In this figure, fluid dynamics is used to illustrate the point. In an appropriate limit, fluid dynamics can be either described by the Navier–Stokes equations or by a discrete model of interacting particles. This corresponds to two different languages to represent the same physical problem.

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}$$

Phenomenon → PDE → discretization → computer solution



mimicking the physical interactions and translating them into a fully discrete universe. Concretely, CA evolution rules are transition functions, which synchronously change the state of each cell according to its value and those of the adjacent cells.

In other words, CA are based on an idealized, virtual microscopic version of the real world. Statistical physics teaches that the macroscopic behavior of many systems depends very little on the details of the microscopic interactions between the elementary constituents. This suggests that, in view of modeling efficiency, one can consider a new, fictitious microscopic universe whose numerical implementation is easy and fast, as long as this fictitious system has the same macroscopic behavior as the real one. The recipe to achieve this goal is to build a model with the right conservation laws and symmetries. These properties are indeed those that are preserved at all scales of description.

The above principles make the design of CA models quite intuitive and natural. Rule-based interactions are often easier to understand and discuss than a PDE, especially for researchers outside mathematics or physics. Since the level of description of a CA model is mesoscopic, the rules are close to the underlying physical interaction and it is rather easy to add new features to a model and to describe systems for which no PDE apply. On the other hand, CA and LB models heavily rely on computer simulations to derive results.

The diagram in [Fig. 1](#) sketches the solution process of a CA–LB approach compared to that of the more traditional PDE approach.

2 Definition of a Cellular Automata

In order to give a definition of a cellular automaton, a simple example is first presented. Although it is very basic, the rule discussed here exhibits a surprisingly rich behavior. It was

proposed initially by Edward Fredkin in the 1970s (Banks 1971) and is defined on a two-dimensional square lattice.

Each site of the lattice is a cell, which is labeled by its position $\mathbf{r} = (i, j)$ where i and j are the row and column indices. A function $\psi(\mathbf{r}, t)$ is associated with the lattice to describe the state of each cell \mathbf{r} at iteration t . This quantity ψ can be either 0 or 1.

The CA rule specifies how the states $\psi(\mathbf{r}, t + 1)$ are to be computed from the states at iteration t . We start from an initial condition at time $t = 0$ with a given configuration of the values $\psi(\mathbf{r}, t = 0)$ on the lattice. The state at time $t = 1$ is obtained as follows

1. Each site \mathbf{r} computes the sum of the values $\psi(\mathbf{r}', 0)$ on the four nearest neighbor sites \mathbf{r}' at north, west, south, and east. The system is supposed to be periodic in both i and j directions (like on a torus) so that this calculation is well defined for all sites.
2. If this sum is even, the new state $\psi(\mathbf{r}, t = 1)$ is 0 (white) and, else, it is 1 (black).

The same rule (steps 1 and 2) is repeated to find the states at time $t = 2, 3, 4, \dots$

From a mathematical point of view, this *parity rule* can be expressed by the following relation

$$\psi(i, j, t + 1) = \psi(i + 1, j, t) \oplus \psi(i - 1, j, t) \oplus \psi(i, j + 1, t) \oplus \psi(i, j - 1, t) \quad (1)$$

where the symbol \oplus stands for the exclusive OR logical operation. It is also the sum modulo 2: $1 \oplus 1 = 0 \oplus 0 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$.

When this rule is iterated, very nice geometric patterns are observed, as shown in [Fig. 2](#). This property of producing complex patterns starting from a simple rule is generic of many CA rules. Here, complexity results from some spatial organization, which builds up as the rule is iterated. The various contributions of successive iterations combine together in a specific way. The spatial patterns that are observed reflect how the terms are combined algebraically.

Based on this example a definition of a cellular automata is now given. Formally, a cellular automata is a tuple $(A, \Psi, R, \mathcal{N})$ where

- i. A is a regular lattice of cells covering a portion of a d -dimensional space.
- ii. $\Psi(\mathbf{r}, t) = \{\Psi_1(\mathbf{r}, t), \Psi_2(\mathbf{r}, t), \dots, \Psi_m(\mathbf{r}, t)\}$ is a set of m Boolean variables attached to each site \mathbf{r} of the lattice and giving the local state of the cells at time t .
- iii. R is a set of rules, $R = \{R_1, R_2, \dots, R_m\}$, which specifies the time evolution of the states $\Psi(\mathbf{r}, t)$ in the following way

$$\Psi_j(\mathbf{r}, t + \delta_t) = R_j(\Psi(\mathbf{r}, t), \Psi(\mathbf{r} + \mathbf{v}_1, t), \Psi(\mathbf{r} + \mathbf{v}_2, t), \dots, \Psi(\mathbf{r} + \mathbf{v}_q, t)) \quad (2)$$

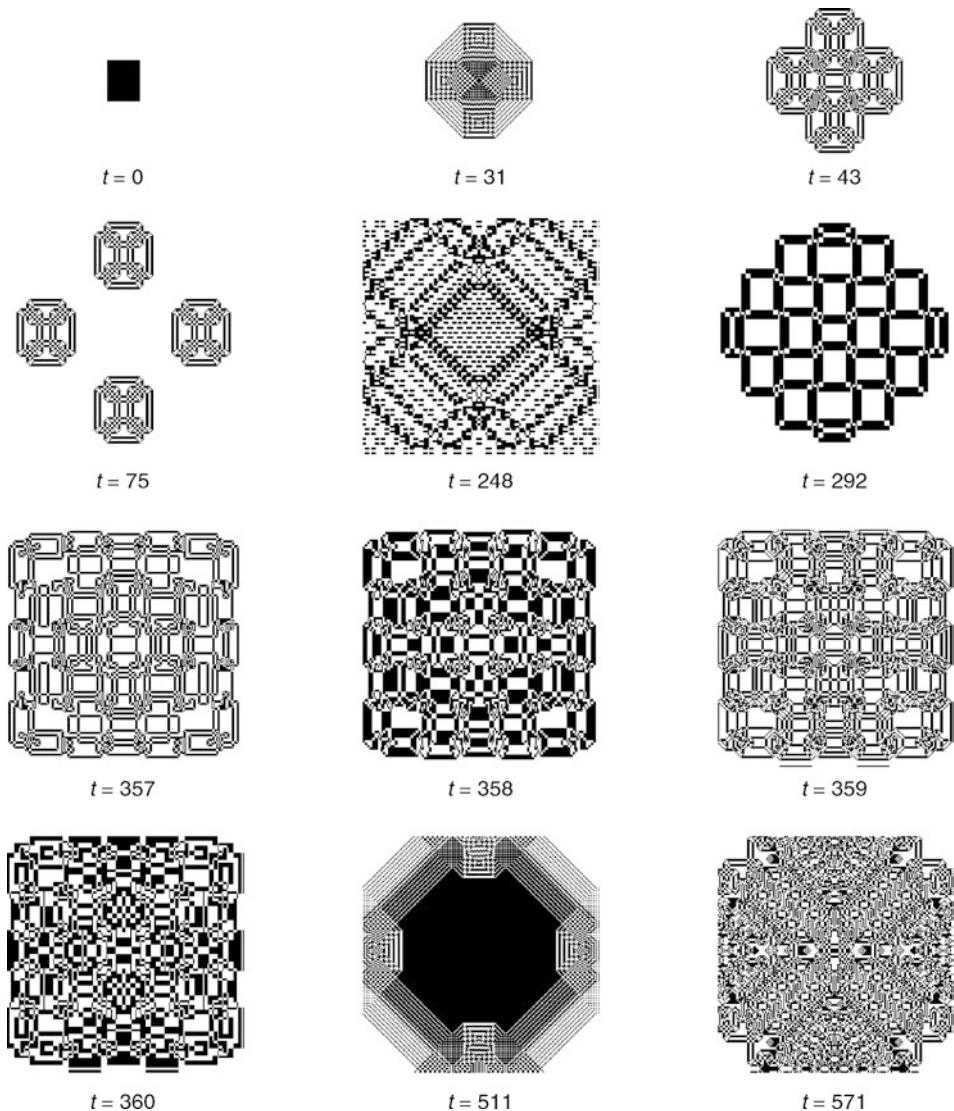
where $\mathbf{r} + \mathbf{v}_k$ designate the cells belonging to the neighborhood \mathcal{N} of cell \mathbf{r} and δ_t is the duration of one time step.

In the above definition, the rule R is identical for all sites and is applied simultaneously to each of them, leading to a synchronous dynamics. As the number of configurations of the neighborhood is finite, it is common to pre-compute all the values of R in a lookup table. Otherwise, an algebraic expression can be used and evaluated at each iteration, for each cell, as in [Eq. 1](#).

It is important to notice that the rule is *homogeneous*, that is, it cannot depend explicitly on the cell position \mathbf{r} . However, spatial (or even temporal) inhomogeneities can be introduced anyway by having some $\Psi_j(\mathbf{r})$ systematically 1 in some given locations of the lattice to mark particular cells on which a different rule applies. Boundary cells are a typical example of spatial inhomogeneities. Similarly, it is easy to alternate between two rules by having a bit, which is 1

Fig. 2

Several snapshots of the parity rule on a 196×196 periodic lattice. The upper left image correspond to an initial configuration make up of a square of 27×34 cells in state 1.

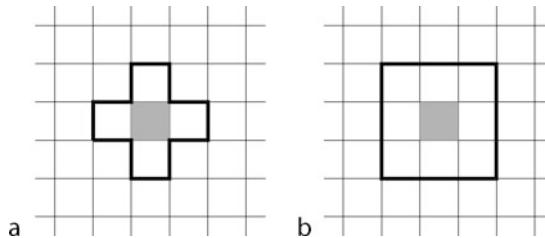


at even time steps and 0 at odd time steps. Finally, memory states can be included by simply copying the current state into a “past state” during the update.

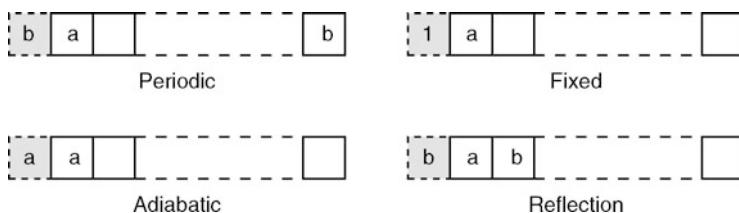
The neighborhood \mathcal{N} of each cell (i.e., the spatial region around each cell used to compute the next state) is usually made of its adjacent cells. It is often restricted to the nearest or next-to-nearest neighbors, to keep the complexity of the rule reasonable. For a two-dimensional cellular automaton, two neighborhoods are often considered, as illustrated in [Fig. 3](#): the *von Neumann neighborhood*, which consists of a central cell (the one which is to be updated)

Fig. 3

(a) Von Neumann and (b) Moore neighborhoods. The shaded region indicates the central cell, which is updated according to the state of the cells located within the domain marked with the bold line.

**Fig. 4**

Various types of boundary conditions obtained by extending the neighborhood. The shaded block represents a virtual cell, which is added at the extremity of the lattice (left extremity, here) to complete the neighborhood.



and its four geographical neighbors north, west, south, and east. The *Moore neighborhood* contains, in addition, the second nearest neighbor north-east, north-west, south-east, and south-west, which is a total of nine cells. See Chopard and Droz (1998) for more details. In practice, when simulating a given CA rule, one cannot deal with an infinite lattice. The system must be finite and have boundaries. Clearly, a site belonging to the lattice boundary does not have the same neighborhood as other internal sites. In order to define the behavior of these sites, a different evolution rule can be considered, which sees the appropriate neighborhood. This means that the information of being a boundary cell or not is coded at the site, using a particular value of Ψ_j , for a chosen j . Depending on this information, a different rule is selected. Following this approach, it is possible to define several types of boundaries, all with a different behavior.

Instead of having a different rule at the limits of the system, another possibility is to extend the neighborhood for the sites at the boundary. For instance, a very common solution is to assume *periodic* (or cyclic) boundary conditions, that is, one supposes that the lattice is embedded in a torus-like topology. In the case of a two-dimensional lattice, this means that the left and right sides are connected, and so are the upper and lower sides.

Other possible types of boundary conditions are illustrated in **Fig. 4**, for a one-dimensional lattice. It is assumed that the lattice is augmented by a set of virtual cells beyond its limits. A *fixed* boundary is defined so that the neighborhood is completed with cells having a preassigned value. An *adiabatic* boundary condition (or zero-gradient) is obtained by

duplicating the value of the site to the extra virtual cells. A reflecting boundary amounts to copying the value of the other neighbor in the virtual cell.

According to the definition, a cellular automaton is deterministic. The rule is some well-defined function and a given initial configuration will always evolve identically. However, it may be very convenient for some applications to have a certain degree of randomness in the rule. For instance, it may be desirable that a rule selects one outcome among several possible states, with a probability p . CA whose updating rule is driven by some external probabilities are called *probabilistic* CA. On the other hand, those which strictly comply with the definition given above, are referred to as *deterministic* CA.

3 Cellular Automata and Complex Systems

3.1 Game of Life and Langton Ant

Complex systems are now an important domain in sciences. They are systems made of many interacting constituents which often exhibit spatiotemporal patterns and collective behaviors.

A standard and successful methodology in research has been to isolate phenomena from each other and to study them independently. This leads to a deep understanding of the phenomena themselves but also leads to a compartmentalized view of nature. The real world is made of interacting processes and these interactions bring new phenomena that are not present in the individual constituents. Therefore, the whole is more than the sum of its parts and new scientific tools and concepts may be required to analyze complex systems.

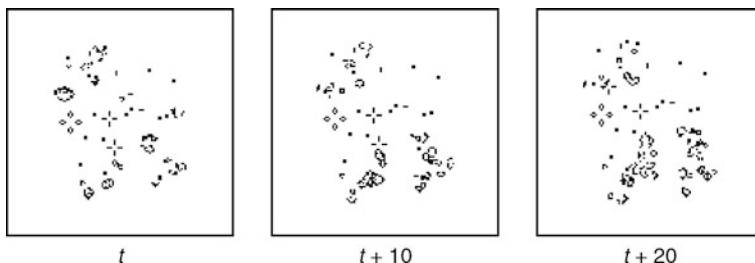
CAs offer such a possibility by being themselves simple, fully discrete complex systems. In this section, two CA, the so-called *Game of Life* and *Langton's Ant* model are introduced. Both illustrate interesting aspects of complex systems.

3.1.1 The Game of Life

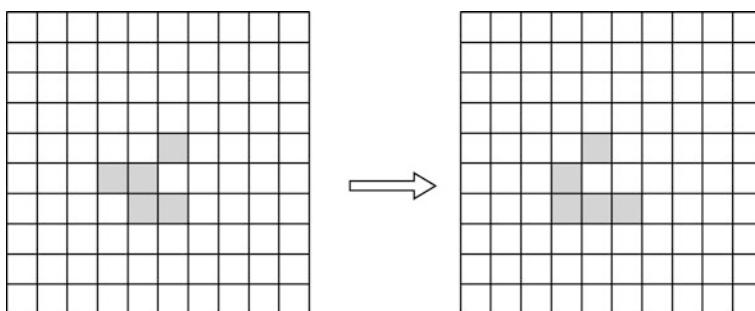
In 1970, the mathematician John Conway proposed the now famous Game of Life (Gardner 1970) CA. The motivation was to find a simple rule leading complex behaviors in a system of fictitious one-cell organisms evolving in a fully discrete universe. The Game of Life rule is defined on a two-dimensional square lattice in which each spatial cell can be either occupied by a living organism (state one) or empty (state zero). The updating rule of the Game of Life is as follows: an empty cell surrounded by exactly three living cells gets alive; a living cell surrounded by less than two or more than three neighbors dies of isolation or overcrowdedness. Here, the surrounding cells correspond to a Moore neighborhood composed of the four nearest cells (north, south, east and west), plus the four second nearest neighbors, along the diagonals. It turns out that the Game of Life automaton has an unexpectedly rich behavior. Complex structures emerge out of a primitive “soup” and evolve so as to develop some new skills (see Fig. 5). For instance a particular spatial assembly of cells has the property to move across the lattice. Such an object, called *glider*, can be seen as higher level organism because it is composed of several simple elementary cells. Its detailed structure is shown in Fig. 6. Thus, by assembling in a clever way cells that are unable to move, it is possible to produce, at a larger scale, a new capability. This is a signature of complex systems. Of course, more complex objects can be built, such as for instance *glider guns*, which are arrangements of cell-producing gliders rhythmically.

Fig. 5

The Game of Life automaton. Black dots represent living cells whereas dead cells are white. The figure shows the evolution of a random initial configuration and the formation of spatial structures, with possibly some emerging functionalities.

**Fig. 6**

The detailed structure of a glider, over two consecutive iterations. A glider is an assembly of cells that has a higher functionality than its constituent, namely, the capability to move in space by changing its internal structure in a periodic way.



The Game of Life is a cellular automata capable of *universal computations*: it is always possible to find an initial configuration of the cellular space reproducing the behavior of any electronic gate and, thus, to mimic any computation process. Although this observation has little practical interest, it is very important from a theoretical point of view since it assesses the ability of CAs to be a nonrestrictive computational technique. As an illustration of this fact, the Game of Life has been used to compute prime numbers.

3.1.2 Langton's Ant

As just discussed, CAs exemplify the fact that a collective behavior can emerge out of the sum of many, simply interacting, components. Even if the basic and local interactions are perfectly known, it is possible that the global behavior obeys new laws that are not obviously extrapolated from the individual properties. The Langton's ant model further illustrates this aspect.

The ant rule is a cellular automata invented by Chris Langton (Stewart 1994) and Greg Turk, which models the behavior of a hypothetical animal (ant) having a very simple algorithm of motion. The ant moves on a square lattice whose sites are either white or gray. When the ant enters a white cell, it turns 90° to the left and paints the cell in gray. Similarly, if it enters a gray cell, it paints it in white and turns 90° to the right. This is illustrated in [Fig. 7](#).

It turns out that the motion of this ant exhibits a very complex behavior. Suppose the ant starts in a completely white space, after a series of about 500 steps where it essentially keeps returning to its initial position, it enters a chaotic phase during which its motion is unpredictable. Then, after about 10,000 steps of this very irregular motion, the ant suddenly performs a very regular motion, which brings it far away from where it started.

[Figure 8](#) illustrates the ant motion. The path the ant creates to escape the chaotic initial region has been called a highway (Propp 1994). Although this highway is oriented at 45° with respect to the lattice direction, it is traveled by the ant in a way, which makes very much think of a sewing machine: the pattern is a sequence of 104 steps that are repeated indefinitely.

The Langton ant is a good example of a cellular automata whose rule is very simple and yet generates a complex behavior, which seems beyond understanding. Somehow, this fact is typical of the cellular automata approach: although everything is known about the fundamental laws governing a system (because the rules are man-made!), it is difficult to explain its macroscopic behavior.

There is anyway a global property of the ant motion: The ant visits an unbounded region of space, *whatever* the initial space texture is (configuration of gray and white cells).

The proof (due to Bunimovich and Troubetzkoy) goes as follows: suppose the region the ant visits is bounded. Then, it contains a finite number of cells. Since the number of iteration is infinite, there is a domain of cells that are visited infinitely often. Moreover, due to the rule of motion, a cell is either entered horizontally (we call it a H cell) or vertically (we call it a V cell). Since the ant turns by 90° after each step, a H cell is surrounded by four V cells and conversely. As a consequence, the H and V cells tile the lattice in a fixed checkerboard pattern. Now, the upper rightmost cell of the domain is considered, that is a cell whose right and upper neighbor

Fig. 7

The Langton's ant rule.

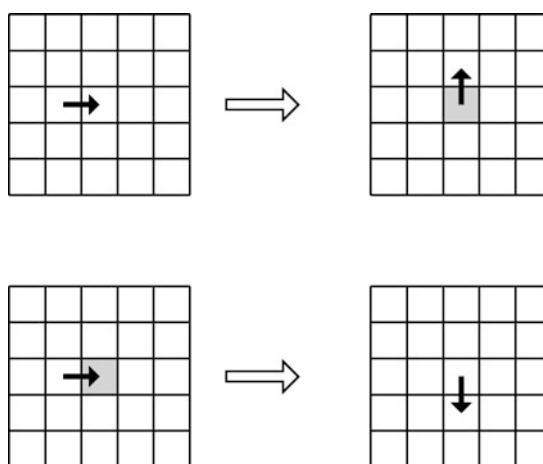
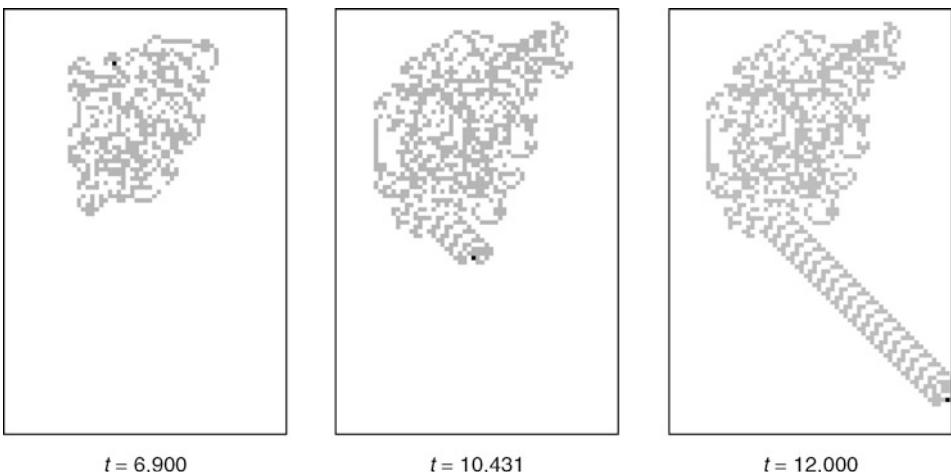


Fig. 8

The Langton's ant rule. The motion of a single ant starts with a chaotic phase of about 10,000 time steps, followed by the formation of a highway. The figure shows the state of each lattice cell (gray or white) and the ant position (marked by the black dot). In the initial condition, all cells are white and the ant is located in the middle of the image.

 $t = 6,900$ $t = 10,431$ $t = 12,000$

is not visited. This cell exists if the trajectory is bounded. If this cell is a H cell (and be so for ever), it has to be entered horizontally from left and exited vertically downward and, consequently, be gray. However, after the ant has left, the cell is white and there is a contradiction. The same contradiction appears if the cell is a V cell. Therefore, the ant trajectory is not bounded.

Beyond the technical aspect of this proof, it is interesting to realize that its conclusion is based on symmetry properties of the rule. Although one is not able to predict the detailed motion of the ant analytically (the only way is to perform the simulation), something can be learned about the global behavior of the system: the ant goes to infinity. This observation illustrates the fact that, often, global features are related to symmetries more than to details.

The case where several ants coexist on the lattice can be now considered. The rule defined in [Fig. 7](#) is only valid for a single ant. It can be generalized for situations where up to four ants enter the same site at the same time, from different sides.

To mathematically describe the ant motion, we introduce $n_i(\mathbf{r}, t)$, a Boolean variable representing the presence ($n_i = 1$) or the absence ($n_i = 0$) of an ant entering site \mathbf{r} at time t along lattice direction \mathbf{c}_i , where $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$, and \mathbf{c}_4 stand for direction right, up, left, and down, respectively. If the color $\mu(\mathbf{r}, t)$ of the site is gray ($\mu = 0$), *all* entering ants turn 90° to the right. On the other hand, if the site is white ($\mu = 1$), then all ants turn 90° to the left. The color of each cell is modified after one or more ants have gone through. Here, we chose to switch $\mu \rightarrow 1 - \mu$ only when an odd number of ants are present.

When several ants travel simultaneously on the lattice, cooperative and destructive behaviors are observed. First, the erratic motion of several ants favors the formation of a local arrangement of colors allowing the creation of a highway. One has to wait much less time before the first highway appears. Second, once a highway is being created, other ants may use it to travel very fast (they do not have to follow the complicated pattern of the highway builder).

In this way, the term “highway” is very appropriate. Third, a destructive effect occurs as the second ant gets to the highway builder. It breaks the pattern and several situations may be observed. For instance, both ants may enter a new chaotic motion; or the highway is traveled in the other direction (note that the rule is time reversal invariant) and destroyed. [Figure 9](#) illustrates the multi-ant behavior.

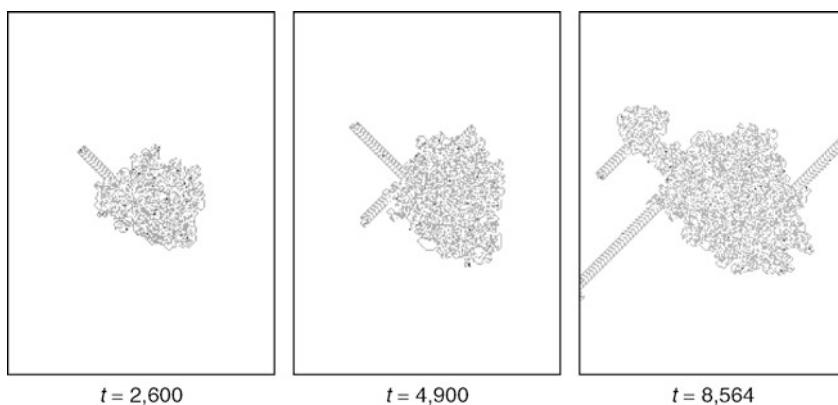
The problem of an unbounded trajectory pauses again with this generalized motion. The assumption of Bunimovich–Troubetzkoy’s proof no longer holds in this case because a cell may be both a H cell or a V cell. Indeed, two different ants may enter the same cell, one vertically and the other horizontally. Actually, the theorem of an unbounded motion is wrong in several cases where two ants are present. Periodic motions may occur when the initial positions are well chosen.

For instance, when the relative location of the second ant with respect to the first one is $(\Delta x, \Delta y) = (2, 3)$, the two ants return to their initial position after 478 iterations of the rule (provided they started in a uniformly white substrate, with the same direction of motion). A very complicated periodic behavior is observed when $(\Delta x, \Delta y) = (1, 24)$: the two ants start a chaotic-like motion for several thousands of steps. Then, one ant builds a highway and escapes from the central region. After a while, the second ant finds the entrance of the highway and rapidly catches the first one. After the two ants meet, they start undoing their previous paths and return to their original position. This complete cycle takes about 30,000 iterations.

More generally, it is found empirically that, when $\Delta x + \Delta y$ is odd and the ants enter their site with the same initial direction, the two-ant motion is likely to be periodic. However, this is not a rule and the configuration $(\Delta x, \Delta y) = (1, 0)$ yields an unbounded motion, a diamond pattern of increasing diameter, which is traveled in the same direction by the two ants.

Fig. 9

Motion of several Langton’s ants. Gray and white indicate the colors of the cells at the current time. Ant locations are marked by the black dots. At the initial time, all cells are white and a few ants are randomly distributed in the central region, with random directions of motion. The first highway appears much earlier than when the ant is alone. In addition the highway can be used by other ants to travel much faster. However, the “highway builder” is usually prevented from continuing its construction as soon as it is reached by the following ants. For instance, the highway heading north-west after 4,900 steps gets destroyed. A new highway emerges later on from the rest, as seen from the snapshot at time $t = 8,564$.



It turns out that the periodic behavior of a two-ant configuration is not so surprising. The rule defined is reversible in time, provided that there is never more than one ant at the same site. Time reversal symmetry means that if the direction of motion of all ants are reversed, they will move backward through their own sequence of steps, with an opposite direction of motion. Therefore, if at some point of their motion the two ants cross each other (on a lattice link, not on a site), the first ant will go through the past of the second one, and vice versa. They will return to the initial situation (the two ants being exchanged) and build a new pattern, symmetrical to the first one, due to the inversion of the directions of motion. The whole process then cycles forever. Periodic trajectories are therefore related to the probability that the two ants will, at some time, cross each other in a suitable way. The conditions for this to happen are fulfilled when the ants sit on a different sublattice (black or white sites on the checkerboard) and exit two adjacent sites against each other. This explains why a periodic motion is likely to occur when $\Delta x + \Delta y$ is odd.

An interesting conclusion is that, again, it is a symmetry of the rule (time reversal invariance) that allows one to draw conclusions about the global behavior. The details of periodic motions are not known but it is known that they are possible.

3.2 Cellular Automata as Simple Dynamical Systems

CA can also be seen as simple prototypes of dynamical systems. In physics, the time evolution of physical quantities is often governed by nonlinear equations. Due to the nonlinearities, solution of these dynamical systems can be very complex. In particular, the solution of these equations can be strongly sensitive to the initial conditions, leading to what is called a chaotic behavior. Similar complications can occur in discrete dynamical systems. Models based on cellular automata provide an alternative approach to study the behavior of dynamical systems. By virtue of their discrete nature, the numerical studies are free of rounding approximations and thus lead to exact results. Also, exhaustive searches in the space of possible rule can be considered for simple CA.

Crudely speaking, two classes of problems can be posed. First, given a cellular automaton rule, predict its properties. Second, find a cellular automaton rule that will have some prescribed properties. These two closely related problems are usually difficult to solve as seen on simple examples.

The simplest cellular automata rules are one-dimensional ones for which each site has only two possible states and the rule involves only the nearest-neighbors sites. They are easily programmable on a personal computer and offer a nice “toy model” to start the study of cellular automata.

A systematic study of these rules was undertaken by Wolfram in 1983 (Wolfram 1986, 1994). Each cell at location r has, at a given time, two possible states $s(r) = 0$ or $s(r) = 1$. The state s at time $t + 1$ depends only on the triplet $(s(r - 1), s(r), s(r + 1))$ at time t :

$$s(r, t + 1) = \Phi(s(r - 1, t), s(r, t), s(r + 1, t)) \quad (3)$$

Thus to each possible values of the triplet $(s(r - 1), s(r), s(r + 1))$, one associates a value $\alpha_k = 0$ or 1 according to the following list:

$$\underbrace{111}_{\alpha_7} \quad \underbrace{110}_{\alpha_6} \quad \underbrace{101}_{\alpha_5} \quad \underbrace{100}_{\alpha_4} \quad \underbrace{011}_{\alpha_3} \quad \underbrace{010}_{\alpha_2} \quad \underbrace{001}_{\alpha_1} \quad \underbrace{000}_{\alpha_0} \quad (4)$$

Each possible CA rule \mathcal{R} is characterized by the values $\alpha_0, \dots, \alpha_7$. There are clearly 256 possible choices. Each rule can be identified by an index $\mathcal{N}_{\mathcal{R}}$ computed as follows

$$\mathcal{N}_{\mathcal{R}} = \sum_{i=0}^7 2^{(i)} \alpha_i \quad (5)$$

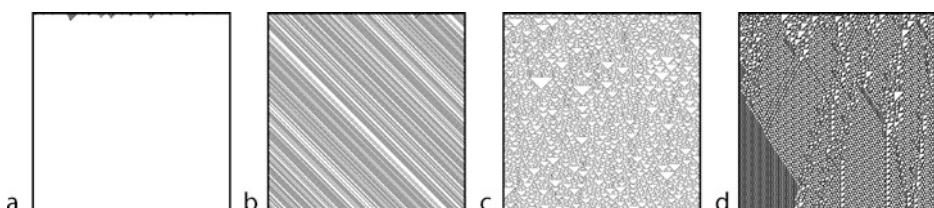
which corresponds to the binary representation $\alpha_7 \alpha_6 \alpha_5 \alpha_4 \alpha_3 \alpha_2 \alpha_1 \alpha_0$.

Giving a rule and an initial state, one can study the time evolution of the system. Some results can be deduced analytically using algebraic techniques, but most of the conclusions follow from numerical iterations of the rules. One can start from a simple initial state (i.e., only one cell in the state 1) or with a typical random initial state. According to their behavior, the different rules have been grouped in four different classes.

1. Class 1. These CA evolve after a finite number of time steps from almost all initial states to a unique homogeneous state (all the sites have the same value). The set of exceptional initial configurations, which behave differently is of measure zero when the number of cells N goes to infinity. An example is given by the rule 40 (see [Fig. 10a](#)). From the point of view of dynamical systems, these automata evolve toward a simple *limit point* in the phase space.
2. Class 2. A pattern consisting of separated periodic regions is produced from almost all the initial states. The simple structures generated are either stable or periodic with small periods. An example is given by rule 56 (see [Fig. 10b](#)) Here again, some particular initial states (set of measure zero) can lead to unbounded growth. The evolution of these automata is analogous to the evolution of some continuous dynamical systems to *limit cycles*.
3. Class 3. These CA evolve from almost all initial states to chaotic, aperiodic patterns. An example is given by rule 18 (see [Fig. 10c](#)). Small changes in the initial conditions almost always lead to increasingly large changes in the later stages. The evolution of these automata is analogous to the evolution of some continuous dynamical systems to *strange attractors*.

Fig. 10

Example of the four Wolfram rules with a random initial configuration. Horizontal lines correspond to consecutive iterations. The initial state is the uppermost line. (a) Rule 40 belonging to class 1 reaches a fixed point (stable configuration) very quickly. (b) Rule 56 of class 2 reaches a pattern composed of stripes that move from left to right. (c) Rule 18 is in class 3 and exhibits a self-similar pattern. (d) Rule 110 is an example of a class 4 cellular automaton. Its behavior is not predictable and, as a consequence, a rupture in the pattern on the left part is observed.



4. Class 4. For these CA, persistent complex structures are formed for a large class of initial states. An example is given by the rule 110 (see [Fig. 10d](#)). The behavior of such CA can generally be determined only by an explicit simulation of their time evolution.

The “toy rules” considered by Wolfram, although very simple in construction, are capable of very complex behavior. The validity of this classification is not restricted to the simple rules described above but is somehow generic for more complicated rules. For example, one can consider rules for which each cell can have k different states and involve a neighborhood of radius ℓ (thus the rule depends on the values of $2\ell + 1$ cells). In this case, the number of possible rules is $k^{(k^{(2\ell+1)})}$. Several cases have been studied in the literature and the different rules can be classified in one of the four above classes. Many of the class 4 CA (starting with $k = 2, \ell = 2$) have the property of computational universality and initial configurations can specify arbitrary algorithmic procedures.

However the above “phenomenological” classification suffers drawbacks, the most serious of which is its non-decidability. See Culick and Yu ([1988](#)) for more details.

3.3 Competition, Cooperation, Contamination

In this section, a few CA rules that are very simple and mimic very natural interactions are briefly described: competition between adjacent cells, cooperation between them, or contamination of neighbors. All these ideas can be implemented within a discrete universe, with cells having only a few possible states. Other models of cooperation–competition are discussed in Galam et al. ([1998](#)).

3.3.1 Cooperation Models

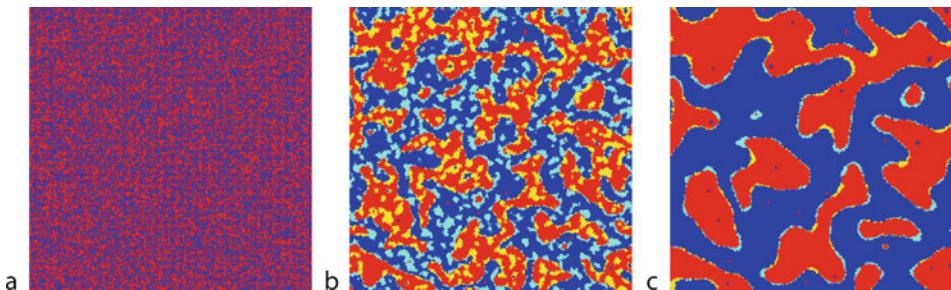
For instance, in a simple cooperation model, a cell may want to evolve by copying the behavior of the majority of its neighbors. If the possible states are 0 or 1, then clearly an all 0’s configuration or an all 1’s configuration are both stable. But what happens if the initial configuration contains cells that are 1 with probability p and 0 with probability $1 - p$. It is likely that such a system will evolve to one of the two stable configuration. It would be very nice if the all 1’s configuration is reached whenever $p > 1/2$ and the all 0’s configuration would be the final stage of the case $p < 1/2$. Then, one would have built a system with only *local* calculation that is able to solve a global problem: deciding if the initial density of cells in state 1 is larger or smaller than 1/2. This problem is known as the *density task* and, in general, a simple majority rule is not able to give a reliable answer. See Sipper ([1997](#)) for more details.

From the point of view of modeling physical systems, a slight variant of the majority rule produces interesting patterns. The *twisted majority rule* proposed by Vichniac ([1984](#)) is defined on a two-dimensional lattice where each cell considers its Moore neighborhood. The evolution rule first computes the sum of the cells in state 1. This sum can be any value between 0 and 9. The new state $s(t + 1)$ of each cell is then determined from this local sum, according to the following table

$$\begin{array}{ll} \text{sum}(t) & 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \\ s(t + 1) & 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \end{array} \quad (6)$$

Fig. 11

Evolution of the twisted majority rule. The inherent “surface tension” present in the rule tends to separate the red phases $s = 1$ from the blue phase $s = 0$. The snapshots (a), (b), and (c) correspond to $t = 0$, $t = 72$, and $t = 270$ iterations, respectively. The other colors indicate how “capes” have been eroded and “bays” filled: light blue shows the blue regions that have been eroded during the last few iterations and yellow marks the red regions that have been filled.



As opposed to the plain majority rule, here, the two middle entries of the table have been swapped. Therefore, when there is a slight majority of 1 around a cell, it turns to 0. Conversely, if there is a slight majority of 0, the cell becomes 1.

Surprisingly enough, this rule describes the interface motion between two phases, as illustrated in Fig. 11. It is observed that the normal velocity of the interface is proportional to its local curvature, as required by several physical theories. Of course, due to its discrete and local nature, the rule cannot detect the curvature of the interface directly. However, as the rule is iterated, local information is propagated to the nearest neighbors and the radius of curvature emerges as a collective effect.

3.3.2 Competition Models

In some sense, the twisted majority rule corresponds to a cooperative behavior between the cells. A quite different situation can be obtained if the cells obey a competitive dynamics. For instance, one may imagine that the cells compete for some resources at the expense of their nearest neighbors. A winner is a cell of state 1 and a loser a cell of state 0. No two winner cells can be neighbors and any loser cell must have at least one winner neighbor (otherwise nothing would have prevented it from winning).

This problem has a direct application in biology, to study cells differentiation. It has been observed in the development of the drosophila that about 25% of the cells forming the embryo are evolving to the state of neuroblast, while the remaining 75% does not. How can we explain this differentiation and the observed fraction since, at the beginning of the process all cells can be assumed equivalent? A possible mechanism (Luthi et al. 1998) is that some competition takes place between the adjacent biological cells. In other words, each cell produces some substance S but the production rate is inhibited by the amount of S already present in the neighboring cells. Differentiation occurs when a cell reaches a level of S above a given threshold.

Following this interpretation, the following CA model of competition can be considered. First, a hexagonal lattice, which is a reasonable approximation of the cell arrangement

observed in the drosophila embryo, is considered. It is assumed that the values of S can be 0 (inhibited) or 1 (active) in each lattice cell.

- A $S = 0$ cell will grow (i.e., turn to $S = 1$) with probability p_{growth} provided that all its neighbors are 0. Otherwise, it stays inhibited.
- A cell in state $S = 1$ will decay (i.e., turn to $S = 0$) with probability p_{annihil} if it is surrounded by at least one active cell. If the active cell is isolated (all the neighbors are in state 0) it remains in state 1.

The evolution stops (stationary process) when no $S = 1$ cell feels any more inhibition from its neighbors and when all $S = 0$ cells are inhibited by their neighborhood. Then, with the biological interpretation, cells with $S = 1$ are those that will differentiate.

What is the expected fraction of these $S = 1$ cells in the final configuration? Clearly, from [Fig. 12](#), the maximum value is 1/3. According to the inhibition condition imposed, this is the close-packed situation on the hexagonal lattice. On the other hand, the minimal value is 1/7, corresponding to a situation where the lattice is partitioned in blocks with one active cell surrounded by six inhibited cells. In practice, any of these two limits are not expected to occur spontaneously after the automaton evolution. On the contrary, one can observe clusters of close-packed active cells surrounded by defects, that is, regions of low density of active cells.

As illustrated in [Fig. 13](#), CA simulations show indeed that the final fraction s of active cells is a mix of the two limiting situations of [Fig. 12](#)

$$.23 \leq s \leq .24$$

almost irrespectively of the values chosen for p_{annihil} and p_{growth} .

This is exactly the value expected from the biological observations made on the drosophila embryo. Thus, cell differentiation can be explained by a geometrical competition without having to specify the inhibitory couplings between adjacent cell and the production rate (i.e., the values of p_{annihil} and p_{growth}): the result is quite robust against any possible choices.

3.3.3 Contamination Models

Finally, after cooperation and competition dynamics, we also consider a contamination process. To make the model more interesting, we consider cells with at least three possible states. These are the resting state, the excited state, and the refractory state.

[Fig. 12](#)

The hexagonal lattice used for the competition-inhibition CA rule. Black cells are cells of state 1 (winners) and white cells are cells of state 0 (losers). The two possible final states with a fully regular structure are illustrated with density 1/3 and 1/7 of winner, respectively.

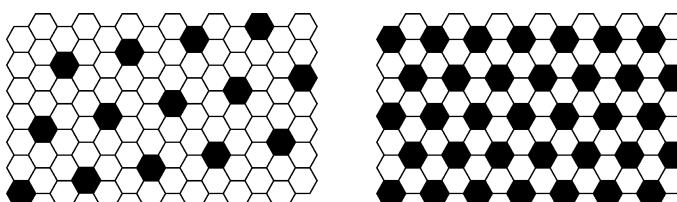
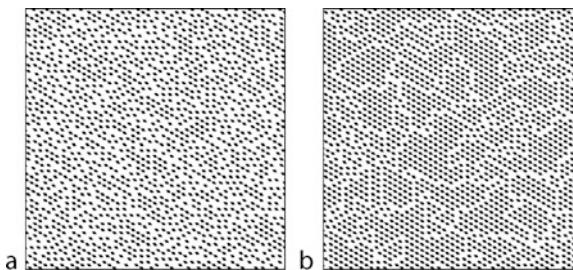


Fig. 13

Final configuration of the competition CA model. (a) A typical situation with about 23% of active cells, obtained with almost any value of p_{anihil} and p_{growth} . (b) Configuration obtained with $p_{\text{anihil}} = 1$ and $p_{\text{growth}} = 0.8$ and yielding a fraction of 28% of active cells; one clearly sees the close-packed regions and the defects.



The resting state is a stable state of the system. But a resting state can respond to a local perturbation and become excited. Then, the excited state evolves to a refractory state where it no longer influences its neighbors and, finally, returns to the resting state.

The Greenberg–Hastings model is an example of a cellular automata model with a contamination mechanism. It is also called a model for an *excitable media* in the context of reactive systems and chemical waves.

The Greenberg–Hastings model can be defined as follows: the state $\psi(\mathbf{r}, t)$ of site \mathbf{r} at time t takes its value in the set $\{0, 1, 2, \dots, n - 1\}$. The state $\psi = 0$ is the resting state. The states $\psi = 1, \dots, n/2$ (n is assumed to be even) correspond to excited states. The rest, $\psi = n/2 + 1, \dots, n - 1$ are the refractory states. The CA evolution rule is the following:

1. If $\psi(\mathbf{r}, t)$ is excited or refractory, then $\psi(\mathbf{r}, t + 1) = \psi(\mathbf{r}, t) + 1 \bmod n$.
2. If $\psi(\mathbf{r}, t) = 0$ (resting state) it remains so, unless there are at least k excited sites in the Moore neighborhood of site \mathbf{r} . In this case $\psi(\mathbf{r}, t + 1) = 1$.

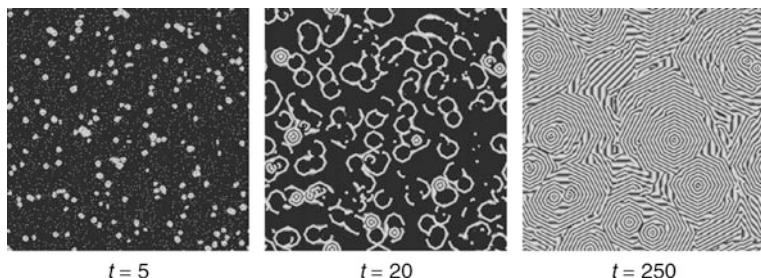
The n states play the role of a clock: An excited state evolves through the sequence of all possible states until it returns to 0, which corresponds to a stable situation.

The behavior of this rule is quite sensitive to the value of n and the excitation threshold k . Figure 14 shows the evolution of this CA for a given set of the parameters n and k . The simulation is started with a uniform configuration of resting states, perturbed by some excited sites randomly distributed over the system. Note that if the concentration of perturbation is low enough, excitation dies out rapidly and the system returns to the rest state. Increasing the number of perturbed states leads to the formation of traveling waves and self-sustained oscillations may appear in the form of ring or spiral waves.

The Greenberg–Hastings model has some similarity with the “tube-worms” rule proposed by Toffoli and Margolus (1987). This rule is intended to model the Belousov–Zhabotinsky reaction and is as follows. The state of each site is either 0 (refractory) or 1 (excited) and a local timer (whose value is 3, 2, 1, or 0) controls the refractory period. Each iteration of the rule can be expressed by the following sequence of operations: (1) where the timer is zero, the state is excited; (2) the timer is decreased by 1 unless it is 0; (3) a site becomes refractory whenever the timer is equal to 2; (4) the timer is reset to 3 for the excited sites, which have two, or more than four, excited sites in their Moore neighborhood.

Fig. 14

Excitable medium: evolution of a configuration with 5% of excited states $\phi = 1$, and 95% of resting states (black), for $n = 8$ and $k = 3$.

**Fig. 15**

The tube-worms rule for an excitable media.

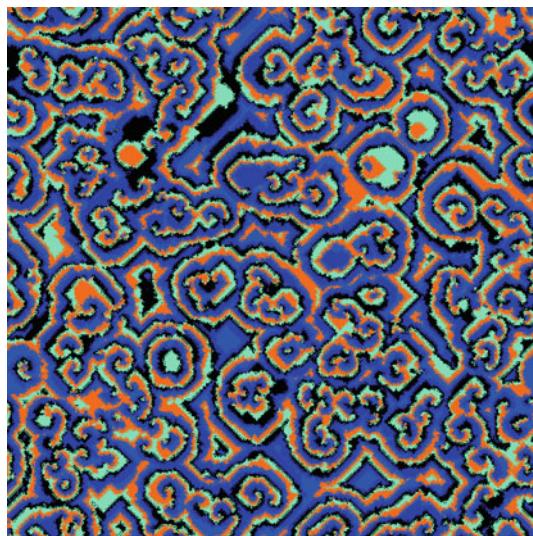
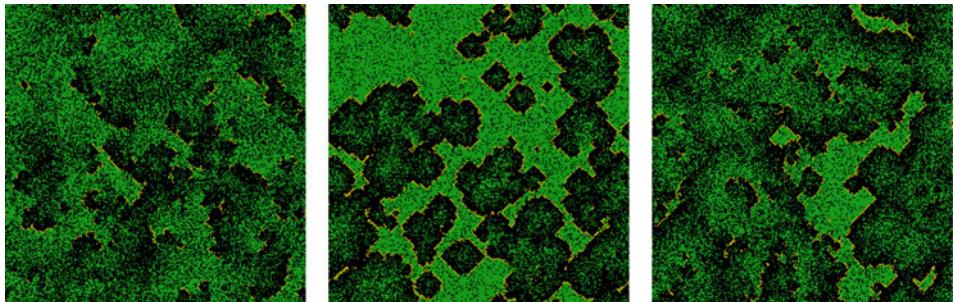


Figure 15 shows a simulation of this automaton, starting from a random initial configuration of the timers and the excited states. One can observe the formation of spiral pairs of excitations. Note that this rule is very sensitive to small modifications (in particular to the order of operations (1)–(4)).

Another rule which is also similar to Greenberg–Hastings and Margolus–Toffoli tube-worm models is the so-called forest-fire model. This rule describes the propagation of a fire or, in a different context, may also be used to mimic contagion in the case of an epidemic. Here, the case of a forest-fire rule is described. The forest-fire rule is a probabilistic CA defined on a two-dimensional square lattice. Initially, each site is occupied by either a tree, a burning tree, or it is empty. The state of the system is parallel updated according to the

Fig. 16

The forest-fire rule: green sites correspond to a grown tree, black pixels represent burned sites, and the yellow color indicates a burning tree. The snapshots given here represents three situations after a few hundred iterations. The parameters of the rule are $p = 0.3$ and $f = 6 \times 10^{-5}$.



following rule: (1) a burning tree becomes an empty site; (2) a green tree becomes a burning tree if at least one of its nearest neighbors is burning; (3) at an empty site, a tree grows with probability p ; (4) A tree without a burning neighbor becomes a burning tree with probability f (so as to mimic an effect of lightning). [Figure 16](#) illustrates the behavior of this rule, in a two-dimensional situation.

3.4 Traffic Models

CA models for road traffic have received a great deal of interest during the past few years (see Yukawa et al. 1994, Schreckenberg and Wolf 1998, Chopard et al. 1996, Schreckenberg et al. 1995, and Nagel and Herrmann 1993 for instance).

CA models for a single lane car motions are quite simple. The road is represented as a line of cells, each of them being occupied or not by a vehicle. All cars travel in the same direction (say to the right). Their positions are updated synchronously. During the motion, each car can be at rest or jump to the nearest neighbor site, along the direction of motion. The rule is simply that a car moves only if its destination cell is empty. This means that the drivers do not know whether the car in front will move or will be blocked by another car. Therefore, the state s_i of each cell at location i is entirely determined by the occupancy of the cell itself and that of its two nearest neighbors s_{i-1} and s_{i+1} . The motion rule can be summarized by the following table, where all eight possible configurations $(s_{i-1}s_is_{i+1})_t \rightarrow (s_i)_{t+1}$ are given

$$\begin{array}{cccccccc} (111) & (110) & (101) & (100) & (011) & (010) & (001) & (000) \\ \underbrace{1} & \underbrace{0} & \underbrace{1} & \underbrace{1} & \underbrace{1} & \underbrace{0} & \underbrace{0} & \underbrace{0} \end{array} \quad (7)$$

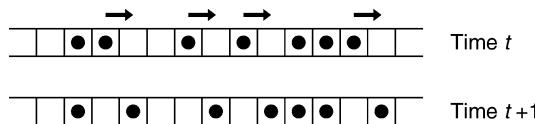
This cellular automaton rule turns out to be Wolfram rule 184 (Wolfram 1986; Yukawa et al. 1994). It is illustrated in [Fig. 17](#).

This simple dynamics captures an interesting feature of real car motion: traffic congestion. Suppose there is a low car density ρ in the system, for instance something like

$$\dots 0010000010010000010\dots \quad (8)$$

Fig. 17

Illustration of the basic traffic rule: car with a free cell in front can move. The other ones stay at rest.



This is a free traffic regime in which all the cars are able to move. The average velocity $\langle v \rangle$ defined as the number of motions divided by the number of cars is then

$$\langle v_{\text{free}} \rangle = 1 \quad (9)$$

On the other hand, in a high density configuration such as

$$\dots 110101110101101110\dots \quad (10)$$

only six cars over 12 will move and $\langle v \rangle = 1/2$. This is a partially jammed regime.

In this case, since a car needs a hole to move to, one can expect that the number of moving cars simply equals the number of empty cells (Yukawa et al. 1994). Thus, the number of motions is $L(1 - \rho)$, where L is the number of cells. Since the total number of cars is ρL , the average velocity in the jammed phase is

$$\langle v_{\text{jam}} \rangle = \frac{1 - \rho}{\rho} \quad (11)$$

From the above relations the so-called fundamental flow diagram can be computed, that is, the relation between the flow of cars $\rho \langle v \rangle$ as a function of the car density ρ : for $\rho \leq 1/2$, one can use the free regime expression and $\rho \langle v \rangle = \rho$. For densities $\rho > 1/2$, one can use the jammed expression and $\rho \langle v \rangle = 1 - \rho$. The resulting diagram is shown in [Fig. 18](#). As in real traffic, we observe that the flow of cars reaches a maximum value before decreasing.

A richer version of the above CA traffic model is due to Nagel and Schreckenberg (1992). The cars may have several possible velocities $u = 0, 1, 2, \dots, u_{\max}$. Let u_i be the velocity of car i and d_i the distance, along the road, separating cars i and $i + 1$. The updating rule is

- The cars accelerate when possible: $u_i \rightarrow u'_i = u_i + 1$, if $u_i < u_{\max}$.
- The cars slow down when required: $u'_i \rightarrow u''_i = d_i - 1$, if $u'_i \geq d_i$.
- The cars have a random behavior: $u''_i \rightarrow u'''_i = u''_i - 1$, with probability p_i if $u''_i > 0$.
- Finally the cars move u'''_i sites ahead.

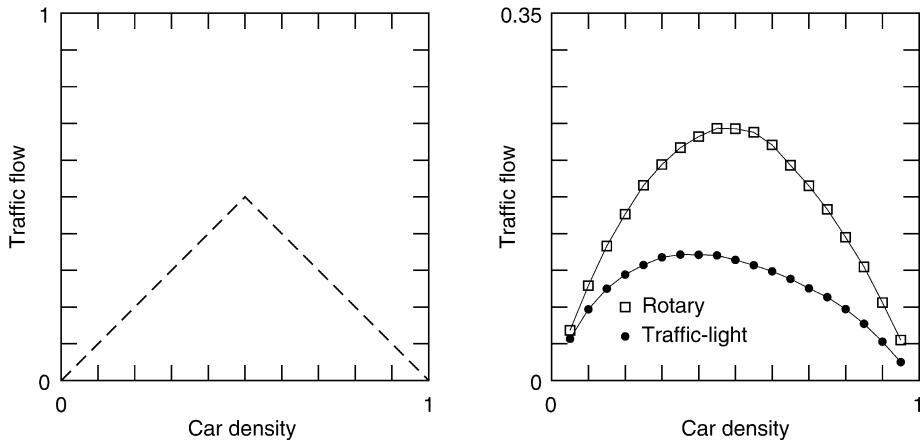
This rule captures some important behaviors of real traffic on a highway: velocity fluctuations due to a nondeterministic behavior of the drivers, and “stop-and-go” waves observed in high density traffic regime.

Note that a street network can also be described using a CA. A possible approach is to couple several 1D CA model at each road intersection using a roundabout (Chopard et al. 1996; Chopard and Dupuis 2003). This is illustrated in [Fig. 19](#) for a Manhattan-like configuration of streets.

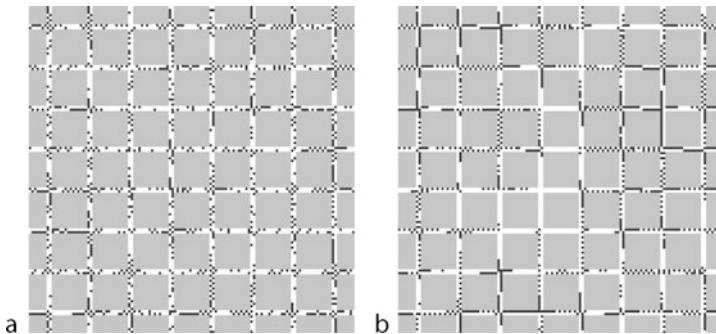
The reader can refer to recent literature for the new developments in this topic. See for instance Kanai et al. (2005, 2006).

Fig. 18

Traffic flow diagram. *Left:* for the simple CA traffic rule. *Right:* for an urban CA traffic model, for a configuration of streets as shown in [Fig. 19](#).

**Fig. 19**

Traffic configuration after 600 iterations, for a car density of 30%. Situation (a) corresponds to a situation where junctions are modeled as roundabouts, whereas image (b) mimics the presence of traffic lights. In the second case, car queues are more likely to form and the global mobility is less than in the first case, as shown in the right part of [Fig. 18](#).



4 A Simple Model for a Gas of Particles

The Hardy, Pomeau, de Pazzis (HPP) rule is a simple example of an important class of cellular automata models: LGA. The basic ingredient of such models are point particles that move on a lattice, according to appropriate rules so as to mimic a fully discrete “molecular dynamics.”

This model is mostly interesting for pedagogical reasons as it illustrates many important features of LGA and LB models in a simple way. However, HPP is of little practical interest because its physical behavior has many flaws (see for instance [Fig. 22](#)) that are cured in more sophisticated models, such as the famous FHP model (Frisch et al. 1986), which has been shown to reproduce the Navier–Stokes equations.

The HPP LGA is defined on a two-dimensional square lattice. Particles can move along the main directions of the lattice, as shown in [Fig. 20](#). The model limits to 1 the number of particles entering a given site with a given direction of motion. This is the exclusion principle, which is common in most LGA (LGA models without exclusion principle are called multiparticle models (Chopard and Droz 1998)).

With at most one particle per site and direction, four bits of information at each site are enough to describe the system during its evolution. For instance, if at iteration t site \mathbf{r} has the following state $s(\mathbf{r}, t) = (1011)$, it means that three particles are entering the site along direction 1, 3 and 4, respectively.

The CA rule describing the evolution of $s(\mathbf{r}, t)$ is often split in two steps: collision and propagation (or streaming). The collision phase specifies how the particles entering the same site will interact and change their trajectories. During the propagation phase, the particles actually move to the nearest neighbor site they are traveling to. This decomposition into two phases is a quite convenient way to partition the space so that the collision rule is purely local.

[Fig. 21](#) illustrates the HPP rules. According to the Boolean representation of the particles at each site, the collision rules for the two-particle head-on collisions are expressed as

$$(1010) \rightarrow (0101) \quad (0101) \rightarrow (1010) \quad (12)$$

All other configurations are unchanged by the collision process.

After the collision, the propagation phase moves information to the nearest neighbors: The first bit of the state variable is shifted to the east neighbor cell, the second bit to the north, and so on. This gives the new state of the system, at time $t + 1$. Remember that both collision and propagation are applied simultaneously to all lattice sites.

The aim of the HPP rule is to reproduce some aspects of the real interactions between particles, namely, that momentum and particle number are conserved during a collision. From [Fig. 21](#), it is easy to check that these properties are obeyed: A pair of zero momentum particles along a given direction is transformed into another pair of zero momentum along the perpendicular axis.

It is easy to express the HPP model in a mathematical form. For this purpose, the so-called occupation number $n_i(\mathbf{r}, t)$ is introduced for each lattice site \mathbf{r} and each time step t . The index i

Fig. 20

Example of a configuration of HPP particles.

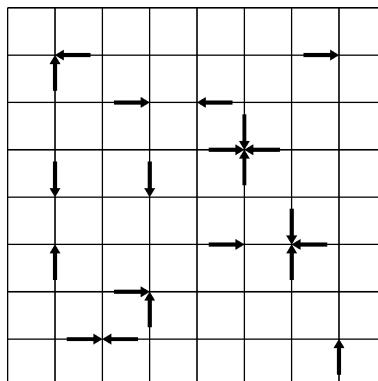
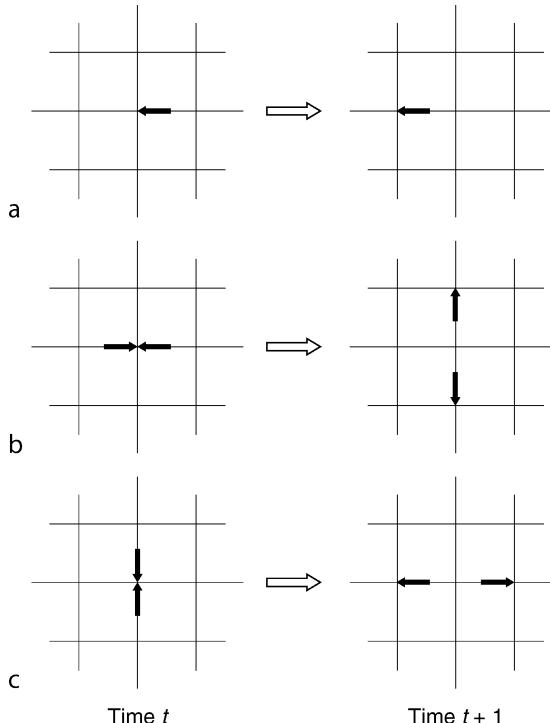


Fig. 21

The HPP rule: (a) a single particle has a ballistic motion until it experiences a collision; (b) and (c) the two nontrivial collisions of the HPP model: two particles experiencing a head-on collision are deflected in the perpendicular direction. In the other situations, the motion is ballistic, that is, the particles are transparent to each other when they cross the same site.



labels the lattice directions (or the possible velocities of the particles). In the HPP model, the lattice has four directions (north, west, south, and east) and i runs from 1 to 4.

By definition and due to the exclusion principle, the n_i 's are Boolean variables

$$n_i(\mathbf{r}, t) = \begin{cases} 1 & \text{if a particle is entering site } \mathbf{r} \text{ at time } t \text{ along lattice direction } i \\ 0 & \text{otherwise} \end{cases}$$

From this definition it is clear that, for HPP, the n_i 's are simply the components of the state s introduced above

$$s = (n_1, n_2, n_3, n_4)$$

In a LGA model, the microdynamics can be naturally expressed in terms of the occupation numbers n_i as

$$n_i(\mathbf{r} + \mathbf{v}_i \delta_t, t + \delta_t) = n_i(\mathbf{r}, t) + \Omega_i(n(\mathbf{r}, t)) \quad (13)$$

where \mathbf{v}_i is a vector denoting the speed of the particle in the i th lattice direction and δ_t is the duration of the time step. The function Ω is called the collision term and it describes the interaction of the particles, which meet at the same time and same location.

Note that another way to express [Eq. 13](#) is through the so-called collision and propagation operators C and P

$$n(t + \delta_t) = PCn(t) \quad (14)$$

where $n(t)$ describe the set of values $n_i(\mathbf{r}, t)$ for all i and \mathbf{r} . The quantities C and P act over the entire lattice. They are defined as

$$(Pn)_i(\mathbf{r}) = n_i(\mathbf{r} - \mathbf{v}_i \delta_t) \quad (Cn)_i(\mathbf{r}) = n_i(\mathbf{r}) + \Omega_i$$

More specifically, for the HPP model, it can be shown (Chopard and Droz 1998) that the collision and propagation phase can be expressed as

$$n_i(\mathbf{r} + \mathbf{v}_i \delta_t, t + \delta_t) = n_i - n_i n_{i+2}(1 - n_{i+1})(1 - n_{i+3}) + n_{i+1} n_{i+3}(1 - n_i)(1 - n_{i+2}) \quad (15)$$

In this equation, the values $i + m$ are wrapped onto the values 1–4 and the right-hand term is computed at position \mathbf{r} and time t . From this relation, it is easy to show that, for any values of n_i ,

$$\sum_{i=1}^4 n_i(\mathbf{r} + \mathbf{v}_i \delta_t, t + \delta_t) = \sum_{i=1}^4 n_i(\mathbf{r}, t) \quad (16)$$

that expresses the conservation of the number of particle during the collision and the propagation. Similarly, it can be shown ($\mathbf{v}_1 = -\mathbf{v}_3$ and $\mathbf{v}_2 = -\mathbf{v}_4$) that

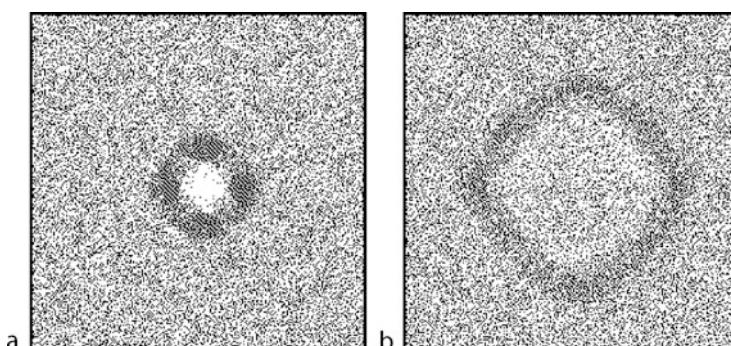
$$\sum_{i=1}^4 n_i(\mathbf{r} + \mathbf{v}_i \delta_t, t + \delta_t) \mathbf{v}_i = \sum_{i=1}^4 n_i(\mathbf{r}, t) \mathbf{v}_i \quad (17)$$

which reflects that momentum is conserved.

The behavior of the HPP model is illustrated in [Fig. 22](#). From this simulation, it is clear that some spatially anisotropic behavior builds up during the time evolution of the rule. A square lattice is actually too poor to represent correctly a fluid system. The FHP model (Frisch et al. 1986; Chopard and Droz 1998), in essence similar to the HPP model, is based on a hexagonal lattice and also includes three-particle collision rules.

Fig. 22

Time evolution of a HPP gas with a density wave. (a) The initial state is a homogeneous gas with a higher density of particles in the middle region (dark area) (b) After several iterations, the initial perturbation propagates as a wave across the system. As can be observed, there is a clear lack of isotropy in this propagation.



5 Lattice Boltzmann Models

Historically LB was developed as an extension of the CA-fluids described in [Sect. 4](#). Another approach is to derive LB models from a discretization of the classical continuous Boltzmann equation (He and Luo 1997; Shan et al. 2006).

Here we stick to the historical approach as it better illustrates the close relation between the numerical scheme and the underlying discrete physical model of interacting particles. The main conceptual difference between LGA and LB models is that in the latter the cell state is no longer the Boolean variable n , but a real-valued quantity f_i for each lattice directions i . Instead of describing the presence or absence of a particle, the interpretation of f_i is the density distribution function of particles traveling in lattice directions i .

From a practical point of view, the advantages of suppressing the Boolean constraint are several: less statistical noise, more numerical accuracy and, importantly, more flexibility to choose the lattice topology, the collision operator and boundary conditions. Thus, for many applications, the LB approach is preferred to the LGA one. On the other hand, LB models do not integrate local fluctuations that are naturally present in a LGA and that can have relevant physical effects (Chopard and Droz 1998).

Several textbooks (Chen and Doolen 1998; Succi 2001; Chopard and Droz 1998; Wolf-Gladrow 2000; Sukop and Thorne 2005) exist, which describe in great detail the LB approach. The method has been used extensively in the literature to simulate complex flows and other physical processes (Chopard and Droz 1998). For hydrodynamics, the LB method is now recognized as a serious competitor to the more traditional approaches based on the computer solution of the Navier–Stokes partial differential equations. Among the advantages of the LB method over more traditional numerical schemes, we mention its simplicity, its flexibility to describe complex flows, and its local nature (no need to solve a Poisson equation). Another feature of the LB method is its extended range of validity when the Knudsen number is not negligible (e.g., in microflows) (Ansumali et al. 2007).

5.1 General Principles

5.1.1 Definitions

The key quantities to define an LB model are the density distributions $f_i(\mathbf{r}, t)$ and the “molecular velocities” \mathbf{v}_i , for $i = 0 \dots z$ where z is the lattice coordination number of the chosen lattice topology and $z + 1$ is the number of discrete velocities. The quantity f_i then denotes the number of particles entering lattice site \mathbf{r} at time t with discrete velocity \mathbf{v}_i . Note that \mathbf{v}_i is a vector so that molecular velocities have a norm and a direction. For instance, a common choice of velocities in 2D problem is

$$\begin{aligned} \mathbf{v}_0 &= (0, 0) & \mathbf{v}_1 &= \nu(1, 0) & \mathbf{v}_2 &= \nu(1, 1) & \mathbf{v}_3 &= \nu(0, 1) & \mathbf{v}_4 &= \nu(-1, 1) \\ \mathbf{v}_5 &= \nu(-1, 0) & \mathbf{v}_6 &= \nu(-1, -1) & \mathbf{v}_7 &= \nu(0, -1) & \mathbf{v}_8 &= \nu(1, -1) \end{aligned} \quad (18)$$

In these expressions, ν is a velocity norm defined as $\nu = \delta_r / \delta_t$, with δ_r being the lattice spacing and δ_t the duration of the time step. Both δ_r and δ_t can be expressed in any desired unit system.

From the f_i 's and the \mathbf{v}_i 's the standard physical quantities such as particle density ρ , particle current $\rho\mathbf{u}$ can be defined, by taking various moments of the distribution

$$\rho(\mathbf{r}, t) = \sum_i f_i(\mathbf{r}, t) \quad \rho(\mathbf{r}, t)\mathbf{u}(\mathbf{r}, t) = \sum_i f_i(\mathbf{r}, t)\mathbf{v}_i \quad (19)$$

The intuitive interpretation of these relations is obvious: The number of particles at point \mathbf{r} and time t is the sum of all particles coming with all velocities; and the total momentum is the sum of momentum carried by each f_i .

In hydrodynamics, it is also important to define higher moments, such as the *momentum tensor*

$$\Pi_{\alpha\beta} = \sum_i f_i(\mathbf{r}, t) v_{i\alpha} v_{i\beta}$$

where Greek subscripts label spatial coordinates. The tensor $\Pi_{\alpha\beta}$ describes the amount of α -momentum transported along the β -axis.

Following the particle interpretation, one can say that in an LB model, all particles entering the same site at the same time from different directions (i.e., particles with different molecular velocities \mathbf{v}_i) collide. As a consequence, a new distribution of particles results. Then, during the next time step δ_t , the particles emerging from the collision move to a new lattice site, according to their new speed. Therefore, the dynamics of an LB model is the alternation of collision and propagation phases.

This is illustrated in [Fig. 23](#) and [Fig. 24](#), for two different lattice topologies in two dimensions (hexagonal and square lattices). In accordance with [Fig. 23](#) and [Fig. 24](#), the LB dynamics can be written as a collision phase

$$f_i^{\text{out}}(\mathbf{r}, t) = f_i^{\text{in}}(\mathbf{r}, t) + \Omega_i(f^{\text{in}}(\mathbf{r}, t)) \quad (20)$$

and a propagation (propagation is often termed *streaming* in the literature) phase

$$f_i^{\text{in}}(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i^{\text{out}}(\mathbf{r}, t) \quad (21)$$

where Ω , the collision term is a model-specific function describing the outcome of the particle collision. When subscript i is omitted, Ω denotes the entire set of Ω_i . Its form will be discussed below. Here, an upper-script in or out to the distribution function f_i is added to distinguish the pre-collision distribution f_i^{in} from the post-collision ones f_i^{out} . When no upper-script is used,

Fig. 23

Illustration of the collision and propagation phases in an LB model defined on a 2D hexagonal lattice, with six possible velocities. The arrows represent the particles, their directions correspond to the \mathbf{v}_i and their length is proportional f_i .

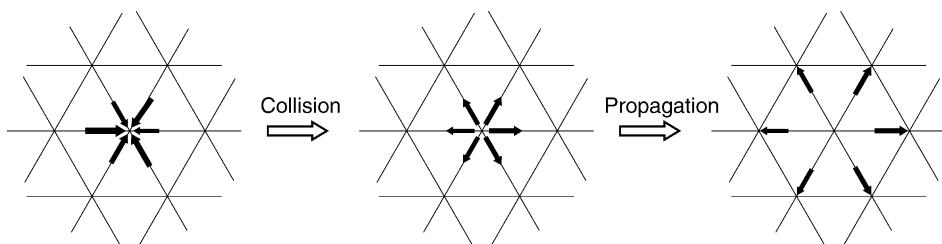
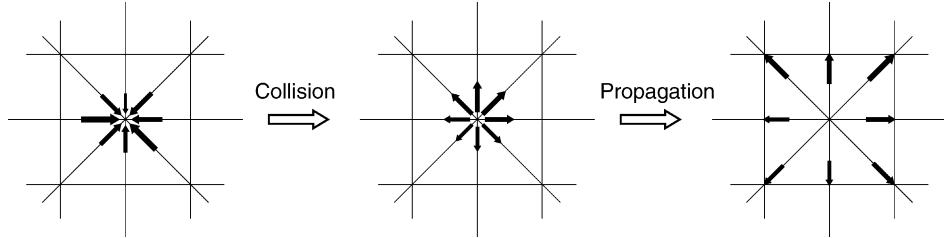


Fig. 24

Illustration of the collision and propagation phases in an LB model defined on a 2D square lattice with eight velocities.



we define that $f = f^{\text{in}}$. Therefore, by combining [Eqs. 20](#) and [21](#), an LB model can also be expressed as

$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \Omega_i(f(\mathbf{r}, t)) \quad (22)$$

Conservation laws play an important role in building an LB model. When some physical quantities are known to be conserved in a given phenomena, this conservation must be reflected exactly by the dynamics of the corresponding LB equation. For instance, if the number of particle ρ is conserved in the collision process, it is required to have

$$\sum_{i=0}^z f_i^{\text{out}}(\mathbf{r}, t) = \sum_{i=0}^z f_i^{\text{in}}(\mathbf{r}, t)$$

for all \mathbf{r} and all t .

From [Eq. 22](#), this means that the collision term must obey

$$\sum_{i=0}^z \Omega_i = 0 \quad (23)$$

Similarly, if momentum is also conserved (as in a fluid), it is required to have

$$\sum_{i=0}^z \mathbf{v}_i f_i^{\text{out}}(\mathbf{r}, t) = \sum_{i=0}^z \mathbf{v}_i f_i^{\text{in}}(\mathbf{r}, t)$$

and then

$$\sum_{i=0}^z \mathbf{v}_i \Omega_i = 0 \quad (24)$$

5.1.2 Lattice Properties

As seen previously, propagation moves particles with velocity \mathbf{v}_i from one lattice site to a neighboring one. As a consequence, $\mathbf{r} + \delta_t \mathbf{v}_i$ must also be a point of the lattice. Thus, there is a tight connection between the spatial lattice and the discrete set of molecular velocities.

In the LB framework, the choice of velocity \mathbf{v}_i (and consequently the corresponding spatial lattice) is commonly labeled as $DdQq$, where d is the spatial dimension ($d = 2$ for a two-dimensional problem) and q is the number of discrete velocities (or quantities).

When q is odd, it is assumed that the model includes a rest speed $\mathbf{v}_0 = 0$. Then velocities \mathbf{v}_i are labeled from $i = 0$ to $q - 1$ and $q = z + 1$, z being the lattice coordination number. When q is even, the model contains no rest speed and the velocities \mathbf{v}_i are labeled from $i = 1$ to $i = q$ and $q = z$. For instance, the D2Q9 lattice corresponds the velocities given in [Fig. 18](#) and illustrated in [Fig. 25](#). A D2Q8 lattice is the same, but without \mathbf{v}_0 . To build a proper LB model, the \mathbf{v}_i should be carefully chosen. In addition to the fact that $\mathbf{r} + \delta_i \mathbf{v}_i$ must correspond to a lattice site, the molecular velocities must have enough symmetry and isotropy properties. In short, tensors built by summing the velocity components should have enough rotational invariance to represent the physical process under consideration. To achieve this goal, it is often necessary to add weights w_i to velocity vector \mathbf{v}_i , as suggested in the right panel of [Fig. 25](#). In practice it is required that

$$\sum_i w_i = 1 \quad \sum_i w_i \mathbf{v}_i = 0 \quad \sum_i w_i v_{ix} v_{i\beta} = c_s^2 \delta_{x\beta} \quad (25)$$

where $\delta_{x\beta}$ is the Kronecker symbol and c_s a coefficient to be determined. For a D2Q5 lattice $\mathbf{v}_0 = 0$, $\mathbf{v}_1 = (\nu, 0)$, $\mathbf{v}_2 = (0, \nu)$, $\mathbf{v}_3 = (-\nu, 0)$, and $\mathbf{v}_4 = (0, -\nu)$, the above set of conditions is easily fulfilled by choosing $w_1 = w_2 = w_3 = w_4 = (1 - w_0)/4 > 0$ and $c_s^2 = \nu^2(1 - w_0)/2$ because \mathbf{v}_i and \mathbf{v}_{i+1} are orthogonal 2D vectors.

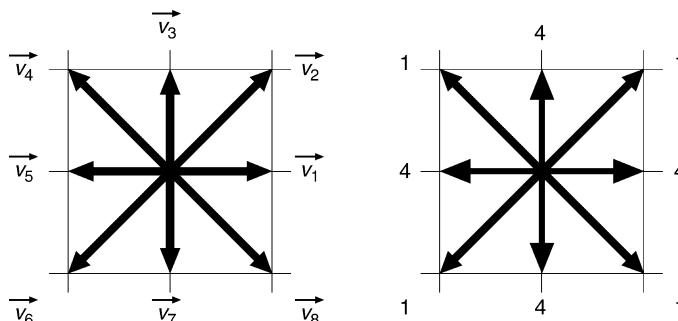
Conditions [\(25\)](#) are sufficient to model diffusion processes or wave propagation (Chopard and Droz 1998). However, to model nonthermal hydrodynamic flows, they are not enough and must be supplemented by conditions on the third-, fourth-, and fifth-order tensors that can be built out of the \mathbf{v}_i 's. These conditions read

$$\begin{aligned} \sum_i w_i v_{ix} v_{i\beta} v_{i\gamma} &= 0 \\ \sum_i w_i v_{ix} v_{i\beta} v_{i\gamma} v_{i\delta} &= c_s^4 (\delta_{x\beta} \delta_{\gamma\delta} + \delta_{x\gamma} \delta_{\beta\delta} + \delta_{x\delta} \delta_{\beta\gamma}) \\ \sum_i w_i v_{ix} v_{i\beta} v_{i\gamma} v_{i\delta} v_{ic} &= 0 \end{aligned} \quad (26)$$

For thermo-hydrodynamic models, even higher order conditions must be considered, forcing one to add more discrete velocities to the system (Shan et al. 2006). The reader is directed to

Fig. 25

The D2Q9 lattice with nine velocities, corresponding to a square lattice, including diagonals. Note that $\mathbf{v}_0 = (0, 0)$ is not shown on the figure. The right panel shows the ratio of the weights associated with every direction: the diagonal directions should have a weight four times smaller than the main directions in order to ensure the isotropy of the fourth-order tensor.



LB textbooks to better understand the origin and meaning of these isotropy conditions. Here, they shall be just accepted as a requirement on a proper choice of velocity sets. They can be satisfied for a D2Q9 model by taking

$$w_0 = 4/9 \quad w_1 = w_3 = w_5 = w_7 = 1/9 \quad w_2 = w_4 = w_6 = w_8 = 1/36$$

for which one gets $c_s^2/v^2 = 1/3$.

Three-dimensional models, such as the well-known D3Q19 model, can also be constructed to satisfy [Eqs. 25](#) and [26](#). In the D3Q19 model, the velocity vectors are defined as

$$\begin{aligned} \mathbf{v}_0 &= 0 \\ \mathbf{v}_1 &= v(-1, 0, 0) & \mathbf{v}_2 &= v(0, -1, 0) & \mathbf{v}_3 &= v(0, 0, -1) \\ \mathbf{v}_4 &= v(-1, -1, 0) & \mathbf{v}_5 &= v(-1, 1, 0) & \mathbf{v}_6 &= v(-1, 0, -1) \\ \mathbf{v}_7 &= v(-1, 0, 1) & \mathbf{v}_8 &= v(0, -1, -1) & \mathbf{v}_9 &= v(0, -1, 1) \\ \mathbf{v}_{10} &= v(1, 0, 0) & \mathbf{v}_{11} &= v(0, 1, 0) & \mathbf{v}_{12} &= v(0, 0, 1) \\ \mathbf{v}_{13} &= v(1, 1, 0) & \mathbf{v}_{14} &= v(1, -1, 0) & \mathbf{v}_{15} &= v(1, 0, 1) \\ \mathbf{v}_{16} &= v(1, 0, -1) & \mathbf{v}_{17} &= v(0, 1, 1) & \mathbf{v}_{18} &= v(0, 1, -1) \end{aligned} \quad (27)$$

and the lattice properties are

$$c_s^2/v^2 = 1/3 \quad w_0 = 1/3 \quad w_{\text{slow}} = 1/18 \quad w_{\text{fast}} = 1/36$$

where w_{slow} concerns the \mathbf{v}_i of norm v and w_{fast} the \mathbf{v}_i of norm $\sqrt{2}v$.

5.2 Lattice BGK Models

We now return to the LB [Eq. 22](#)

$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \Omega_i(f(\mathbf{r}, t)) \quad (28)$$

and consider a special family of collision terms: the so-called *single relaxation time* models, also termed lattice BGK models (LBGK) for its correspondence with the BGK form of the continuous Boltzmann equation (Bhatnager et al. 1954).

Although more sophisticated models exist (d'Humières et al. 2002; Latt and Chopard 2006; Chikatamarla et al. 2006), the LBGK is still the most popular version of an LB model. It reads

$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \frac{1}{\tau} (f_i^{\text{eq}} - f_i) \quad (29)$$

where f^{eq} is called the *local equilibrium* distribution; it is a given function, which depends on the phenomena that one wants to model (note that when one refers to all f_i or all f_i^{eq} , the subscript i is dropped). The quantity τ is the so-called *relaxation time*. It is a parameter of the model, which is actually related to the transport coefficient of the model: viscosity for a fluid model, diffusion constant in case of a diffusion model.

In [Eq. 29](#), it is important to note that the local equilibrium distribution f^{eq} depends on space and time only through the conserved quantities. This is a common assumption of statistical physics. In a hydrodynamic process, where both mass and momentum are conserved, f^{eq} will then be a function of ρ and \mathbf{u} .

Thus, in [Eq. 29](#), to compute $f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t)$ from the $f_i(\mathbf{r}, t)$ one first has to compute $\rho = \sum f_i$ and $\mathbf{u} = (1/\rho) \sum f_i \mathbf{v}_i$ before computing $f_i^{\text{eq}}(\rho, \mathbf{u})$. Only then can f_i be updated.

It is beyond the scope of this chapter to show the equivalence between the LB model and the differential equations representing the corresponding physical phenomena. This derivation requires rather heavy mathematical calculations and can be found in several textbooks. See for instance Chopard and Droz (1998), Chopard et al. (2002), and Lätt (2007) for a derivation based on the so-called multiscale *Chapman–Enskog* formalism. Or, see Junk et al. (2005) for a derivation based on the *asymptotic expansion*. Here the important results are simply given, without demonstration.

5.2.1 LBGK Fluid Models

A first central ingredient of LB models is to properly enforce the physical conservation laws in the collision term. Hydrodynamics is characterized by mass and momentum conservation, which, in the differential equation language, are expressed by the *continuity* and *Navier–Stokes* equations.

From [Eqs. 23](#) and [24](#), conservation laws impose conditions on f_i^{eq} when an LBGK model is considered, namely

$$\sum_i f_i^{\text{eq}} = \sum_i f_i = \rho \quad \sum_i \mathbf{v}_i f_i^{\text{eq}} = \sum_i \mathbf{v}_i f_i = \rho \mathbf{u} \quad (30)$$

In addition, in order to recover a hydrodynamic behavior, one imposes that $\Pi_{\alpha\beta}^{\text{eq}}$, the second moment of f^{eq} , which is the non-dissipative part of the momentum tensor, has the standard Euler form

$$\Pi_{\alpha\beta}^{\text{eq}} = \sum_i f_i^{\text{eq}} v_{i\alpha} v_{i\beta} = p \delta_{\alpha\beta} + \rho u_\alpha u_\beta \quad (31)$$

where p is the pressure.

Using [Eqs. 25](#) and [26](#), it is easy to show that the following expression for f^{eq} satisfies the conservation laws ([30](#))

$$f_i^{\text{eq}} = f_i^{\text{eq}}(\rho, \mathbf{u}) = \rho w_i \left(1 + \frac{v_{i\alpha} u_\alpha}{c_s^2} + \frac{1}{2c_s^4} Q_{i\alpha\beta} u_\alpha u_\beta \right) \quad (32)$$

where the Q_i 's are tensors whose spatial components are

$$Q_{i\alpha\beta} = v_{i\alpha} v_{i\beta} - c_s^2 \delta_{\alpha\beta} \quad (33)$$

Note that in this equation and in what follows, the Einstein summation convention is used over repeated Greek indices

$$v_{i\alpha} u_\alpha = \sum_{\alpha=x,y,z} v_{i\alpha} u_\alpha$$

and

$$Q_{i\alpha\beta} u_\alpha u_\beta \equiv \sum_{\alpha,\beta \in \{x,y,z\}} Q_{i\alpha\beta} u_\alpha u_\beta$$

Note that f_i^{eq} can also be interpreted as a discretization of the Maxwell–Boltzmann distribution function of statistical physics.

One can also check that the second moment of \bullet Eq. 32 gives the correct expression for the Euler momentum tensor (\bullet Eq. 31), provided that the pressure is related to the density ρ through an ideal gas relation

$$p = \rho c_s^2$$

From this expression, one can interpret the lattice parameter c_s as the speed of sound.

The fact that, in an LB model, the pressure is directly obtained from the density is an important observation. It means that in an LB fluid model, there is no need to solve a (nonlocal) Poisson equation for the pressure, as is the case when solving Navier–Stokes equations.

Using expression (\bullet 32) for f_i^{eq} , the behavior of the LB model (\bullet 29) can be analyzed mathematically with, for instance a Chapman–Enskog method. Several important results are obtained.

It is found that, to order δ_t^2 and δ_r^2 , and for small Mach number ($\mathbf{u} \ll c_s$), the LB dynamics implies that ρ and \mathbf{u} obey the continuity equation

$$\partial_t \rho + \partial_x \rho u_x = 0 \quad (34)$$

and the Navier–Stokes equation

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + v \nabla^2 \mathbf{u} \quad (35)$$

with a kinematic viscosity v depending on the relaxation time τ as

$$v = c_s^2 \delta_t (\tau - 1/2)$$

The last question that needs to be addressed here is how to obtain the expression of f in terms of the hydrodynamic quantities. It is easy to obtain ρ and \mathbf{u} from the f_i using \bullet Eq. 19. But the inverse relations, expressing f_i as a function of ρ and \mathbf{u} is more difficult. There are $z+1$ variables f_i and only $1+d$ hydrodynamic quantities in d dimensions. However, in the hydrodynamic limit, it turns out that the derivatives of \mathbf{u} are precisely the missing pieces of information.

The first step is to split the density distributions f_i as

$$f_i = f_i^{\text{eq}} + f_i^{\text{neq}} \quad \text{assuming } f_i^{\text{neq}} \ll f_i^{\text{eq}}$$

where f_i^{eq} , by its definition (\bullet 32) is already a function of ρ and \mathbf{u}

$$f_i^{\text{eq}} = f_i^{\text{eq}}(\rho, \mathbf{u}) = \rho w_i \left(1 + \frac{\mathbf{v}_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4} Q_{iz\beta} u_z u_\beta \right) \quad (36)$$

The Chapman–Enskog expansion then gives

$$f_i^{\text{neq}} = -\delta_t \tau \frac{w_i}{c_s^2} \rho Q_{iz\beta} \partial_z u_\beta = -\delta_t \tau \frac{w_i}{c_s^2} \rho Q_{iz\beta} S_{z\beta} \quad (37)$$

where $S_{z\beta} = (1/2)(\partial_z u_\beta - \partial_\beta u_z)$ is the so-called strain rate tensor. As one can see from this relation, the derivatives of \mathbf{u} are part of the LB variables.

By taking the second moment of \bullet Eq. 37 one can obtain $\Pi_{z\beta}^{\text{neq}}$, the nonequilibrium part of the momentum tensor. Due to the lattice properties (\bullet 25) and (\bullet 26),

$$\Pi_{z\beta}^{\text{neq}} = \sum_i v_{iz} v_{i\beta} f_i^{\text{neq}} = -2\delta_t \tau c_s^2 \rho S_{z\beta} \quad (38)$$

Thus, the strain rate tensor can be directly obtained from $f^{\text{neq}} = f - f^{\text{eq}}$, without the need to compute finite differences.

Therefore, with \blacktriangleright Eqs. 19, \blacktriangleright 36, and \blacktriangleright 37, a relation that allows one to translate the hydrodynamic quantities to LB quantities and vice-versa is established

$$\begin{pmatrix} \rho \\ \mathbf{u} \\ S_{\alpha\beta} \end{pmatrix} \leftrightarrow (f_i) \quad (39)$$

They are valid in the hydrodynamic regime, with $u \ll c_s$ and for $\delta_t \rightarrow 0$ and $\delta_r \rightarrow 0$.

In order to illustrate the LBGK method, \blacktriangleright Fig. 26 shows an example of an LB fluid simulation, the flow past an obstacle.

5.3 Diffusion and Reaction–Diffusion LBGK Models

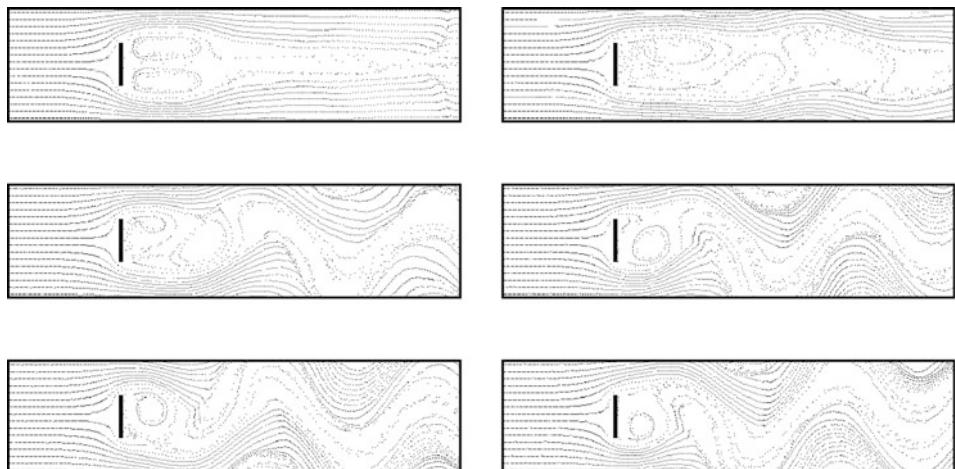
It is actually very easy to devise an LBGK model to describe other physical processes. The basic equation remains the same, namely

$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \frac{1}{\tau} (f_i^{\text{eq}} - f_i) \quad (40)$$

What changes is the expression for f^{eq} and also the isotropy requirements of the lattice. For instance, for a diffusion model, \blacktriangleright Eq. 25 is sufficient because diffusion does not involve fourth-order isotropy constraints. In 2D, a D2Q4 lattice (having $w_i = 1/4$, $c_s^2/v^2 = 1/2$) is enough to model a diffusion process and, in 3D, a D3Q6 lattice ($w_i = 1/6$, $c_s^2/v^2 = 1/2$) has enough isotropy.

Fig. 26

Several stages (from left to right and top to bottom) of the evolution of a flow past an obstacle, using a D2Q9 LBGK fluid model.



5.3.1 Diffusion

To build the local equilibrium distribution corresponding to a diffusion model, we notice that only the density $\rho = \sum f_i$ is conserved in the process. Then, it is found that

$$f_i^{\text{eq}} = w_i \rho \quad (41)$$

produces the diffusion equation

$$\partial_t \rho = D \nabla^2 \rho$$

where

$$D = c_s^2 \delta_t \left(\tau - \frac{1}{2} \right)$$

is the diffusion coefficient. Also, it is found that the particle current $\mathbf{j} = -D \nabla \rho$ can be computed from the non-equilibrium part of the distribution function, $f^{\text{neq}} = f - f^{\text{eq}} = -\tau \delta_t w_i v_{ix} \partial_x \rho$ as

$$\mathbf{j} = \left(1 - \frac{1}{2\tau} \right) \sum_i \mathbf{v}_i f_i^{\text{neq}} = \left(1 - \frac{1}{2\tau} \right) \sum_i \mathbf{v}_i f_i$$

Advection–diffusion processes

$$\partial_t \rho + \partial_x \rho u_x = D \nabla^2 \rho$$

where $\mathbf{u}(\mathbf{r}, t)$ is a given velocity field, can be modeled by adding an additional term to the local equilibrium (❸ Eq. 41)

$$f_i^{\text{eq}} = w_i \rho \left(1 + \frac{1}{c_s^2} \mathbf{u} \cdot \mathbf{v}_i \right) \quad (42)$$

If \mathbf{u} is the solution of a fluid flow, it is appropriate to add a term $w_i \rho (1/(2c_s^4)) Q_{i\alpha\beta} u_\alpha u_\beta$ to ❸ Eq. 42. See Van der Sman and Ernst (2000), Suga (2001), Chopard et al. (2009), Ginzburg (2005), and Servan-Camas and Tsai (2008) for more details.

5.3.2 Reaction–Diffusion

In order to simulate a reaction–diffusion process

$$\partial_t \rho = D \nabla^2 \rho + R(\rho)$$

where R is any reaction term, the LB diffusion model can be modified as

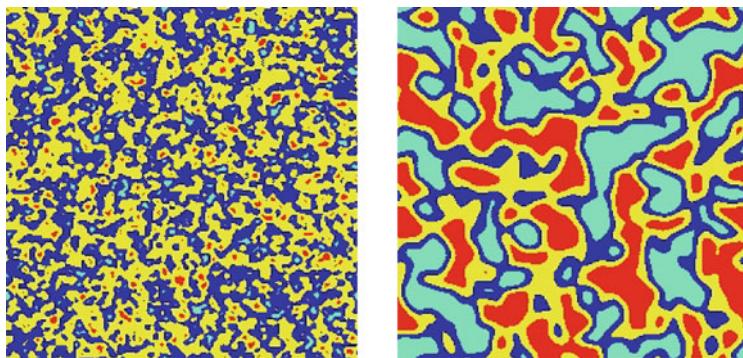
$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \frac{1}{\tau} (f_i^{\text{eq}} - f_i) + \delta_t w_i R(\rho) \quad (43)$$

For instance, the reaction $A + B \rightarrow C$ where C is some inert product can be simulated by two sets of equations (❸ Eq. 43), one for species A and one for species B. The reaction term is chosen as $R = -k \rho_A \rho_B$. ❸ Figure 27 shows the evolution of the concentrations of both species when they are initially mixed.

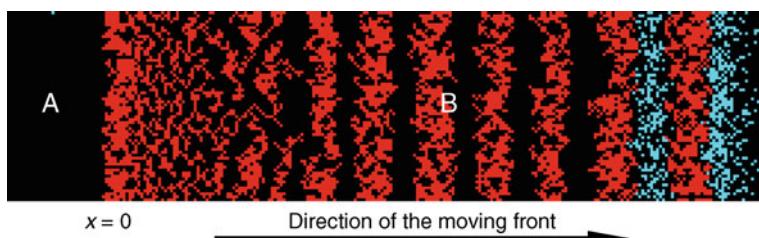
Another example of a reaction–diffusion process is shown in ❸ Fig. 28. See Chopard et al. (1994) for more details.

Fig. 27

A reaction–diffusion model simulating the $A + B \rightarrow C$ reaction. The left panel shows an early stage and the right panel a later stage of the reaction process. Red denotes regions very rich in A, yellow indicates regions where A dominates over B. Similarly, green regions are those very rich in B and blue those where B is slightly more abundant than A.

**Fig. 28**

A reaction–diffusion model simulating the formation of Liesegang bands in an $A + B \rightarrow C$ process, where C can precipitate and forms bands at positions that have interesting geometrical properties.



5.4 Wave Propagation

Finally, *wave propagation* can also be described by an LBGK model (Chopard and Droz 1998; Chopard et al. 2002). In this case, it is a must to choose $\tau = 1/2$, which ensures the time reversibility of the LB dynamics (Chopard and Droz 1998), a required symmetry of the wave equation.

Wave processes have two conserved quantities, $\rho = \sum f_i$, which can be any scalar quantity obeying a wave process, and its current $\mathbf{j} = \sum f_i \mathbf{v}_i$. However, as opposed to flow and diffusion models, the f_i are no longer positive quantities and they oscillate between a minimal negative value and a maximum positive value.

The appropriate form of the local equilibrium is found to be

$$\begin{aligned} f_i^{\text{eq}} &= w_i \rho + w_i \frac{\mathbf{j} \cdot \mathbf{v}_i}{c_s^2} \\ f_0^{\text{eq}} &= w_0 \rho \end{aligned} \tag{44}$$

In the continuous limit, \otimes Eq. 40 with \otimes Eq. 44 yield

$$\begin{aligned}\partial_t \rho + \partial_\beta j_\beta &= 0 \\ \partial_t j_\alpha - c_s^2 \partial_\alpha \rho &= 0\end{aligned}\tag{45}$$

When combined, these two equations give the wave equation

$$\partial_t^2 \rho - c_s^2 \nabla^2 \rho = 0\tag{46}$$

The rest population f_0 allows us to adjust the speed of the wave from place to place by having the value of w_0 depend on spatial location \mathbf{r} or time t .

As for the diffusion case, a second-order isotropy is enough for the wave equation. Therefore D2Q5 and D3Q7 lattices are appropriate. For other topologies, it is important to use the same weight $w_i = w$ for all nonzero velocities. It means that the second-order isotropy condition now reads

$$\sum_{i \geq 1} v_{i\alpha} v_{i\beta} = z c_{\max}^2 \delta_{\alpha\beta}\tag{47}$$

for some coefficient c_{\max} and for z the lattice coordination number. For D2Q5 and D3Q7, it is easy to show that $c_{\max}^2/v^2 = 2/z$.

When $w_i = w$, one can conclude from the condition

$$1 = \sum_{i \geq 0} w_i = w_0 + zw$$

that

$$w = \frac{1 - w_0}{z}\tag{48}$$

Therefore

$$\sum_{i \geq 1} w_i v_{i\alpha} v_{i\beta} = (1 - w_0) c_{\max}^2 \delta_{\alpha\beta}\tag{49}$$

and we obtain

$$c_s^2 = (1 - w_0) c_{\max}^2\tag{50}$$

A consequence of this relation is that $w_0 < 1$ must be imposed. Note that this way of adjusting c_s is only possible for processes that do not require fourth-order isotropy. For a fluid model, there is however a way to adjust the speed of sound (Alexander et al. 1992; Yu and Zhao 2000; Chopard et al. 2002).

The numerical stability of the LBGK wave model is guaranteed because a quantity, termed *energy*,

$$E = \frac{w}{w_0} f_0^2 + \sum_{i \geq 1} f_i^2\tag{51}$$

is conserved during the collision step

$$E^{\text{out}} = \frac{w}{w_0} (f_0^{\text{out}})^2 + \sum_{i \geq 1} (f_i^{\text{out}})^2 = \frac{w}{w_0} (f_0^{\text{in}})^2 + \sum_{i \geq 1} (f_i^{\text{in}})^2 = E^{\text{in}}\tag{52}$$

The proof of this condition also requires that $w_i = w$ for $i \neq 0$. When $(w/w_0) \geq 0$, the numerical stability of the model is guaranteed because, with E given, the f_i cannot diverge to $\pm\infty$.

With $0 \leq w_0 < 1$ we obtain from (☞ Eq. 50) that the speed of the wave is such that

$$0 \leq c_s \leq c_{\max}$$

the maximum speed being achieved with $w_0 = 0$ that is, without a rest population f_0 . The refraction index is defined as

$$n = \frac{c_{\max}}{c_s}$$

Using ☞ Eqs. 48 and ☞ 50, one then obtains

$$w = \frac{1}{z} \frac{c_s^2}{c_{\max}^2} = \frac{1}{n^2 z} \quad w_0 = 1 - \frac{c_s^2}{c_{\max}^2} = \frac{n^2 - 1}{n^2} \quad (53)$$

With the above value of w_i and the fact that $\tau = 1/2$, the LBGK wave model can also be written as

$$\begin{aligned} f_i^{\text{out}} &= \frac{2}{n^2 z} \rho + \frac{2}{z c_{\max}^2} \mathbf{v}_i \cdot \mathbf{j} - f_i \\ f_0^{\text{out}} &= 2 \frac{n^2 - 1}{n^2} \rho - f_0 \end{aligned} \quad (54)$$

☞ *Figure 29* illustrates this model with a D2Q5 lattice. A plane wave is produced on the left side of the domain and propagates to the right where it penetrates in a lens-shaped media with slower propagation speed.

Note that in Chopard and Droz (1998), Chopard et al. (2002), Marconi and Chopard (2003) the above model is investigated for its capability to model deformable elastic solids. ☞ *Figure 30* illustrates an application of the wave model to describe a fracture process.

Finally, note that the LBGK wave model is also known in the literature as the transmission line matrix model (Hoeffner 1985) and has been derived in several different contexts (Vanneste et al. 1992).

■ **Fig. 29**

LB model simulating the propagation of a wave in a lens. The colors represent the energy E .

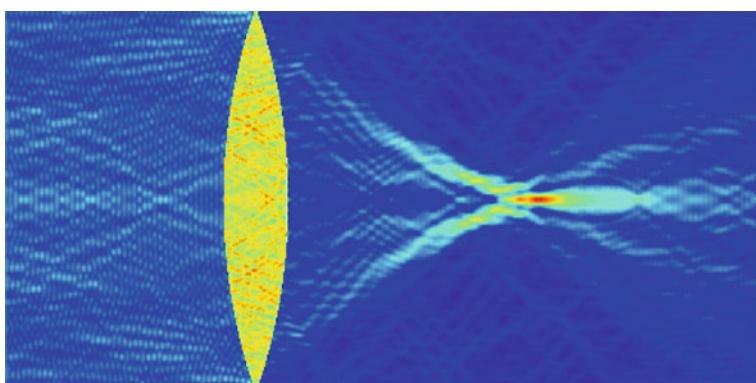
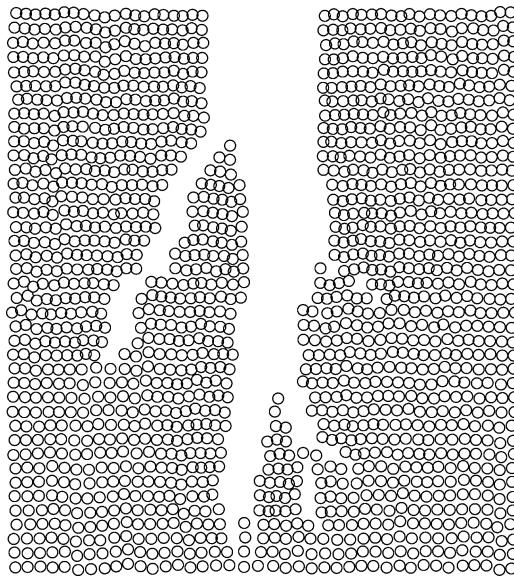
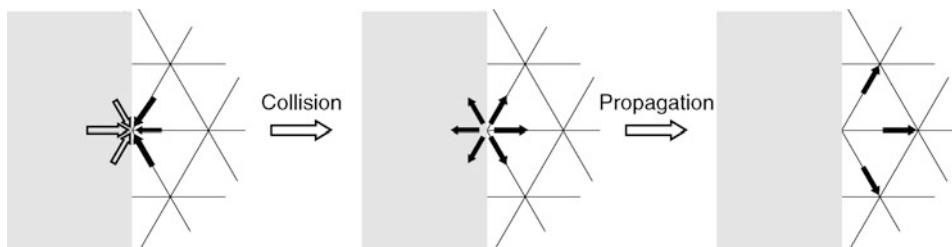


Fig. 30

Fracture process in an LB model for an elastic solid body.

**Fig. 31**

At the boundary of the domain, the density distributions coming from outside the system are not known. Specific boundary conditions must be used to define them.



5.5 Boundary Conditions

Boundary conditions are an important aspect of an LB model. It is not an easy question to properly specify the values of the distributions f_i at the limit of the computational domain. Clearly, to apply the collision phase, all f_i must be defined. But at a boundary cell, the propagation phase does not provide any information from outside the domain. This is illustrated in [Fig. 31](#). These unknown distributions must be specified according to the desired behavior of the system at the boundary.

Following this procedure, the time evolution of an LB model can then be represented by the following loop in a computer program:

```
for t=0 to tmax
    boundary
    observation
    collision
    propagation
endfor
```

Note that here a new operation, termed *observation* in the LB execution loop, has been introduced. This is where measurement can be done on the system. It is important to remember that f_i actually denotes f^{in} so that the theoretical results showing the correspondence between LB and physical quantities are only valid after the propagation step and before collision. Obviously, quantities such as ρ that are conserved during the collision can be measured after collision too. But, in hydrodynamics, this is not the case of the strain rate $S_{\alpha\beta}$.

In what follows, we focus on the discussion on simple hydrodynamical LB models. In practice, the way to impose a boundary condition is to use the correspondence expressed in [Eq. 39](#) to build the missing f_i from the desired values of the fluid variables at the boundary.

A standard situation is shown in [Fig. 32](#), for a D2Q9 lattice. The density distributions f_2 , f_3 , and f_4 must be determined before applying the collision operator. Assuming that the y -axis is vertical and pointing upward, we have the following relations (in lattice units where $v = 1$)

$$\begin{aligned}\rho &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \\ \rho u_y &= f_2 + f_3 + f_4 - f_6 - f_7 - f_8\end{aligned}\tag{55}$$

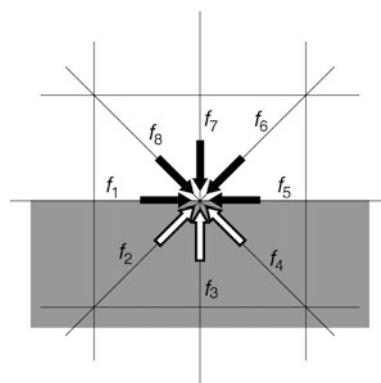
From these equations, we get

$$\rho - \rho u_y = f_0 + f_1 + f_5 + 2(f_6 + f_7 + f_8)\tag{56}$$

The right-hand side of this equation is fully known.

Fig. 32

For a flat wall at the lower boundary of a D2Q9 lattice, the distributions f_2 , f_3 , and f_4 are unknown and must be computed according to the desired boundary condition.



In case $\mathbf{u} = (u_x, u_y)$ is specified at the boundary (velocity boundary condition), we can compute ρ consistently (Inamuro et al. 1995)

$$\rho = \frac{f_0 + f_1 + f_5 + 2(f_6 + f_7 + f_8)}{1 - u_y}$$

Once ρ and \mathbf{u} are known, $f_i^{\text{eq}}(\rho, \mathbf{u})$ is also known for all i , due to \blacktriangleright Eq. 36. Then, we can certainly compute $f_i^{\text{neq}} = f_i - f_i^{\text{eq}}$ for $i \in \{0, 1, 5, 6, 7, 8\}$.

From these quantities, we can now compute f_i^{neq} for the missing directions $i = 2, 3, 4$ by imposing that their nonequilibrium part obeys \blacktriangleright Eq. 37, which tells us that $f_i^{\text{neq}} = f_{\text{opp}(i)}^{\text{neq}}$, where $j = \text{opp}(i)$ is the index such that $\mathbf{v}_j = -\mathbf{v}_i$.

Unfortunately, there is no guarantee that this choice of the nonequilibrium parts of the distributions is globally consistent. For instance, one may well find that $\sum_i f_i^{\text{neq}} \neq 0$, which is inconsistent with \blacktriangleright Eq. 55. The solution to this problem is a *regularization* step in which the f_i^{neq} are redistributed over all directions. This step can be explained as follows: first we compute

$$\Pi_{\alpha\beta}^{\text{neq}} = \sum_i v_{iz} v_{i\beta} f_i^{\text{neq}}$$

from the f_i^{neq} obtained above. Second, by combining \blacktriangleright Eqs. 37 and \blacktriangleright 38 we get

$$\begin{aligned} f_i^{\text{neq}} &= -\delta_t \tau \frac{w_i}{c_s^2} \rho Q_{iz\beta} S_{z\beta} \\ &= \frac{w_i}{2c_s^4} Q_{iz\beta} \Pi_{\alpha\beta}^{\text{neq}} \end{aligned} \quad (57)$$

This equation allows one to recompute all $f_i^{\text{neq}} (i = 0, \dots, z)$ from the previous ones. All the f_i are then redefined to their regularized value

$$f_i = f_i^{\text{eq}} + f_i^{\text{neq}} \quad i = 0, \dots, z$$

This terminates the calculation of the boundary condition for \mathbf{u} imposed at the wall and guarantees the proper values of ρ and \mathbf{u} at the wall since $\sum_i w_i Q_{iz\beta} = \sum_i w_i \mathbf{v}_i Q_{iz\beta} = 0$.

If, instead of \mathbf{u} , ρ is the prescribed quantity at the boundary (pressure boundary condition), the consistent flow speed u_y can be determined from \blacktriangleright Eq. 56

$$u_y = 1 - \frac{f_0 + f_1 + f_5 + 2(f_6 + f_7 + f_8)}{\rho}$$

By choosing for instance $u_x = 0$ we then obtain f_i^{eq} , for all i . Then, the same regularization procedure as explained above can be used to compute all the f_i at the wall.

The reader can refer to Zou and He (1997) and Latt et al. (2008) for a detailed discussion of the above on-site boundary condition and to Bouzidi et al. (2001), Guo et al. (2002b), Lallemand and Luo (2003), and Kao and Yang (2008) for a discussion of boundary conditions that are not located on lattice sites, or moving boundary conditions.

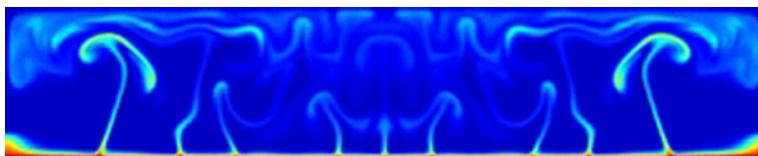
However, another simple solution to specify a boundary condition is to exploit the mesoscopic interpretation of the f_i as particles traveling with velocity \mathbf{v}_i . Following this idea, a very popular way to impose a boundary with zero velocity (no-slip condition) is to bounce back the particles from where they came. This is illustrated in \blacktriangleright Fig. 33b. Using such

Fig. 33

Boundary conditions based on the particle interpretation of the LB method. The white arrows represent f^{in} and the black ones indicate f^{out} . (a) A free slip condition (specular reflection). (b) The bounce back rule to create a no-slip boundary condition on a wall.

**Fig. 34**

A lattice Boltzmann simulation of a Rayleigh–Benard convection.



a bounce-back condition actually means a redefinition of the collision operator of the LB model on the boundary cells

$$f_i^{\text{out}} = f_{\text{opp}(i)}^{\text{in}}$$

where $\text{opp}(i)$ is the direction such that $\mathbf{v}_i = -\mathbf{v}_{\text{opp}(i)}$.

In addition to the bounce-back rule, periodic boundary condition can be used when appropriate. For instance, when simulating the flow in a straight tube, one can sometimes say that the particles leaving the tube through the outlet are reinjected in the inlet. In such a situation it is also convenient to add a body force \mathbf{F} to create and maintain the flow in the tube. The LBGK fluid model can then be modified as follows

$$f_i(\mathbf{r} + \delta_t \mathbf{v}_i, t + \delta_t) = f_i(\mathbf{r}, t) + \frac{1}{\tau} (f_i^{\text{eq}} - f_i) + w_i \frac{\delta_t}{C_s^2} \rho \mathbf{v}_i \cdot \mathbf{F} \quad (58)$$

in order to reproduce a Navier–Stokes equation with external force

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F} \quad (59)$$

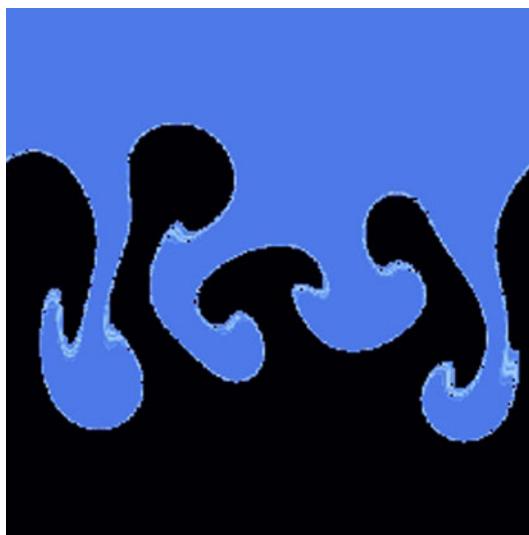
Note that this way of adding a body force on an LBGK fluid model is only possible if \mathbf{F} is constant in space and time. For a more general body force, refer to Guo et al. (2002a).

5.6 Examples of LB Modeling

The LB method has been used in many cases, from sediment transport to blood flow and clotting processes, to shallow water flow, just to mention a few applications. For the sake of illustration, the result of some simulations obtained with an LB model for complex flows is shown in this section without detailed explanations. More examples and on-line animations can be found on the Web (<http://cui.unige.ch/~chopard/CA/Animations/root.html> and <http://www.lbmethod.org>) and in a vast body of literature on LB models.

Fig. 35

Rayleigh–Taylor instability with two immiscible LB fluids. The blue fluid is heavier than the black one.

**Fig. 36**

Evolution of bubbles in a shear flow.

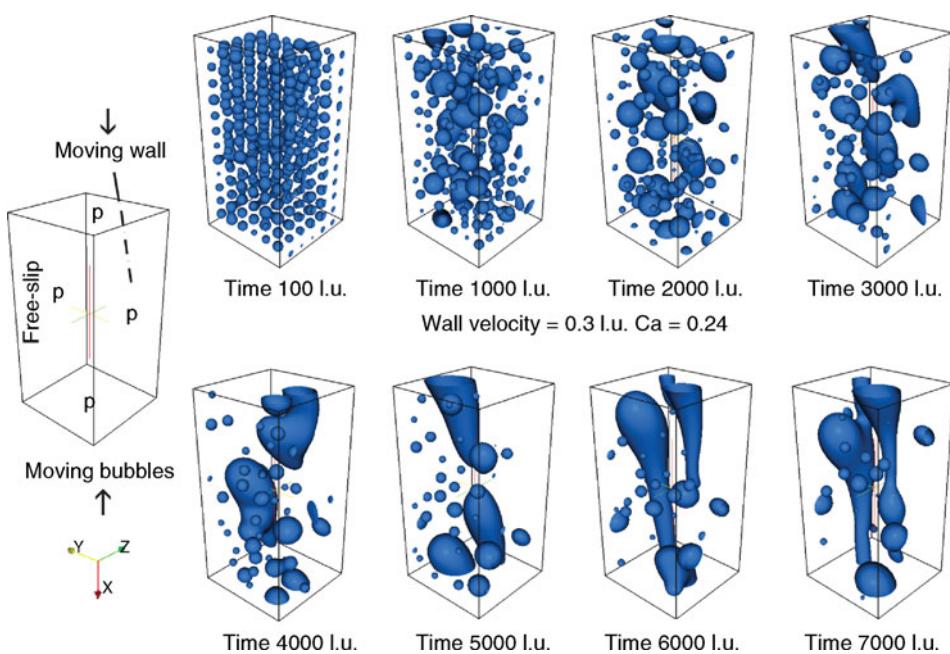
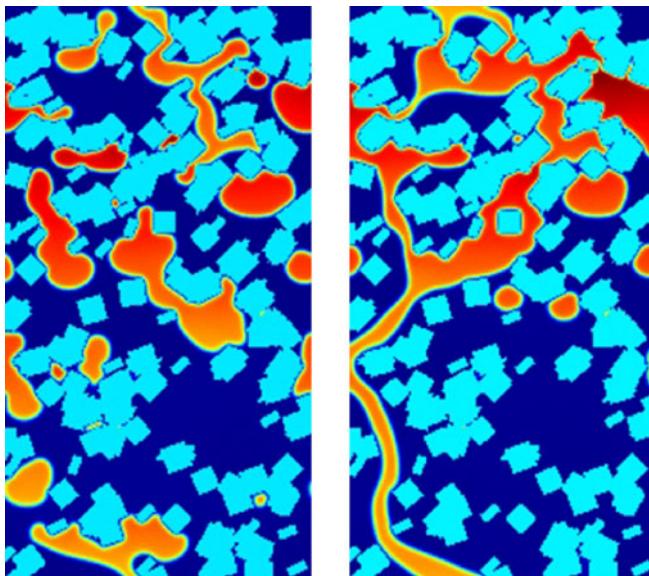


Fig. 37

Bubbles rising in a fluid inside a porous media.



By coupling an advection–diffusion LB model for a temperature field with a fluid LB model with gravity force, it is possible to model a thermal fluid in the Boussinesq approximation and, in particular, the well-known Rayleigh–Benard convection (instability that appears when a fluid is heated from below). This is illustrated in [Fig. 34](#).

An interesting capability of the LB fluid models is to consider multicomponent flows. In short, two-fluid models are implemented with, in addition, a repulsive coupling between them. Several ways exist to create the separation of the components (Martis and Chen 1996; Swift et al. 1996). [Figure 35](#) shows a simulation of the Rayleigh–Taylor instability, which occurs when a heavy fluid layer is on top of a lighter one. [Figure 36](#) represents the evolution of droplets in suspension in another fluid subject to a shear flow.

[Figure 37](#) illustrates a two-component system (gas+liquid), in a porous media, with a buoyancy force field. Bubbles of gas are produced at the lower part of the system and they rise inside a porous media.

6 Conclusions

In this chapter, the concepts underlying the CA and LB approaches are presented. The ideas have been developed using a mesoscopic level of description. Physical systems are represented by idealized entities evolving on a discrete space–time universe. Implementing the correct conservation laws and symmetry in this virtual discrete mesoscopic universe ensures that the observed macroscopic behavior of the system is close enough to that of the corresponding physical system.

The clear relation that exists between the model and the real process makes CA and LB methods very intuitive. As such, they offer a new language to describe physical systems (as illustrated in Fig. 1), which has proven quite powerful in interdisciplinary research.

The simplicity and flexibility of the approach makes it very appropriate to describe many complex systems for which more traditional numerical methods are difficult to apply.

CA are by definition fully discrete numerical models. Usually, they have only a few possible states per cell and they can be implemented very efficiently on low cost dedicated hardware. Since CA models only require integer values, their numerical implementation is exact. No numerical instabilities or truncation error affect the simulation.

LB models offer a higher level of abstraction and consider real-valued variables. They offer much more flexibility than CA to implement a given interaction rule. However, they can be numerically unstable. LB methods are now acknowledged as a powerful way to simulate hydrodynamics and specifically complex fluids in time-dependent regimes. The reader interested to use the LB method can learn more in the several textbooks mentioned earlier and by exploring the Web site www.lbmmethod.org.

A current challenging issue in computational science is multiscale, multiscience modeling. Many real-life problems cover a wide range of spatial and temporal scales and include different processes. Biomedical applications are a good example where physical processes (e.g., hemodynamics) interact with much slower biological ones at a microscopic scale (e.g., tissue modifications). Often, scales can be separated in the sense that the full system can be represented as several coupled submodels, each of them corresponding to a given scale and a given process.

Coupling different CA and LB models together is possible and has been investigated recently (Hoekstra et al. 2008a, b) using the concept of CxA (*Complex Automata*). A CxA is a graph whose nodes are CA or LB models and the edges implement the coupling strategies. A software framework (Hegewald et al. 2008) offers a way to realize such a graph of submodels and to simulate challenging biomedical applications (Evans et al. 2008).

Acknowledgments

I thank Jonas Latt, Orestis Malaspinas, Andrea Parmigiani, and Chris Huber for stimulating discussions and for providing many of the figures illustrating Sect. 5.6.

References

- Alexander FJ, Chen H, Chen S, Doolen GD (Aug 1992) Lattice Boltzmann model for compressible fluids. *Phys Rev A* 46(4):1967–1970
- Ansumali S, Karlin IV, Arcidiacono S, Abbas A, Prasianakis NI (2007) Hydrodynamics beyond Navier-Stokes: exact solution to the lattice Boltzmann hierarchy. *Phys Rev Lett* 98:124502
- Banks E (1971) Information processing and transmission in cellular automata. Technical report, MIT. MAC TR-81
- Bhatnager P, Gross EP, Krook MK (1954) A model for collision process in gases. *Phys Rev* 94:511
- Bouzidi M, Firdauss M, Lallemand P (2001) Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Phys Fluids* 13(11):3452–3459
- Burks AW (1970) Von Neumann's self-reproducing automata. In: Burks AW (ed) Essays on cellular automata, University of Illinois Press, Urbana, IL, pp 3–64
- Chen S, Doolen GD (1998) Lattice Boltzmann methods for fluid flows. *Annu Rev Fluid Mech* 30:329
- Chikatamarla SS, Ansumali S, Karlin IV (2006) Entropic lattice Boltzmann models for hydrodynamics in three dimensions. *Phys Rev Lett* 97:010201

- Chopard B, Droz M (1998) Cellular automata modeling of physical systems. Cambridge University Press, Cambridge, UK
- Chopard B, Dupuis A (2003) Cellular automata simulations of traffic: a model for the city of Geneva. *Netw Spatial Econ* 3:9–21
- Chopard B, Falcone J-L, Latt J (2009) The lattice Boltzmann advection-diffusion model revisited. *Eur Phys J* 171:245–249
- Chopard B, Luthi P, Droz M (1994) Reaction-diffusion cellular automata model for the formation of Liesegang patterns. *Phys Rev Lett* 72(9): 1384–1387
- Chopard B, Luthi PO, Queloz P-A (1996) Cellular automata model of car traffic in two-dimensional street networks. *J Phys A* 29:2325–2336
- Chopard B, Luthi P, Masselot A, Dupuis A (2002) Cellular automata and lattice Boltzmann techniques: an approach to model and simulate complex systems. *Adv Complex Syst* 5(2):103–246. <http://cui.unige.ch/~chopard/FTP/CA/acis.pdf>
- Culick K, Yu S (1988) Undecidability of CA classification scheme. *Complex Syst* 2:177–190
- Deutsch A, Dormann S (2005) Cellular automaton modeling of biological pattern formation. Birkhäuser, Boston, MA
- d'Humières D, Ginzburg I, Krafczyk M, Lallemand P, Luo L-S (2002) Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Phil Trans R Soc A* 360:437–451
- Evans D, Lawford P-V, Gunn J, Walker D, Hose D-R, Smallwood RH, Chopard B, Krafczyk M, Bernsdorf J, Hoekstra A (2008) The application of multi-scale modelling to the process of development and prevention of stenosis in a stented coronary artery. *Phil Trans R Soc* 366(1879):3343–3360
- Frisch U, Hasslacher B, Pomeau Y (1986) Lattice-gas automata for the Navier-Stokes equation. *Phys Rev Lett* 56:1505
- Galam S, Chopard B, Masselot A, Droz M (1998) Competing species dynamics: qualitative advantage versus geography. *Eur Phys J B* 4:529–531
- Gardner M (1970) The fantastic combinations of John Conway's new solitaire game "Life." *Sci Am* 220(4):120
- Gaylord RJ, Nishidate K (1996) Modeling nature with cellular automata using Mathematica. Springer, New York
- Ginzburg I (2005) Equilibrium-type and link-type lattice Boltzmann models for generic advection and anisotropic-dispersion equation. *Adv Water Resour* 28(11):1171–1195
- Guo Z, Zheng C, Shi B (2002a) Discrete lattice effects on forcing terms in the lattice Boltzmann method. *Phys Rev E* 65:046308
- Guo Z, Zheng C, Shi B (2002b) An extrapolation method for boundary conditions in lattice Boltzmann method. *Phys Fluids* 14:2007–2010
- He X, Luo L-S (1997) A priori derivation of the lattice Boltzmann equation. *Phys Rev E* 55:R6333–R6336
- Hegewald J, Krafczyk M, Tölke J, Hoekstra A, Chopard B (2008) An agent-based coupling platform for complex automata. In: Bubak M et al (ed) ICCS 2008, vol LNCS 5102, Springer, Berlin, Germany, pp 291–300
- Hoefler WJR (Oct 1985) The transmission-line matrix method. theory and applications. *IEEE Trans Microw Theory Tech MTT-33(10)*:882–893
- Hoekstra A, Falcone J-L, Caiazzo A, Chopard B (2008a) Multi-scale modeling with cellular automata: the complex automata approach. In: Umeo H et al (ed) ACRI 2008, vol LNCS 5191, Springer, Berlin, Germany, pp 192–199
- Hoekstra A, Lorenz E, Falcone J-L, Chopard B (2008b) Towards a complex automata formalism for multi-scale modeling. *Int J Multiscale Comput Eng* 5(6):491–502
- Ilachinski A (2001) Cellular automata: a discrete universe. World Scientific, River Edge, NJ
- Inamuro T, Yoshino M, Ogino F (1995) A non-slip boundary condition for lattice Boltzmann simulations. *Phys Fluids* 7(12):2928–2930
- Junk M, Klar A, Luo L-S (2005) Asymptotic analysis of the lattice Boltzmann equation. *J Comput Phys* 210(2)
- Kanai M, Nishinari K, Tokihiro T (2005) Stochastic optimal velocity model and its long-lived metastability. *Phys Rev E* 72:035102(R)
- Kanai M, Nishinari K, Tokihiro T (2006) Stochastic cellular automaton model for traffic flow. In: El Yacoubi S, Chopard B, Bandini S (eds) Cellular automata: 7th ACRI conference, Perpignan, France, vol 4173. LNCS, Springer, pp 538–547
- Kao P-H, Yang R-J (2008) An investigation into curved and moving boundary treatments in the lattice Boltzmann method. *J Comput Phys* 227(11):5671–5690
- Lallemand P, Luo L-S (2003) Lattice Boltzmann method for moving boundaries. *J Comput Phys* 184(2): 406–421
- Latt J (2007) Hydrodynamic limit of lattice Boltzmann equations. Ph.D. thesis, University of Geneva, Switzerland. <http://www.unige.ch/cyberdocuments/theses2007/LattJ/meta.html>
- Latt J, Chopard B (2006) Lattice Boltzmann method with regularized non-equilibrium distribution functions. *Math Comp Sim* 72:165–168
- Latt J, Chopard B, Malaspina O, Deville M, Michler A (2008) Straight velocity boundaries in the lattice Boltzmann method. *Phys Rev E* 77:056703

- Luthi PO, Preiss A, Ramsden J, Chopard B (1998) A cellular automaton model for neurogenesis in drosophila. *Physica D* 118:151–160
- Marconi S, Chopard B (2003) A lattice Boltzmann model for a solid body. *Int J Mod Phys B* 17(1/2):153–156
- Martis NS, Chen H (1996) Simulation of multi-component fluids in complex 3D geometries by the lattice Boltzmann method. *Phys Rev E* 53:743–749
- Nagel K, Herrmann HJ (1993) Deterministic models for traffic jams. *Physica A* 199:254
- Nagel K, Schreckenberg M (1992) Cellular automaton model for freeway traffic. *J Phys I (Paris)* 2:2221
- Prop J (1994) Trajectory of generalized ants. *Math Intell* 16(1):37–42
- Rothman D, Zaleski S (1997) Lattice-gas cellular automata: simple models of complex hydrodynamics. Collection Aléa. Cambridge University Press, Cambridge, UK
- Schreckenberg M, Schadschneider A, Nagel K, Ito N (1995) Discrete stochastic models for traffic flow. *Phys Rev E* 51:2939
- Schreckenberg M, Wolf DE (ed) (1998) Traffic and granular flow '97. Springer, Singapore
- Servan-Camas B, Tsai FTC (2008) Lattice Boltzmann method for two relaxation times for advection-diffusion equation: third order analysis and stability analysis. *Adv Water Resour* 31:1113–1126
- Shan X, Yuan X-F, Chen H (2006) Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation. *J Fluid Mech* 550:413–441
- Sipper M (1997) Evolution of parallel cellular machines: the cellular programming approach. Springer, Berlin, Germany. Lecture notes in computer science, vol 1194
- Stewart I (July 1994) The ultimate in anty-particle. *Sci Am* 270:88–91
- Succi S (2001) The lattice Boltzmann equation, for fluid dynamics and beyond. Oxford University Press, New York
- Suga S (2006) Numerical schemes obtained from lattice Boltzmann equations for advection diffusion equations. *Int J Mod Phys C* 17(11):1563–1577
- Sukop MC, Thorne DT (2005) Lattice Boltzmann modeling: an introduction for geoscientists and engineers. Springer, Berlin, Germany
- Swift MR, Olrendini E, Osborn WR, Yeomans JM (1996) Lattice Boltzmann simulations of liquid-gas and binary fluid systems. *Phys Rev E* 54:5041–5052
- Toffoli T, Margolus N (1987) Cellular automata machines: a new environment for modeling. The MIT Press, Cambridge, MA
- Van der Sman RGM, Ernst MH (2000) Convection-diffusion lattice Boltzmann scheme for irregular lattices. *J Comp Phys* 160:766–782
- Vanneste C, Sebbah P, Sornette D (1992) A wave automaton for time-dependent wave propagation in random media. *Europhys Lett* 17:715
- Vichniac G (1984) Simulating physics with cellular automata. *Physica D* 10:96–115
- Weimar JR (1998) Simulation with cellular automata. Logos, Berlin, Germany
- Wolf-Gladrow DA (2000) Lattice-gas cellular automata and lattice Boltzmann models: an introduction. Lecture notes in mathematics, 1725. Springer, Berlin, Germany
- Wolfram S (1986) Theory and application of cellular automata. World Scientific, Singapore
- Wolfram S (1994) Cellular automata and complexity. Addison-Wesley, Reading MA
- Wolfram S (2002) A new kind of science. Wolfram Sciences, New York
- Yu H, Zhao K (Apr 2000) Lattice Boltzmann method for compressible flows with high Mach numbers. *Phys Rev E* 61(4):3867–3870
- Yukawa S, Kikuchi M, Tadaki S (1994) Dynamical phase transition in one-dimensional traffic flow model with blockage. *J Phys Soc Jpn* 63(10): 3609–3618
- Zou Q, He X (1997) On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Phys Fluids* 7:2998

Section II

Neural Computation

Tom Heskes and Joost N. Kok

10 Computing with Spiking Neuron Networks

Hélène Paugam-Moisy^{1,2} · Sander Bohte³

¹Laboratoire LIRIS – CNRS, Université Lumière Lyon 2, Lyon, France

²INRIA Saclay – Ile-de-France, Université Paris-Sud, Orsay, France

helene.paugam-moisy@univ-lyon2.fr

hpaugam@lri.fr

³Research Group Life Sciences, CWI, Amsterdam, The Netherlands

s.m.bohte@cwi.nl

1	<i>From Natural Computing to Artificial Neural Networks</i>	336
2	<i>Models of Spiking Neurons and Synaptic Plasticity</i>	343
3	<i>Computational Power of Neurons and Networks</i>	351
4	<i>Learning in Spiking Neuron Networks</i>	357
5	<i>Discussion</i>	367

Abstract

Spiking Neuron Networks (SNNs) are often referred to as the third generation of neural networks. Highly inspired by natural computing in the brain and recent advances in neurosciences, they derive their strength and interest from an accurate modeling of synaptic interactions between neurons, taking into account the time of spike firing. SNNs overcome the computational power of neural networks made of threshold or sigmoidal units. Based on dynamic event-driven processing, they open up new horizons for developing models with an exponential capacity to memorize and a strong ability to do fast adaptation. Today, the main challenge is to discover efficient learning rules that might take advantage of the specific features of SNNs while keeping the nice properties (general-purpose, easy-to-use, available simulators, etc.) of traditional connectionist models. This chapter relates the history of the “spiking neuron” in [Sect. 1](#) and summarizes the most currently-in-use models of neurons and synaptic plasticity in [Sect. 2](#). The computational power of SNNs is addressed in [Sect. 3](#) and the problem of learning in networks of spiking neurons is tackled in [Sect. 4](#), with insights into the tracks currently explored for solving it. Finally, [Sect. 5](#) discusses application domains, implementation issues and proposes several simulation frameworks.

1 From Natural Computing to Artificial Neural Networks

1.1 Traditional Neural Networks

Since the human brain is made up of a great many intricately connected neurons, its detailed workings are the subject of interest in fields as diverse as the study of neurophysiology, consciousness, and of course artificial intelligence. Less grand in scope, and more focused on the functional detail, artificial neural networks attempt to capture the essential computations that take place in these dense networks of interconnected neurons making up the central nervous systems in living creatures.

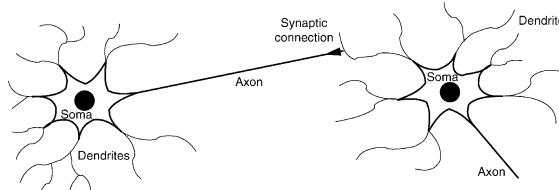
The original work of McCulloch and Pitts ([1943](#)) proposed a neural network model based on simplified “binary” neurons, where a single neuron implements a simple thresholding function: a neuron’s state is either “active” or “not active,” and at each neural computation step, this state is determined by calculating the weighted sum of the states of all the afferent neurons that connect to the neuron. For this purpose, connections between neurons are directed (*from* neuron N_i *to* neuron N_j), and have a weight (w_{ij}). If the weighted sum of the states of all the neurons N_i connected to a neuron N_j exceeds the characteristic threshold of N_j , the state of N_j is set to active, otherwise it is not ([Fig. 1](#), where index j has been omitted).

Subsequent neuronal models evolved where inputs and outputs were real-valued, and the nonlinear threshold function (Perceptron) was replaced by a linear input–output mapping (Adaline) or nonlinear functions like the sigmoid (Multilayer Perceptron). Alternatively, several connectionist models (e.g., RBF networks, Kohonen self-organizing maps (Kohonen [1982](#); Van Hulle [2000](#))) make use of “distance neurons” where the neuron output results from applying a transfer function to the (usually quadratic) distance $\| X - W \|$ between the weights W and inputs X , instead of the dot product, usually denoted by $\langle X, W \rangle$ ([Fig. 2](#)).

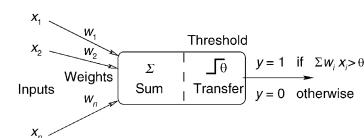
Remarkably, networks of such simple, connected computational elements can implement a wide range of mathematical functions relating input states to output states: With algorithms for setting the weights between neurons, these artificial neural networks can “learn” such relations.

Fig. 1

The first model of a neuron picked up the most significant features of a natural neuron: All-or-none output resulting from a nonlinear transfer function applied to a weighted sum of inputs.



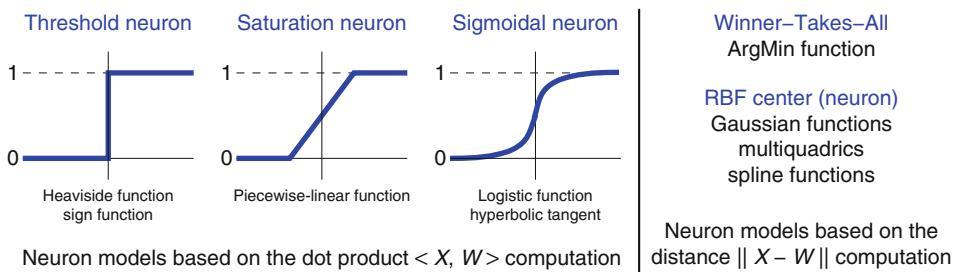
Elementary scheme of biological neurons



First mathematical model of artificial neuron

Fig. 2

Several variants of neuron models, based on a dot product or a distance computation, with different transfer functions.



A large number of learning rules have been proposed, both for teaching a network explicitly to perform some task (supervised learning), and for learning interesting features “on its own” (unsupervised learning). Supervised learning algorithms include gradient descent algorithms (e.g., error backpropagation) (Rumelhart et al. 1986) that fit the neural network behavior to some target function. Many ideas on local unsupervised learning in neural networks can be traced back to the original work on synaptic plasticity by Hebb (1949), and his famous, oft-repeated quote:

- When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

Unsupervised learning rules inspired by this type of natural neural processing are referred to as Hebbian rules (e.g., in Hopfield's (1982) network model).

In general, artificial neural networks (NNs) have been proved to be very powerful as engineering tools in many domains (pattern recognition, control, bioinformatics, and robotics), and also in many theoretical cases:

- Calculability: NNs computational power outperforms a Turing machine (Siegelmann 1999).
- Complexity: The “loading problem” is NP-complete (Blum and Rivest 1989; Judd 1990).

- Capacity: Multilayer Perceptrons (MLP), Radial Basis Function (RBF) network, and Wavelet Neural Networks (WNN) are universal approximators (Cybenko 1988; Funahashi 1989; Hornik et al. 1989).
- Regularization theory (Poggio and Girosi 1989); Probably Approximately Correct learning (PAC-learning) (Valiant 1984); Statistical learning theory, Vapnik–Chervonenkis dimension (VC-dimension), and Support Vector Machines (SVM) (Vapnik 1998).

Nevertheless, traditional neural networks suffer from intrinsic limitations, mainly for processing large amounts of data or for fast adaptation to a changing environment. Several characteristics, such as iterative learning algorithms or artificially designed neuron models and network architectures, are strongly restrictive compared with biological processing in natural neural networks.

1.2 The Biological Inspiration, Revisited

A new investigation in natural neuronal processing is motivated by the evolution of thinking regarding the basic principles of brain processing. When the first neural networks were modeled, the prevailing belief was that intelligence is based on reasoning, and that logic is the foundation of reasoning. In 1943, McCulloch and Pitts designed their model of the neuron in order to prove that the elementary components of the brain were able to compute elementary logic functions: Their first application of thresholded binary neurons was to build networks for computing Boolean functions. In the tradition of Turing’s work (1939, 1950), they thought that complex, “intelligent” behavior could emerge from a very large network of neurons, combining huge numbers of elementary logic gates. History shows that such basic ideas have been very productive, even if effective learning rules for large networks (e.g., backpropagation for MLP) were discovered only at the end of the 1980s, and even if the idea of Boolean decomposition of tasks has been abandoned for a long time.

Separately, neurobiological research has greatly progressed. Notions such as associative memory, learning, adaptation, attention, and emotions have unseated the notion of logic and reasoning as being fundamental to understand how the brain processes information. *Time* has become a central feature in cognitive processing (Abeles 1991). Brain imaging and a host of new technologies (microelectrode, LFP (local field potential) or EEG (electroencephalogram) recordings, fMRI (functional magnetic resonance imaging)) can now record rapid changes in the internal activity of the brain, and help elucidate the relation between the brain activity and the perception of a given stimulus. The current consensus is that cognitive processes are most likely based on the activation of transient assemblies of neurons (see Sect. 3.2), although the underlying mechanisms are not yet well understood.

With these advances in mind, it is worth recalling some neurobiological detail: real neurons spike, at least most biological neurons rely on pulses as an important part of information transmission from one neuron to another neuron. In a rough and non-exhaustive outline, a neuron can generate an action potential – the *spike* – at the soma, the cell body of the neuron. This brief electric pulse (1 or 2 ms duration) then travels along the neuron’s axon, which in turn is linked to the receiving end of other neurons, the dendrites (see Fig. 1, left view). At the end of the axon, synapses connect one neuron to another, and at the arrival of each individual spike the synapses may release neurotransmitters along the *synaptic cleft*. These neurotransmitters are taken up by the neuron at the receiving end, and modify the state of that

postsynaptic neuron, in particular the *membrane potential*, typically making the neuron more or less likely to fire for some duration of time.

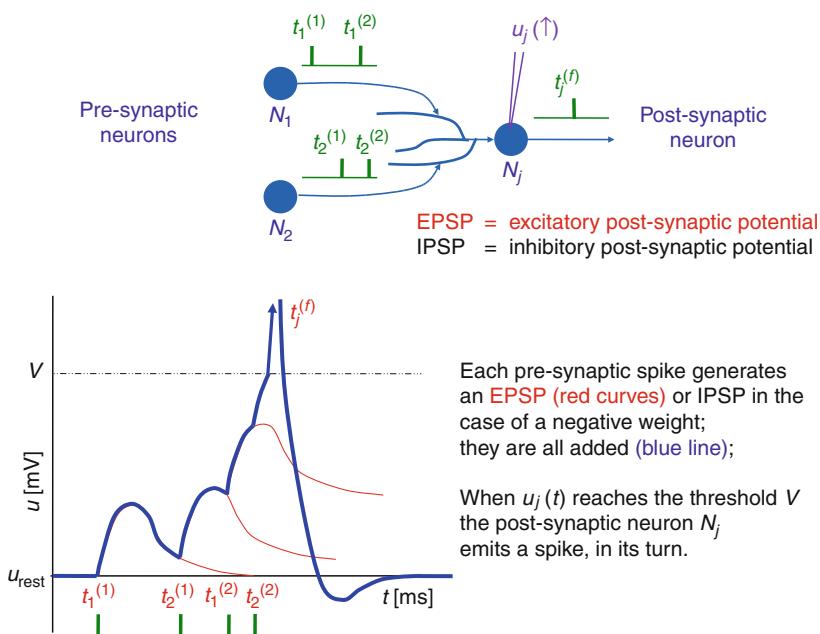
The transient impact a spike has on the neuron's membrane potential is generally referred to as the *postsynaptic potential*, or *PSP*, and the PSP can either inhibit the future firing – inhibitory postsynaptic potential, *IPSP* – or excite the neuron, making it more likely to fire – an excitatory postsynaptic potential, *EPSP*. Depending on the neuron, and the specific type of connection, a PSP may directly influence the membrane potential for anywhere between tens of microseconds and hundreds of milliseconds. A brief sketch of the typical way a *spiking neuron* processes is depicted in Fig. 3. It is important to note that the firing of a neuron may be a deterministic or stochastic function of its internal state.

Many biological details are omitted in this broad outline, and they may or may not be relevant for computing. Examples are the stochastic release of neurotransmitter at the synapses: depending on the firing history, a synaptic connection may be more or less reliable, and more or less effective. Inputs into different parts of the dendrite of a neuron may sum nonlinearly, or even multiply. More detailed accounts can be found in, for example, Maass and Bishop (1999).

Evidence from the field of neuroscience has made it increasingly clear that in many situations information is carried in the individual action potentials, rather than aggregate

Fig. 3

A model of spiking neuron: N_j fires a spike whenever the weighted sum of incoming EPSPs generated by its presynaptic neurons reaches a given threshold. The graphic (bottom) shows how the membrane potential of N_j varies through time, under the action of the four incoming spikes (top).



measures such as “firing rate.” Rather than the form of the action potential, it is the number and the timing of spikes that matter. In fact, it has been established that *the exact timing of spikes* can be a means for coding information, for instance in the electrosensory system of electric fish (Heiligenberg 1991), in the auditory system of echo-locating bats (Kuwabara and Suga 1993), and in the visual system of flies (Bialek et al. 1991).

1.3 Time as Basis of Information Coding

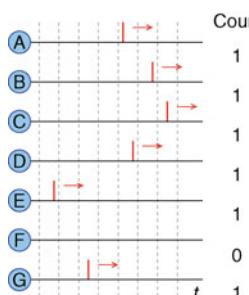
The relevance of the timing of individual spikes has been at the center of the debate about rate coding versus spike coding. Strong arguments against rate coding have been given by Thorpe et al. (1996) and van Rullen and Thorpe (2001) in the context of visual processing. Many physiologists subscribe to the idea of a Poisson-like rate code to describe the way neurons transmit information. However, as pointed out by Thorpe et al., Poisson rate codes seem hard to reconcile with the impressively efficient rapid information transmission required for sensory processing in human vision. Only 100–150 ms are sufficient for a human to respond selectively to complex visual stimuli (e.g., faces or food), but due to the feedforward architecture of the visual system, made up of multiple layers of neurons firing at an average rate of 10 ms, realistically only one spike or none could be fired by each neuron involved in the process during this time window. A pool of neurons firing spikes stochastically as a function of the stimulus could realize an instantaneous rate code: a *spike density code*. However, maintaining such a set of neurons is expensive, as is the energetic cost of firing so many spikes to encode a single variable (Olshausen 1996). It seems clear from this argument alone that the presence and possibly timing of individual spikes is likely to convey information, and not just the number, or rate, of spikes.

From a combinatorial point of view, precisely timed spikes have a far greater encoding capacity, given a small set of spiking neurons. The representational power of alternative coding schemes was pointed out by Recce (1999) and analyzed by Thorpe et al. (2001). For instance, consider that a stimulus has been presented to a set of n spiking neurons and that each of them fires at most one spike in the next T (ms) time window (Fig. 4).

Fig. 4

Comparing the representational power of spiking neurons, for different coding schemes.

Count code: 6/7 spike per 7 ms, that is ≈ 122 spikes per s; Binary code: 1111101; Timing code: latency, here with a 1 ms precision; Rank order code: $E \geq G \geq A \geq D \geq B \geq C \geq F$.



	Count	Latency	Rank
A	1	5	3
B	1	6	5
C	1	7	6
D	1	5	4
E	1	1	1
F		-	-
G	0	-	-
	t	1	2

Numeric examples:	Count code	Binary code	Timing code	Rank order
Left (opposite) figure $n = 7, T = 7$ ms	3	7	≈ 19	12.3
Thorpe et al. (2001) $n = 10, T = 10$ ms	3.6	10	≈ 33	21.8

Number of bits that can be transmitted by n neurons in a T time window.

Consider some different ways to decode the temporal information that can be transmitted by the n spiking neurons. If the code is to *count* the overall number of spikes fired by the set of neurons (population rate coding), the maximum amount of available information is $\log_2(n + 1)$, since only $n + 1$ different events can occur. In the case of a *binary code*, the output is an n -digits binary number, with obviously n as the information-coding capacity. A greater amount of information is transmitted with a *timing code*, provided that an efficient decoding mechanism is available for determining the precise times of each spike. In practical cases, the available code size depends on the decoding precision, for example for a 1 ms precision, an amount of information of $n \times \log_2(T)$ can be transmitted in the T time window. Finally, in *rank order coding*, information is encoded in the order of the sequence of spike emissions, that is one among the $n!$ orders that can be obtained from n neurons, thus $\log_2(n!)$ bits can be transmitted, meaning that the order of magnitude of the capacity is $n \log(n)$. However, this theoretical estimate must be reevaluated when considering the unavoidable bound on precision required for distinguishing two spike times, even in computer simulation (Cessac et al. 2010).

1.4 Spiking Neuron Networks

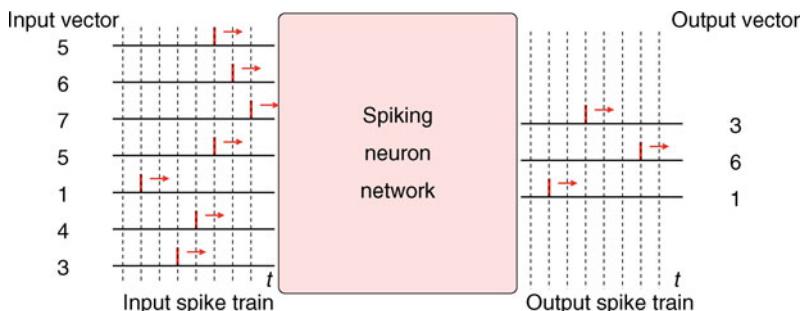
In *Spiking Neuron Networks* (SNNs), sometimes referred to as Pulsed-Coupled Neural Networks (PCNNs) in the literature, the presence and *timing* of individual spikes is considered as the means of communication and neural computation. This compares with traditional neuron models where analog values are considered, representing the *rate* at which spikes are fired.

In SNNs, new input–output notions have to be developed that assign meaning to the presence and timing of spikes. One example of such coding that easily compares to traditional neural coding is *temporal coding* (sometimes referred to as “latency coding” or “time-to-first-spike”). Temporal coding is a straightforward method for translating a vector of real numbers into a spike train, for example, for simulating traditional connectionist models using SNNs, as in Maass (1997a). The basic idea is biologically well founded: the more intensive the input, the earlier the spike transmission (e.g., in visual systems). Hence, a network of spiking neurons can be designed with n input neurons N_i whose firing times are determined through some external mechanism. The network is fed by successive n -dimensional input patterns $\mathbf{x} = (x_1, \dots, x_n)$ – with all x_i inside a bounded interval of \mathbb{R} , for example $[0, 1]$ – that are translated into spike trains through successive temporal windows (comparable to successive steps of traditional NNs computation). In each time window, a pattern \mathbf{x} is temporally coded relative to a fixed time T_{in} by one spike emission of neuron N_i at time $t_i = T_{in} - x_i$, for all i (► Fig. 5). It is straightforward to show that with such temporal coding, and some mild assumptions, any traditional neural network can be emulated by an SNN. However, temporal coding obviously does not apply readily to more continuous computing where neurons fire multiple spikes, in spike trains.

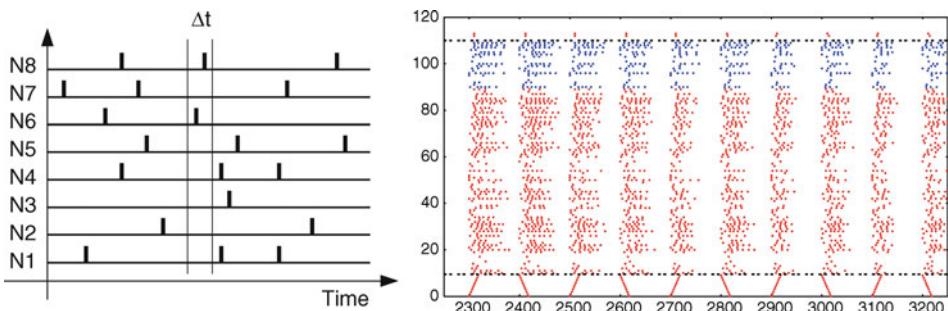
Many SNN approaches focus on the continuous computation that is carried out on such spike trains. Assigning meaning is then less straightforward, and depends on the approach. However, a way to visualize the temporal computation processed by an SNN is by displaying a complete representation of the network activity on a *spike raster plot* (► Fig. 6): With time on the abscissa, a small bar is plotted each time a neuron fires a spike (one line per neuron, numbered on the Y-axis). Variations and frequencies of neuronal activity can be observed in such diagrams, in the same way as natural neurons activities can be observed in spike raster

Fig. 5

Illustration of the temporal coding principle for encoding and decoding real vectors in spike trains.

**Fig. 6**

On a spike raster plot, a small bar is plotted each time (in abscissa) that a neuron (numbered in ordinates) fires a spike. For computational purposes, time is often discretized in temporal Δt units (*left*). The dynamic answer of an SNN, stimulated by an input pattern in temporal coding – diagonal patterns, bottom – can be observed on a spike raster plot (*right*). (From Paugam-Moisy et al. 2008.)



plots drawn from multielectrode recordings. Likewise, other representations (e.g., time-frequency diagrams) can be drawn from simulations of artificial networks of spiking neurons, as is done in neuroscience from experimental data.

Since the basic principle underlying SNNs is so radically different, it is not surprising that much of the work on traditional neural networks, such as learning rules and theoretical results, has to be adapted, or even has to be fundamentally rethought. The main purpose of this chapter is to give an exposition on important state-of-the-art aspects of computing with SNNs, from theory to practice and implementation.

The first difficult task is to define “the” model of neuron, as there exist numerous variants already. Models of spiking neurons and synaptic plasticity are the subject of [Sect. 2](#). It is worth mentioning that the question of network architecture has become less important in SNNs than in traditional neural networks. [Section 3](#) proposes a survey of theoretical results (capacity, complexity, and learnability) that argue for SNNs being a new generation of neural networks that are more powerful than the previous ones, and considers some of the ideas on

how the increased complexity and dynamics could be exploited. [Section 4](#) addresses different methods for learning in SNNs and presents the paradigm of *Reservoir Computing*. Finally, [Sect. 5](#) focuses on practical issues concerning the implementation and use of SNNs for applications, in particular with respect to temporal pattern recognition.

2 Models of Spiking Neurons and Synaptic Plasticity

A spiking neuron model accounts for the impact of impinging action potentials – spikes – on the targeted neuron in terms of the internal state of the neuron, as well as how this state relates to the spikes the neuron fires. There are many models of spiking neurons, and this section only describes some of the models that have so far been most influential in Spiking Neuron Networks.

2.1 Hodgkin–Huxley Model

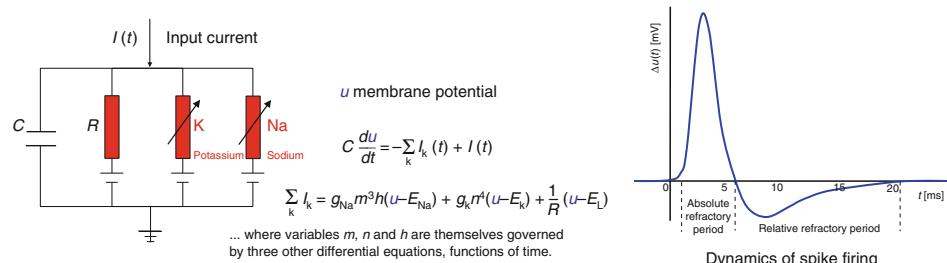
The fathers of the spiking neurons are the conductance-based neuron models, such as the well-known electrical model defined by Hodgkin and Huxley (1952) ([Fig. 7](#)). Hodgkin and Huxley modeled the electrochemical information transmission of natural neurons with electrical circuits consisting of capacitors and resistors: C is the capacitance of the membrane, g_{Na} , g_K , and g_L denote the conductance parameters for the different ion channels (sodium Na, potassium K, etc.) and E_{Na} , E_K , and E_L are the corresponding equilibrium potentials. The variables m , h , and n describe the opening and closing of the voltage-dependent channels.

$$\begin{aligned} C \frac{du}{dt} &= -g_{\text{Na}} m^3 h (u - E_{\text{Na}}) - g_K n^4 (u - E_K) - g_L (u - E_L) + I(t) \quad (1) \\ \tau_n \frac{dn}{dt} &= -[n - n_0(u)], \quad \tau_m \frac{dm}{dt} = -[m - m_0(u)], \quad \tau_h \frac{dh}{dt} = -[h - h_0(u)] \end{aligned}$$

Appropriately calibrated, the Hodgkin–Huxley model has been successfully compared to numerous data from biological experiments on the giant axon of the squid. More generally, it has been shown that the Hodgkin–Huxley neuron is able to model biophysically meaningful

Fig. 7

Electrical model of “spiking” neuron as defined by Hodgkin and Huxley. The model is able to produce realistic variations of the membrane potential and the dynamics of a spike firing, for example in response to an input current $I(t)$ sent during a small time, at $t < 0$.



properties of the membrane potential, respecting the behavior recordable from natural neurons: an abrupt, large increase at firing time, followed by a short period where the neuron is unable to spike again, the *absolute refractoriness*, and a further time period where the membrane is depolarized, which makes renewed firing more difficult, that is, the *relative refractory period* (❷ Fig. 7).

The Hodgkin–Huxley model (HH) is realistic but far too complex for the simulation of SNNs. Although ordinary differential equations (ODE) solvers can be applied directly to the system of differential equations, it would be intractable to compute temporal interactions between neurons in a large network of Hodgkin–Huxley models.

2.2 Integrate-and-Fire Model and Variants

2.2.1 Integrate-and-Fire (I&F) and Leaky-Integrate-and-Fire (LIF)

Simpler than the Hodgkin–Huxley neuron model are Integrate-and-Fire (I&F) neuron models, which are much more computationally tractable (see ❷ Fig. 8 for equation and electrical model).

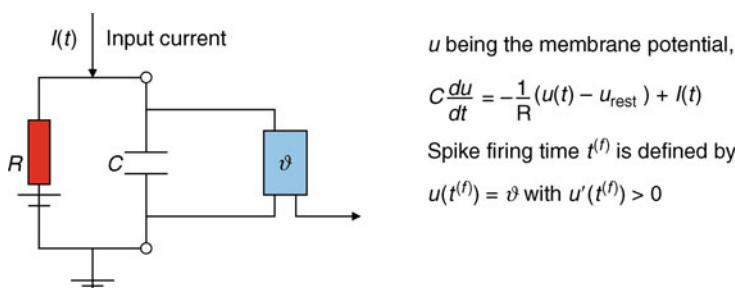
An important I&F neuron type is the *Leaky-Integrate-and-Fire* (LIF) neuron (Abbott 1999; Stein 1965). Compared to the Hodgkin–Huxley model, the most important simplification in the LIF neuron implies that the shape of the action potentials is neglected and every spike is considered as a uniform event defined only by the time of its appearance. The electrical circuit equivalent for a LIF neuron consists of a capacitor C in parallel with a resistor R driven by an input current $I(t)$. In this model, the dynamics of the membrane potential in the LIF neuron are described by a single first-order linear differential equation:

$$\tau_m \frac{du}{dt} = u_{\text{rest}} - u(t) + RI(t) \quad (2)$$

where $\tau_m = RC$ is taken as the time constant of the neuron membrane, modeling the voltage leakage. Additionally, the firing time $t^{(f)}$ of the neuron is defined by a threshold crossing equation $u(t^{(f)}) = \vartheta$, under the condition $u'(t^{(f)}) > 0$. Immediately after $t^{(f)}$, the potential is reset to a given value u_{rest} (with $u_{\text{rest}} = 0$ as a common assumption). An absolute refractory period can be modeled by forcing the neuron to a value $u = -u_{\text{abs}}$ during a time d_{abs} after a spike emission, and then restarting the integration with initial value $u = u_{\text{rest}}$.

❷ Fig. 8

Electrical circuit and equation of the Integrate-and-Fire model (I&F).



2.2.2 Quadratic-Integrate-and-Fire (QIF) and Theta Neuron

Quadratic-Integrate-and-Fire (QIF) neurons, a variant where $\frac{du}{dt}$ depends on u^2 , may be a somewhat better, and still computationally efficient, compromise. Compared to LIF neurons, QIF neurons exhibit many dynamic properties such as delayed spiking, bi-stable spiking modes, and activity-dependent thresholding. They further exhibit a frequency response that better matches biological observations (Brunel and Latham 2003). Via a simple transformation of the membrane potential u to a phase θ , the QIF neuron can be transformed to a Theta-neuron model (Ermentrout and Kopell 1986).

In the Theta-neuron model, the neuron's state is determined by a *phase*, θ . The Theta neuron produces a spike with the phase passing through π . Being one-dimensional, the Theta-neuron dynamics can be plotted simply on a phase circle (☞ Fig. 9).

The phase trajectory in a Theta neuron evolves according to:

$$\frac{d\theta}{dt} = (1 - \cos(\theta)) + \alpha I(t)(1 + \cos(\theta)) \quad (3)$$

where θ is the neuron phase, α is a scaling constant, and $I(t)$ is the input current.

The main advantage of the Theta-neuron model is that neuronal spiking is described in a continuous manner, allowing for more advanced gradient approaches, as illustrated in ☞ Sect. 4.1.

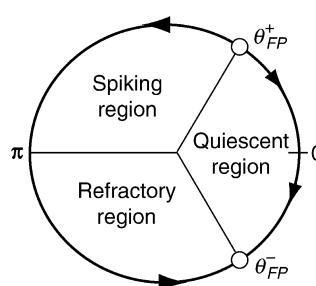
2.2.3 Izhikevich's Neuron Model

In the class of spiking neurons defined by differential equations, the two-dimensional *Izhikevich neuron model* (Izhikevich 2003) is a good compromise between biophysical plausibility and computational cost. It is defined by the coupled equations

$$\begin{aligned} \frac{du}{dt} &= 0.04u(t)^2 + 5u(t) + 140 - w(t) + I(t) & \frac{dw}{dt} &= a(bu(t) - w(t)) \\ \text{with after-spike resetting :} & \quad \text{if } u \geq \vartheta \text{ then } u \leftarrow c \text{ and } w \leftarrow w + d \end{aligned} \quad (4)$$

Fig. 9

Phase circle of the Theta-neuron model, for the case where the baseline current $I(t) < 0$. When the phase goes through π , a spike is fired. The neuron has two fixed points: a saddle point θ_{FP}^+ , and an attractor θ_{FP}^- . In the spiking region, the neuron will fire after some time, whereas in the quiescent region, the phase decays back to θ_{FP}^- unless the input pushes the phase into the spiking region. The refractory phase follows after spiking, and in this phase it is more difficult for the neuron to fire again.



This neuron model is capable of reproducing many different firing behaviors that can occur in biological spiking neurons (☞ Fig. 10). (An electronic version of the original figure and reproduction permission are freely available at www.izhikevich.com.)

2.2.4 On Spiking Neuron Model Variants

Besides the models discussed here, there exist many different spiking neuron models that cover the complexity range between the Hodgkin–Huxley model and LIF models, with decreasing biophysical plausibility, but also with decreasing computational cost (see, e.g., Izhikevich (2004) for a comprehensive review, or Standage and Trappenberg (2005) for an in-depth comparison of Hodgkin–Huxley and LIF subthreshold dynamics).

Whereas the Hodgkin–Huxley models are the most biologically realistic, the LIF and – to a lesser extent – QIF models have been studied extensively due to their low complexity, making them relatively easy to understand. However, as argued by Izhikevich (2004), LIF neurons are a simplification that no longer exhibit many important spiking neuron properties. Where the full Hodgkin–Huxley model is able to reproduce many different neuro-computational properties and firing behaviors, the LIF model has been shown to be able to reproduce only three out of the 20 firing schemes displayed in ☞ Fig. 10: the “tonic spiking” (A), the “class 1 excitable” (G) and the “integrator” (L). Note that although some behaviors are mutually exclusive for a particular instantiation of a spiking neuron model – for example (K) “resonator” and (L) “integrator” – many such behaviors may be reachable with different parameter choices, for the same neuron model. The QIF model is already able to capture more realistic behavior, and the Izhikevich neuron model can reproduce all of the 20 firing schemes displayed in ☞ Fig. 10. Other intermediate models are currently being studied, such as the gIF model (Rudolph and Destexhe 2006).

The complexity range can also be expressed in terms of the computational requirements for simulation. Since it is defined by four differential equations, the Hodgkin–Huxley model requires about 1,200 floating point computations (FLOPS) per 1 ms simulation. Simplified to two differential equations, the Morris–Lecar or FitzHugh–Nagumo models still have a computational cost of one to several hundred FLOPS. Only five FLOPS are required by the LIF model, around 10 FLOPS for variants such as LIF-with-adaptation and quadratic or exponential Integrate-and-Fire neurons, and around 13 FLOPS for Izhikevich’s model.

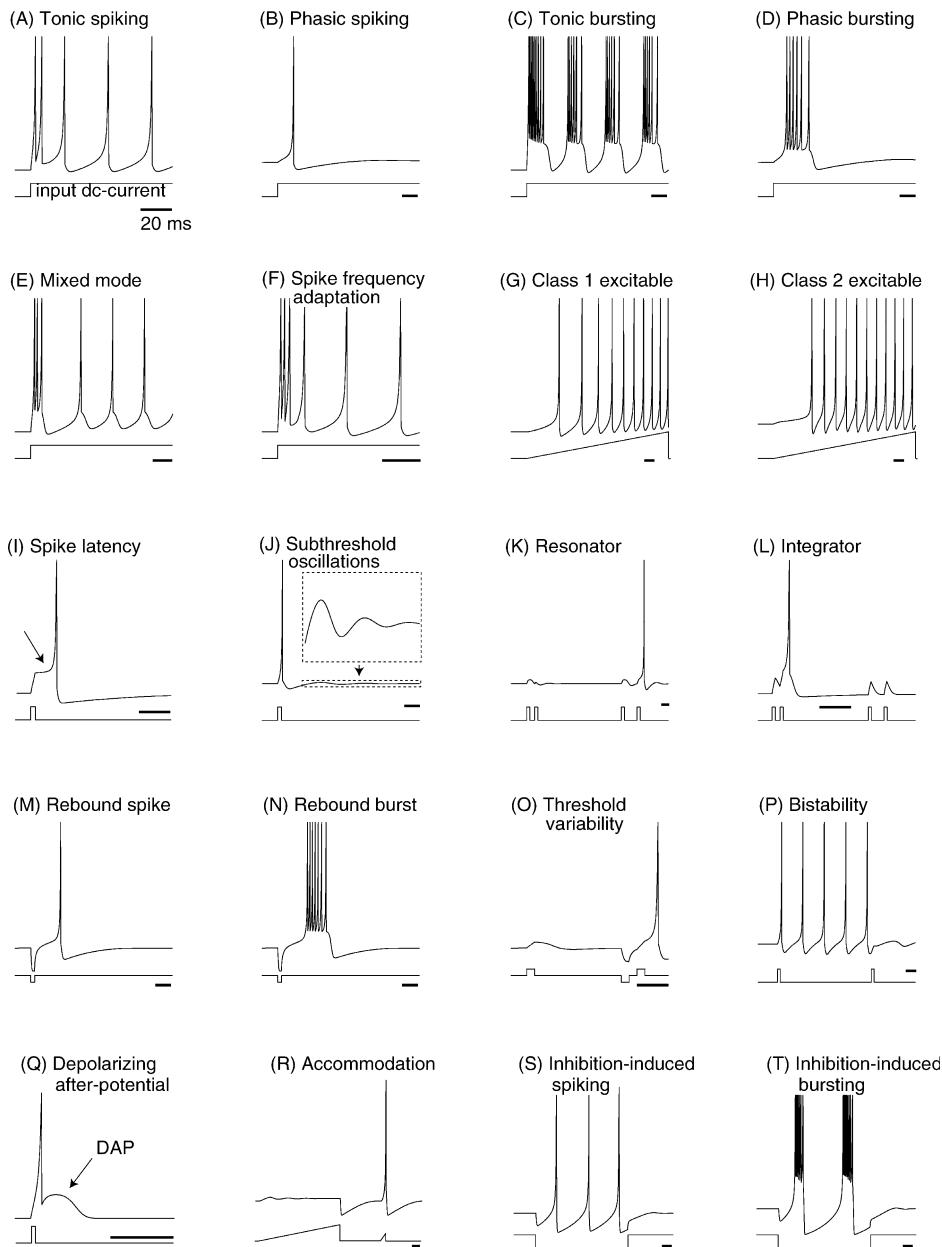
2.3 Spike Response Model

Compared to the neuron models governed by coupled differential equations, the *Spike Response Model* (SRM) as defined by Gerstner (1995) and Kistler et al. (1997) is more intuitive and more straightforward to implement. The SRM model expresses the membrane potential u at time t as an integral over the past, including a model of refractoriness. The SRM is a phenomenological model of neuron, based on the occurrence of spike emissions. Let $\mathcal{F}_j = \left\{ t_j^{(f)}; 1 \leq f \leq n \right\} = \left\{ t | u_j(t) = \vartheta \wedge u'_j(t) > 0 \right\}$ denote the set of all firing times of neuron N_j , and $\Gamma_j = \{i | N_i \text{ is presynaptic to } N_j\}$ define its set of presynaptic neurons. The state $u_j(t)$ of neuron N_j at time t is given by

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_j(t - t_j^{(f)}) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in \mathcal{F}_i} w_{ij} \varepsilon_{ij}(t - t_i^{(f)}) + \underbrace{\int_0^{\infty} \kappa_j(r) I(t - r) dr}_{\text{if external input current}} \quad (5)$$

Fig. 10

Many firing behaviors can occur in biological spiking neurons. Shown are simulations of the Izhikevich neuron model, for different external input currents (displayed under each temporal firing pattern). (From Izhikevich [2004].)



with the following kernel functions: η_j is non-positive for $s > 0$ and models the potential reset after a spike emission, ε_{ij} describes the membrane's potential response to presynaptic spikes, and κ_j describes the response of the membrane potential to an external input current (☞ Fig. 11). Some common choices for the kernel functions are:

$$\eta_j(s) = -\vartheta \exp\left(-\frac{s}{\tau}\right) \mathcal{H}(s)$$

or, somewhat more involved,

$$\eta_j(s) = -\eta_0 \exp\left(-\frac{s - \delta^{\text{abs}}}{\tau}\right) \mathcal{H}(s - \delta^{\text{abs}}) - K \mathcal{H}(s) \mathcal{H}(\delta^{\text{abs}} - s)$$

where \mathcal{H} is the Heaviside function, ϑ is the threshold and τ a time constant, for neuron N_j . Setting $K \rightarrow \infty$ ensures an absolute refractory period δ^{abs} and η_0 scales the amplitude of relative refractoriness.

Kernel ε_{ij} describes the generic response of neuron N_j to spikes coming from presynaptic neurons N_i , and is generally taken as a variant of an α -function. (An α -function is like $\alpha(x) = x \exp^{-x}$.)

$$\varepsilon_{ij}(s) = \frac{s - d_{ij}^{\text{ax}}}{\tau_s} \exp\left(-\frac{s - d_{ij}^{\text{ax}}}{\tau_s}\right) \mathcal{H}(s - d_{ij}^{\text{ax}})$$

or, in a more general description,

$$\varepsilon_{ij}(s) = \left[\exp\left(-\frac{s - d_{ij}^{\text{ax}}}{\tau_m}\right) - \exp\left(-\frac{s - d_{ij}^{\text{ax}}}{\tau_s}\right) \right] \mathcal{H}(s - d_{ij}^{\text{ax}})$$

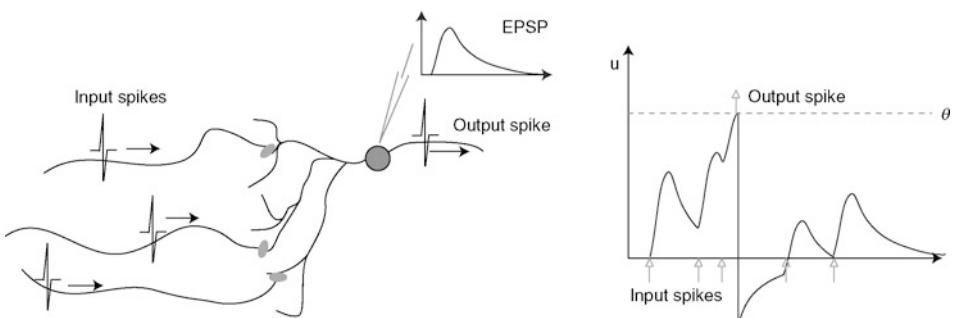
where τ_m and τ_s are time constants, and d_{ij}^{ax} describes the axonal transmission delay.

For the sake of simplicity, $\varepsilon_{ij}(s)$ can be assumed to have the same form $\varepsilon(s - d_{ij}^{\text{ax}})$ for any pair of neurons, only modulated in amplitude and sign by the weight w_{ij} (excitatory EPSP for $w_{ij} > 0$, inhibitory IPSP for $w_{ij} < 0$).

A short-term memory variant of SRM results from assuming that only the last firing \hat{t}_j of N_j contributes to refractoriness, $\eta_j(t - \hat{t}_j)$ replacing the sum in formula (☞ Eq. 5) by a

Fig. 11

The Spike Response Model (SRM) is a generic framework to describe the spike process. (Redrawn after Gerstner and Kistler 2002b.)



single contribution. Moreover, integrating the equation on a small time window of 1 ms and assuming that each presynaptic neuron fires at most once in the time window (reasonable because of the refractoriness of presynaptic neurons) reduces the SRM to the simplified SRM_0 model:

$$u_j(t) = \eta_j(t - \hat{t}_j) + \sum_{i \in \Gamma_j} w_{ij} e(t - \hat{t}_i - d_{ij}^{ax}) \quad \text{next firing time} \quad t_j^{(f)} = t \iff u_j(t) = \vartheta \quad (6)$$

Despite its simplicity, the SRM is more general than Integrate-and-Fire neuron models and is often able to compete with the Hodgkin–Huxley model for simulating complex neuro-computational properties.

2.4 Synaptic Plasticity and STDP

In all the models of neurons, most of the parameters are constant values, and specific to each neuron. The exception is the synaptic connections that are the basis of adaptation and learning, even in traditional neural network models where several synaptic weight updating rules are based on Hebb's law (Hebb 1949) (see [Sect. 1](#)). *Synaptic plasticity* refers to the adjustments and even formation or removal of synapses between neurons in the brain. In the biological context of natural neurons, the changes of synaptic weights with effects lasting several hours are referred to as Long-Term Potentiation (LTP) if the weight values (also called *efficacies*) are strengthened, and Long-Term Depression (LTD) if the weight values are decreased. On the timescale of seconds or minutes, the weight changes are denoted as Short-Term Potentiation (STP) and Short-Term Depression (STD). Abbott and Nelson (2000) give a good review of the main synaptic plasticity mechanisms for regulating levels of activity in conjunction with Hebbian synaptic modification, for example, redistribution of synaptic efficacy (Markram and Tsodyks 1996) or synaptic scaling. Neurobiological research has also increasingly demonstrated that synaptic plasticity in networks of spiking neurons is sensitive to the presence and precise timing of spikes (Markram et al. 1997; Bi and Poo 1998; Kempter et al. 1999).

One important finding that is receiving increasing attention is *Spike-Timing Dependent Plasticity*, STDP, as discovered in neuroscientific studies (Markram et al. 1997; Kempter et al. 1999), especially in detailed experiments performed by Bi and Poo (1998, 2001). Often referred to as a *temporal Hebbian rule*, STDP is a form of synaptic plasticity sensitive to the precise timing of spike firing relative to impinging presynaptic spike times. It relies on local information driven by backpropagation of action potential (BPAP) through the dendrites of the postsynaptic neuron. Although the type and amount of long-term synaptic modification induced by repeated pairing of pre- and postsynaptic action potential as a function of their relative timing vary from one neuroscience experiment to another, a basic computational principle has emerged: a maximal increase of synaptic weight occurs on a connection when the presynaptic neuron fires a short time before the postsynaptic neuron, whereas a late presynaptic spike (just after the postsynaptic firing) leads to a decrease in the weight. If the two spikes (pre- and post-) are too distant in time, the weight remains unchanged. This type of LTP/LTD timing dependency should reflect a form of causal relationship in information transmission through action potentials.

For computational purposes, STDP is most commonly modeled in SNNs using temporal windows for controlling the weight LTP and LTD that are derived from neurobiological

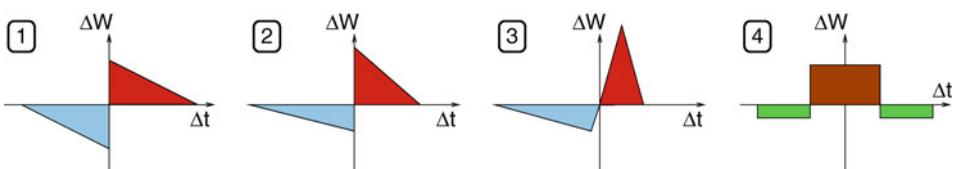
experiments. Different shapes of STDP windows have been used in recent literature (Markram et al. 1997; Kempfer et al. 1999; Song et al. 2000; Senn et al. 2001; Buchs and Senn 2002; Izhikevich et al. 2004; Kistler 2002; Gerstner and Kistler 2002a; Nowotny et al. 2003; Izhikerich and Desai 2003; Saudargiene et al. 2004; Meunier and Paugam-Moisy 2005; Mouraud and Paugam-Moisy 2006): They are smooth versions of the shapes schematized by polygons in Fig. 12. The spike timing (X-axis) is the difference $\Delta t = t_{\text{post}} - t_{\text{pre}}$ of firing times between the pre- and postsynaptic neurons. The synaptic change ΔW (Y-axis) operates on the weight update. For excitatory synapses, the weight w_{ij} is increased when the presynaptic spike is supposed to have a causal influence on the postsynaptic spike, that is when $\Delta t > 0$ and is close to zero (windows 1–3 in Fig. 12), and decreased otherwise. The main differences in shapes 1–3 concern the symmetry or asymmetry of the LTP and LTD subwindows, and the discontinuity or not of the ΔW function of Δt , near $\Delta t = 0$. For inhibitory synaptic connections, it is common to use a standard Hebbian rule, just strengthening the weight when the pre- and postsynaptic spikes occur close in time, regardless of the sign of the difference $t_{\text{post}} - t_{\text{pre}}$ (window 4 in Fig. 12).

There exist at least two ways to compute with STDP: The modification ΔW can be applied to a weight w according to either an additive update rule $w \leftarrow w + \Delta W$ or a multiplicative update rule $w \leftarrow w(1 + \Delta W)$.

The notion of temporal Hebbian learning in the form of STDP appears as a possible new direction for investigating innovative learning rules in SNNs. However, many questions arise and many problems remain unresolved. For example, weight modifications according to STDP windows cannot be applied repeatedly in the same direction (e.g., always potentiation) without fixing bounds for the weight values, for example an arbitrary fixed range $[0, w_{\max}]$ for excitatory synapses. Bounding both the weight increase and decrease is necessary to avoid either silencing the overall network (when all weights are down) or have “epileptic” network activity (all weights are up, causing disordered and frequent firing of almost all neurons). However, in many STDP driven SNN models, a saturation of the weight values to 0 or w_{\max} has been observed, which strongly reduces further adaptation of the network to new events. Among other solutions, a regulatory mechanism, based on a triplet of spikes, has been described by Nowotny et al. (2003), for a smooth version of the temporal window 3 of Fig. 12, with an additive STDP learning rule. On the other hand, applying a multiplicative weight update also effectively applies a self-regulatory mechanism. For deeper insights into the influence of the nature of the update rule and the shape of STDP windows, the reader could refer to Song et al. (2000), Rubin et al. (2000), and Câteau and Fukai (2003).

Fig. 12

Various shapes of STDP windows with LTP in blue and LTD in red for excitatory connections (1–3). More realistic and smooth ΔW function of Δt are mathematically described by sharp rising slope near $\Delta t = 0$ and fast exponential decrease (or increase) toward $\pm\infty$. Standard Hebbian rule (window 4) with brown LTP and green LTD are usually applied to inhibitory connections.



3 Computational Power of Neurons and Networks

Since information processing in spiking neuron networks is based on the precise timing of spike emissions (pulse coding) rather than the average numbers of spikes in a given time window (rate coding), there are two straightforward advantages of SNN processing. First, SNN processing allows for the very fast decoding of sensory information, as in the human visual system (Thorpe et al. 1996), where real-time signal processing is paramount. Second, it allows for the possibility of multiplexing information, for example, like the auditory system combines amplitude and frequency very efficiently over one channel. More abstractly, SNNs add a new dimension, the temporal axis, to the representation capacity and the processing abilities of neural networks. Here, different approaches to determining the computational power and complexity of SNNs are described, and current thinking on how to exploit these properties is outlined, in particular with regard to dynamic cell assemblies.

In 1997, Maass (1997b, 2001) proposed to classify neural networks as follows:

- *1st generation:* Networks based on McCulloch and Pitts' neurons as computational units, that is, threshold gates, with only digital outputs (e.g., Perceptron, Hopfield network, Boltzmann machine, multilayer networks with threshold units).
- *2nd generation:* Networks based on computational units that apply an activation function with a continuous set of possible output values, such as sigmoid or polynomial or exponential functions (e.g., MLP and RBF networks). The real-valued outputs of such networks can be interpreted as *firing rates* of natural neurons.
- *3rd generation of neural network models:* Networks which employ spiking neurons as computational units, taking into account the precise *firing times* of neurons for information coding. Related to SNNs are also pulse stream very large-scale integrated (VLSI) circuits, new types of electronic solutions for encoding analog variables using time differences between pulses.

Exploiting the full capacity of this new generation of neural network models raises many fascinating and challenging questions that will be discussed in the following sections.

3.1 Complexity and Learnability Results

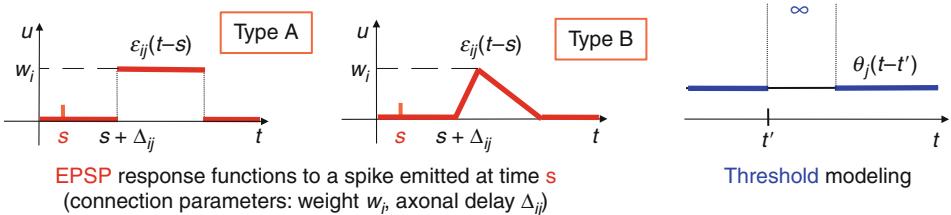
3.1.1 Tractability

To facilitate the derivation of theoretical proofs on the complexity of computing with spiking neurons, Maass proposed a simplified spiking neuron model with a rectangular EPSP shape, the “type_A spiking neuron” (❶ Fig. 13). The type_A neuron model can for instance be justified as providing a link to silicon implementations of spiking neurons in analog VLSI neural microcircuits. Central to the complexity results is the notion of transmission delays: different transmission delays, d_{ij} , can be assigned to different presynaptic neurons, N_i , connected to a postsynaptic neuron N_j .

Let Boolean input vectors, (x_1, \dots, x_n) , be presented to a spiking neuron by a set of input neurons (N_1, \dots, N_n) such that N_i fires at a specific time T_{in} if $x_i = 1$ and does not fire if $x_i = 0$. A type_A neuron is at least as powerful as a threshold gate (Maass 1997b; Schmitt 1998). Since spiking neurons can behave as coincidence detectors (for a proper choice of weights, a spiking neuron can only fire when two or more input spikes are effectively coincident in time) it is

Fig. 13

Very simple versions of spiking neurons: “type_A spiking neuron” (rectangular-shaped pulse) and “type_B spiking neuron” (triangular-shaped pulse), with elementary representation of refractoriness (threshold goes to infinity), as defined in Maass (1997b).



straightforward to prove that the Boolean function CD_n (*Coincidence Detection function*) can be computed by a single spiking neuron of type_A (the proof relies on a suitable choice of the transmission delays d_{ij}):

$$CD_n(x_1, \dots, x_n, y_1, \dots, y_n) = \begin{cases} 1, & \text{if } (\exists i) x_i = y_i \\ 0, & \text{otherwise} \end{cases}$$

In previous neural network generations, the computation of the Boolean function CD_n required many more neurons: at least $\frac{n}{\log(n+1)}$ threshold gates and at least an order of magnitude of $\Omega(n^{1/4})$ sigmoidal units.

Of special interest is the *Element Distinctness function*, ED_n :

$$ED_n(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } (\exists i \neq j) x_i = x_j \\ 0, & \text{if } (\forall i \neq j) |x_i - x_j| \geq 1 \\ \text{arbitrary}, & \text{otherwise} \end{cases}$$

Let real-valued inputs (x_1, \dots, x_n) be presented to a spiking neuron by a set of input neurons (N_1, \dots, N_n) such that N_i fires at time $T_{in} - cx_i$ (cf. temporal coding, defined in [Sect. 1.4](#)). With positive real-valued inputs and a binary output, the ED_n function can be computed by a single type_A neuron, whereas at least $\Omega(n \log(n))$ threshold gates and at least $\frac{n-4}{2} - 1$ sigmoidal hidden units are required.

However, for arbitrary real-valued inputs, type_A neurons are no longer able to compute threshold circuits. For such settings, the “type_B spiking neuron” ([Fig. 13](#)) has been proposed, as its triangular EPSP can shift the firing time of a targeted postsynaptic neuron in a continuous manner. It is easy to see that any threshold gate can be computed by $O(1)$ type_B spiking neurons. Furthermore, at the network level, any threshold circuit with s gates, for real-valued inputs $x_i \in [0, 1]$, can be simulated by a network of $O(s)$ type_B spiking neurons.

From these results, Maass concludes that spiking neuron networks are computationally more powerful than both the 1st and the 2nd generations of neural networks.

Schmitt develops a deeper study of type_A neurons with programmable delays in Schmitt (1998) and Maass and Schmitt (1997). Some results are:

- Every Boolean function of n variables, computable by a single spiking neuron, can be computed by a disjunction of at most $2n-1$ threshold gates.

- There is no $\Sigma\Pi$ -unit with fixed degree that can simulate a spiking neuron.
- The *threshold number* of a spiking neuron with n inputs is $\Theta(n)$.
- The following relation holds: $(\forall n \geq 2) \exists$ a Boolean function on n variables that has threshold number 2 and cannot be computed by a spiking neuron.
- The *threshold order* of a spiking neuron with n inputs is $\Omega(n^{1/3})$.
- The *threshold order* of a spiking neuron with $n \geq 2$ inputs is at most $n-1$.

3.1.2 Capacity

Maass (2001) considers *noisy spiking neurons*, a neuron model close to the SRM (cf. [Sect. 2.3](#)), with a probability of *spontaneous* firing (even under threshold) or not firing (even above threshold) governed by the difference:

$$\sum_{i \in T_j} \sum_{s \in \mathcal{T}_i, s < t} w_{ij} \varepsilon_{ij}(t-s) - \underbrace{\eta_j(t-t')}_{\text{threshold function}}$$

The main result from Maass (2001) is that for any given $\varepsilon, \delta > 0$ one can simulate any given feedforward sigmoidal neural network \mathcal{N} of s units with linear saturated activation function by a network $\mathcal{N}_{\varepsilon,\delta}$ of $s + O(1)$ noisy spiking neurons, in temporal coding. An immediate consequence of this result is that SNNs are *universal approximators*, in the sense that any given continuous function $F : [0, 1]^n \rightarrow [0, 1]^k$ can be approximated within any given precision $\varepsilon > 0$ with arbitrarily high reliability, in temporal coding, by a network of noisy spiking neurons with a single hidden layer.

With regard to synaptic plasticity, Legenstein et al. (2005) studied STDP learnability. They define a *Spiking Neuron Convergence Conjecture* (SNCC) and compare the behavior of STDP learning by teacher-forcing with the Perceptron convergence theorem. They state that a spiking neuron can learn with STDP, basically, any map from input to output spike trains that it could possibly implement in a stable manner. They interpret the result as saying that STDP endows spiking neurons with *universal learning capabilities* for Poisson input spike trains.

Beyond these and other encouraging results, Maass (2001) points out that SNNs are able to encode time series in spike trains, but there are, in computational complexity theory, no standard reference models yet for analyzing computations on time series.

3.1.3 VC-Dimension

The first attempt to estimate the VC-dimension (see http://en.wikipedia.org/wiki/VC_dimension for a definition) of spiking neurons is probably the work of Zador and Pearlmutter (1996), where they studied a family of integrate-and-fire neurons (cf. [Sect. 2.2](#)) with threshold and time constants as parameters. Zador and Pearlmutter proved that for an Integrate-and-Fire (I&F) model, $VC_{dim}(I\&F)$ grows as $\log(B)$ with the input signal bandwidth B , which means that the VC_{dim} of a signal with infinite bandwidth is unbounded, but the divergence to infinity is weak (logarithmic).

More conventional approaches (Maass and Schmitt 1997; Maass 2001) estimate bounds on the VC-dimension of neurons as functions of their programmable/learnable parameters, such as the synaptic weights, the transmission delays and the membrane threshold:

- With m variable positive delays, $VC_{\text{dim}}(\text{type_A neuron})$ is $\Omega(m \log(m))$ – even with fixed weights – whereas, with m variable weights, $VC_{\text{dim}}(\text{threshold gate})$ is $\Omega(m)$.
- With n real-valued inputs and a binary output, $VC_{\text{dim}}(\text{type_A neuron})$ is $O(n \log(n))$.
- With n real-valued inputs and a real-valued output, $pseudo_{\text{dim}}(\text{type_A neuron})$ is $O(n \log(n))$.

The implication is that the learning complexity of a single spiking neuron is greater than the learning complexity of a single threshold gate. As Maass and Schmitt (1999) argue, this should not be interpreted as saying that supervised learning is impossible for a spiking neuron, but rather that it is likely quite difficult to formulate rigorously provable learning results for spiking neurons.

To summarize Maass and Schmitt's work: let the class of Boolean functions, with n inputs and 1 output, that can be computed by a spiking neuron be denoted by \mathcal{S}_n^{xy} , where x is b for Boolean values and a for analog (real) values and idem for y . Then the following holds:

- The classes \mathcal{S}_n^{bb} and \mathcal{S}_n^{ab} have VC-dimension $\Theta(n \log(n))$.
- The class \mathcal{S}_n^{aa} has pseudo-dimension $\Theta(n \log(n))$.

At the network level, if the weights and thresholds are the only programmable parameters, then an SNN with temporal coding seems to be nearly equivalent to traditional Neural Networks (NNs) with the same architecture, for traditional computation. However, transmission delays are a new relevant component in spiking neural computation and SNNs with programmable delays appear to be more powerful than NNs.

Let \mathcal{N} be an SNN of neurons with rectangular pulses (e.g., type_A), where all delays, weights and thresholds are programmable parameters, and let E be the number of edges of the \mathcal{N} directed acyclic graph. (The directed acyclic graph is the network topology that underlies the spiking neuron network dynamics.) Then $VC_{\text{dim}}(\mathcal{N})$ is $O(E^2)$, even for analog coding of the inputs (Maass and Schmitt 1999). Schmitt (2004) derived more precise results by considering a feedforward architecture of depth D , with nonlinear synaptic interactions between neurons.

It follows that the sample sizes required for the networks of fixed depth are not significantly larger than traditional neural networks. With regard to the generalization performance in pattern recognition applications, the models studied by Schmitt can be expected to be at least as good as traditional network models (Schmitt 2004).

3.1.4 Loading Problem

In the framework of PAC-learnability (Valiant 1984; Blumer et al. 1989), only hypotheses from \mathcal{S}_n^{bb} may be used by the learner. Then, the computational complexity of training a spiking neuron can be analyzed within the formulation of the *consistency* or *loading problem* (cf. Judd 1990):

- Given a training set T of labeled binary examples (X, b) with n inputs, do there exist parameters defining a neuron \mathcal{N} in \mathcal{S}_n^{bb} such that $(\forall (X, b) \in T) y_{\mathcal{N}} = b$?

In this PAC-learnability setting, the following results are proved in Maass and Schmitt (1999):

- The *consistency problem* for a spiking neuron with binary delays is *NP*-complete.
- The *consistency problem* for a spiking neuron with binary delays and fixed weights is *NP*-complete.

Several extended results were developed by Šíma and Sgall (2005), such as:

- The *consistency problem* for a spiking neuron with nonnegative delays is *NP*-complete ($d_{ij} \in \mathbb{R}^+$). The result holds even with some restrictions (see Šíma and Sgall (2005) for precise conditions) on bounded delays, unit weights or fixed threshold.
- A single spiking neuron with programmable weights, delays and threshold does not allow robust learning unless $RP = NP$. The *approximation problem* is not better solved even if the same restrictions as above are applied.

3.1.5 Complexity Results Versus Real-World Performance

Non-learnability results, such as those outlined above, have of course been derived for classic NNs already, for example in Blum and Rivest (1989) and Judd (1990). Moreover, the results presented in this section apply only to a restricted set of SNN models and, apart from the programmability of transmission delays of synaptic connections, they do not cover all the capabilities of SNNs that could result from computational units based on firing times. Such restrictions on SNNs can rather be explained by a lack of practice for building proofs in such a context or, even more, by an incomplete and unadapted computational complexity theory or learning theory. Indeed, learning in biological neural systems may employ rather different mechanisms and algorithms than common computational learning systems. Therefore, several characteristics, especially the features related to computing in continuously changing time, will have to be fundamentally rethought to develop efficient learning algorithms and ad hoc theoretical models to understand and master the computational power of SNNs.

3.2 Cell Assemblies and Synchrony

One way to take a fresh look at SNNs complexity is to consider their dynamics, especially the spatial localization and the temporal variations of their activity. From this point of view, SNNs behave as *complex systems*, with emergent macroscopic-level properties resulting from the complex dynamic interactions between neurons, but hard to understand just looking at the microscopic level of each neuron processing. As biological studies highlight the presence of a specific organization in the brain (Sporns et al. 2005; Eguíluz 2005; Achard and Bullmore 2007), the complex networks research area appears to provide valuable tools (“Small-Word” connectivity (Watts and Strogatz 1998), presence of clusters (Newman and Girvan 2004; Meunier and Paugam-Moisy 2006), presence of hubs (Barabasi and Albert 1999), etc., see Newman (2003) for a survey) for studying topological and dynamic complexities of SNNs, both in natural and artificial networks of spiking neurons. Another promising direction for research takes its inspiration from the area of dynamic systems: Several methods and

measures, based on the notions of phase transition, edge-of-chaos, Lyapunov exponents, or mean-field predictors, are currently proposed to estimate and control the computational performance of SNNs (Legenstein and Maass 2005; Verstraeten et al. 2007; Schrauwen et al. 2009). Although these directions of research are still in their infancy, an alternative is to revisit older and more biological notions that are already related to the network topology and dynamics.

The concept of the *cell assembly* was introduced by Hebb (1949), more than half a century ago. (The word “cell” was used at that time, instead of “neuron.”) However, the idea was not further developed, neither by neurobiologists – since they could not record the activity of more than one or a few neurons at a time, until recently – nor by computer scientists. New techniques of brain imaging and recording have boosted this area of research in neuroscience only recently (cf. Wenneker et al. 2003). In computer science, a theoretical analysis of assembly formation in spiking neuron network dynamics (with SRM neurons) has been discussed by Gerstner and van Hemmen (1994), where they contrast ensemble code, rate code, and spike code as descriptions of neuronal activity.

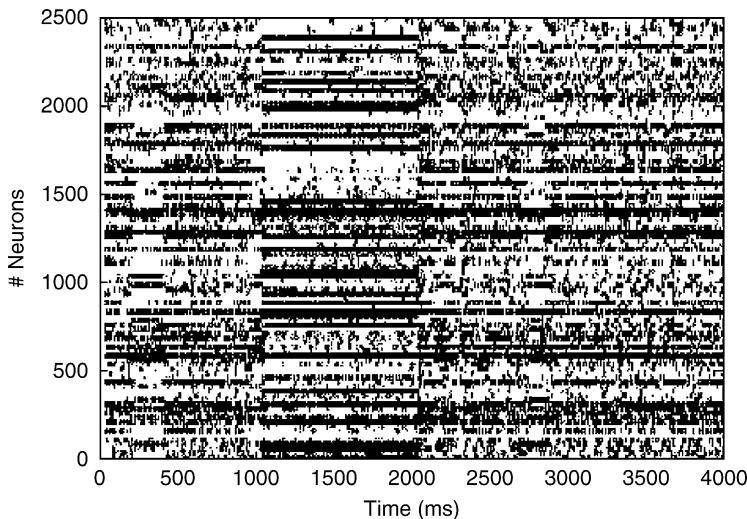
A cell assembly can be defined as a group of neurons with strong mutual excitatory connections. Since a cell assembly, once a subset of its neurons are stimulated, tends to be activated as a whole, it can be considered as an operational unit in the brain. An association can be viewed as the activation of an assembly by a stimulus or another assembly. Then, short-term memory would be a persistent activity maintained by reverberations in assemblies, whereas long-term memory would correspond to the formation of new assemblies, for example, by a Hebb’s rule mechanism. Inherited from Hebb, current thinking about cell assemblies is that they could play the role of “grandmother neural groups” as a basis for memory encoding, instead of the old controversial notion of “grandmother cell,” and that material entities (e.g., a book, a cup, or a dog) and even more abstract entities such as concepts or ideas could be represented by cell assemblies.

Within this context, synchronization of firing times for subsets of neurons inside a network has received much attention. Abeles (1991) developed the notion of *synfire chains*, which describes activity in a pool of neurons as a succession of synchronized firing by specific subsets of these neurons. Hopfield and Brody demonstrated *transient synchrony* as a means for collective spatiotemporal integration in neuronal circuits (Hopfield and Brody 2000, 2001). The authors claim that the event of collective synchronization of specific pools of neurons in response to a given stimulus may constitute a basic computational building block, at the network level, for which there is no resemblance in traditional neural computing (❷ Fig. 14).

However, synchronization per se – even transient synchrony – appears to be too restrictive a notion for fully understanding the potential capabilities of information processing in cell assemblies. This has been comprehensively pointed out by Izhikevich (2006) who proposes the extended notion of *polychronization* within a group of neurons that are sparsely connected with various axonal delays. Based on the connectivity between neurons, a polychronous group is a possible stereotypical time-locked firing pattern. Since the neurons in a polychronous group have matching axonal conduction delays, the group can be activated in response to a specific temporal pattern triggering very few neurons in the group, other ones being activated in a chain reaction. Since any given neuron can be activated within several polychronous groups, the number of coexisting polychronous groups can be far greater than the number of neurons in the network. Izhikevich argues that networks with delays are “infinite-dimensional” from a purely mathematical point of view, thus resulting in much greater information capacity as compared to synchrony-based assembly coding. Polychronous groups represent good

Fig. 14

A spike raster plot showing the dynamics of an artificial SNN: Erratic background activity is disrupted by a stimulus presented between 1,000 and 2,000 ms. (By courtesy of D. Meunier, reprint from his PhD thesis (2007), Université Lyon 2, France)



candidates for modeling multiple trace memory and they could be viewed as a computational implementation of cell assemblies.

Notions of cell assemblies and synchrony, derived from natural computing in the brain and biological observations, are inspiring and challenging computer scientists and theoretical researchers to search for and define new concepts and measures of complexity and learnability in dynamic systems. This will likely bring a much deeper understanding of neural computations that include the time dimension, and will likely benefit both computer science as well as neuroscience.

4 Learning in Spiking Neuron Networks

Traditionally, neural networks have been applied to pattern recognition, in various guises. For example, carefully crafted layers of neurons can perform highly accurate handwritten character recognition (LeCun et al. 1995). Similarly, traditional neural networks are preferred tools for function approximation, or regression. The best-known learning rules for achieving such networks are of course the class of error-backpropagation rules for supervised learning. There also exist learning rules for unsupervised learning, such as Hebbian learning, or distance-based variants like Kohonen self-organizing maps.

Within the class of computationally oriented spiking neuron networks, two main directions are distinguished. First, there is the development of learning methods equivalent to those developed for traditional neural networks. By substituting traditional neurons with spiking neuron models, augmenting weights with delay lines, and using temporal coding, algorithms for supervised and unsupervised learning have been developed. Second, there are networks and computational algorithms that are uniquely developed for networks of spiking

neurons. These networks and algorithms use the temporal domain as well as the increased complexity of SNNs to arrive at novel methods for temporal pattern detection with spiking neuron networks.

4.1 Simulation of Traditional Models

Maass and Natschläger (1997) propose a theoretical model for emulating arbitrary Hopfield networks in temporal coding (see [Sect. 1.4](#)). Maass (1997a) studies a “relatively realistic” mathematical model for biological neurons that can simulate arbitrary feedforward sigmoidal neural networks. Emphasis is put on the fast computation time that depends only on the number of layers of the sigmoidal network, and no longer on the number of neurons or weights. Within this framework, SNNs are validated as universal approximators (see [Sect. 3.1](#)), and traditional supervised and unsupervised learning rules can be applied for training the synaptic weights.

It is worth remarking that, to enable theoretical results, Maass and Natschläger’s model uses static reference times T_{in} and T_{out} and auxiliary neurons. Even if such artifacts can be removed in practical computation, the method rather appears to be an artificial attempt to make SNNs computing like traditional neural networks, without taking advantage of SNNs intrinsic abilities to compute with time.

4.1.1 Unsupervised Learning in Spiking Neuron Networks

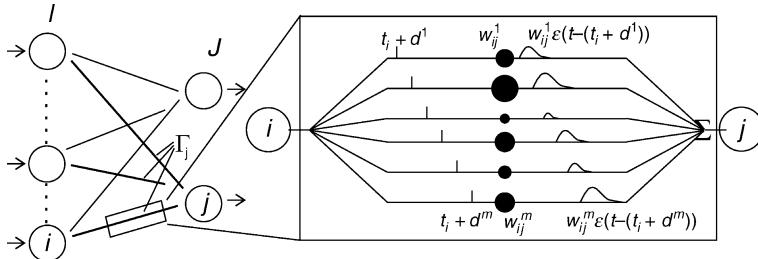
Within this paradigm of computing in SNNs equivalently to traditional neural network computing, a number of approaches for unsupervised learning in spiking neuron networks have been developed, based mostly on variants of Hebbian learning. Extending on Hopfield’s (1995) idea, Natschläger and Ruf (1998a) propose a learning algorithm that performs unsupervised clustering in spiking neuron networks, akin to RBF network, using spike times as input. Natschläger and Ruf’s spiking neural network for unsupervised learning is a simple two-layer network of SRM neurons, with the addition of multiple delays between the neurons: An individual connection from a neuron i to a neuron j consists of a fixed number of m synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay d^k and weight w_{ij}^k ([Fig. 15](#)). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the presynaptic neuron i , and the time the postsynaptic potential of neuron j starts rising.

A winner-takes-all learning rule modifies the weights between the source neurons and the neuron first to fire in the target layer using a time-variant of Hebbian learning: If the start of the PSP at a synapse slightly precedes a spike in the target neuron, the weight of this synapse is increased, as it exerted significant influence on the spike time via a relatively large contribution to the membrane potential. Earlier and later synapses are decreased in weight, reflecting their lesser impact on the target neuron’s spike time. With such a learning rule, input patterns can be encoded in the synaptic weights such that, after learning, the firing time of an output neuron reflects the distance of the evaluated pattern to its learned input pattern thus realizing a kind of RBF neuron (Natschläger and Ruf 1998a).

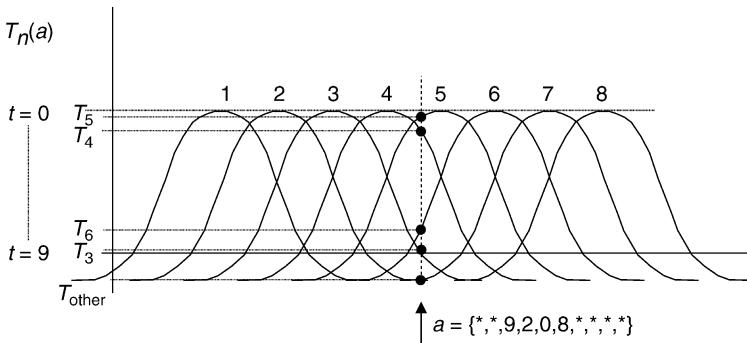
Bohte et al. (2002b) extend on this approach to enhance the precision, capacity, and clustering capability of a network of spiking neurons by developing a temporal version of population coding. To extend the encoding precision and clustering capacity, input data is

Fig. 15

Unsupervised learning rule in SNNs: Any single connection can be considered as being multisynaptic, with random weights and a set of increasing delays, as defined in Natschläger and Ruf (1998b).

**Fig. 16**

Encoding with overlapping Gaussian receptive fields. An input value a is translated into firing times for the input-neurons encoding this input-variable. The highest stimulated neuron (neuron 5), fires at a time close to $T = 0$, whereas less-stimulated neurons, as for instance neuron 3, fire at increasingly later times.



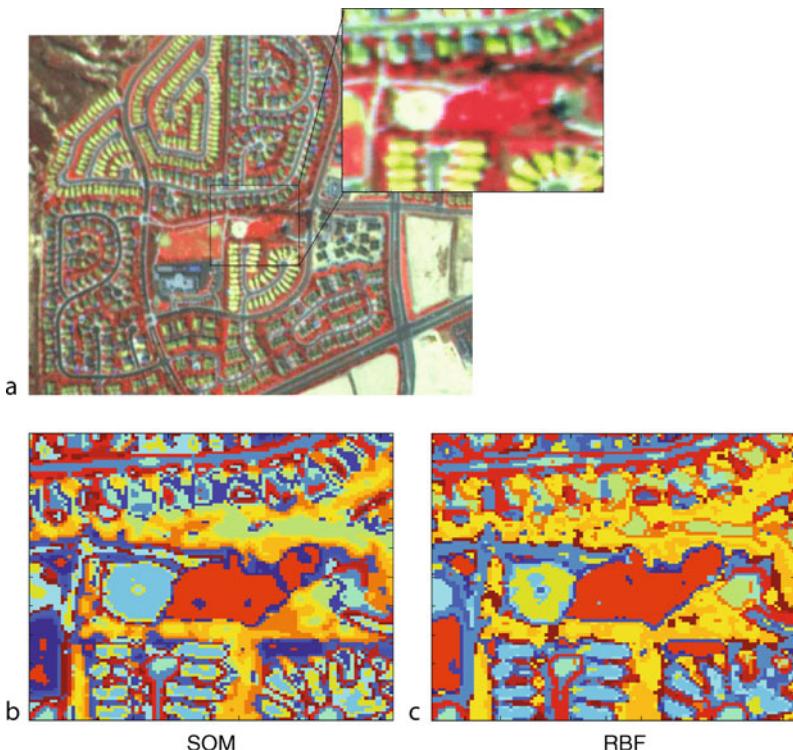
encoded into temporal spike-time patterns by population coding, using multiple local receptive fields like Radial Basis Functions. The translation of inputs into relative firing times is straightforward: An optimally stimulated neuron fires at $t = 0$, whereas a value up to say $t = 9$ is assigned to less optimally stimulated neurons (depicted in [Fig. 16](#)). With such encoding, spiking neural networks were shown to be effective for clustering tasks, for example [Fig. 17](#).

4.1.2 Supervised Learning in Multilayer Networks

A number of approaches for supervised learning in standard multilayer feedforward networks have been developed based on gradient descent methods, the best known being error backpropagation. As developed in Bohte et al. (2002a), *SpikeProp* starts from error backpropagation to derive a supervised learning rule for networks of spiking neurons that transfer the information in the timing of a single spike. This learning rule is analogous to the derivation rule by Rumelhart et al. (1986), but SpikeProp applies to spiking neurons of the SRM type. To overcome the discontinuous nature of spiking neurons, the thresholding function is

Fig. 17

Unsupervised classification of remote sensing data. (a) The full image. Inset: image cutout that is actually clustered. (b) Classification of the cutout as obtained by clustering with a Self-Organizing Map (SOM). (c) Spiking Neuron Network RBF classification of the cutout image.



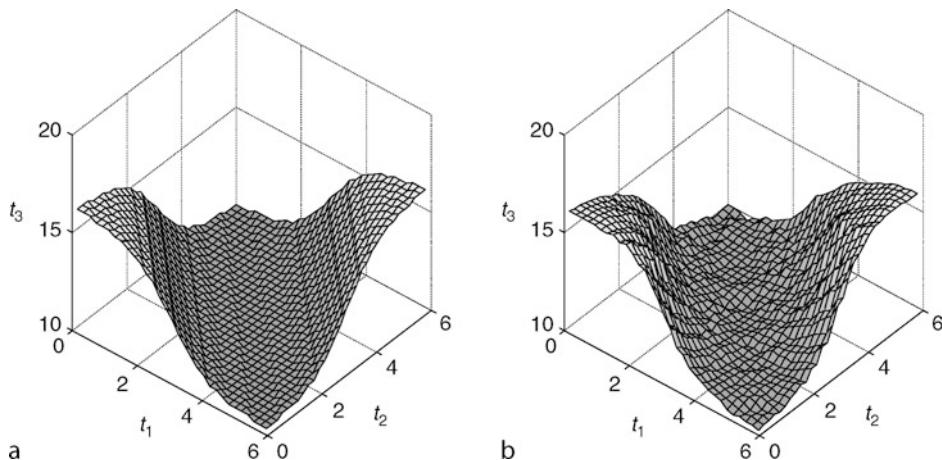
approximated, thus linearizing the model at a neuron's output spike times. As in the unsupervised SNN described above, each connection between neurons may have multiple delayed synapses with varying weights (see [Fig. 15](#)). The SpikeProp algorithm has been shown to be capable of learning complex nonlinear tasks in spiking neural networks with similar accuracy as traditional sigmoidal neural networks, including the archetypal XOR classification task ([Fig. 18](#)).

The SpikProp method has been successfully extended to adapt the synaptic delays along the error gradient, as well as the decay for the α -function and the threshold (Schrauwen and Van Campenhout 2004a, b). Xin and Embrechts (2001) have further shown that the addition of a simple momentum term significantly speeds up convergence of the SpikeProp algorithm. Booij and Nguyen (2005) have, analogously to the method for BackPropagation-Through-Time, extended SpikeProp to account for neurons in the input and hidden layer to fire multiple spikes.

McKenna et al. (2009) derived a supervised Theta-learning rule for multilayer networks of Theta neurons. By mapping QIF neurons to the canonical Theta-neuron model (a nonlinear phase model, see [Sect. 2.2](#)), a more dynamic spiking neuron model is placed at the heart of the spiking neuron network. The Theta-neuron phase model is cyclic and allows for a continuous reset. Derivatives can then be computed without any local linearization assumptions.

Fig. 18

Interpolated XOR function $f(t_1, t_2) : [0, 6] \rightarrow [10, 16]$. (a) Target function. (b) Spiking Neuron Network output after training.



Some sample results showing the performance of both SpikeProp and the Theta Neuron learning rule as compared to error backpropagation in traditional neural networks is shown in [Table 1](#). The more complex Theta-neuron learning allows for a smaller neural network to optimally perform classification.

As with SpikeProp, Theta learning requires some careful fine-tuning of the network. In particular, both algorithms are sensitive to *spike loss*, in that no error gradient is defined when the neuron does not fire for any pattern, and hence will never recover. McKennoch et al. (2009) heuristically deal with this issue by applying alternating periods of coarse learning, with a greater learning rate, and fine tuning, with a small learning rate.

As demonstrated in Belatreche et al. (2007), non-gradient-based methods like evolutionary strategies do not suffer from these tuning issues. For MLP networks based on various spiking neuron models, performance comparable to SpikeProp is shown. An evolutionary strategy is, however, very time consuming for large-scale networks.

4.2 Reservoir Computing

Clearly, the architecture and dynamics of an SNN can be matched, by temporal coding, to traditional connectionist models, such as multilayer feedforward networks or recurrent networks. However, since networks of spiking neurons behave decidedly different as compared to traditional neural networks, there is no pressing reason to design SNNs within such rigid schemes.

According to biological observations, the neurons of biological SNNs are sparsely and irregularly connected in space (network topology) and the variability of spike flows implies that they communicate irregularly in time (network dynamics) with a low average activity. It is important to note that the network topology becomes a simple underlying support to the neural dynamics, but that only active neurons contribute to information processing. At a given time t , the sub-topology defined by active neurons can be very sparse and different from the

Table 1

Classification results for the SpikeProp and Theta neuron supervised learning methods on two benchmarks, the Fisher Iris dataset and the Wisconsin Breast Cancer dataset. The results are compared to standard error backpropagation, BP A and BP B denoting the standard Matlab backprop implementation with default parameters, where their respective network sizes are set to correspond to either the SpikeProp or the Theta-neuron networks (taken from McKennoch et al. (2009))

Learning method	Network size	Epochs	Train (%)	Test (%)
<i>Fisher Iris Dataset</i>				
SpikeProp	$50 \times 10 \times 3$	1,000	97.4	96.1
BP A	$50 \times 10 \times 3$	2.6e6	98.2	95.5
BP B	$4 \times 8 \times 1$	1e5	98.0	90.0
Theta-Neuron BP	$4 \times 8 \times 1$	1,080	100	98.0
<i>Wisconsin Breast Cancer Dataset</i>				
SpikeProp	$64 \times 15 \times 2$	1,500	97.6	97.0
BP A	$64 \times 15 \times 2$	9.2e6	98.1	96.3
BP B	$9 \times 8 \times 1$	1e5	97.2	99.0
Theta-Neuron BP	$9 \times 8 \times 1$	3,130	98.3	99.0

underlying network architecture (e.g., local clusters, short or long path loops, and synchronized cell assemblies), comparable to the active brain regions that appear colored in brain imaging scanners. Clearly, an SNN architecture has no need to be regular. A network of spiking neurons can even be defined randomly (Maass et al. 2002b; Jaeger 2002) or by a loosely specified architecture, such as a set of neuron groups that are linked by projections, with a given probability of connection from one group to the other (Meunier and Paugam-Moisy 2005). However, the nature of a connection has to be prior defined as an excitatory or inhibitory synaptic link, without subsequent change, except for the synaptic efficacy. That is, the weight value can be modified, but not the weight sign.

With this in mind, a new family of networks has been developed that is specifically suited to processing temporal input/output patterns with spiking neurons. The new paradigm is named *Reservoir Computing* as a unifying term for which the precursor models are Echo State Networks (ESNs) and Liquid State Machines (LSMs). Note that the term “reservoir computing” is not reserved to SNNs, since ESN was first designed with sigmoidal neurons, but this chapter mainly presents reservoir computing with SNNs.

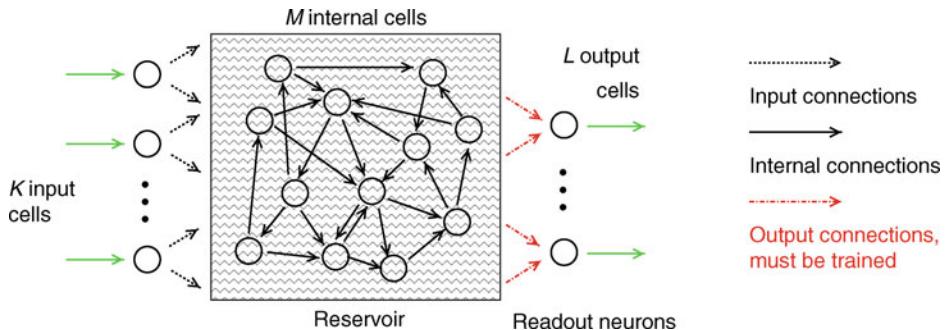
4.2.1 Main Characteristics of Reservoir Computing Models

The topology of a reservoir computing model (Fig. 19) can be defined as follows:

- A layer of K neurons with input connections toward the reservoir
- A recurrent network of M neurons, interconnected by a random and sparse set of weighted links: the so-called *reservoir*, which is usually left untrained
- A layer of L *readout neurons* with trained connections from the reservoir

Fig. 19

Architecture of a reservoir computing network: the “reservoir” is a set of M internal neurons, with random and sparse connectivity.



The early motivation of reservoir computing is the well-known difficulty to find efficient supervised learning rules to train recurrent neural networks, as attested by the limited success of methods like Backpropagation Through Time (BPTT), Real-Time Recurrent Learning (RTRL) or Extended Kalman Filtering (EKF). The difficulty stems from lack of knowledge of how to control the behavior of the complex dynamic system resulting from the presence of cyclic connections in the network architecture. The main idea of reservoir computing is to renounce training the internal recurrent network and only to pick out, by way of the readout neurons, the relevant part of the dynamic states induced in the reservoir by the network inputs. Only the reading out of this information is subject to training, usually by very simple learning rules, such as linear regression. The success of the method is based on the high power and accuracy of self-organization inherent to a random recurrent network.

In SNN versions of reservoir computing, a soft kind of unsupervised, local training is often added by applying a synaptic plasticity rule like STDP inside the reservoir. Since STDP was directly inspired by the observation of natural processing in the brain (see [Sect. 2.4](#)), its computation does not require supervised control or an understanding of the network dynamics.

The paradigm of “reservoir computing” is commonly referred to as such since approximately 2007, and it encompasses several seminal models in the literature that predate this generalized notion by a few years. The next section describes the two founding models that were designed concurrently in the early 2000s, by Jaeger ([2001](#)) for the ESN and by Maass et al. ([2002b](#)) for the LSM.

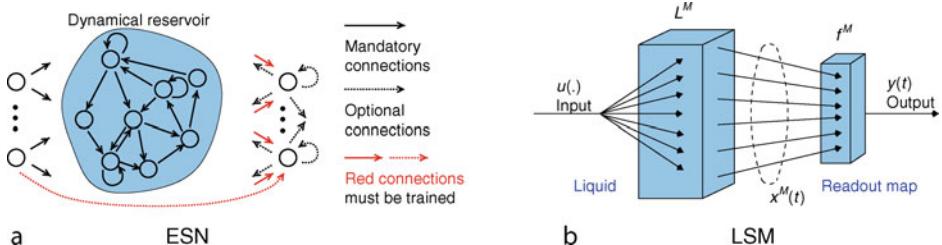
4.2.2 Echo State Network (ESN) and Liquid State Machine (LSM)

The original design of *Echo State Network*, proposed by Jaeger ([2001](#)), was intended to learn time series $\mathbf{u}(1), \mathbf{d}(1), \dots, \mathbf{u}(T)$, and $\mathbf{d}(T)$ with recurrent neural networks. The internal states of the reservoir are supposed to reflect, as an “echo,” the concurrent effect of a new teacher input $u(t+1)$ and a teacher-forcing output $d(t)$, related to the previous time. Therefore, Jaeger’s model includes backward connections from the output layer toward the reservoir (see [Fig. 20a](#)) and the network training dynamics is governed by the following equation:

$$\mathbf{x}(t+1) = f(W^{\text{in}}\mathbf{u}(t+1) + W\mathbf{x}(t) + W^{\text{back}}\mathbf{d}(t)) \quad (7)$$

Fig. 20

Architecture of the two founding models of reservoir computing: ESN and LSM.



where $x(t+1)$ is the new state of the reservoir, W^{in} is the input weight matrix, W the matrix of weights in the reservoir, and W^{back} the matrix of feedback weights, from the output layer to the reservoir. The learning rule for output weights W^{out} (feedforward connections from reservoir to output) consists of a linear regression algorithm, for example least mean squares: At each step, the network states $x(t)$ are collected into a matrix M , after a washout time t_0 , and the sigmoid-inverted teacher output $\tanh^{-1} \mathbf{d}(n)$ into a matrix T , in order to obtain $(W^{\text{out}})^t = M^{\dagger} T$ where M^{\dagger} is the pseudo-inverse of M . In the exploitation phase, the network is driven by novel input sequences $u(t)$, with $t \geq T$ (desired output $d(t)$ is unknown), and it produces the computed output $y(t)$ with coupled equations like:

$$\mathbf{x}(t+1) = f(W^{\text{in}} \mathbf{u}(t+1) + W \mathbf{x}(t) + W^{\text{back}} \mathbf{y}(t)) \quad (8)$$

$$\mathbf{y}(t+1) = f^{\text{out}}(W^{\text{out}}[\mathbf{u}(t+1), \mathbf{x}(t+1), \mathbf{y}(t)]) \quad (9)$$

For the method to be efficient, the network must have the “Echo State Property,” that is, the property of being state contracting, state forgetting, and input forgetting, which gives it a behavior of “fading memory.” Choosing a reservoir weight matrix W with a spectral radius $|\lambda_{\max}|$ slightly lower than 1 is neither a necessary nor a sufficient condition (as discussed in Lukoševičius and Jaeger 2009) but can be applied as a rule of thumb for most practical cases: the common practice is to rescale W after randomly initializing the network connections. An important remark must be made: the condition on the spectral radius is no longer clearly relevant when the reservoir is an SNN with fixed weights, and it totally vanishes when an STDP rule is applied to the reservoir. A comparative study of several measures for the reservoir dynamics, with different neuron models, can be found in Verstraeten et al. (2007).

ESNs have been successfully applied in many experimental settings, with networks no larger than 20–400 internal units, for example, in mastering the benchmark task of learning the Mackey–Glass chaotic attractor (Jaeger 2001). Although the first design of ESN was for networks of sigmoid units, Jaeger has also introduced spiking neurons (LIF model) in the ESNs (Jaeger 2002; Jaeger et al. 2007). Results improve substantially over standard ESNs, for example, in the task of generating a slow sinewave ($\mathbf{d}(n) = 1/5 \sin(n/100)$), which becomes easy with a leaky integrator network (Jaeger 2002).

The basic motivation of the *Liquid State Machine*, defined by Maass et al. (2002b), was to explain how a continuous stream of inputs $u(\cdot)$ from a rapidly changing environment can be processed in real time by recurrent circuits of Integrate-and-Fire neurons (Fig. 20b). The solution they propose is to build a “liquid filter” L^M – the reservoir – that operates similarly to water undertaking the transformation from the low-dimensional space of a set of motors

stimulating its surface into a higher-dimensional space of waves in parallel. The liquid states, $x^M(t)$, are transformed by a readout map, f^M , to generate an output, $y(t)$, that can appear to be stable and appropriately scaled responses given by the network, even if the internal state never converges to a stable attractor. Simulating such a device on neural microcircuits, Maass et al. have shown that a readout neuron receiving inputs from hundreds or thousands of neurons can learn to extract salient information from the high-dimensional transient states of the circuit and can transform transient circuit states into stable outputs.

In mathematical terms, the liquid state is simply the current output of some operator L^M that maps input functions $u(\cdot)$ onto functions $x^M(t)$. The L^M operator can be implemented by a randomly connected recurrent neural network. The second component of an LSM is a “memoryless readout map” f^M that transforms, at every time t , the current liquid state into the machine output, according to the equations:

$$x^M(t) = (L^M(u))(t) \quad (10)$$

$$y(t) = f^M(x^M(t)) \quad (11)$$

The readout is usually implemented by one or several Integrate-and-Fire neurons that can be trained to perform a specific task using very simple learning rules, such as a linear regression or the p-delta rule (Auer et al. 2008).

Often, in implementation, the neural network playing the role of the liquid filter is inspired by biological modeling cortical columns. Therefore, the reservoir has a 3D topology, with a probability of connection that decreases as a Gaussian function of the distance between the neurons.

The readout map is commonly task-specific. However, the hallmark of neural microcircuits is their ability to carry out several parallel real-time computations within the same circuitry. It appears that a readout neuron is able to build a sort of equivalence class among dynamical states, and then to well recognize similar (but not equal) states. Moreover, several readout neurons, trained to perform different tasks, may enable parallel real-time computing.

LSMs have been successfully applied to several nonlinear problems, such as the XOR and many others. LSMs and ESNs are very similar models of reservoir computing that promise to be convenient for both exploiting and capturing most temporal features of spiking neuron processing, especially for time series prediction and for temporal pattern recognition. Both models are good candidates for engineering applications that process temporally changing information.

4.2.3 Related Reservoir Computing Work

An additional work that was linked to the family of “reservoir computing” models after being published is the Backpropagation DeCorrelation rule (BPDC), proposed by Steil (2004). As an extension of the Atiya–Parlos learning rule in recurrent neural networks (Atiya and Parlos 2000), the BPDC model is based on a multilayer network with fixed weights until the last layer. Only this layer has learnable weights both from the reservoir (the multilayer network) to the readout (the last layer) and recurrently inside the readout layer. However, the BPDC model has not been proposed with spiking neurons so far, even if that appears to be readily feasible.

Another approach, by Paugam-Moisy et al. (2008), takes advantage of the theoretical results proving the importance of delays in computing with spiking neurons (see [Sect. 3](#)) for defining a supervised learning rule acting on the delays of connections (instead of weights) between the reservoir and the readout neurons. The reservoir is an SNN, with an STDP rule for adapting the weights to the task at hand, where it can be observed that polychronous groups (see [Sect. 3.2](#)) are activated more and more selectively as training goes on. The learning rule of the readout delays is based on a temporal margin criterion inspired by Vapnik's theory.

There exist reservoir computing networks that make use of evolutionary computation for training the weights of the reservoir, such as Evolino (Schmidhuber et al. 2007), and several other models that are currently proposed, with or without spiking neurons (Devert et al. 2007; Jiang et al. 2008a, b). Although the research area is in rapid expansion, several papers (Verstraeten et al. 2007; Schrauwen et al. 2007b; Lukoševičius and Jaeger 2009) offer valuable surveys.

4.3 Other SNN Research Tracks

Besides trying to apply traditional learning rules to SNNs and the development of reservoir computing, there are many research efforts that relate to learning with spiking neurons.

Much research is, for instance, being carried out on deriving theoretically principled learning rules for spiking neurons, for instance on Information Bottleneck learning rules that attempt to maximize measures of mutual information between input and output spike trains (Barber 2003; Chechik 2003; Pfister et al. 2003, 2006; Bell and Parra 2005; Toyoizumi et al. 2005a, b; Pfister and Gerstner 2006; Bohte and Mozer 2007; Büsing and Maass 2008). The aim of this work on theoretically principled learning is, typically, to arrive at easily understood methods that have spiking neurons carry out some form of Independent Component Analysis (ICA) (Toyoizumi et al. 2005a; Klampfl et al. 2009), or Principal Component Analysis (PCA) (Büsing and Maass 2008), or focus on sparse efficient coding (Olshausen 1996; Volkmer 2004; Lewicki 2002; Smith 2006).

These methods have variable applicability to real world problems, though some have demonstrated excellent performance: Smith and Lewicki (2006) develop an efficient encoding of auditory signals in spike-trains based on sparse over-complete dictionaries that outperforms many standard, filter-based approaches. Forms of reinforcement learning have been developed based on the combination of reward modulation and STDP (Xie and Seung 2001; Izhikevich 2007; Legenstein et al. 2008). Many of these algorithms are highly technical and much of this research is still converging can practical algorithms. We only mention these directions here, the reader can pursue the state of the art in these areas.

Just as advances in neurosciences have contributed to the reevaluation of the significance of the timing and presence of single spikes in neuronal activity, advances in neuropsychology suggest that brain-like systems are able to carry out at least some forms of Bayesian inference (Koerding and Wolpert 2004; Daw and Courville 2008). As a result, the implementation of Bayesian inference algorithms in neural networks has received much attention, with a particular emphasis on networks of spiking neurons.

In this line of research, the activity in neural networks is somehow related to representing probability distributions. Much of this research, however, relies on noisy, stochastic spiking neurons that are characterized by a spike density, and Bayesian inference is implicitly carried out

by large populations of such neurons (Barber et al. 2005; Zemel et al. 1998; Sahani and Dayar 2003; Wu et al. 2003; Rao 2005; Gerwinn 2007; Hays et al. 2007; Ma et al. 2008). As noted by Deneve (2008a), coding probabilities with stochastic neurons “has two major drawbacks. First, [...], it adds uncertainty, and therefore noise, to an otherwise deterministic probability computation. Second, [...], the resulting model would not be self-consistent since the input and output firing rates have different meanings and different dynamics.”

In Deneve (2008a, b), an alternative approach is developed for binary log-likelihood estimation in an SNN. Such binary log-likelihood estimation in an SNN has some known limitations: It can only perform exact inference in a limited family of generative models, and in a hierarchical model only the objects highest in the hierarchy truly have temporal dynamics. Interestingly, in this model neurons still exhibit a Poisson-like distribution of synaptic events. However, rather than reflecting stochasticity due to a noisy firing mechanism, it reflects the sensory input noise. Still, this type of SNN is at the forefront of current developments and many advances in this direction are to be expected.

5 Discussion

This chapter has given an outline of some of the most important ideas, models, and methods in the development of Spiking Neuron Networks, with a focus on pattern recognition and temporal data processing, such as time series. By necessity, many related subjects are not treated in detail here. Variants of the models and methods described in this chapter, and variants thereof, are increasingly being applied to real world pattern recognition.  Section 4.1 listed some results on SNN algorithms applied to traditional pattern recognition, where a dataset of numeric vectors is mapped to a classification. However, as was emphasized in the section on reservoir computing, many interesting application domains have an additional temporal dimension: Not just the immediate data are important, but the *sequence* of data. An increasing amount of work deals with applying SNN concepts to various application domains with important temporal dynamics, such as speech processing, active vision for computers, and autonomous robotics.

5.1 Pattern Recognition with SNNs

Considerable work has focused on developing SNNs that are suitable for speech processing (Verstraeten et al. 2005; Holmberg et al. 2005; Wang and Parel 2005; Loiselle et al. 2005). Verstraeten et al. (2005) have developed a model based on Liquid State Machines that is trained to recognize isolated words. They compare several front-end signal encoding methods, and find that a nature-inspired front-end like a “Lyon Passive Ear” outperforms other methods when an LSM is applied. Similarly, Holmberg et al. (2005) developed a method for automatic speech recognition grounded in SNNs. As a “front end,” they simulated a part of the inner ear, and then simulated octopus spiking neurons to encode the inner-ear signal in a spike train. They subsequently used a fairly simple classifier to recognize speech from both the inner-ear simulation and the spiking neuron spike trains. Wang and Pavel (2005) used an SNN to represent auditory signals based on using the properties of the spiking neuron refractory period. In their SNN, they converted amplitude to temporal code while maintaining the phase information of the carrier. They proposed that for auditory signals the narrow band envelope information could be encoded simply in the temporal inter-spike intervals. Rank order coding

with spiking neural networks has been explored for speech recognition by Loiselle et al. (2005). They show it is an efficient method (fast response/adaptation ability) when having only small training sets.

One important fact that the speech-processing case studies highlight is that traditional preprocessing techniques do not provide optimal front ends and back ends for subsequent SNN processing. Still, many promising features have been pointed out, like robustness to noise, and combining SNN processing with other methods is proposed as a promising research area.

In parallel, a number of SNN-based systems have been developed for computer vision, for example, using spike asynchrony (Thorpe and Gautrais 1997); sparse image coding using an asynchronous spiking neural network (Perrinet and Samuelides 2002); a synchronization-based dynamic vision model for image segmentation (Azhar 2005); saliency extraction with a distributed spiking neural network (Chevallier et al. 2006; Masquelier et al. 2007; Chavallier and Tarroux 2008); and SNNs applied to character recognition (Wysoski et al. 2008).

SNN-based systems also develop increasingly in the area of robotics, where fast processing is a key issue (Maass et al. 2002a; Tenore 2004; Floreano et al. 2005, 2006; Panchev and Wermter 2006; Hartland and Bredeche 2007), from wheels to wings, or legged locomotion. The special abilities of SNNs for fast computing transient temporal patterns make them the first choice for designing efficient systems in the area of autonomous robotics. This perspective is often cited but not yet fully developed.

Other research domains mention the use of SNNs, such as Echo State Networks for motor control (e.g., Salmen and Plöger 2005), prediction in the context of wireless telecommunications (e.g., Jaeger and Haas 2004), or neuromorphic approaches to rehabilitation in medicine (e.g., Kutch 2004). In this context, the *Neuromorphic Engineer* newsletter often publishes articles on applications developed with SNNs (Institute of Neuromorphic Engineering newsletter: <http://www.ine-web.org/>).

It is worth remarking that SNNs are ideal candidates for designing *multimodal interfaces*, since they can represent and process very diverse information in a unifying manner based on time, from such different sources as visual, auditory, speech, or other sensory data. An application of SNNs to audiovisual speech recognition was proposed by Séguier and Mercier (2002). Crépet et al. (2000) developed, with traditional NNs, a modular connectionist model of multimodal associative memory including temporal aspects of visual and auditory data processing (Bouchut et al. 2003). Such a multimodal framework, applied to a virtual robotic prey–predator environment, with spiking neuron networks as functional modules, has proved capable of simulating high-level natural behavior such as cross-modal priming (Meunier and Paugam-Moisy 2004) or real-time perceptive adaptation to changing environments (Chevallier et al. 2005).

5.2 Implementing SNNs

Since SNNs perform computations in such a different way as compared to traditional NNs, the way to program an SNN model for application purposes has to be revised also. The main interest of SNN simulation is to take into account the precise timing of the spike firing, hence the width of the time window used for discrete computation of the successive network states must remain narrow (see  Sect. 1.4), and consequently only a few spike events occur at each time step: In  Fig. 6, only two spikes were fired inside the Δt time range, among the 64 potential connections linking the eight neurons. Hence, inspecting all the neurons and

synapses of the network at each time step is exceedingly time consuming: In this example, a clock-based simulation (i.e., based on a time window) computes zero activity in 97% of the computations! An event-driven simulation is clearly more suitable for sequential simulations of spiking neural networks (Watts 1994; Mattia and Del Giudice 2000; Makino 2003; Rochel and Martinez 2003; Reutimann et al. 2003; McKennoch 2009), as long as the activity of an SNN can be fully described by a set of dated spikes. Nevertheless, event-driven programming that requires the next spike time can be explicitly computed in reasonable time, so that not all models of neurons can be used.

At the same time, SNN simulation can highly benefit from parallel computing, substantially more so than traditional NNs. Unlike a traditional neuron in rate coding, a spiking neuron does not need to receive weight values from each presynaptic neuron at each computation step. Since at each time step only a few neurons are active in an SNN, the classic bottleneck of message passing is removed. Moreover, computing the updated state of the membrane potential (e.g., for an SRM or LIF model neuron) is more complex than computing a weighted sum (e.g., for the threshold unit). Therefore, a communication time and computation cost are much better balanced in SNN parallel implementation as compared to traditional NNs, as proved by the parallel implementation of the *SpikeNET* software (Delorme 1999).

Well-known simulators of spiking neurons include *GENESIS* (Bower and Beeman 1998) and *NEURON* (Hines and Carnevale 1997), but they were designed principally for programming detailed biophysical models of isolated neurons rather than for fast simulation of very large-scale SNNs. However, *NEURON* has been updated with an event-driven mechanism on the one hand (Hines and Carnevale 2004) and a version for parallel machines on the other hand (Hines and Carnevale 2008). *BRIAN* (<http://brian.di.ens.fr/>) is a mainly clock-based simulator with an optional event-driven tool, whereas *MVASpike* (<http://mvaspoke.gforge.inria.fr/>) is a purely event-driven simulator (Roche and Martinez 2003). *DAMNED* is a parallel event-driven simulator (Mouraud and Puzenat 2009). A comparative and experimental study of several SNN simulators can be found in Brette et al. (2007). Most simulators are currently programmed in C or C++. Others are Matlab toolboxes, such as Jaeger's toolbox for ESNs available from the web page (http://www.faculty.jacobs-university.de/hjaeger/esn_research.html), or the *Reservoir Computing Toolbox*, available at <http://snn.elis.ugent.be/node/59> and briefly presented in the last section of Verstraeten (2007). A valuable tool for developers could be *PyNN* (<http://neuralensemble.org/trac/PyNN>), a Python package for simulator-independent specification of neuronal network models.

Hardware implementations of SNNs are also being actively pursued. Several chapters in the book by Maass and Bishop (1999) are dedicated to this subject, and more recent work can be found in Upegui et al. (2004), Hellmich et al. (2005), Johnston et al. (2005), Chicca et al. (2003), Oster et al. (2005), Mitra et al. (2006), and Schrauwen et al. (2007a).

5.3 Conclusion

This chapter has given an overview of the state of the art in Spiking Neuron Networks: its biological inspiration, the models that underlie the networks, some theoretical results on computational complexity and learnability, learning rules, both traditional and novel, and some current application areas and results. The novelty of the concept of SNNs means that many lines of research are still open and are actively being pursued.

References

- Abbott LF (1999) Brain Res Bull 50(5/6):303–304
- Abbott LF, Nelson SB (2000) Synaptic plasticity: taming the beast. Nat Neurosci 3:1178–1183
- Abeles M (1991) Corticonics: neural circuits of the cerebral cortex. Cambridge University Press, Cambridge
- Achard S, Bullmore E (2007) Efficiency and cost of economical brain functional networks. PLoS Comput Biol 3(2):e17
- Atiya A, Parlos AG (2000) New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Trans Neural Netw 11(3): 697–709
- Auer P, Burgsteiner H, Maass W (2008) A learning rule for very simple universal approximators consisting of a single layer of perceptrons. Neural Netw 21(5): 786–795
- Azhar H, Iftekharuddin K, Kozma R (2005) A chaos synchronization-based dynamic vision model for image segmentation. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS, Montreal, pp 3075–3080
- Barabasi AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
- Barber D (2003) Learning in spiking neural assemblies. In: Becker S, Thrun S, Obermayer K (eds) NIPS 2002, Advances in neural information processing systems, vol 15. MIT Press, Cambridge, MA, pp 165–172
- Barber MJ, Clark JW, Anderson CH (2005) Neural representation of probabilistic information, vol 15. MIT Press, Cambridge, MA
- Belatreche A, Maguire LP, McGinnity M (2007) Advances in design and application of spiking neural networks. Soft Comput-A Fusion Found Methodol Appl 11:239–248
- Bell A, Parra L (2005) Maximising information yields spike timing dependent plasticity. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 121–128
- Bi G-q, Poo M-m (1998) Synaptic modification in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and polysynaptic cell type. J Neurosci 18(24):10464–10472
- Bi G-q, Poo M-m (2001) Synaptic modification of correlated activity: Hebb's postulate revisited. Annu Rev Neurosci 24:139–166
- Bialek W, Rieke F, de Ruyter R, van Steveninck RR, Warland D (1991) Reading a neural code. Science 252:1854–1857
- Blum A, Rivest R (1989) Training a 3-node neural net is NP-complete. In: Proceedings of NIPS 1988, advances in neural information processing systems. MIT Press, Cambridge, MA, pp 494–501
- Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1989) Learnability and the Vapnik-Chervonenkis dimension. J ACM 36(4):929–965
- Bobrowski O, Meir R, Shoham S, Eldar YC (2007) A neural network implementing optimal state estimation based on dynamic spike train decoding. In: Schölkopf B, Platt JC, Hoffman T (eds) NIPS 2006, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA, pp 145–152
- Bohte SM, Mozer MC (2007) Reducing the variability of neural responses: a computational theory of spike-timing-dependent plasticity. Neural Comput 19:371–403
- Bohte SM, Kok JN, La Poutre H (2002a) Spike-prop: error-backpropagation in multi-layer networks of spiking neurons. Neurocomputing 48:17–37
- Bohte SM, La Poutre H, Kok JN (2002b) Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. IEEE Trans Neural Netw 13:426–435
- Booij O, tat Nguyen H (2005) A gradient descent rule for spiking neurons emitting multiple spikes. Info Process Lett 95:552–558
- Bouchut Y, Paugam-Moisy H, Puzenat D (2003) Asynchrony in a distributed modular neural network for multimodal integration. In: PDCS 2003, International conference on parallel and distributed computing and systems. ACTA Press, Calgary, pp 588–593
- Bower JM, Beeman D (1998) The book of GENESIS: exploring realistic neural models with the General Simulation system, 2nd edn. Springer, New York
- Brette R, Rudolph M, Hines T, Beeman D, Bower JM et al. (2007) Simulation of networks of spiking neurons: a review of tools and strategies. J Comput Neurosci 23(3):349–398
- Brunel N, Latham PE (2003) Firing rate of the noisy quadratic integrate-and-fire neuron, vol 15. MIT Press, Cambridge
- Buchs NJ, Senn W (2002) Spike-based synaptic plasticity and the emergence of direction selective simple cells: simulation results. J Comput Neurosci 13:167–186
- Büsning L, Maass W (2008) Simplified rules and theoretical analysis for information bottleneck optimization and PCA with spiking neurons. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge
- Câteau H, Fukai T (2003) A stochastic method to predict the consequence of arbitrary forms of spike-timing-dependent plasticity. Neural Comput 15(3):597–620

- Cessac B, Paugam-Moisy H, Viéville T (2010) Overview of facts and issues about neural coding by spikes. *J Physiol (Paris)* 104:5–18
- Chechik G (2003) Spike-timing dependent plasticity and relevant mutual information maximization. *Neural Comput* 15(7):1481–1510
- Chevallier S, Tarroux P (2008) Covert attention with a spiking neural network. In: ICVS'08, Computer Vision Systems, Santorini, 2008. Lecture notes in computer science, vol 5008. Springer, Heidelberg, pp 56–65
- Chevallier S, Paugam-Moisy H, Lemaître F (2005) Distributed processing for modelling real-time multimodal perception in a virtual robot. In: PDCN 2005, International conference on parallel and distributed computing and networks. ACTA Press, Calgary, pp 393–398
- Chevallier S, Tarroux P, Paugam-Moisy H (2006) Saliency extraction with a distributed spiking neuron network. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 209–214
- Chicca E, Badoni D, Dante V, d'Andreagiovanni M, Salina G, Carota L, Fusi S, Del Giudice P (2003) A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Trans Neural Netw* 14 (5):1297–1307
- Crépet A, Paugam-Moisy H, Reynaud E, Puzenat D (2000) A modular neural model for binding several modalities. In: Arabnia HR (ed) IC-AI 2000, International conference on artificial intelligence. CSREA Press, Las Vegas, pp 921–928
- Cybenko G (1988) Approximation by superpositions of a sigmoidal function. *Math Control Signal Syst* 2:303–314
- Daw ND, Courville AC (2008) The pigeon as particle filter. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA
- Delorme A, Gautrais J, Van Rullen R, Thorpe S (1999) SpikeNET: a simulator for modeling large networks of integrate and fire neurons. *Neurocomputing* 26–27:989–996
- Deneve S (2008a) Bayesian spiking neurons. I. Inference. *Neural Comput* 20:91–117
- Deneve S (2008b) Bayesian spiking neurons. II. Learning. *Neural Comput* 20:118–145
- Devert A, Bredeche N, Schoenauer M (2007) Unsupervised learning of echo state networks: a case study in artificial embryogeny. In: Montmarché N et al. (eds) Artificial evolution, selected papers, Lecture Notes in Computer Science, vol 4926/2008, pp 278–290
- Eguiluz VM, Chialvo GA, Cecchi DR, Baliki M, Apkarian AV (2005) Scale-free brain functional networks. *Phys Rev Lett* 94(1):018102
- Ermentrout GB, Kopell N (1986) Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM J Appl Math* 46:233
- Floreano D, Zufferey JC, Nicoud JD (2005) From wheels to wings with evolutionary spiking neurons. *Artif Life* 11(1–2):121–138
- Floreano D, Epars Y, Zufferey J-C, Mattiussi C (2006) Evolution of spiking neural circuits in autonomous mobile robots. *Int J Intell Syst* 21(9):1005–1024
- Funahashi K (1989) On the approximate realization of continuous mappings by neural networks. *Neural Netw* 2(3):183–192
- Gerstner W (1995) Time structure of the activity in neural network models. *Phys Rev E* 51:738–758
- Gerstner W, Kistler WM (2002a) Mathematical formulations of Hebbian learning. *Biol Cybern* 87(5–6): 404–415
- Gerstner W, Kistler W (2002b) Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press, Cambridge
- Gerstner W, van Hemmen JL (1994) How to describe neuronal activity: spikes, rates or assemblies? In: Cowan JD, Tesauro G, Alspector J (eds) NIPS 1993, Advances in neural information processing system, vol 6. MIT Press, Cambridge, MA, pp 463–470
- Gerwinn S, Macke JH, Seeger M, Bethge M (2007) Bayesian inference for spiking neuron models with a sparsity prior. In: NIPS 2006, Advances in neural information processing systems, vol 19. MIT Press, Cambridge, MA
- Hartland C, Bredeche N (2007) Using echo state networks for robot navigation behavior acquisition. In: ROBO'07, Sanya, China
- Hebb DO (1949) The organization of behavior. Wiley, New York
- Heiligenberg W (1991) Neural nets in electric fish. MIT Press, Cambridge, MA
- Hellmich HH, Geike M, Griep P, Rafanelli M, Klar H (2005) Emulation engine for spiking neurons and adaptive synaptic weights. In: IJCNN 2005, International joint conference on neural networks. IEEE-INNS, Montreal, pp 3261–3266
- Hines ML, Carnevale NT (1997) The NEURON simulation environment. *Neural Comput* 9:1179–1209
- Hines ML, Carnevale NT (2004) Discrete event simulation in the NEURON environment. *Neurocomputing* 58–60:1117–1122
- Hines ML, Carnevale NT (2008) Translating network models to parallel hardware in NEURON. *J Neurosci Methods* 169:425–455
- Hodgkin AL, Huxley AF (1952) A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *J Physiol* 117:500–544
- Holmberg M, Gelbart D, Ramacher U, Hemmert W (2005) Automatic speech recognition with neural

- spike trains. In: Interspeech 2005 – Eurospeech, 9th European conference on speech communication and technology, Lisbon, pp 1253–1256
- Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci* 79(8):2554–2558
- Hopfield JJ (1995) Pattern recognition computation using action potential timing for stimulus representation. *Nature* 376:33–36
- Hopfield JJ, Brody CD (2000) What is a moment? “Cortical” sensory integration over a brief interval. *Proc Natl Acad Sci* 97(25):13919–13924
- Hopfield JJ, Brody CD (2001) What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proc Natl Acad Sci* 98(3):1282–1287
- Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2(5):359–366
- Huys QJM, Zemel RS, Natarajan R, Dayan P (2007) Fast population coding. *Neural Comput* 19:404–441
- Izhikevich EM (2003) Simple model of spiking neurons. *IEEE Trans Neural Netw* 14(6):1569–1572
- Izhikevich EM (2004) Which model to use for cortical spiking neurons? *IEEE Trans Neural Netw* 15(5): 1063–1070
- Izhikevich EM (2006) Polychronization: computation with spikes. *Neural Comput* 18(2):245–282
- Izhikevich EM (2007) Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb Cortex* 17(10):2443–2452
- Izhikevich EM, Desai NS (2003) Relating STDP and BCM. *Neural Comput* 15(7):1511–1523
- Izhikevich EM, Gally JA, Edelman GM (2004) Spike-timing dynamics of neuronal groups. *Cereb Cortex* 14:933–944
- Jaeger H (2001) The “echo state” approach to analysing and training recurrent neural networks. Technical Report TR-GMD-148, German National Research Center for Information Technology
- Jaeger H (2002) Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the “echo state network” approach. Technical Report TR-GMD-159, German National Research Center for Information Technology
- Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science* 304(5667):78–80
- Jaeger H, Lukoševičius M, Popović D, Siewert U (2007) Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Netw* 20(3):335–352
- Jiang F, Berry H, Schoenauer M (2008a) Supervised and evolutionary learning of echo state networks. In: Rudolph G et al. (eds) *Proceedings of the 10th international conference on parallel problem solving from nature: PPSN X. Lecture notes in computer science*, vol 5199. Springer, pp 215–224
- Jiang F, Berry H, Schoenauer M (2008b) Unsupervised learning of echo state networks: balancing the double pole. In: GECCO’08: Proceedings of the 10th annual conference on genetic and evolutionary computation, ACM, New York, pp 869–870
- Johnston S, Prasad G, Maguire L, McGinnity T (2005) Comparative investigation into classical and spiking neuron implementations on FPGAs. In: ICANN 2005, International conference on artificial neural networks. *Lecture notes in computer science*, vol 3696. Springer, New York, pp 269–274
- Judd JS (1990) *Neural network design and the complexity of learning*. MIT Press, Cambridge, MA
- Kempter R, Gerstner W, van Hemmen JL (1999) Hebbian learning and spiking neurons. *Phys Rev E* 59(4): 4498–4514
- Kistler WM (2002) Spike-timing dependent synaptic plasticity: a phenomenological framework. *Biol Cybern* 87(5–6):416–427
- Kistler WM, Gerstner W, van Hemmen JL (1997) Reduction of Hodgkin–Huxley equations to a single-variable threshold model. *Neural Comput* 9:1015–1045
- Klampfl S, Legenstein R, Maass W (2009) Spiking neurons can learn to solve information bottleneck problems and extract independent components. *Neural Comput* 21(4):911–959
- Koerding KP, Wolpert DM (2004) Bayesian integration in sensorimotor learning. *Nature* 427:244–247
- Kohonen T (1982) Self-organized formation of topologically correct feature maps. *Biol Cybern* 43:59–69
- Kutch JJ (2004) Neuromorphic approaches to rehabilitation. *Neuromorphic Eng* 1(2):1–2
- Kuwabara N, Suga N (1993) Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. *J Neurophysiol* 69:1713–1724
- LeCun Y, Jackel LD, Bottou L, Cortes C, Denker JS, Drucker H, Guyon I, Muller UA, Sackinger E, Simard P (1995) Learning algorithms for classification: a comparison on handwritten digit recognition, vol 276. World Scientific, Singapore
- Legenstein R, Maass W (2005) What makes a dynamical system computationally powerful? In: Haykin S, Principe JC, Sejnowski TJ, McWhirter JG (eds) *New directions in statistical signal processing: from systems to brain*. MIT Press, Cambridge, MA
- Legenstein R, Näger C, Maass W (2005) What can a neuron learn with spike-time-dependent plasticity? *Neural Comput* 17(11):2337–2382
- Legenstein R, Pecevski D, Maass W (2008) Theoretical analysis of learning with reward-modulated

- spike-timing-dependent plasticity. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA
- Lewicki MS (2002) Efficient coding of natural sounds. *Nat Neurosci* 5:356–363
- Loiselle S, Rouat J, Pressnitzer D, Thorpe S (2005) Exploration of rank order coding with spiking neural networks for speech recognition. In: IJCNN 2005, International joint conference on neural networks. IEEE-INNS Montreal, pp 2076–2080
- Lukoševičius M, Jaeger H (July 2007) Overview of reservoir recipes. Technical Report 11, Jacobs University Bremen
- Ma WJ, Beck JM, Pouget A (2008) Spiking networks for Bayesian inference and choice. *Curr Opin Neurobiol* 18(2):217–222
- Maass W (1997a) Fast sigmoidal networks via spiking neurons. *Neural Comput* 10:1659–1671
- Maass W (1997b) Networks of spiking neurons: the third generation of neural network models. *Neural Netw* 10:1659–1671
- Maass W (2001) On the relevance of time in neural computation and learning. *Theor Comput Sci* 261:157–178 (extended version of ALT'97, in LNAI 1316:364–384)
- Maass W, Bishop CM (eds) (1999) Pulsed neural networks. MIT Press, Cambridge, MA
- Maass W, Natschläger T (1997) Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding. *Netw: Comput Neural Syst* 8(4):355–372
- Maass W, Schmitt M (1997) On the complexity of learning for a spiking neuron. In: COLT'97, Conference on computational learning theory. ACM Press, New York, pp 54–61
- Maass W, Schmitt M (1999) On the complexity of learning for spiking neurons with temporal coding. *Info Comput* 153:26–46
- Maass W, Steinbauer G, Kohlholz R (2002a) Autonomous fast learning in a mobile robot. In: Hager GD, Christensen HI, Bunke H, Klein R (eds) Sensor based intelligent robots, vol 2238. Springer, Berlin, pp 345–356
- Maass W, Natschläger T, Markram H (2002b) Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput* 14(11):2531–2560
- Makino T (2003) A discrete event neural network simulator for general neuron model. *Neural Comput Appl* 11(2):210–223
- Markram H, Tsodyks MV (1996) Redistribution of synaptic efficacy between neocortical pyramidal neurones. *Nature* 382:807–809
- Markram H, Lübke J, Frotscher M, Sakmann B (1997) Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science* 275:213–215
- Masquelier T, Thorpe SJ, Friston KJ (2007) Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput Biol* 3:e31
- Mattia M, Del Giudice P (2000) Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Comput* 12: 2305–2329
- McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 5:115–133
- McKenna S, Voeglin T, Bushnell L (2009) Spike-timing error backpropagation in theta neuron networks. *Neural Comput* 21(1):9–45
- Meunier D (2007) Une modélisation évolutionniste du liaage temporel (in French). PhD thesis, University Lyon 2, http://demeter.univ-lyon2.fr/sdx/theses/lyon2/2007/meunier_d, 2007
- Meunier D, Paugam-Moisy H (2004) A “spiking” bidirectional associative memory for modeling intermodal priming. In: NCI 2004, International conference on neural networks and computational intelligence. ACTA Press, Calgary, pp 25–30
- Meunier D, Paugam-Moisy H (2005) Evolutionary supervision of a dynamical neural network allows learning with on-going weights. In: IJCNN 2005, International joint conference on neural networks. IEEE-INNS, Montreal, pp 1493–1498
- Meunier D, Paugam-Moisy H (2006) Cluster detection algorithm in neural networks. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 19–24
- Mitra S, Fusi S, Indiveri G (2006) A VLSI spike-driven dynamic synapse which learns only when necessary. In: Proceedings of IEEE international symposium on circuits and systems (ISCAS) 2006. IEEE Press, New York, p 4
- Mouraud A, Paugam-Moisy H (2006) Learning and discrimination through STDP in a top-down modulated associative memory. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 611–616
- Mouraud A, Paugam-Moisy H, Puzenat D (2006) A distributed and multithreaded neural event driven simulation framework. In: PDCN 2006, International conference on parallel and distributed computing and networks, Innsbruck, Austria, February 2006. ACTA Press, Calgary, 2006
- Mouraud A, Puzenat D (2009) Simulation of large spiking neuron networks on distributed architectures, the “DAMNED” simulator. In: Palmer-Brown D, Draganova C, Pimenidis E, Mouratidis H (eds) EANN 2009, Engineering applications of neural networks. Communications in computer

- and information science, vol 43. Springer, pp 359–370
- Natschläger T, Ruf B (1998a) Online clustering with spiking neurons using radial basis functions. In: Hamilton A, Smith LS (eds) Neuromorphic systems: engineering silicon from neurobiology. World Scientific, Singapore, Chap 4
- Natschläger T, Ruf B (1998b) Spatial and temporal pattern analysis via spiking neurons. *Netw: Comp Neural Syst* 9(3):319–332
- Newman MEJ (2003) The structure and function of complex networks. *SIAM Rev* 45:167–256
- Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. *Phys Rev E* 69:026113
- Nowotny T, Zhigulin VP, Selverston AI, Abarbanel HDI, Rabinovich MI (2003) Enhancement of synchronization in a hybrid neural circuit by spike-time-dependent plasticity. *J Neurosci* 23(30):9776–9785
- Olshausen BA, Fields DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381:607–609
- Oster M, Whatley AM, Liu S-C, Douglas RJ (2005) A hardware/software framework for real-time spiking systems. In: ICANN 2005, International conference on artificial neural networks. Lecture notes in computer science, vol 3696. Springer, New York, pp 161–166
- Panchev C, Wermter S (2006) Temporal sequence detection with spiking neurons: towards recognizing robot language instructions. *Connect Sci* 18:1–22
- Paugam-Moisy H, Martinez R, Bengio S (2008) Delay learning and polychronization for reservoir computing. *Neurocomputing* 71(7–9):1143–1158
- Perrinet L, Samuelides M (2002) Sparse image coding using an asynchronous spiking neural network. In: Verleysen M (ed) ESANN 2002, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 313–318
- Pfister J-P, Gerstner W (2006) Beyond pair-based STDP: a phenomenological rule for spike triplet and frequency effects. In: NIPS 2005, Advances in neural information processing systems, vol 18. MIT Press, Cambridge, MA, pp 1083–1090
- Pfister J-P, Barber D, Gerstner W (2003) Optimal Hebbian learning: a probabilistic point of view. In: Kaynak O, Alpaydin E, Oja E, Xu L (eds) ICANN/ICONIP 2003, International conference on artificial neural networks. Lecture notes in computer science, vol 2714. Springer, Heidelberg, pp 92–98
- Pfister J-P, Toyozumi T, Barber D, Gerstner W (2006) Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Comput* 18(6):1318–1348
- Poggio T, Girosi F (1989) Networks for approximation and learning. *Proc IEEE* 78(9):1481–1497
- Rao RPN (2005) Hierarchical Bayesian inference in networks of spiking neurons. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 1113–1120
- Reccre M (1999) Encoding information in neuronal activity. In: Maass W, Bishop CM (eds) Pulsed neural networks. MIT Press, Cambridge
- Reutimann J, Giugliano M, Fusi S (2003) Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Comput* 15(4):811–830
- Rochel O, Martinez D (2003) An event-driven framework for the simulation of networks of spiking neurons. In: Verleysen M (ed) ESANN'03, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 295–300
- Rubin J, Lee DD, Sompolinsky H (2001) Equilibrium properties of temporal asymmetric Hebbian plasticity. *Phys Rev Lett* 86:364–366
- Rudolph M, Destexhe A (2006) Event-based simulation strategy for conductance-based synaptic interactions and plasticity. *Neurocomputing* 69: 1130–1133
- Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by back-propagating errors. *Nature* 323:533–536
- Sahani M, Dayan P (2003) Doubly distributional population codes: Simultaneous representation of uncertainty and multiplicity. *Neural Comput* 15:2255–2279
- Salmen M, Plöger PG (2005) Echo state networks used for motor control. In: ICRA 2005, International joint conference on robotics and automation. IEEE, New York, pp 1953–1958
- Saudargiene A, Porr B, Wörgötter F (2004) How the shape of pre- and postsynaptic signals can influence STDP: a biophysical model. *Neural Comput* 16(3): 595–625
- Schmidhuber J, Wiestra D, Gagliolo D, Gomez M (2007) Training recurrent networks by Evolino. *Neural Comput* 19(3):757–779
- Schmitt M (1998) On computing Boolean functions by a spiking neuron. *Ann Math Artif Intell* 24: 181–191
- Schmitt M (2004) On the sample complexity of learning for networks of spiking neurons with nonlinear synaptic interactions. *IEEE Trans Neural Netw* 15(5): 995–1001
- Schrauwen B, Van Campenhout J (2004a) Extending SpikeProp. In: Proceedings of the international joint conference on neural networks, vol 1. IEEE Press, New York, pp 471–476
- Schrauwen B, Van Campenhout J (2004b) Improving spikeprop: enhancements to an error-backpropagation rule for spiking neural networks. In: Proceedings of the 15th ProRISC workshop, vol 11

- Schrauwen B, D'Haene M, Verstraeten D, Van Campenhout J (2007a) Compact hardware for real-time speech recognition using a liquid state machine. In: IJCNN 2007, International joint conference on neural networks, 2007, pp 1097–1102
- Schrauwen B, Verstraeten D, Van Campenhout J (2007b) An overview of reservoir computing: theory, applications and implementations. In: Verleysen M (ed) ESANN'07, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 471–482
- Schrauwen B, Büsing L, Legenstein R (2009) On computational power and the order-chaos phase transition in reservoir computing. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) NIPS'08, advances in neural information processing systems, vol 21. MIT Press, Cambridge, MA, pp 1425–1432
- Séguié R, Mercier D (2002) Audio-visual speech recognition one pass learning with spiking neurons. In: ICANN'02, International conference on artificial neural networks. Springer, Berlin, pp 1207–1212
- Senn W, Markram H, Tsodyks M (2001) An algorithm for modifying neurotransmitter release probability based on pre- and post-synaptic spike timing. *Neural Comput* 13(1):35–68
- Siegelmann HT (1999) Neural networks and analog computation, beyond the Turing limit. Birkhäuser, Boston, MA
- Sima J, Sgall J (2005) On the nonlearnability of a single spiking neuron. *Neural Comput* 17(12):2635–2647
- Smith EC, Lewicki MS (2006) Efficient auditory coding. *Nature* 439:978–982
- Song S, Miller KD, Abbott LF (2000) Competitive Hebbian learning through spike-time dependent synaptic plasticity. *Nat Neurosci* 3(9):919–926
- Sporns O, Tononi G, Kotter R (2005) The human connectome: a structural description of the human brain. *PLoS Comp Biol* 1(4):e42
- Standage DI, Trappenberg TP (2005) Differences in the subthreshold dynamics of leaky integrate-and-fire and Hodgkin-Huxley neuron models. In: IJCNN 2005, International joint conference on neural networks. IEEE-INNS, Montreal, pp 396–399
- Steil JJ (2004) Backpropagation-decorrelation: Online recurrent learning with $O(n)$ complexity. In: IJCNN 2004, International joint conference on neural networks, vol 1. IEEE-INNS, Montreal, pp 843–848
- Stein RB (1965) A theoretical analysis of neuronal variability. *Biophys J* 5:173–194
- Tenore F (2004) Prototyping neural networks for legged locomotion using custom aVLSI chips. *Neuromorphic Eng* 1(2):4, 8
- Thorpe SJ, Gautrais J (1997) Rapid visual processing using spike asynchrony. In: Mozer M, Jordan MI, Petsche T (eds) NIPS 1996, Advances in neural information processing systems, volume 9. MIT Press, Cambridge, MA, pp 901–907
- Thorpe S, Fize D, Marlot C (1996) Speed of processing in the human visual system. *Nature* 381(6582): 520–522
- Thorpe S, Delorme A, Van Rullen R (2001) Spike-based strategies for rapid processing. *Neural Netw* 14:715–725
- Toyoizumi T, Pfister J-P, Aihara K, Gerstner W (2005a) Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. *Proc Natl Acad Sci USA* 102(14): 5239–5244
- Toyoizumi T, Pfister J-P, Aihara K, Gerstner W (2005b) Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 1409–1416
- Turing AM (1939) Systems of logic based on ordinals. *Proc Lond Math Soc* 45(2):161–228
- Turing AM (1950) Computing machinery and intelligence. *Mind* 59:433–460
- Upegui A, Peña Reyes CA, Sanchez E (2004) An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocess Microsyst* 29:211–223
- Valiant LG (1984) A theory of the learnable. *Commun ACM* 27(11):1134–1142
- van Hulle M (2000) Faithful representations and topographic maps: from distortion- to information-based self-organization. Wiley, New York
- Van Rullen R, Thorpe S (2001) Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Comput* 13: 1255–1283
- Vapnik VN (1998) Statistical learning theory. Wiley, New York
- Verstraeten D, Schrauwen B, Stroobandt D (2005) Isolated word recognition using a liquid state machine. In: Verleysen M (ed) ESANN'05, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 435–440
- Verstraeten D, Schrauwen B, D'Haene M, Stroobandt D (2007) An experimental unification of reservoir computing methods. *Neural Netw* 20(3):391–403
- Viéville T, Crahay S (2004) Using an Hebbian learning rule for multi-class SVM classifiers. *J Comput Neurosci* 17(3):271–287
- Volkmer M (2004) A pulsed neural network model of spectro-temporal receptive fields and population coding in auditory cortex. *Nat Comput* 3: 177–193
- Wang G, Pavel M (2005) A spiking neuron representation of auditory signals. In: IJCNN 2005, International

- joint conference on neural networks. IEEE-INNS, Montreal, pp 416–421
- Watts L (1994) Event-driven simulation of networks of spiking neurons. In: Cowan JD, Tesauro G, Alspector J (eds) NIPS 1993, Advances in neural information processing systems, vol 6. MIT Press, Cambridge, MA, pp 927–934
- Watts D, Strogatz S (1998) Collective dynamics of “small-world” networks. *Nature* 393:440–442
- Wennekers T, Sommer F, Aertsen A (2003) Editorial: cell assemblies. *Theory Biosci (special issue)* 122:1–4
- Wu S, Chen D, Niranjan M, Amari S (2003) Sequential Bayesian decoding with a population of neurons. *Neural Comput* 15:993–1012
- Wysoski SG, Benuskova L, Kasabov N (2008) Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing* 71(13–15):2563–2575
- Xie X, Seung HS (2004) Learning in neural networks by reinforcement of irregular spiking. *Phys Rev E* 69 (041909)
- Xin J, Embrechts MJ (2001) Supervised learning with spiking neuron networks. In: Proceedings of the IJCNN 2001 IEEE international joint conference on neural networks, Washington, DC, vol 3. IEEE Press, New York, pp 1772–1777
- Zador AM, Pearlmuter BA (1996) VC dimension of an integrate-and-fire neuron model. *Neural Comput* 8(3):611–624
- Zemel RS, Dayan P, Pouget A (1998) Probabilistic interpretation of population codes. *Neural Comput* 10:403–430

11 Image Quality Assessment — A Multiscale Geometric Analysis-Based Framework and Examples

Xinbo Gao¹ · Wen Lu² · Dacheng Tao³ · Xuelong Li⁴

¹Video and Image Processing System Lab, School of Electronic Engineering, Xidian University, China
xbgao@ieee.org

²Video and Image Processing System Lab, School of Electronic Engineering, Xidian University, China
luwen@mail.xidian.edu.cn

³School of Computer Engineering, Nanyang Technological University, Singapore
dacheng.tao@gmail.com

⁴Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, Shaanxi, China
xuelong_li@opt.ac.cn

1	<i>Introduction</i>	378
2	<i>Multiscale Geometric Analysis</i>	383
3	<i>Multiscale Geometric Analysis for Image Quality Assessment</i>	385
4	<i>Performance Evaluation</i>	391
5	<i>Conclusion</i>	397

Abstract

This chapter is about objective *image quality assessment* (IQA), which has been recognized as an effective and efficient way to predict the visual quality of distorted images. Basically, IQA has three different dependent degrees on original images, namely, *full-reference* (FR), *no-reference* (NR), and *reduced-reference* (RR). To begin with, we introduce the fundamentals of IQA and give a broad treatment of the state-of-the-art. We focus on a novel framework for IQA to mimic the *human visual system* (HVS) by incorporating the merits from *multiscale geometric analysis* (MGA), *contrast sensitivity function* (CSF), and Weber's law of *just noticeable difference* (JND). Thorough empirical studies were carried out using the laboratory for image and video engineering (LIVE) database against subjective *mean opinion score* (MOS), and these demonstrate that the proposed framework has good consistency with subjective perception values and the objective assessment results well reflect the visual quality of the images.

1 Introduction

With the rapid development of information technology, digital image as a media for representing and communicating has witnessed tremendous growth. A huge number of processing methods have been proposed to treat images for different purposes. The performance of these methods highly depends on the quality of the images after processing. Therefore, how to evaluate image quality has become a burning question in recent years. Problems of *image quality assessment* (IQA) (Wang and Bovik 2006) occur in many applications, such as image compression, enhancement, communication, storage and watermarking, etc. In the process of image compression, lossy compression techniques may introduce artificial block, blurring and ringing effects, which can lead to image degradation. In poor transmission channels, transmission errors or data dropping may happen, which can lead to imperfect quality and distortion of the received video data. The past 5 years have witnessed tremendous demands for IQA methods in the following three ways: (1) they can be used to monitor image quality for quality control systems; (2) they can be employed to benchmark image processing systems and algorithms; (3) they can also be embedded into image processing systems to optimize algorithms and parameter settings.

Existing IQA metrics can be categorized into subjective (Rec. ITU-R) and objective methods (Wang and Bovik 2006). The former is based on quality, which is assessed by human observers; the latter depends on quantified parameters, which are obtained from the model to measure the image quality.

Because human observers are the ultimate receivers of the visual information contained in an image, a subjective method whose results are directly given by human observers is probably the best way to assess the quality of an image. The subjective method is one in which the observers are asked to evaluate the picture quality of sequences using a continuous grading scale and to give one score for each sequence. A number of different subjective methods are represented by ITU-R Recommendation BT.500 (Rec. ITU-R). The subjective quality measurement has been used for many years as the most reliable form of quality measurement. However, subjective experiment requires human viewers working over a long period and repeated experiments are needed for many image objects. Furthermore, it is expensive, time consuming, and cannot be easily and routinely performed for many scenarios, for example,

real-time systems. Moreover, there currently is no precise mathematical model for subjective assessment. As a consequence, there is a need for an objective quality metric that accurately matches the subjective quality and can be easily implemented in various image systems, leading to the emergence of objective IQA.

Objective IQA makes use of the variation of several original or distorted image characteristics caused by degradation to represent the variation of image perceptual quality. Many objective quality metrics for predicting image distortions have been investigated. According to the availability of original image information, there is a general agreement (Wang and Bovik 2006) that objective quality metrics can be classified into three categories: *full-reference* (FR), *no-reference* (NR), and *reduced-reference* (RR).

To evaluate the quality of a distorted image, FR metrics, which have access to both whole original and reconstructed information, provide the most precise evaluation results compared with NR and RR. Generally, FR metrics can be divided into two steps: one tends to construct the errors between original and distorted images, then produce distortion maps; the other provides the global IQA by pooling the errors. Conventional FR IQA methods (Avcibas et al. 2002; Eskicioglu and Fisher 1995) calculate pixel-wise distances, for example, *peak signal-to-noise ratio* (PSNR) and *mean square error* (MSE), between the distorted image and the corresponding reference. These measurements are attractive because of their simplicity and good performance when images with the same type of degradation are compared. However, when images with different types of degradations are compared, the results may not be consistent with that of subjective IQA. Besides, images with various types of degradations but the same value could have very different subjective qualities. This is mainly because they are based on pixel-to-pixel difference calculations, which disregard image content and some characteristics of human visual perception.

Recently, a great deal of effort has been made to develop visual models (Mannos and Sakrison 1974) that take advantage of the known characteristics of the *human visual system* (HVS). The aim of HVS-based IQA is to evaluate how strong the error signal is perceived by the HVS according to the characteristics of human visual error sensitivity. A number of IQA methods have been proposed to evaluate the perceptual quality. Two famous models, the Daly *Visible Differences Predictor* (VDP) and Sarnoff *Visual Discrimination Model* (VDM), were proposed by Daly (1993) and Lubin (1995), respectively. The Daly VDP receives the distorted and corresponding reference as input and produces a difference map as output, which predicts the probability of detection for dissimilarities throughout the whole image. If two images vary substantially, the probability of prediction will be one, and as the differences aggravate, the probability does not increase further. The Sarnoff VDM was designed for physiological plausibility as well as for speed and simplicity. While the Daly VDP is an example of a frequency domain visual model, the Sarnoff VDM operates in the spatial domain. In Karunasekera and Kingsbury (1995), these two excellent models were comparatively evaluated. The Watson (1993) metric uses the DCT-based perceptual error measurement. The quantified errors for every coefficient in every block are scaled by the corresponding visual sensitivities of every DCT basis function in each block. Miyahara et al. (1998) reported a new methodology for the determination of an objective metric. This methodology is applied to obtain a *picture quality scale* (PQS) for the coding of achromatic images over the full range of the image quality presented by the *mean opinion score* (MOS). Damera-Venkata et al. (2000) modeled the degraded image as a reference image polluted by linear frequency distortion and additive noise contamination. Since the psycho-visual effects of frequency distortion and noise

contamination are independent, they decouple these two sources of degradation and measure their effect on the HVS. Instead of computing a residual image, they compute a model restored image by applying the restoration algorithm on the original image, using the same parameters as those used while restoring the degraded image. A number of limitations of HVS-based IQA methods were discussed in Wang and Bovik (2006), because they must rely on several strong assumptions and generalizations.

Different from traditional HVS-based error-sensitivity IQA, structural similarity-based IQA (Wang and Bovik 2006) is based on the following philosophy: *The main function of the HVS is to extract structural information from the viewing field, and the HVS is highly adapted for this purpose.* Therefore, measurements of structural information loss can provide a good approximation to image perceived distortion. Wang and Bovik (2006) introduces a complementary framework for IQA based on the degradation of structural information and develops a structural similarity (SSIM) index to demonstrate its promise through a set of intuitive examples. Extensive experimental results have demonstrated that SSIM achieves better performance compared with the traditional methods. Sheikh et al. proposed to quantify the loss of image information in the distortion process and explore the relationship between image information and visual quality. They modeled natural images in the wavelet domain using *Gaussian scale mixtures* (GSM) (Wang and Bovik 2006) and employed the *natural scene statistics* (NSS) model for IQA (Wang and Bovik 2006). The *visual information fidelity* (VIF) method was derived from the combination of a statistical model for natural scenes, an image distortion model, and a HVS model in an information-theoretic setting. The experimental results demonstrate that it outperformed traditional IQA methods by a sizable margin. Sheikh et al. also furnished a statistical evaluation of recent FR algorithms (Tao et al. 2007b).

Since FR metrics need full information of images on the comparison between corresponding regions of the original image and the degraded image, this requirement makes the metrics less than optimal for some applications, which require images and videos to be broadcasted or transmitted through the data network. For these applications, FR metrics might be impossible or too expensive to allocate the extra bandwidth that is required to send information of images. Therefore, researchers are seeking a rational IQA method that could work without any prior information. NR IQA emerges in demand, it does not require any information of the original image and only uses a distortion analysis of the distorted image to assess its quality. It addresses a fundamental distinction between fidelity and quality, that is, HVS usually does not need any reference to determine the subjective quality of a target image. This kind of metric is most promising in the context of a video broadcast scenario, since the original images or videos are not accessible to terminal users in practice.

NR IQA is a relatively new research direction with a promising future. However, NR is also a difficult task, and most NR metrics are designed for one or a set of predefined specific distortion types and are unlikely to generalize for images with other types of distortions, for example, the blocking effect in JPEG, the ringing and blurring artifacts in JPEG2000. In practical applications, they are useful only when the distortion type of the distorted images are fixed and known. So there is a big gap between NR metrics and real scenarios. Wang and Bovik (2006) proposed a computational and memory efficient quality model for JPEG images. Li (2002) proposed to appraise the image quality by three objective measures: edge-sharpness level, random noise level, and structural noise level, which jointly provide a heuristic approach to characterizing most important aspects of visual quality. Sheikh et al. (2006) use NSS models to blindly measure the quality of the image compressed using the JPEG2000 scheme.

They claim that natural scenes contain nonlinear dependencies that are disturbed by the compression process, and this disturbance can be quantified and related to human perceptual quality.

FR metrics may not be applicable because of the absence of original images. On the other hand, NR or “blind” IQA is such an extremely difficult task that it is impossible to apply a universal NR metric to practical applications. Although there is an urgent demand for NR QA methods that are applicable to a wide variety of distortions, unfortunately no such method has been proposed and extensively tested. RR IQA provides an alternative solution that compromises the FR and NR methods. It only makes use of partial information from the original images to evaluate the perceptual quality of the distorted image. In general, certain features or physical measures are extracted from the original image and then transmitted to the receiver as extra information for evaluating the quality of the image or video.

RR metrics may be less accurate for evaluating image quality than the FR metrics, but they are less complicated and make real-time implementations more affordable. RR metrics are very useful for monitoring quality on transmission network. In such contexts, the features are transmitted with the coded sequence if they correspond to a reasonable overhead. They also can be used to track image quality degradations and control the streaming resources in real-time visual communication systems. Recently, the VQEG (2000a) report has included RR image and video quality assessment as one of its directions for future development. RR methods are useful in a number of applications. Masry et al. (2006) presented a computationally efficient video distortion metric that can operate in FR and RR model as required. This metric is based on a model of the HVS and implemented using the wavelet transform and separable filters. The visual model is parameterized using a set of video frames and the associated quality scores. Wolf and Pinson (2005) presented a new RR video quality monitoring system under low bandwidth. This system utilizes less than 10 kbits/s of reference information from the source video stream.

In the RR model, the key issue is to determine which features have the best ability to capture distortion between the original image and the distorted image. The best features are the ones that are able to produce the highest correlation between objective quality scores and subjective ones. In order to select the best feature to design a reduced description IQA method, Lu et al. (Gao et al. 2008a; Lu et al. 2008; Li et al. 2009) utilized some transforms to extract features for representing images based on HVS. Wang and Bovik (2006) proposed an RR IQA method called the *wavelet-domain natural image statistic metric* (WNISM), which achieves promising performance for image visual perception quality evaluation. The underlying factor in WNISM is that the marginal distribution of wavelet coefficients of a natural image conforms to the generalized Gaussian distribution. Based on this fact, WNISM measures the quality of a distorted image by the fitting error between the wavelet coefficients of the distorted image and the Gaussian distribution of the reference. They use the Kullback–Leibler distance to represent this error, so that only a relatively small number of RR features are needed for the evaluation of image quality.

In recent years, neural computing has emerged as a practical technology, with successful applications in many fields (Petersena et al. 2002). The majority of these applications are concerned with problems in image processing. They can address most of the various steps that are involved in the image processing chain: from the preprocessing to the image understanding level. For example, in the process of image construction, Wang and Wahl trained a Hopfield *artificial neural network* (ANN) for the reconstruction of 2D images from pixel data obtained

from projections (Wang and Wahl 1997). In image enhancement, Chandrasekaran et al. (1996) used a novel feed-forward architecture to classify an input window as containing an edge or not. The weights of the network were set manually instead of being obtained from training. In the phase of image understanding, it is considered as part of artificial intelligence or perception, which involves recognition, classification, and relational matching. So it seems that neural networks can be used to advantage in certain problems of image understanding, especially in feature extraction. Feature extraction can be seen as a special kind of data reduction, of which the goal is to find a subset of informative variables based on image data. In image processing, an effective approach to extract feature can be employed to represent image sparsely. A well-known feature-extraction ANN is Oja's neural implementation of a one-dimensional *principal component analysis* (PCA) (Oja 1982). To aim at IQA, feature extraction is probably the most important stage, and effective features can well reflect the quality of digital images and vice versa. So the neural network for feature extraction is used to evaluate image quality in RR mode. In Bishop (1995) and Lampinen and Oja (1998), Callet et al. (2006) use a *convolutional neural network* (CNN) that allows a continuous time scoring of the video to complete the QA in MPEG-2 video. Objective features are extracted on a frame-by-frame basis on both the reference and the distorted sequences. They are derived from a perceptual-based representation and integrated along the temporal axis using a *time-delay neural network* (TDNN). By realizing a nonlinear mapping between nonsubjective features extracted from the video frames and subjective scores, experimental results demonstrated that TDNN can be useful to assess the perceived quality of video sequences. Gastaldo and Zunino (2004) use a *circular back propagation* (CBP) feed-forward network to process objective features extracted from JPEG images and to return the associated quality scores. Gastaldo et al. (2002) feed the CBP network, estimating the corresponding perceived quality; objective features are continuously extracted from compressed video streams on a frame-by-frame basis. The resulting adaptive modeling of subjective perception supports a real-time system for monitoring displayed video quality.

Although the aforementioned RR metrics achieve a good solution for IQA problems, there is still much room to further improve the performance of RR IQA, because the existing methods fail to consider the statistical correlations of transformed coefficients in different subbands and the visual response characteristics of the mammalian cortical simple cells. Moreover, wavelet transforms cannot explicitly extract the image geometric information; for example, lines, curves, and wavelet coefficients are dense for smooth image edge contours.

To target the aforementioned problems, to further improve the performance of RR IQA, and to broaden RR IQA related applications, a novel HVS driven framework is proposed. The new framework is consistent with HVS: MGA decomposes images for feature extraction to mimic the multichannel structure of HVS, CSF reweights MGA decomposed coefficients to mimic the nonlinearities inherent in HVS, and JND produces a noticeable variation in sensory experience. This framework contains a number of different ways for IQA because MGA offers a series of transforms including wavelet (Mallet 1989), contourlet (Do and Vetterli 2005), *wavelet-based contourlet transform* (WBCT) (Eslami and Radha 2004), and *hybrid wavelets and directional filter banks* (HWD) (Eslami and Radha 2005), and different transforms capture different types of image geometric information. Extensive experiments based on the laboratory for image and video engineering (LIVE) database (Sheikh et al. 2003) against subjective MOS (VQEG 2000a) have been conducted to demonstrate the effectiveness of the new framework.

2 Multiscale Geometric Analysis

Multiscale geometric analysis (MGA) (Romberg 2003) is such a framework for optimally representing high-dimensional function. It is developed, enhanced, formed, and perfected in signal processing, computer vision, machine learning, and statistics. MGA can detect, organize, represent, and manipulate data, for example, edges, which nominally span a high-dimensional space but contain important features approximately concentrated on lower-dimensional subsets, for example, curves.

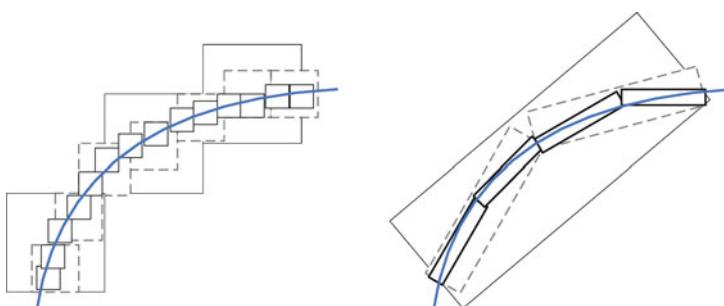
In recent decades, scientists devoted themselves to finding a simple way to present images. With the advances in science, MGA has formed a big family with a large number of members. Wavelet is the first one that can analyze images by multiscale and multidirection transforms, and can recover the original image in a lossless way. Due to their good *nonlinear approximation* (NLA) performance for piecewise smooth functions in one dimensions, wavelets have been successfully applied to many image processing tasks, such as low bit-rate compression and denoising. To utilize wavelets, we can catch point or zero-dimensional discontinuities effectively. However, in two-dimension or other higher dimensions, the wavelet is not able to depict the singulars efficiently and effectively. In essence, wavelets in two-dimension obtained by a tensor-product of one-dimensional wavelets will be good at isolating the discontinuities at edge points, but will not see the smoothness along the contours.

Taking [Fig. 1](#) as an example (Do and Vetterli 2005) for approximating the contour efficiently, the wavelet is limited to using brushes of square shapes along the contour, with different sizes corresponding to the multiresolution of wavelets. As the resolution gets finer, the limitation of the wavelet scheme can be clearly seen since it requires many finer dots to capture the contour. The X-let styles, which are the expected transforms, on the other hand, have more freedom in making brush strokes in different directions and rectangular shapes that follow the contour. As hoped, the expected NLA methods are much more efficient than the wavelet.

In addition, wavelets have only three directions and lack the important feature of directionality; hence, they are not efficient in retaining textures and fine details in these applications. There have been several efforts toward developing geometrical image transforms. According to the above analysis, the expected multiscale geometrical image transforms should contain the following features (Do and Vetterli 2005). Multi-resolution, localization, critical sampling, directionality, and anisotropy.

Fig. 1

The wavelet versus other lets approximating the 2D contour.



Recently, some more powerful MGA representations were created. In 1998, Cands pioneered a nonadaptive method for high dimension functions representation, named Ridgelet (Cands 1998). And in the same year, Donoho gave a method for constructing the orthonormal ridgelet. The ridgelet can approximate the multivariable functions containing the line-singularity effectively, but for the others with curve-singularity, it performs only as well as the wavelet. In 2000, Pennec and Mallat proposed a new system of representation, called bandelet (Pennec and Mallat 2000). It can represent the images based on the edges and track the geometrically regular directions of images. Thus, if one knew the geometrically regular directions of the images, bandelet would lead to optimal sparse representations for images and have great potential in image compression. In order to achieve optimal approximation behavior in a certain sense for 2D piecewise smooth functions in R^2 where the discontinuity curve is a C^2 function, Cands and Donoho constructed the curvelet (Cands and Donoho 2000) transform. More specifically, an M-term NLA for such piecewise smooth function using curvelets has L^2 square error decaying. An attractive property of the curvelet system is that such correct approximation behavior is simply obtained via thresholding a fixed transform. The key features of the curvelet elements are that they exhibit very high directionality and anisotropy. However, the original construction of the curvelet transform is intended for functions defined in the continuum space; when the critical sampling is desirable, the development of discrete transforms for sampled images remains a challenge.

Based on the key features, namely directionality and anisotropy, which make curvelets an efficient representation for 2D piecewise smooth functions with smooth discontinuity curves, Do and Vetterli proposed a new image transform, contourlet, which is also called the *pyramidal directional filter bank* (PDFB) (Bamberger and Smith 1992). The contourlet first uses the Laplacian pyramid (LP) (Burt and Adelson 1983) to decompose the image and capture point singularity. Then, it combines all the singular points in every direction into one coefficient by directional filter bank. The contourlet provides a multiscale and directional decomposition for images with a small redundancy factor, and a frame expansion for images with frame elements like contour segments, which is the reason that it is named contourlet. The connection between the developed discrete- and continuous-domain constructions is made precise via a new directional multi-resolution analysis, which provides successive refinements at both spatial and directional resolutions. The contourlet transform can be designed to satisfy the anisotropy scaling relation for curves and thus it provides a curvelet-like decomposition for images. The contourlet transform aims to achieve an optimal approximation rate of piecewise smooth functions with discontinuities along twice continuously differentiable curves. Therefore, it captures areas with subsection smooth contours. WBCT and HWD are designed to optimize the representation of image features without redundancy.

For IQA, one needs to find MGA transforms that perform excellently for reference image reconstruction, have perfect perception of orientation, are computationally tractable, and are sparse and effective for image representation. Among all requirements for IQA, the effective representation of visual information is especially important. Natural images are not simple stacks of 1D piecewise smooth scan-lines, and points of discontinuity are typically located along smooth curves owing to smooth boundaries of physical objects. As a result of a separable extension from 1D bases, the 2D wavelet transform is not good at representing visual information of images. Consequently, when we dispose of the image with linear features the wavelet transform is not effective. However, MGA transforms can capture the characteristics of images, for example, lines, curves, cuneiforms, and the contours of object. As mentioned in ➤ [Table 1](#), different transforms of MGA capture different features of an image and

Table 1**The performance of PSNR, MSSIM, and WNISM on the LIVE database**

Metric	Type	JPEG					JPEG2000				
		CC	ROCC	OR	MAE	RMSE	CC	ROCC	OR	MAE	RMSE
PSNR	FR	0.9229	0.8905	0.1886	7.118	9.154	0.933	0.9041	0.0947	6.407	8.313
MSSIM	FR	0.9674	0.9485	0.04	4.771	5.832	0.949	0.9368	0.0651	5.422	6.709
WNISM	RR	0.9291	0.9069	0.1486	6.0236	8.2446	0.9261	0.9135	0.1183	6.135	7.9127

complement each other. Based on the mentioned requirements for IQA, it is reasonable to consider a wide range of explicit interactions between multiscale methods and geometry, for example, contourlet (Do and Vetterli 2005), WBCT (Eslami and Radha 2004), and HWD (Eslami and Radha 2005).

3 Multiscale Geometric Analysis for Image Quality Assessment

As discussed in [Sect. 2](#), MGA contains a series of transforms, which can analyze and approximate geometric structure while providing near optimal sparse representations. The image sparse representation means we can represent the image by a small number of components, so small visual changes of the image will affect these components significantly. Therefore, sparse representations can be well utilized for IQA. In this chapter, a novel framework for IQA (Gao et al. 2008b) is developed by applying MGA transforms to decompose images and extract effective features. This framework (Wang and Bovik 2006) quantifies the errors between the distorted and the reference images by mimicking the error sensitivity function in the HVS. The objective of this framework is to provide IQA results, which have good consistency with subjective perception values. [Figure 2](#) shows the framework for IQA.

3.1 MGA-Based Feature Extraction

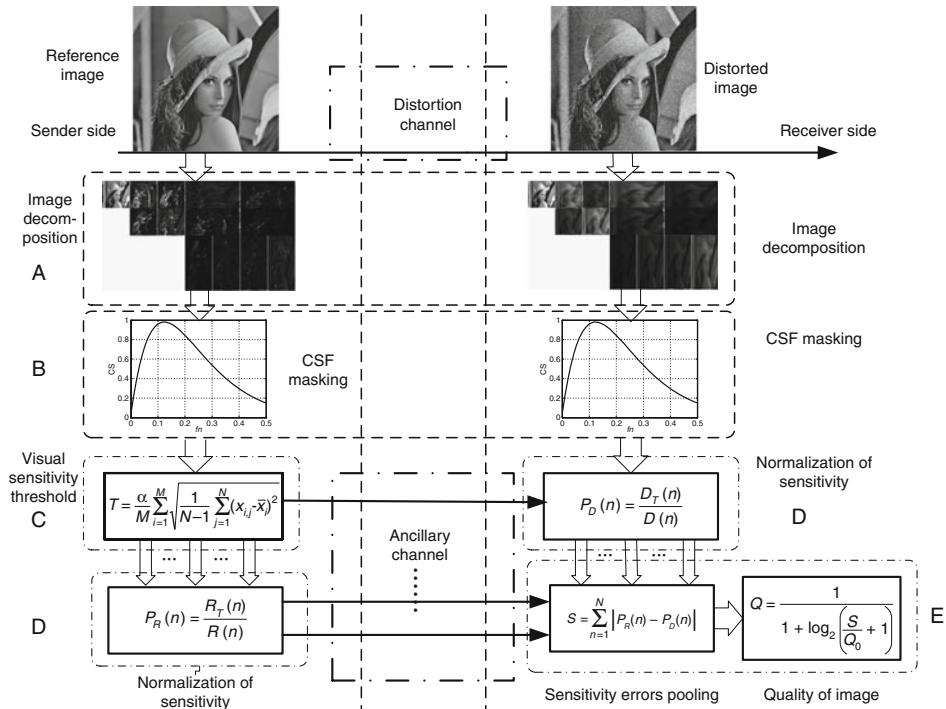
A number of MGA transforms (Do 2001), which are contourlet, WBCT, and HWD, are considered for image decomposition and feature extraction. Moreover, the wavelet is utilized as the baseline for comparison. In this framework, MGA is utilized to decompose images and then extract features to mimic the multichannel structure of HVS. Moreover, there is a wide application of neural computing that can be applied to perform feature extraction for image representation sparsely. Some methods of neural computing are selected to extract features of image data further. Feature extraction (Bishop 1995) is treated as a means for reducing dimensionality of the image data and preserving feature data separability well, for example, feed-forward ANN, *self-organizing feature map* (SOMs) and Hopfield ANN.

3.1.1 Wavelet Transform

The wavelet transform (Mallat 1989) is well known as an approach to analyze signals in both time and frequency domains simultaneously and adaptively. Features are extracted effectively

Fig. 2

MGA-based IQA framework. (Sender side is applied to extract the normalized histogram of the reference image. Receiver side is applied to extract the normalized histogram of the distorted image. Ancillary channel is applied to transmit the extracted normalized histogram.)



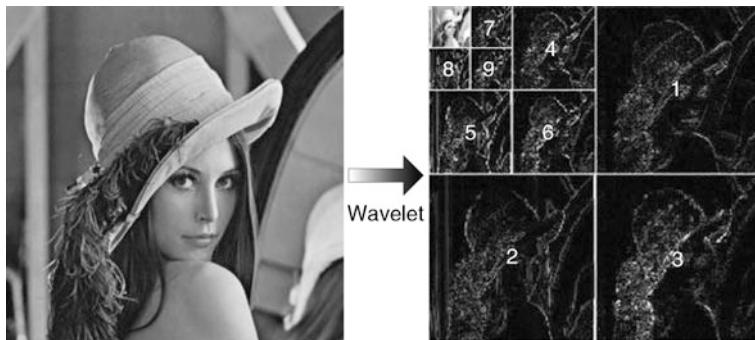
from signals, especially non-stationary signals, by multiscale operation. In this article, three levels of wavelet transform are applied to decompose the image into nine highpass subbands and a lowpass residual subband. Then all the highpass subbands are selected for feature extraction in the proposed framework. **Figure 3** shows decomposition of the image using wavelet transforms and a set of selected subbands (marked with white dashed boxes and white numerals) are used for feature extraction in the proposed framework.

3.1.2 Contourlet Transform

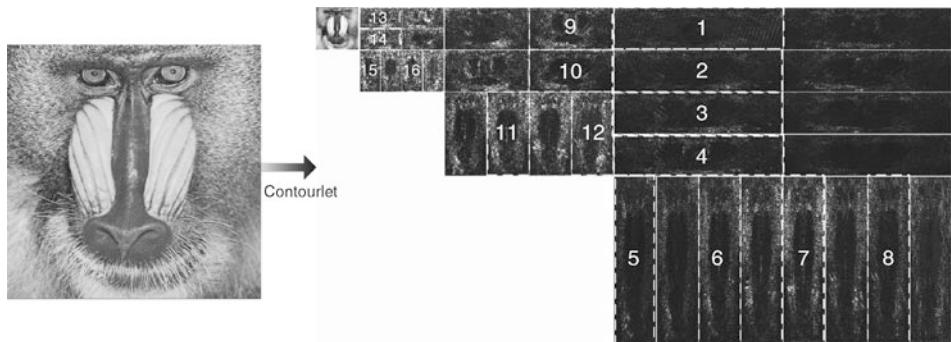
The contourlet transform (Mallat 1989) can capture the intrinsic geometrical structure that is key in visual information. Contourlet consists of two major stages: the multiscale decomposition and the direction decomposition. At the first stage, it uses a *Laplacian pyramid* (LP) (Bishop 1995) to capture the point discontinuities. For the second stage, it uses *directional filter banks* (DFB) (Burt and Adelson 1983) to link point discontinuities into linear structures. Here, every image is decomposed into three pyramidal levels. Based on the characteristics of DFB for decomposition, half of the directional subbands are selected for the feature extraction. The decomposition of images using the contourlet transform and a set of selected subbands

Fig. 3

Wavelet transform-based image decomposition.

**Fig. 4**

Contourlet-transform-based image decomposition.



(marked with white dashed boxes and numerals) are used for feature extraction in the proposed framework, as shown in [Fig. 4](#).

3.1.3 WBCT

WBCT (Eslami and Radha 2004) has a construction similar to the contourlet transform, which consists of two filter bank stages. The first stage is subband decomposition by wavelet transform. The second stage of the WBCT is angular decomposition. In this stage, WBCT employs DFB in the contourlet transform. Here, each image is decomposed into two wavelet levels, and the number of DFB decomposition levels at each highpass of the finest and finer scales is equal to 3. The image is decomposed into 48 high-frequency directional subbands and a lowpass residual subband in all. Like the contourlet transform, half of the subbands at each fine scale are selected to extract the features of the image. The decomposition of the image using the WBCT transform and a set of selected subbands (marked with numerals and gray blocks) are applied for feature extraction in the proposed framework, as demonstrated in [Fig. 5](#).

Fig. 5

Wavelet-based contourlet transform (WBCT)-based image decomposition.

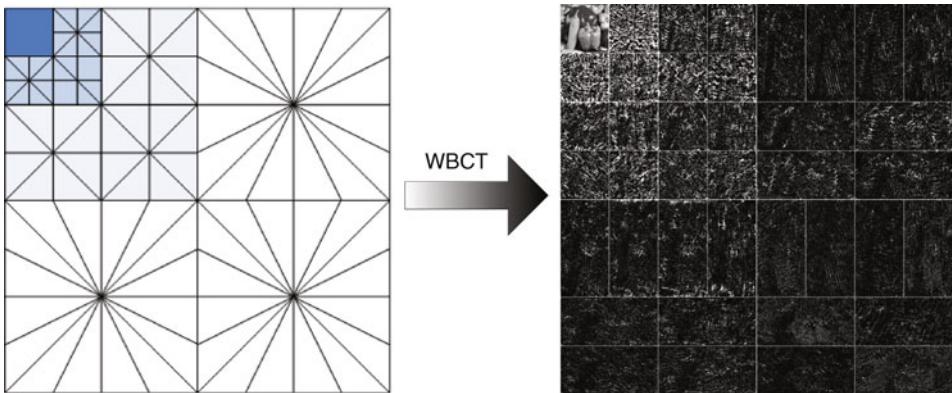
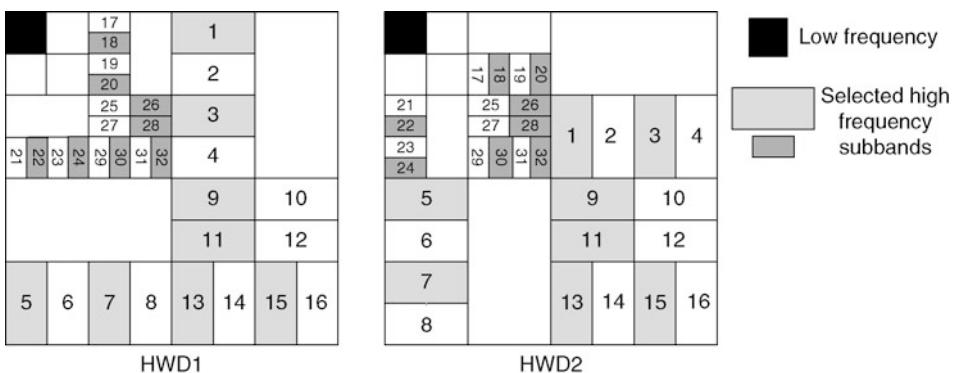


Fig. 6

Hybrid wavelets and directional filter banks (HWD) transform-based image decomposition.



3.1.4 HWD Transform

Like WBCT, the HWD transform (Eslami and Radha 2005) employs wavelets as the multiscale decomposition. The DFB and modified DFB are applied to some of the wavelets subbands. When the HWD transform (HWD1 and HWD2 respectively) is employed in the proposed framework, each image is decomposed into two wavelet levels, and the number of DFB decomposition levels at each highpass of the finest and finer scales is equal to 3. Thus, the image is decomposed into 36 high-frequency directional subbands and a lowpass residual subband in all. Half of the directional subbands at each fine scale (for HWD1 and HWD2, 16 directional subbands in all, respectively) are selected to extract features of the image. The HWD decomposition of the image and a set of selected subbands (marked with numerals and gray blocks) are applied for feature extraction in the proposed frameworks, as demonstrated in  Fig. 6.

3.2 CSF Masking

MGA is introduced to decompose images and then extract features to mimic the multichannel structure of HVS; that is, HVS (Wandell 1995) works similar to a filter bank (containing filters with various frequencies). CSF (Miloslavski and Ho 1998) measures how sensitive we are to the various frequencies of visual stimuli; that is, we are unable to recognize a stimuli pattern if its frequency of visual stimuli is too high. For example, given an image consisting of horizontal black and white stripes, we will perceive it as a gray image if stripes are very thin; otherwise, these stripes can be distinguished. Because coefficients in different frequency subbands have different perceptual importance, it is essential to balance the MGA decomposed coefficients via a weighting scheme, CSF masking. In this framework, the CSF masking coefficients are obtained using the *modulation transfer function* (MTF) (Miloslavski and Ho 1998), that is,

$$H(f) = a(b + cf)e^{(-cf)^d} \quad (1)$$

where $f = f_n \times f_s$, the center frequency of the band, is the radial frequency in cycles/degree of the visual angle subtended, f_n is the normalized spatial frequency with units of cycles/pixel, and f_s is the sampling frequency with units of pixels/degree. According to Bamberger and Smith (1992), a, b, c, and d are 2.6, 0.192, 0.114, and 1.1, respectively. The sampling frequency f_s is defined as (Nadenau et al. 2003)

$$f_s = \frac{2\pi r \cdot \tan(0.5^\circ)}{0.0254} \quad (2)$$

where r is the viewing distance with units of meters and is the resolution power of the display with units of pixels/in. In this framework, r is 0.8 m (about 2–2.5 times the height of the display), the display is 21 in. with a resolution of 1024×768 . According to the Nyquist sampling theorem, if changes from 0 to $f_s/2$, so f_n changes from 0 to 0.5. Because MGA is utilized to decompose an image into three scales from coarse to fine, we have three normalized spatial frequencies, $f_{n1} = 3/32$, $f_{n2} = 3/16$, $f_{n3} = 3/8$. Weighting factors are identical for coefficients in an identical scale.

For example, if the contourlet transform is utilized to decompose an image, a series of contourlet coefficients $c_{i,j}^k$ is obtained, where k denotes the level index (the scale sequence number) of the contourlet transform, i stands for the serial number of the directional subband index at the k th level, and j represents the coefficient index. By using CSF masking, the coefficient $c_{i,j}^k$ is scaled to $x_{i,j}^k = H(f_k) \cdot c_{i,j}^k$.

3.3 JND Threshold

Because HVS is sensitive to coefficients with larger magnitude, it is valuable to preserve visually sensitive coefficients. JND, a research result in psychophysics, is a suitable means for this. It measures the minimum amount by which stimulus intensity must be changed to produce a noticeable variation in the sensory experience. In our framework, MGA is introduced to decompose an image and highpass subbands contain the primary contours and textures information of the image. CSF masking makes coefficients have similar perceptual importance in different frequency subbands, and then JND is calculated to obtain a threshold to remove visually insensitive coefficients. The number of visually sensitive coefficients reflects

the visual quality of the reconstructed images. The lower the JND threshold is, the more coefficients are utilized for image reconstruction and the better the visual quality of the reconstructed image is. Therefore, the normalized histogram reflects the visual quality of an image. Here, the JND threshold is defined as

$$T = \frac{\alpha}{M} \sum_{i=1}^M \sqrt{\frac{1}{N_i - 1} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)^2} \quad (3)$$

where $x_{i,j}$ is the j th coefficient of the i th subband in the finest scale and \bar{x}_i is the mean value of the i th subband coefficients, M is the number of selected subbands in the finest scale, N_i is the number of coefficients of the i th subband, and α is a tuning parameter corresponding to different types of distortion.

Using the JND threshold technique, some parameters are so delicate that they have to be selected using empirical values. Others may be tunable by trial and error or a cross validation process by observing the overall performance of the system. For optimizing the JND threshold technique, some methods of neural computing (Bishop 1995) might be applied to train parameters or variables for autofit thresholds. Neural networks provides a whole family of divergent formalisms to optimize corresponding values for improving the overall performance, for example, the *radial basis function* (RBF) network.

3.4 Normalization of Sensitivity

By using the JND threshold T , we can count the number of visually sensitive coefficients in the n th selected subband and define the value as $C_T(n)$, which means the number of coefficients in the n th selected subband that are larger than T obtained from [Eq. 5](#). The number of coefficients in the n th selected subband is $C(n)$. Therefore, for a given image, we can obtain the normalized histogram with L bins (L subbands are selected) for representation and the n th entry is given by

$$P(n) = \frac{C_T(n)}{C(n)} \quad (4)$$

3.5 Sensitivity Errors Pooling

Based on [Eq. 4](#), the normalized histograms can be obtained for both the reference and the distorted images as $P_R(n)$ and $P_D(n)$, respectively. In this framework, the metrics of the distorted image quality can be defined as

$$Q = \frac{1}{1 + \log_2 \left(\frac{S}{Q_0} + 1 \right)} \quad (5)$$

where $S = \sum_{n=1}^L |P_R(n) - P_D(n)|$ is the city-block distance between $P_R(n)$ and $P_D(n)$, and Q_0 is a constant used to control the scale of the distortion measure. In this framework, we set Q_0 as 0.1. The log function is introduced here to reduce the effects of large S and enlarge the effects of small S , so that we can analyze a large scope of S conveniently. There is no particular reason to choose the city-block distance, which can be replaced by others, for example, the Euclidean norm. This is also the case for the base 2 for the logarithm. The entire function preserves the monotonic property of S .

4 Performance Evaluation

In this section, we compare the performance of the proposed framework based on different MGA transforms with standard IQA methods, that is, PSNR, WNISM, and MSSIM, based on the following experiments: the consistency experiment, the cross-image and cross-distortion experiment, and the rationality experiment. At the beginning of this section, the image database for evaluation is first briefly described.

The LIVE database (Sheikh et al. 2003) is widely used to evaluate the image quality measures, in which 29 high-resolution RGB color images are compressed at a range of quality levels using either JPEG or JPEG2000, producing a total of 175 JPEG images and 169 JPEG2000 images. To remove any nonlinearity due to the subjective rating process and to facilitate comparison of the metrics in a common analysis space, following the procedure given in the *video quality experts group* (VQEG) (VQEG 2000) test, variance-weighted regression analysis is used in a fitting procedure to provide a nonlinear mapping between the objective and subjective MOS. After the nonlinear mapping, the following three metrics are used as evaluation criteria (VQEG 2000): Metric 1 is the Pearson linear *correlation coefficient* (CC) between the objective and MOS after the variance-weighted regression analysis, which provides an evaluation of prediction accuracy; Metric 2 is the Spearman *rank-order correlation coefficient* (ROCC) between the objective and subjective scores, which is considered as a measure of the prediction monotonicity; Metric 3 is the *outlier ratio* (OR), the percentage of the number of predictions outside the range of twice the standard deviation of the predictions after the nonlinear mapping, which is a measure of the prediction consistency. In addition, the *mean absolute error* (MAE) and the *root mean square error* (RMSE) of the fitting procedure are calculated after the nonlinear mapping.

In the following parts, we compare the performance of different IQA methods based on the aforementioned image database and metrics.

4.1 The Consistency Experiment

In this subsection, the performance of the proposed IQA framework will be compared with PSNR (Avcibas et al. 2002), WNISM (Wang and Bovik 2006), and the well-known full reference assessment metric, MSSIM (Wang and Bovik 2006). The evaluation results for all IQA methods being compared are given as benchmarks in [Table 1](#). [Table 2](#) shows the

Table 2

The performance of the proposed IQA framework with different MGA transforms on the LIVE database

Metric	JPEG					JPEG2000						
	CC	ROCC	OR	MAE	RMSE	CC	ROCC	OR	MAE	RMSE		
Wavelet	1	0.9587	0.9391	0.0914	5.1012	6.5812	6	0.9487	0.9333	0.0473	5.3486	6.8453
Contourlet	3	0.9493	0.9309	0.1029	5.3177	6.6941	2	0.9451	0.9273	0.0710	5.5818	6.9616
WBCT	3	0.9728	0.9527	0.0457	4.1162	5.3750	6	0.9565	0.9390	0.0414	4.9891	6.4718
HWD1	2	0.9704	0.9526	0.0514	4.3305	5.5646	5	0.9540	0.9333	0.0493	5.2857	6.7972
HWD2	3	0.9728	0.9543	0.0400	4.1135	5.3206	5	0.9540	0.9362	0.0473	5.1357	6.6312

evaluation results of the proposed IQA framework with different MGA transforms, e.g., wavelet, contourlet, WBCT, HWD1, and HWD2. ➤ Figures 7 and ➤ 8 present the scatter plots of MOS versus the predicted score using objective metrics after the nonlinear mapping.

➤ Table 2 shows that the results of the MGA transforms using the proposed framework provide a higher effectiveness for IQA for JPEG and JPEG2000 images, respectively.

■ Fig. 7

Scatter plots of mean opinion score (MOS) versus different image quality assessment (IQA) methods for JPEG and JPEG2000 images.

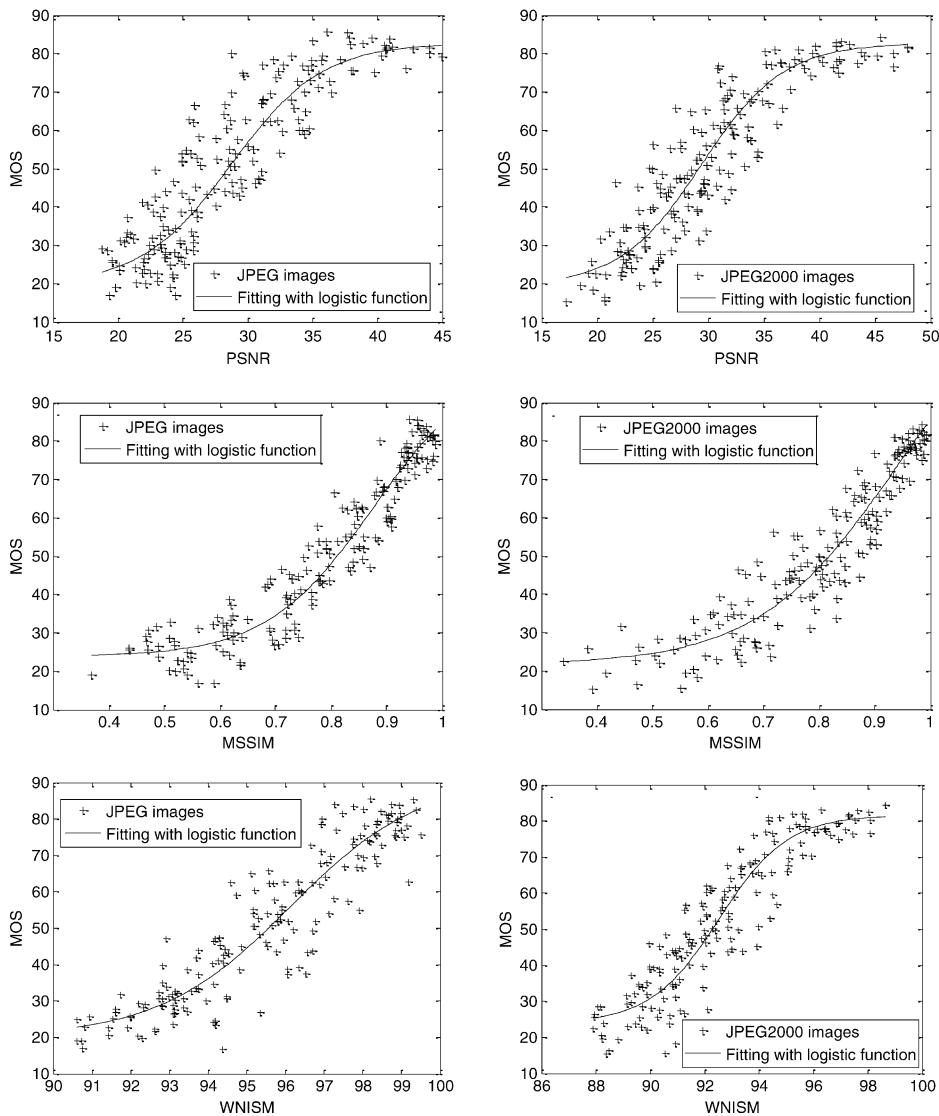
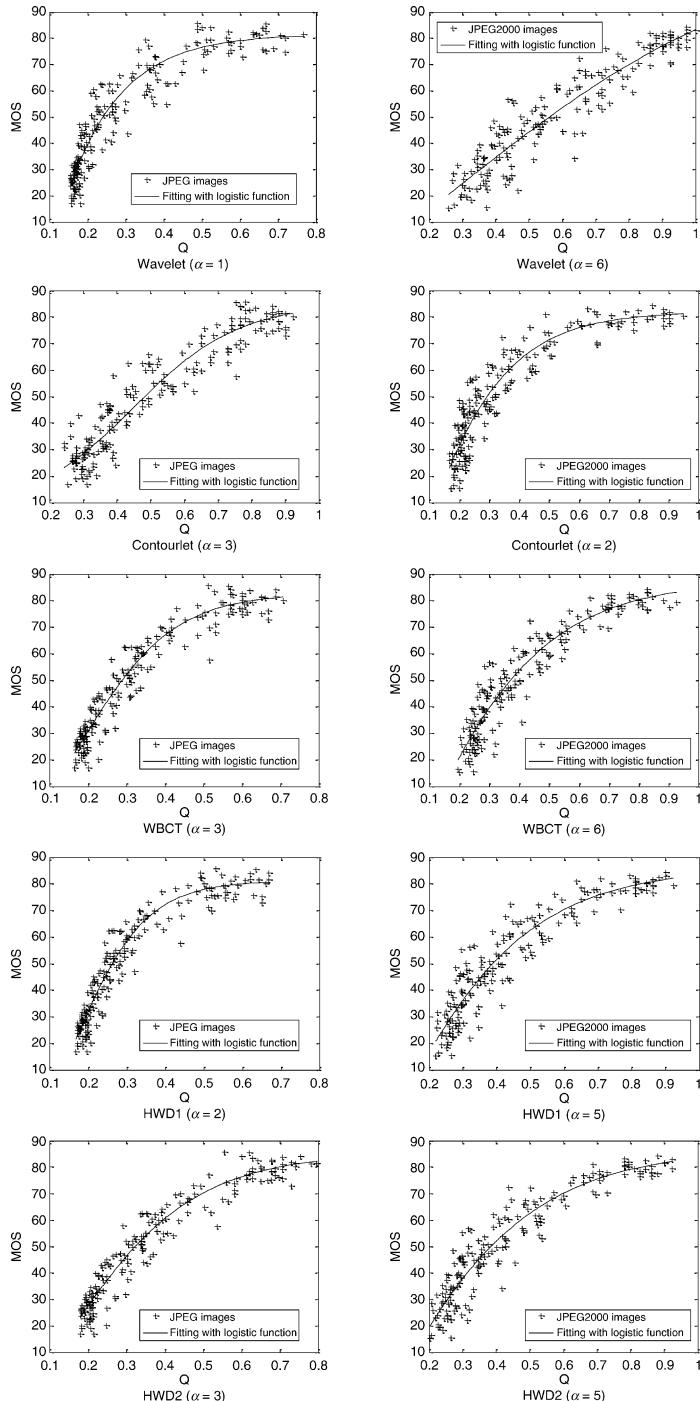


Fig. 8

Trend plots of Lena with different distortions using the proposed framework with contourlet.



Comparing [Table 2](#) with [Table 1](#), one can observe that the metric obtained by the proposed framework provides much better performance than WNISM, including better prediction accuracy (higher CC), better prediction monotonicity (higher ROCC), and better prediction consistency (lower OR, MAE, and RMSE). Particularly when HWD is employed in the proposed framework, better performance will be achieved than with the MSSIM index. The only tuning parameter α in the proposed framework responds to different distortions.

Since the key stage of IQA is how to represent images effectively and efficiently, it is necessary to investigate different transforms. The wavelet is localized in the spatial domain and the frequency domain and can extract local information with high efficiency, so it optimally approximates a target function with 1D singularity. However, the wavelet cannot achieve the sparse representation of edges even if it captures the point singularity effectively. In order to represent edges in an image sparsely, the contourlet approach analyzes the scale and the orientation respectively and reduces the redundancy by approximating images with line-segments similar basis ultimately. To further reduce the redundancy, HWD is proposed to represent images sparsely and precisely. Redundancy is harmful for IQA. If an image is represented by a large number of redundant components, small visual changes will affect the quality of the image slightly. Both transforms are nonredundant, multiscale, and multi-orientation for image decomposition. [Table 3](#) shows the proposed IQA framework with HWD works much better than previous standards.

4.2 The Rationality Experiment

To verify the rationality of the proposed framework, contourlet and WBCT are chosen to test the Lena image with different distortions: blurring (with smoothing window of W^*W), additive Gaussian noise (mean = 0, variance = V), impulsive salt-pepper noise (density = D), and JPEG compression (compression rate = R).

[Figures 9](#) and [10](#) (all images are 8 bits/pixel and resized from 512×512 to 128×128 for visualization) show the relationships between the Lena image with different distortions and the IQA methods prediction trend for the corresponding image.

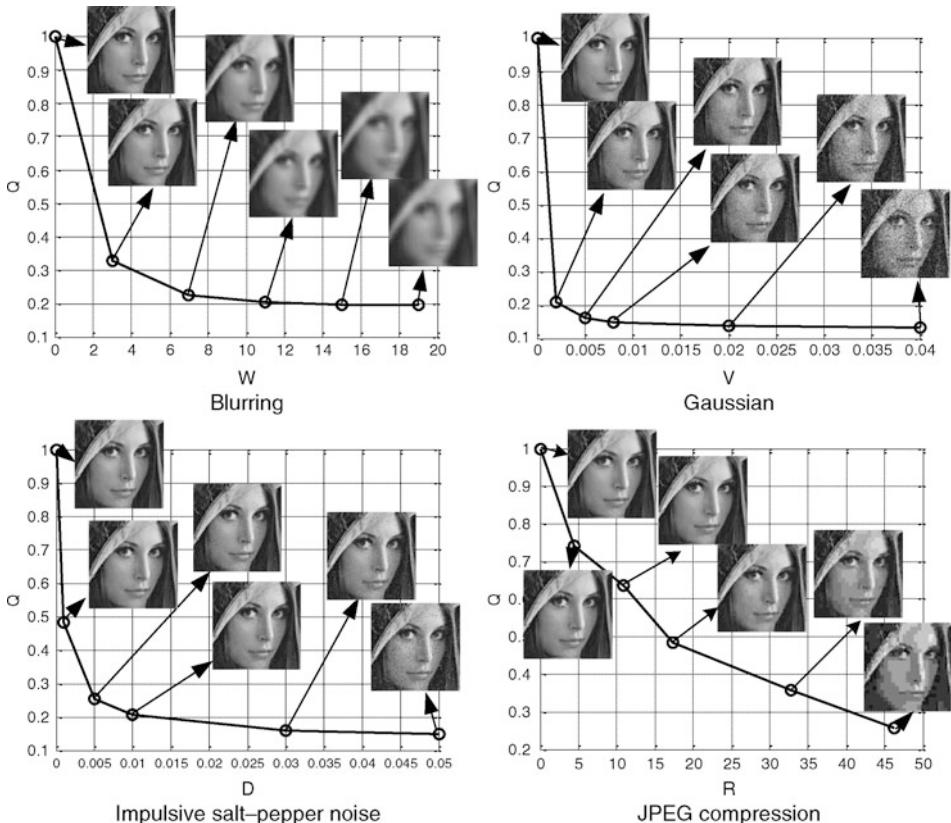
Table 3

The value of different IQA metrics for images in [Fig. 11](#)

Metric	(b)	(c)	(d)
PSNR	24.8022	24.8013	24.8041
MSSIM	0.9895	0.9458	0.6709
Wavelet	1.0000	0.3208	0.2006
Contourlet	1.0000	0.2960	0.2423
WBCT	1.0000	0.2345	0.1929
HWD1	1.0000	0.2740	0.2166
HWD2	1.0000	0.2704	0.2094

Fig. 9

Trend plots of Lena with different distortions using the proposed framework with contourlet transform (all images are 8 bits/pixel and resized from 512×512 to 128×128 for visibility).

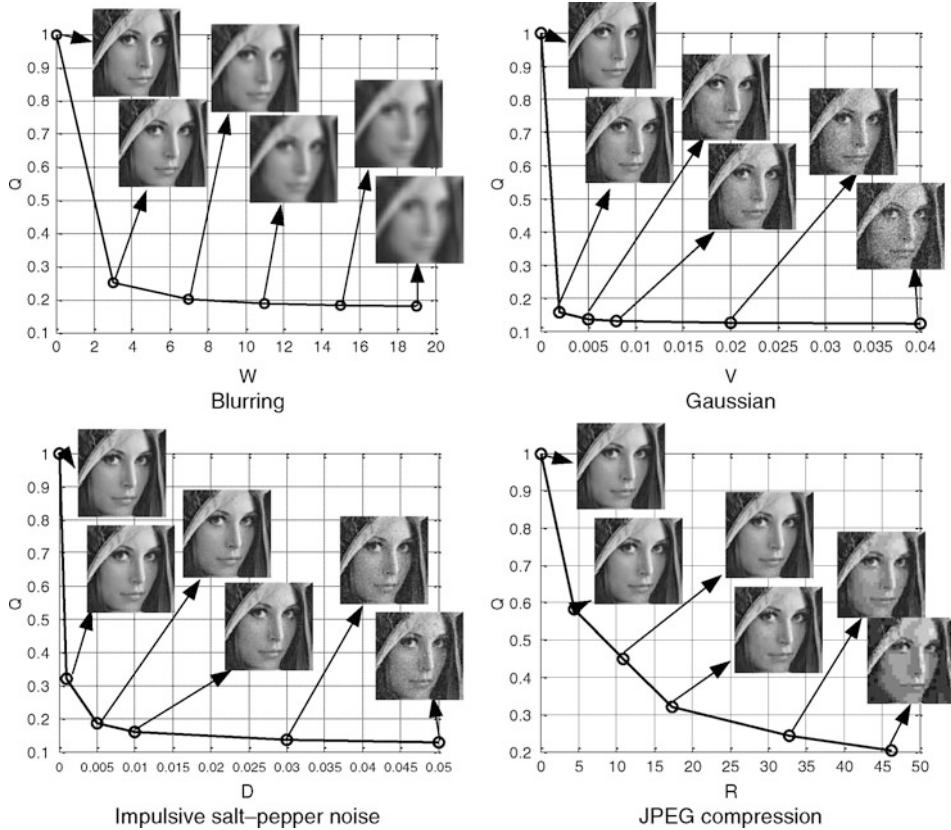


For JPEG compression, we find that Q for IQA drops with increasing intensity of different distortions, which is consistent with the tendency of the decreasing image quality. That is the proposed IQA framework works well for the JPEG distortion. The coding scheme of JPEG (JPEG2000) is based on the discrete cosine transform (discrete wavelet transform). In the JPEG (JPEG2000) coding stage, the lowpass subband is compressed in a high compression rate, and highpass subbands are compressed in a low compression rate to achieve a good compression rate for the whole image while preserving the visual quality of the image. This procedure is similar to the IQA framework; that is, information in the lowpass subband is not considered because most information in the lowpass subband is preserved; and the output value Q is obtained from highpass subbands only because low compression rates are utilized on them and the quality of the reconstructed image is strongly relevant to highpass subbands (the defined JND threshold). Therefore, the proposed scheme adapts well for JPEG and JPEG2000 distortions.

For blurring, additive Gaussian noise distortion, and impulsive salt-pepper noise, Q for IQA drops sharply initially and then slowly because MGA transforms cannot explicitly extract

Fig. 10

Trend plots of Lena with different distortions using the proposed framework with WBCT transform (all images are 8 bits/pixel and resized from 512×512 to 128×128 for visibility).



effective information from images with these distortions. However, based on these figures, Q can still reflect the quality of distorted images with blurring, additive Gaussian noise distortion, and impulsive salt-pepper noise, although the performance is not good.

4.3 Sensitivity Test Experiment of the Proposed Framework

Currently, MSE and PSNR are the most commonly used objective quality metrics for images. However, they do not correlate well with the perceived quality measurement. **Figure 11** shows three degraded “Lena” images with different types of distortion but with the same PSNR; however, their perceived qualities are obviously varied. As shown in **Table 3**, the proposed framework with different MGA transforms can distinguish them very well. It is noted that Q is insensitive to the changes of intensity, in other words, a slight change of gray value at the image level will not affect the image quality when Q equals 1, so it is consistent with the HVS character that only when the distorted image and the reference image have the same variance of light can the two images have the same visual quality.

Fig. 11

Lena image with the same PSNR but different perceived quality. (a) The reference image; (b) mean shift; (c) contrast stretching; (d) JPEG compression.



5 Conclusion

This chapter has described a RR IQA framework by incorporating merits of MGA and HVS. In this framework, sparse image representation based on MGA is used to mimic the multichannel structure of HVS, and then CSF is used to balance the magnitude of the coefficients obtained by MGA mimic nonlinearities of HVS, and JND is used to produce a noticeable variation in sensory experience. The quality of a distorted image is measured by comparing the normalized histogram of the distorted image and that of the reference image. Thorough empirical studies show that the novel framework with a suitable image decomposition method performs better than the conventional standard RR IQA method. Since RR methods require full limited information of the reference image, it could be a serious impediment for many applications. It is essential to develop NR image quality metrics that blindly estimate the quality of images. Recently, tensor-based approaches (Tao et al. 2006, 2007a, b) have been demonstrated to be effective for image representation in classification problems, so it is valuable to introduce them for image quality.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (60771068, 60702061, 60832005), the Ph.D. Programs Foundation of the Ministry of Education of China (No. 20090203110002), the Natural Science Basic Research Plan in the Shaanxi Province of

China (2009JM8004), the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) in China, and the National Laboratory of Automatic Target Recognition, Shenzhen University, China.

References

- Avcibas I, Sankur B, Sayood K (2002) Statistical evaluation of image quality measures. *J Electron Imaging* 11(2):206–213
- Bamberger RH, Smith JT (1992) A filter bank for the directional decomposition of images: theory and design. *IEEE Trans Signal Process* 40(4):882–893
- Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, London
- Burt PJ, Adelson EH (1983) The Laplacian pyramid as a compact image code. *IEEE Trans Commun* 31(4):532–540
- Callet PL, Christian VG, Barba D (2006) A convolutional neural network approach for objective video quality assessment. *IEEE Trans Neural Netw* 17(5): 1316–1327
- Cands EJ (1998) Ridgelets: theory and applications. PhD thesis, Department of statistics, Stanford University
- Cands EJ, Donoho DL (2000) Curvelets – a surprising effective nonadaptive representation for objects with edges. *Curves and surfaces*, Vanderbilt University Press, Nashville, TN, pp 105–120
- Chandrasekaran V, Palaniswami M, Caelli TM (1996) Range image segmentation by dynamic neural network architecture. *Pattern Recogn* 29(2):315–329
- Daly S (1993) The visible difference predictor: an algorithm for the assessment of image fidelity. In: Watson AB (ed) *Digital images and human vision*, The MIT Press, Cambridge, MA, pp 179–206
- Damera-Venkata N, Thomas DK, Wilson SG, Brian LE, Bovik AC (2000) Image quality assessment based on a degradation model. *IEEE Trans Image Process* 9(4): 636–650
- Do MN (2001) Directional multiresolution image representations. PhD thesis, École Polytechnique Fédérale de Lausanne
- Do MN, Vetterli M (2005) The contourlet transform: an efficient directional multiresolution image representation. *IEEE Trans Image Process* 14(12):2091–2106
- Eskicioglu AM, Fisher PS (1995) Image quality measures and their performance. *IEEE Trans Commun* 43 (12):2959–2965
- Eslami R, Radha H (2004) Wavelet-based contourlet transform and its application to image coding. In: *Proceedings of the IEEE international conference on image processing*, Singapore, pp 3189–3192
- Eslami R, Radha H (2005) New image transforms using hybrid wavelets and directional filter banks: analysis and design. In: *Proceedings of the IEEE international conference on image processing*, Singapore, pp 11–14
- Gao XB, Lu W, Li XL, Tao DC (2008a) Wavelet-based contourlet in quality evaluation of digital images. *Neurocomputing* 72(12):378–385
- Gao XB, Lu W, Li XL, Tao DC (2008b) Image quality assessment based on multiscale geometric analysis. Accepted to *IEEE Trans Image Process*
- Gastaldo P, Rovetta S, Zunino R (2002) Objective quality assessment of MPEG-2 video streams by using CBP neural networks. *IEEE Trans Neural Netw* 13 (4):939–947
- Gastaldo P, Zunino R (2004) No-reference quality assessment of JPEG images by using CBP neural networks. In: *Proceedings of the IEEE international symposium on circuits and systems*, Vancouver, Canada, pp 772–775
- Karunasekera SA, Kingsbury NG (1995) A distortion measure for blocking artifacts in images based on human visual sensitivity. *IEEE Trans Image Process* 4(6):713–724
- Lampinen J, Oja E (1998) Neural network systems, techniques and applications. *Image processing and pattern recognition*, Academic Press, New York
- Li X (2002) Blind image quality assessment. In: *Proceedings of the IEEE international conference on image processing*, Singapore, pp 449–452
- Li XL, Tao DC, Gao XB, Lu W (2009) A natural image quality evaluation metric based on HWD. *Signal Process* 89(4):548–555
- Lu W, Gao XB, Tao DC, Li XL (2008) A wavelet-based image quality assessment method. *Int J Wavelets Multiresolut Inf Process* 6(4):541–551
- Lubin J (1995) A visual discrimination mode for image system design and evaluation. In: Peli E (ed) *Visual models for target detection and recognition*, Singapore, World Scientific, pp 207–220
- Mallat S (1989) A theory for multiresolution decomposition: the wavelet representation. *IEEE Trans Pattern Anal Mach Intell* 11(7):674–693
- Mannos JL, Sakrison DJ (1974) The effect of visual fidelity criterion on the encoding of images. *IEEE Trans Inf Theory* 20(2):525–536
- Masry M, Hemami SS, Sermadevi Y (2006) A scalable wavelet-based video distortion metric and applications. *IEEE Trans Circuits Syst Video Technol* 16(2): 260–273

- Miloslavski M, Ho YS (1998) Zerotree wavelet image coding based on the human visual system model. In: Proceedings of the IEEE Asia-Pacific conference on circuits and systems, pp 57–60
- Miyahara M, Kotani K, Algazi RV (1998) Objective picture quality scale (PQS) for image coding. *IEEE Trans Commun* 46(9):1215–1225
- Nadenau MJ, Reichel J, Kunt M (2003) Wavelet-based color image compression: exploiting the contrast sensitivity. *IEEE Trans Image Process* 12(1):58–70
- Oja E (1982) A simplified neuron model as a principal component analyzer. *J Math Biol* 15(3):267–273
- Pennec EL, Mallat S (2000) Image compression with geometrical wavelets. In: Proceedings of the IEEE international conference on image processing, Vancouver, Canada, pp 661–664
- Petersen E, Ridder DD, Handels H (2002) Image processing with neural networks – a review. *Pattern Recogn* 35:2279C2301
- Rec. ITU-R Methodology for the subjective assessment of the quality of television pictures, Recommendation ITU-R Rec. BT. 500-11
- Romberg JK (2003) Multiscale geometric image processing. PhD thesis, Rice University
- Sheikh HR, Sabir MF, Bovik AC (2006) A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Trans Image Process* 15(11):3440–3451
- Sheikh HR, Wang Z, Cormack L, Bovik AC (2003) LIVE image quality assessment database. <http://live.ece.utexas.edu/research/quality>
- Tao DC, Li XL, Wu XD, Maybank SJ (2006) Human carrying status in visual surveillance. In: Proceedings of the IEEE international conference on computer vision and pattern recognition, Hong Kong, China, pp 1670–1677
- Tao DC, Li XL, Wu XD, Maybank SJ (2007a) General tensor discriminant analysis and Gabor features for gait recognition. *IEEE Trans Pattern Anal Mach Intell* 29(10):1700–1715
- Tao DC, Li XL, Wu XD, Maybank SJ (2007b) Supervised tensor learning. *Knowl Inf Syst* (Springer) 13(1): 1–42
- VQEG (2000a) Final report from the video quality experts group on the validation of objective models of video quality assessment, phase II VQEG, 2003. <http://www.vqeg.org>
- VQEG (2000b) Final report from the video quality experts group on the validation of objective models of video quality assessment. <http://www.vqeg.org>
- Wandell BA (1995) Foundations of vision, 1st edn. Sinauer, Sunderland, MA
- Wang Z, Bovik AC (2006) Modern image quality assessment. Morgan and Claypool, New York
- Wang YM, Wahl FM (1997) Vector-entropy optimization based neural-network approach to image reconstruction from projections. *IEEE Trans Neural Netw* 8(5):1008–1014
- Watson AB (1993) DCT quantization matrices visually optimized for individual images. In: Proceedings of the SPIE human vision, visual processing and digital display IV. Bellingham, WA, 1913(14):202–216
- Wolf S, Pinson MH (2005) Low bandwidth reduced reference video quality monitoring system. In: First international workshop on video processing and quality metrics for consumer electronics. Scottsdale, AZ

12 Nonlinear Process Modelling and Control Using Neurofuzzy Networks

Jie Zhang

School of Chemical Engineering and Advanced Materials, Newcastle University, Newcastle upon Tyne, UK

jie.zhang@newcastle.ac.uk

1	<i>Introduction</i>	402
2	<i>Neurofuzzy Networks for Nonlinear Process Modeling</i>	404
3	<i>Modeling of a Neutralization Process Using a Recurrent Neurofuzzy Network</i>	411
4	<i>Nonlinear Model-Based Control Through Combination of Local Linear Controllers Based on Neurofuzzy Network Models</i>	418
5	<i>Modeling and Optimal Control of Batch Processes Using Neurofuzzy Networks</i>	423
6	<i>Conclusions</i>	432

Abstract

This chapter presents neurofuzzy networks for nonlinear process modeling and control. The neurofuzzy network uses local linear models to model a nonlinear process and the local linear models are combined using center of gravity (COG) defuzzification. In order to be able to provide accurate long-range predictions, a recurrent neurofuzzy network structure is developed. An advantage of neurofuzzy network models is that they are easy to interpret. Insight about the process characteristics at different operating regions can be easily obtained from the neurofuzzy network parameters. Based on the neurofuzzy network model, nonlinear model predictive controllers can be developed as a nonlinear combination of several local linear model predictive controllers that have analytical solutions. Applications to the modeling and control of a neutralization process and a fed-batch process demonstrate that the proposed recurrent neurofuzzy network is very effective in the modeling and control of nonlinear processes.

1 Introduction

Advanced process control, optimization, and process monitoring require accurate process models. Process models can be broadly divided into two categories: first principles models and empirical models. First principles models are developed based upon process knowledge and, hence, they are generally reliable. However, the development of first principles models is usually time consuming and effort demanding, especially for complex processes. To overcome this difficulty, empirical models based upon process operational data should be utilized.

Neural networks have been shown to possess good function approximation capability (Cybenko 1989; Girosi and Poggio 1990; Park and Sandberg 1991) and have been applied to process modeling by many researchers (Bhat and McAvoy 1990; Bulsari 1995; Morris et al. 1994; Tian et al. 2002; Zhang 2004). A neural network can learn the underlying process model from a set of process operation data. Conventional network modeling typically results in a “black box” model. One potential limitation of conventional neural network models is that they can be difficult to interpret and can lack robustness when applied to unseen data.

One approach to improve model robustness and open up the “black box” models is through the combined use of both process knowledge and process operation data. Process knowledge can, for example, be used to decompose the process operation into a number of local operating regions such that, within each region, a reduced order linear model can be used to approximate the local behavior of the process. In fact, a nonlinear process can always be locally linearized around a particular operating point and the locally linearized model is valid within a region around that operating point. Fuzzy sets provide an appropriate means for defining operating regions since the definition of local operating regions is often vague in nature and there usually exists overlapping among different regions. This leads to the fuzzy modeling approach (Yager and Filev 1994).

One fuzzy modeling approach was developed by Takagi and Sugeno (1985). In this approach, each model input is assigned several fuzzy sets with the corresponding membership functions being defined. Through logical combinations of these fuzzy inputs, the model input space is partitioned into several fuzzy regions. A local linear model is used within each region and the global model output is obtained through the center of gravity (COG) defuzzification which is essentially the interpolation of local model outputs. This modeling approach is very powerful in that it decomposes a complex system into several less complex subsystems. Based on a

similar principle, Johansen and Foss (1993) proposed an approach to construct NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous inputs) models using local ARMAX (AutoRegressive Moving Average with eXogenous inputs) models. In their approach, a nonlinear system is decomposed into several operating regimes and, within each regime, a local ARMAX model is developed. A model validate function is defined for each local model and the global NARMAX model is then obtained by interpolating these local ARMAX models.

Fuzzy models can be implemented by using neurofuzzy networks. Neurofuzzy network representations have emerged as a powerful approach to the solution of many engineering problems (Blanco et al. 2001b; Brown and Harris 1994; Harris et al. 1996; Horikawa et al. 1992; Jang 1992; Jang and Sun 1995; Jang et al. 1997; Nie and Linkens 1993; Omlin et al. 1998; Wang 1994; Zhang and Morris 1994; Zhang et al. 1998; Zhang and Morris 1999; Zhang 2005, 2006). Jang (Jang 1992; Jang and Sun 1995; Jang et al. 1997) proposed the ANFIS (adaptive-network-based fuzzy inference system) architecture to represent fuzzy models. Through back propagation training, ANFIS is adapted to refine, or derive, the fuzzy if-then rules using system input-output data. Fuzzy reasoning is capable of handling imprecise and uncertain information while neural networks can be identified using real plant data. Neurofuzzy networks combine the advantages of both fuzzy reasoning and neural networks. Process knowledge can also be embedded into neurofuzzy networks in terms of both fuzzy membership partitions and as reduced order local models. Zhang and Morris propose two types of feed forward neurofuzzy networks for nonlinear processes modeling (Zhang and Morris 1995). Most of the reported neurofuzzy network models are, however, one-step-ahead prediction models in that the current process output is used as a network input to predict the process output at the next sampling time step.

In some process-control applications, such as long-range predictive control and batch process optimal control, models capable of providing accurate multistep-ahead prediction are more appropriate. It is well known that nonlinear multistep-ahead prediction models can be built using dynamic neural networks. Good examples are globally recurrent neural networks (e.g., Su et al. 1992; Werbos 1990), Elman networks (Elman 1990; Scott and Ray 1993), dynamic feed forward networks with filters (Morris et al. 1994), and locally recurrent networks (Frasconi et al. 1992; Tsoi and Back 1994; Zhang et al. 1998). In globally recurrent networks, the network outputs are fed back to the network inputs through time delay units. Su et al. (1992) showed that a feed forward neural network trained for one-step-ahead predictions has difficulties in providing acceptable long-term predictions, unlike a globally recurrent network. In an Elman network, the hidden neuron outputs at the previous time step are fed back to all the hidden neurons. Such a topology is similar to a nonlinear state space representation in dynamic systems. Scott and Ray (1993) demonstrated the performance of an Elman network for nonlinear process modeling. Filter networks incorporate filters of the form $N(s)/D(s)$ into the network interconnections (Morris et al. 1994). In locally recurrent networks, the output of a hidden neuron is fed back to its input through one or several time delay units. Zhang et al. (1998) proposed a sequential orthogonal training strategy which allows hidden neurons to be gradually added to avoid an unnecessarily large network structure.

A type of recurrent neurofuzzy network is proposed by Zhang and Morris (1999) and it allows the construction of a “global” nonlinear multistep-ahead prediction model from the fuzzy conjunction of a number of “local” dynamic models. In this recurrent neurofuzzy network, the network output is fed back to the network input through one or more time delay units. This particular structure ensures that predictions from a recurrent neurofuzzy network are multistep-ahead or long-range predictions. Both process knowledge and process input–output

data are used to build multistep-ahead prediction models. Process knowledge is used to initially partition the process nonlinear characteristics into several local operating regions and to aid the initialization of the corresponding network weights. Process input–output data are then used to train the network. Membership functions of the local regimes are identified and local models discovered through network training. In the training of a recurrent neurofuzzy network, the training objective is to minimize the multistep-ahead prediction errors. Therefore, a successfully trained recurrent neurofuzzy network is able to provide good long-range predictions.

A recurrent neurofuzzy network model can be used to develop a novel type of nonlinear model based on a long-range predictive controller. Based upon the local linear models contained in a recurrent neurofuzzy network, local linear model-based predictive controllers can be developed. These local controllers can be combined through COG defuzzification to form a global nonlinear model-based long-range predictive controller. The advantage of this approach is that analytical solutions usually exist for the local model predictive controllers (except for the cases where nonlinear constraints are present), hence avoiding computation that requires numerical optimization procedures and the uncertainty in converging to the global minimum which are typically seen in previously reported nonlinear model-based predictive control strategies. When nonlinear constraints are present, the difficulty in computation and analysis can also be eased through the combination of local controllers. For instance, a nonlinear constraint may never be active in a particular operating region and can therefore be relaxed in that region. Within a particular operating region, a nonlinear constraint may be approximated by a linear constraint. Furthermore, control actions calculated from local linear models in incremental form contain integral actions which can naturally eliminate static control offsets.

This chapter is structured as follows. [Section 2](#) presents neurofuzzy networks and recurrent neurofuzzy networks for nonlinear process modeling. Applications of the proposed recurrent neurofuzzy networks to the modeling of pH dynamics in a continuous stirred tank reactor (CSTR) are presented in [Sect. 3](#). A long-range model-based predictive control technique using recurrent neurofuzzy network models is discussed in [Sect. 4](#). [Section 5](#) presents the modeling and multi-objective optimization control of a fed-batch process using recurrent neurofuzzy networks. Finally, [Sect. 6](#) draws some concluding remarks.

2 Neurofuzzy Networks for Nonlinear Process Modeling

2.1 Fuzzy Models

The global operation of a nonlinear process is divided into several local operating regions. Within each local region, R_i , a reduced order linear model in ARX (AutoRegressive with eXogenous inputs) form is used to represent the process behavior. Fuzzy sets are used to define process operating regions such that the fuzzy dynamic model of a nonlinear process can be described in the following way:

R_i : IF operating condition i

THEN

$$\hat{y}_i(t) = \sum_{j=1}^{no} a_{ij} y(t-j) + \sum_{j=1}^{ni} b_{ij} u(t-j) \quad (1)$$

$$(i = 1, 2, \dots, nr)$$

The final model output is obtained through COG defuzzification as:

$$\hat{y}(t) = \frac{\sum_{i=1}^{nr} \mu_i \hat{y}_i(t)}{\sum_{i=1}^{nr} \mu_i} \quad (2)$$

In the above model, y is the process output, u is the process input, \hat{y}_i is the prediction of the process output in the i th operating region, nr is the number of fuzzy operating regions, ni and no are the time lags in u and y , respectively, μ_i is the membership function for the i th model, a_{ij} and b_{ij} are the ARX model parameters, and t represents discrete time.

The above model represents a one-step-ahead prediction model in that the process output at time $t - 1$, $y(t - 1)$ is used to predict the process output at time t , $y(t)$. A long-term prediction fuzzy dynamic model of a nonlinear process can be described as follows:

R_i : IF operating condition i

THEN

$$\hat{y}_i(t) = \sum_{j=1}^{no} a_{ij} \hat{y}(t-j) + \sum_{j=1}^{ni} b_{ij} u(t-j) \quad (3)$$

$$(i = 1, 2, \dots, nr)$$

The final model output is obtained through COG defuzzification as indicated in [Eq. 2](#). In the above model, \hat{y}_i is the prediction of the process output in the i th operating region. The above model is a multistep-ahead prediction model since it uses previous model outputs, $\hat{y}(t-j)$, instead of previous process outputs, $y(t-j)$, to predict the present process output.

Operating regions of a process can usually be defined by one or several process variables. A number of fuzzy sets, such as “low,” “medium,” and “high,” can be defined for each of these process variables. An operating region can then be defined through logical combinations of the fuzzy sets of those variables. Suppose that x and y are the process variables used to define the process-operating regions and they are assigned the fuzzy sets: “low,” “medium,” and “high.” The i th operating region can be defined, for example, as

x is high AND y is medium

The membership function for this operating region can be obtained in a number of ways. One approach to calculate it is

$$\mu_i = \min(\mu_h(x), \mu_m(y)) \quad (4)$$

and another approach is

$$\mu_i = \mu_h(x)\mu_m(y) \quad (5)$$

In the above equations, μ_i is the membership function of the i th operating region, $\mu_h(x)$ is the membership function of x being “high,” and $\mu_m(y)$ is the membership function of y being “medium.” The second approach allows the calculation of model error gradients which are required in gradient-based network training. For the convenience of using gradient-based network training algorithms, the second approach is adopted here. However, if gradient free training algorithms, such as those based on genetic algorithms, are used, then either approach can be adopted.

The local linear model used in this work can be transformed into incremental form in terms of the model input variables which are used as the manipulated variables in control applications. Consider the following discrete time local linear model:

$$\begin{aligned} y(t) = & a_1 y(t-1) + a_2 y(t-2) + \dots + a_{no} y(t-no) + b_1 u(t-1) + b_2 u(t-2) \\ & + \dots + b_{ni} u(t-ni) \end{aligned} \quad (6)$$

The prediction of y at time $t-1$ is:

$$\begin{aligned} y(t-1) = & a_1 y(t-2) + a_2 y(t-3) + \dots + a_{no} y(t-no-1) \\ & + b_1 u(t-2) + b_2 u(t-3) + \dots + b_{ni} u(t-ni-1) \end{aligned} \quad (7)$$

Subtracting $\textcircled{7}$ Eq. 7 from $\textcircled{6}$ Eq. 6 and rearranging gives the model in incremental form:

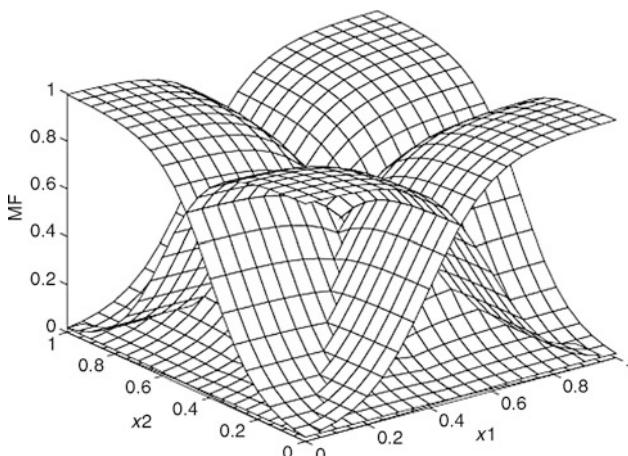
$$\begin{aligned} y(t) = & (a_1 + 1)y(t-1) + (a_2 - a_1)y(t-2) + \dots + (a_{no} - a_{no-1})y(t-no) \\ & - a_{no}y(t-no-1) + b_1\Delta u(t-1) + b_2\Delta u(t-2) + \dots + b_{ni}\Delta u(t-ni) \end{aligned} \quad (8)$$

where $\Delta u(t-1) = u(t-1) - u(t-2)$. Incremental models are particularly useful for controller development since control actions calculated from them naturally contain integral actions which eliminate static control offsets.

$\textcircled{8}$ Figure 1 illustrates the idea of how membership functions span across the operating regions. Process operating regions are determined by two process variables, x_1 and x_2 , each being partitioned into two fuzzy sets: “small” and “large.” There exist four operating regions and they are: (1) x_1 is small and x_2 is small; (2) x_1 is small and x_2 is large; (3) x_1 is large and x_2 is small; and (4) x_1 is large and x_2 is large. $\textcircled{8}$ Figure 1 shows the membership functions of the four operating regions. It can be seen that there exists overlapping among operating regions and the transition from one operating region to another can be achieved in a smooth fashion. The fuzzy approach in representing process-operating regions can handle uncertain and imprecise information.

Fig. 1

Membership functions for operating regions.



Fuzzy modeling is particularly suitable for processes which can be divided into several operating regions. An example would be some batch processes which have different dynamics at different batch stages. Another example would be processes producing different grades of products. These processes can have different dynamics associated with the production of different grades of products. Inside an operating region, the process dynamics could be represented by a fairly simple model, such as a reduced order linear model. In different operating regions, the local models may be very different from each other. By means of fuzzy modeling, a complex global model can be developed in terms of several fairly simple local models.

2.2 Neurofuzzy Networks

The above fuzzy model can be represented by the neurofuzzy networks shown in [Fig. 2](#) and [Fig. 3](#). [Figure 2](#) shows a feed forward neurofuzzy network (Zhang and Morris 1995) while [Fig. 3](#) shows a recurrent neurofuzzy network (Zhang and Morris 1999). A recurrent neurofuzzy network is more appropriate for building long-range prediction models (Zhang and Morris 1999). Both types of neurofuzzy networks contain five layers: an input layer, a fuzzification layer, a rule layer, a function layer, and a defuzzification layer. Several different types of neurons are employed in the network. Inputs to the fuzzification layer are process variables used for defining fuzzy operating regions. Each of these variables is transformed into several fuzzy sets in the fuzzification layer where each neuron corresponds to a particular fuzzy set with the actual membership function being given by the neuron output. Three types of neuron activation functions are used and they are: the sigmoidal function, the Gaussian function, and the complement sigmoidal function. [Figure 4](#) shows the shapes of the membership functions. The two fuzzy sets at the left and right sides in [Fig. 4](#) are represented

Fig. 2

The structural approach neurofuzzy network.

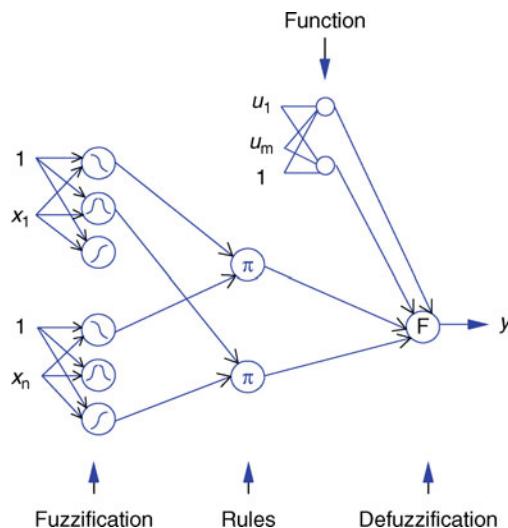
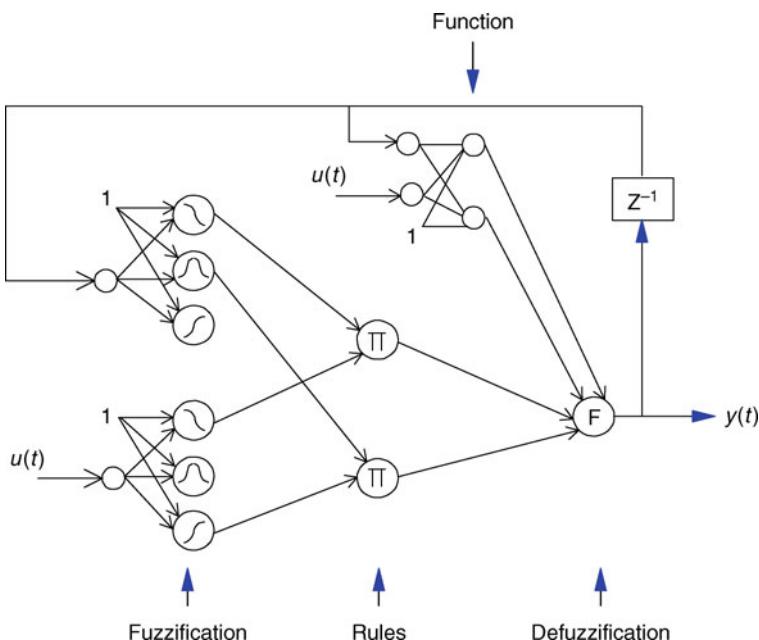
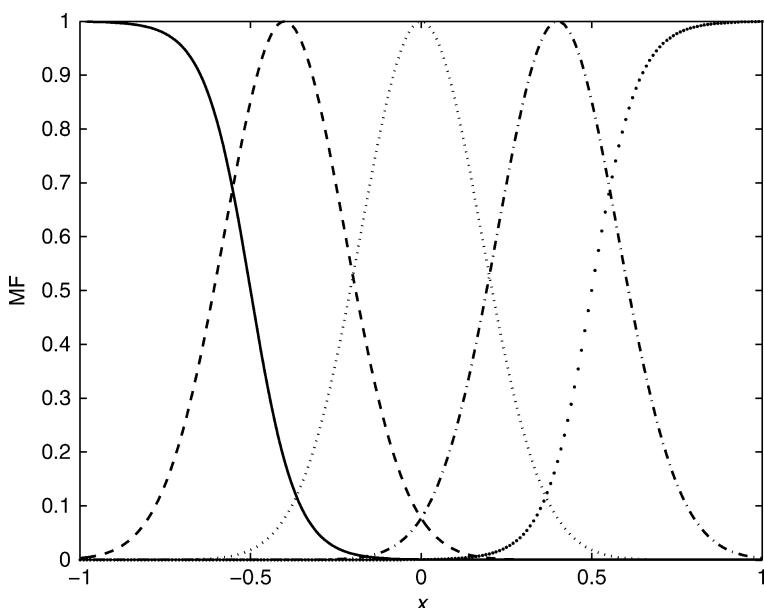


Fig. 3

The structural approach recurrent neurofuzzy network.

**Fig. 4**

Shapes of membership functions.



by neurons with the complement sigmoidal activation function and the sigmoidal activation function, respectively. Other fuzzy sets are represented by neurons with the Gaussian activation function. Weights in this layer determine the membership functions of the corresponding fuzzy sets. By changing the weights in this layer, appropriately shaped membership functions can be obtained. Weights in this layer are initialized based on process knowledge and are refined during network training.

Neurons corresponding to the fuzzy set on the right end of [Fig. 4](#) use the sigmoid function and their outputs are of the form:

$$y = \frac{1}{1 + e^{-(w_1 x + w_0)}} \quad (9)$$

where y is the neuron output, x is the neuron input, w_1 is the input weight, and w_0 is a bias. Neurons corresponding to the fuzzy set on the left end of [Fig. 4](#) use the complement sigmoid function and their outputs are given as:

$$y = 1 - \frac{1}{1 + e^{-(w_1 x + w_0)}} = \frac{e^{-(w_1 x + w_0)}}{1 + e^{-(w_1 x + w_0)}} \quad (10)$$

The activation function used by neurons corresponding to fuzzy sets between the two terminal fuzzy sets in [Fig. 4](#) is the Gaussian function. Outputs of such neurons are of the form:

$$y = e^{-(w_1 x + w_0)^2} \quad (11)$$

The shapes and positions of the membership functions are determined by the values of the fuzzification layer weights. By appropriately selecting the fuzzification layer weights, desired membership functions can be obtained.

Each neuron in the rule layer corresponds to a fuzzy operating region of the process being modeled. Its inputs are the fuzzy sets which determine this operating region. Its output is the product of its input and is the membership function of this fuzzy operating region. Neurons in the rule layer implement the fuzzy intersection defined by [Eq. 5](#). The construction of the rule layer is guided by some knowledge about the process. Knowledge on the number of operating regions and how each region is established is used to set up the rule layer.

Neurons in the function layer implement the local linear models in the fuzzy operating regions. Each neuron corresponds to a particular operating region and is a linear neuron. Its output is a summation of its weighted inputs and a bias which represents the constant term in a local model. Weights in the function layer are the local model parameters. The defuzzification layer performs the COG defuzzification and gives the final network output. Inputs to the defuzzification neuron are membership functions of the fuzzy operating regions and the local model outputs in these regions. The neuron activation function used is given in [Eq. 2](#). With some processes, it may be possible to define the operating regions by using just one process variable. In such cases, the rule layer in [Figs. 2](#) and [3](#) can be removed since the fuzzification layer directly gives the fuzzy operating regions.

2.3 Training of Neurofuzzy Networks

Recurrent neurofuzzy networks can be trained using any of the number of training methods, such as the back propagation method (Rumelhart et al. 1986), the conjugate gradient method

(Leonard and Kramer 1990), Levenberg–Marquardt optimization (Marquardt 1963), or methods based on genetic algorithms (Blanco et al. 2001a; Mak et al. 1999). In this study, recurrent neurofuzzy networks are trained using the Levenberg–Marquardt algorithm with regularization. The training objective function can be defined as:

$$J = \frac{1}{N} \sum_{t=1}^N (\hat{y}(t) - y(t))^2 + \lambda ||W||^2 \quad (12)$$

where N is the number of data points, \hat{y} is the network prediction, y is the target value, t represents the discrete time, W is a vector of network weights, and λ is the regularization parameter.

The objective of regularized training is to penalize excessively large network weights, which do not contribute significantly to the reduction of model errors, so that the trained neural network has a smooth function surface. An intuitive interpretation of this modified objective function is that a weight that does not influence the first term very much will be kept close to zero by the second term. A weight that is important for model fit will, however, not be affected very much by the second term. The appropriate value of λ is obtained through a cross validation procedure. Data for building a recurrent neurofuzzy network model is partitioned into a training data set and a testing data set. Several values of λ are considered and the one resulting in the smallest error on the testing data is adopted. For linear models, minimization of \bullet Eq. 12 leads to the well-known ridge regression formula. Regularization has been widely used in statistical model building and a variety of techniques have been developed, such as ridge regression (Hoerl and Kennard 1970), principal component regression (Geladi and Kowalski 1986), and partial least squares regression (Geladi and Kowalski 1986).

In the Levenberg–Marquardt training method, network weights are adjusted as follows.

$$\Delta W(k+1) = -\eta \left(\frac{1}{N} \sum_{t=1}^N \frac{\partial \hat{y}(t)}{\partial W(k)} \left(\frac{\partial \hat{y}(t)}{\partial W(k)} \right)^T + \delta I \right)^{-1} \frac{\partial J}{\partial W(k)} \quad (13)$$

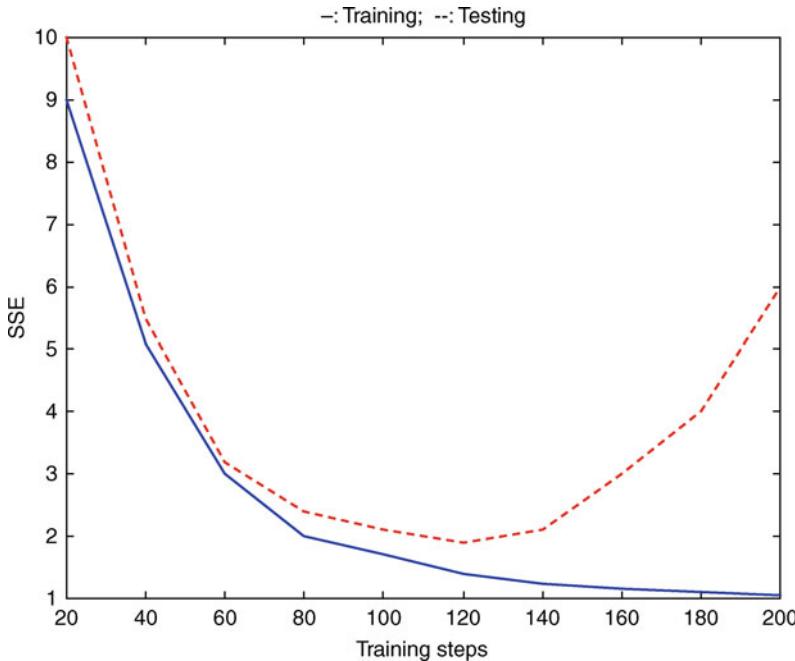
$$W(k+1) = W(k) + \Delta W(k+1) \quad (14)$$

where $W(k)$ and $W(k+1)$ are vectors of weights and weight adaptations at training step k , respectively, η is the learning rate, δ is a parameter to control the searching step size. A large value of δ gives a small step in the gradient direction and a small value of δ gives a searching step close to the Gauss–Newton step. Like other types of gradient-based training methods, training can be terminated when the error gradient is less than a prespecified value. Training can also be terminated by a cross validation-based stopping criterion as described below. For different weights, the gradient $\partial J / \partial W$ can be calculated accordingly. Due to the recurrence in network connections, the gradient $\partial J / \partial W$ needs to be calculated in the “back propagation through time” fashion (Werbos 1990).

A cross validation-based “early stopping” mechanism is used to minimize the problem of over-fitting. Using the early stopping mechanism, network training is stopped at a point beyond which over-fitting would obviously occur. This can be explained using \bullet Fig. 5 which represents a typical neural network learning curve. In \bullet Fig. 5, the vertical axis represents the sum of squared errors (SSE), the horizontal axis represents the network training steps, the solid line represents the network error on training data, and the dashed line represents the network error on testing data. During the initial training stage, both training error and testing error decrease quite quickly. As training progresses, the training error will decrease

Fig. 5

Neural network learning curves.



slowly, and the testing error will start to increase, sometimes very quickly, after certain training steps. The appropriate point to stop training is that point at which the testing error is at its minimum. During network training, both training and testing errors are continuously monitored to detect the appropriate stopping point. “Early stopping” has an implicit effect on regularization as shown by Sjoberg et al. (1995).

3 Modeling of a Neutralization Process Using a Recurrent Neurofuzzy Network

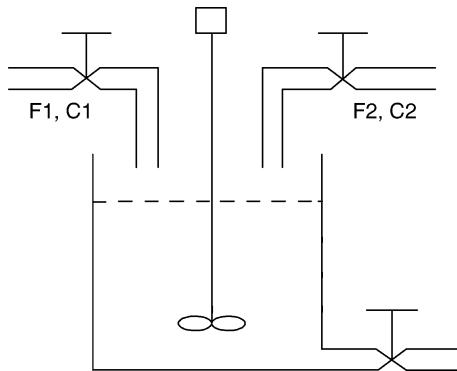
A recurrent neurofuzzy network was used to model a neutralization process which was taken from McAvoy et al. (1972). The neutralization process takes place in a CSTR which is shown in [Fig. 6](#). There are two input streams to the CSTR. One is acetic acid of concentration C_1 at a flow rate F_1 and the other sodium hydroxide of concentration C_2 at a flow rate F_2 . The mathematical equations of the CSTR can be described as follows by assuming that the tank level is perfectly controlled (McAvoy et al. 1972).

$$V \frac{d\zeta}{dt} = F_1 C_1 - (F_1 + F_2)\zeta \quad (15)$$

$$V \frac{d\zeta}{dt} = F_2 C_2 - (F_1 + F_2)\zeta \quad (16)$$

Fig. 6

A continuous stirred tank reactor for pH neutralization.



$$[H^+]^3 + (K_a + \zeta)[H^+]^2 + (K_a(\zeta - \xi) - K_w)[H^+] - K_w K_a = 0 \quad (17)$$

$$\text{pH} = -\log_{10}[H^+] \quad (18)$$

where

$$\zeta = [HAC] + [AC^-] \quad (19)$$

$$\xi = [Na^+] \quad (20)$$

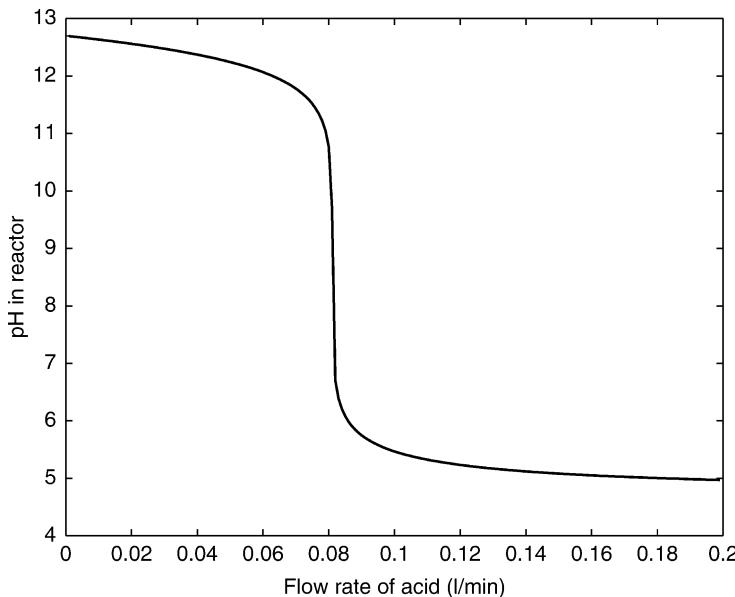
Table 1 gives the meaning and the initial setting of each variable. It is well-known that pH dynamics are highly nonlinear. The steady-state relationship between acid flow rate and pH in the reactor (the titration curve) is plotted in Fig. 7. It can be seen from Fig. 7 that the process gain is very high in the medium pH region while being quite low in both low- and high-pH regions. A recurrent neurofuzzy network model was developed to model the nonlinear dynamic relationships between the acetic acid flow rate and the pH in the reactor.

To generate training, testing, and unseen validation data, multilevel random perturbations were added to the flow rate of acetic acid while other inputs to the reactor were kept constant. Three sets of data were generated. One set was used as training data, another set was used as testing data, and the remaining set was used as unseen validation data. The simulated pH values in the reactor were corrupted with random noise in the range $(-0.3, 0.3)$ representing measurement noise. Training, testing, and validation data are plotted in Figs. 8–10, respectively.

A recurrent neurofuzzy network was used to build a multistep-ahead prediction model for the neutralization process. Based on the process characteristics, the nonlinear pH-operating region is divided into three local regions: pH low, pH medium, and pH high. The fuzzification layer weights were initialized based on the titration curve. Weights for the function layer were initialized as random numbers in the range $(-0.2, 0.2)$. Initially, within each local region a second-order linear model was selected. If the identified model is not sufficiently adequate then the local model order and/or the number of fuzzy operating regions are increased.

Table 1**The physical parameters used in simulation**

Variable	Meaning	Initial setting
V	Volume of tank	1 l
F_1	Flow rate of acid	0.081 l/min
F_2	Flow rate of base	0.512 l/min
C_1	Concentration of acid in F_1	0.32 moles/l
C_2	Concentration of acid in F_2	0.05005 moles/l
K_a	Acid equilibrium constant	1.8×10^{-5}
K_w	Water equilibrium constant	1.0×10^{-14}

Fig. 7**Titration curve.**

The recurrent neurofuzzy network was trained using the Levenberg–Marquardt training method together with regularization and a cross-validation-based stopping criterion. After network training, the following multistep-ahead prediction fuzzy model was identified:

IF pH low

THEN $\hat{y}(t) = 0.6367\hat{y}(t-1) + 0.2352\hat{y}(t-2) - 3.56u(t-1) - 1.81u(t-2) - 1.214$

IF pH medium

THEN $\hat{y}(t) = 0.3309\hat{y}(t-1) + 0.2798\hat{y}(t-2) - 94.5u(t-1) - 22.53u(t-2) + 13.117$

IF pH high

THEN $\hat{y}(t) = 0.6631\hat{y}(t-1) + 0.2595\hat{y}(t-2) - 6.16u(t-1) - 1.12u(t-2) + 1.244$

Fig. 8
Training data.

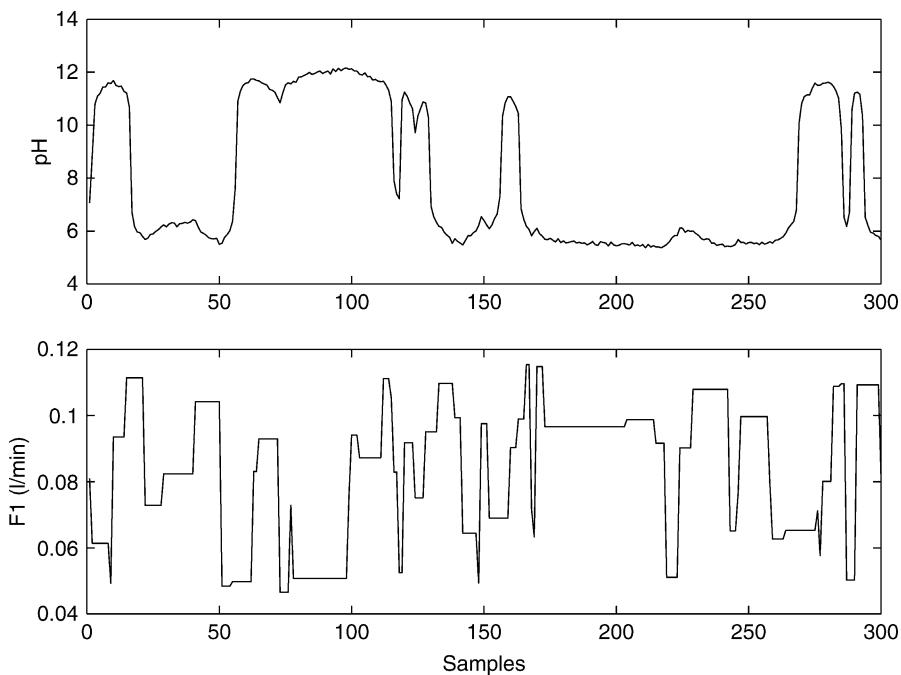


Fig. 9
Testing data.

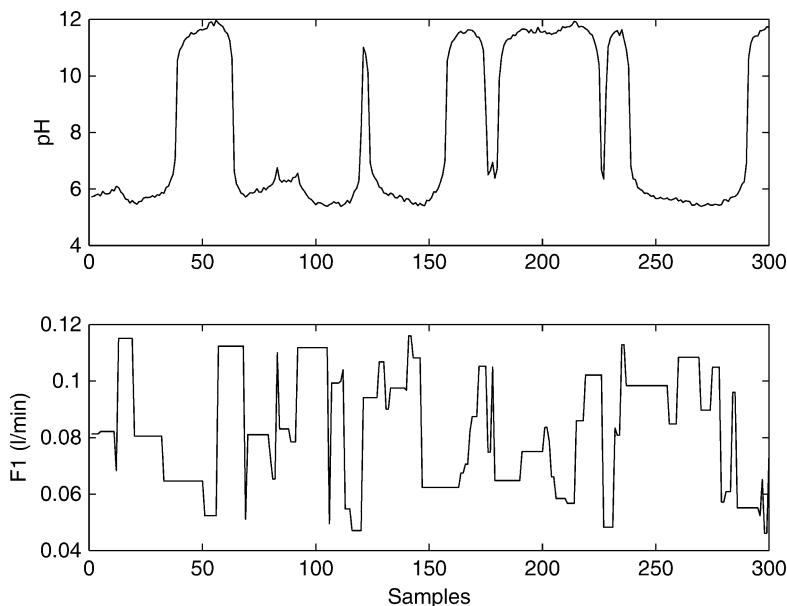
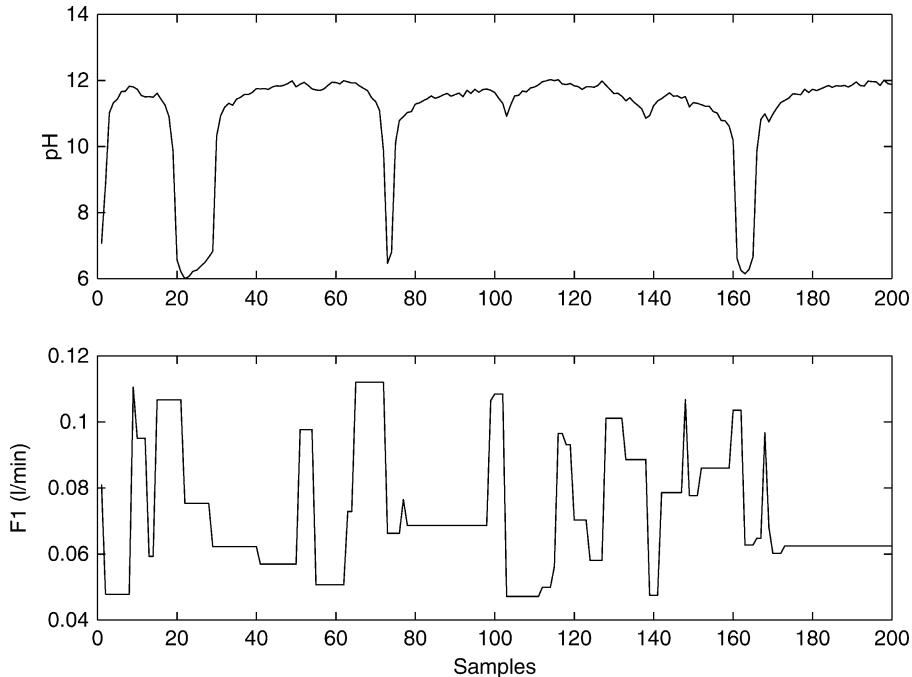


Fig. 10

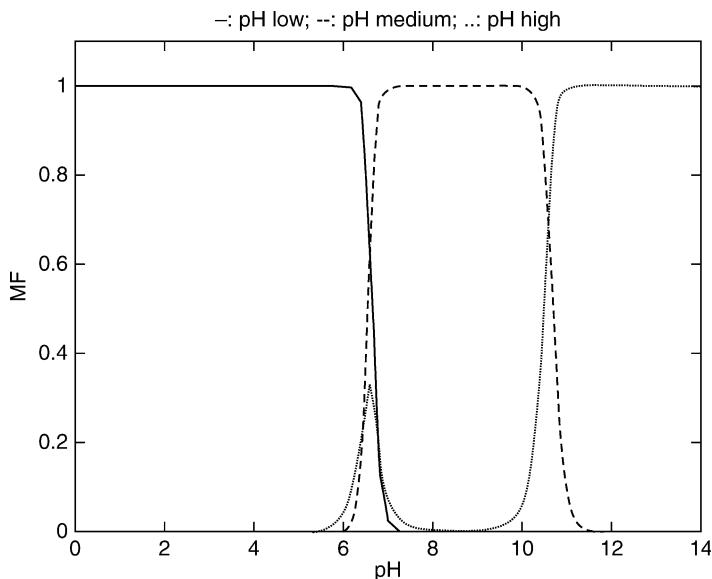
Validation data.



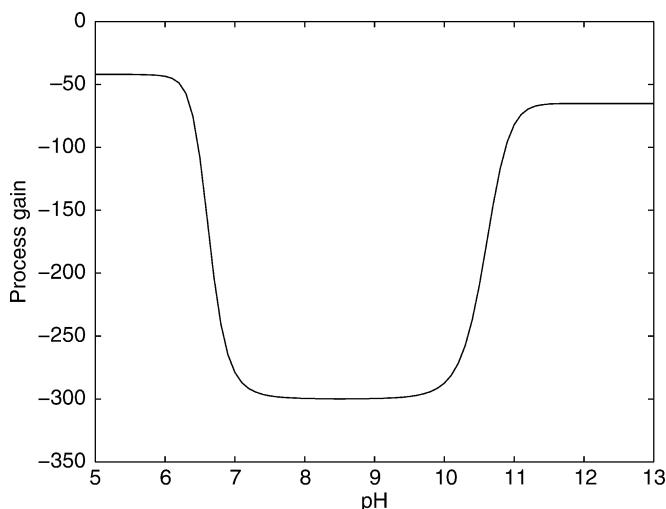
In the above model, \hat{y} is the predicted pH in the reactor, u is the flow rate of acetic acid, and t is the discrete time. The identified membership functions (normalized) represented by the fuzzification layer weights, for the three fuzzy sets: pH low, pH medium, and pH high, are plotted in [Fig. 11](#). It can be seen that the identified partition of process-operating regions agrees with the operating regions indicated by the titration curve. The static process gains in the pH low, pH medium, and pH high regions are easily calculated from the local model parameters as -41.92 , -300.61 , and -65.12 , respectively. Thus, the local model associated with the operating region, pH medium, has much higher gain than the local models for the other two operating regions. This is exactly what one would expect from the titration curve. Neurofuzzy network models can, hence, provide certain insight into the modeled processes and are easy to interpret. Using COG defuzzification, the process gain can be calculated from the local model gains and the membership functions of the local operating regions. The calculated process gain is plotted in [Fig. 12](#) and it matches with what is indicated by the titration curve. The above recurrent neurofuzzy network model is much easier to interpret than strictly black box models, with the fuzzification layer weights indicating how the process operation is partitioned into different operating regions and the function layer weights showing the local dynamic behavior of the process. This again is in marked contrast to conventional neural network models where the modeling function of the network weights are difficult, if not impossible, to understand. The local linear dynamic models can also be transformed into frequency-domain transfer function forms and interpreted using pole-zero positions.

Fig. 11

Membership functions learnt by the recurrent neurofuzzy network.

**Fig. 12**

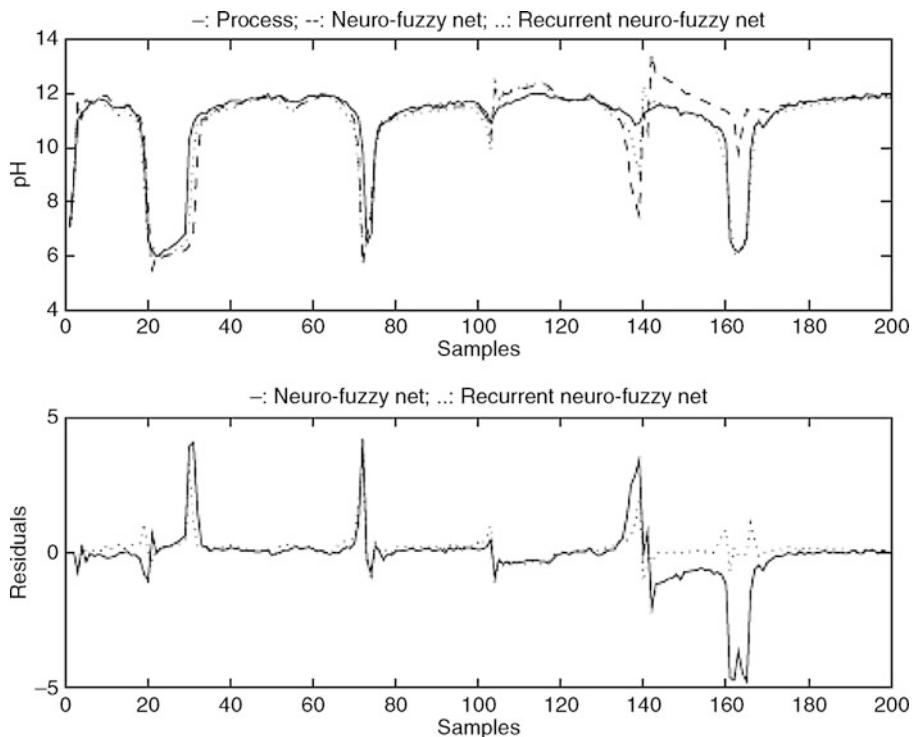
Identified process gains at different operating conditions.



Long-term predictions of pH from the recurrent neurofuzzy network on the unseen validation data are plotted in [Fig. 13](#). It can be seen that the long-term predictions are quite accurate. The SSE of multistep-ahead predictions from the recurrent neurofuzzy network model on training, testing, and validation data are given in [Table 2](#). For the purpose

Fig. 13

Multistep-ahead predictions on validation data.

**Table 2**

SSE for multistep-ahead predictions

	Training	Testing	Validation
Feed forward neurofuzzy net	307	419	211
Recurrent neurofuzzy net	36	37	13

of comparison, a feed forward neurofuzzy network was also used to model the pH dynamics and the following one-step-ahead prediction model was identified.

IF pH low

$$\text{THEN } \hat{y}(t) = 0.5651y(t-1) + 0.2712y(t-2) - 4.92u(t-1) - 0.52u(t-2) + 1.424$$

IF pH medium

$$\text{THEN } \hat{y}(t) = 0.3011y(t-1) + 0.2289y(t-2) - 100.39u(t-1) \\ - 30.07u(t-2) + 14.695$$

IF pH high

$$\text{THEN } \hat{y}(t) = 0.667y(t-1) + 0.2463y(t-2) - 0.78u(t-1) - 3.87u(t-2) + 1.323$$

This model can give accurate one-step-ahead predictions with the SSE on training, testing, and validation data being 18.5, 24.4, and 11.1, respectively. However, if this model is used for multistep-ahead prediction, the prediction accuracy deteriorates significantly as is shown in \blacktriangleright [Table 2](#). Multistep-ahead predictions of pH from the feed forward neurofuzzy network are plotted in \blacktriangleright [Fig. 13](#). Both \blacktriangleright [Fig. 13](#) and \blacktriangleright [Table 2](#) show that the recurrent neurofuzzy network can achieve much accurate multistep-ahead predictions than the feed forward neurofuzzy network.

A number of model-validity tests for nonlinear model identification procedures have been developed, for example, the statistical chi-square test (Leontaris and Billings [1987](#)), the Akaike Information Criterion (AIC) (Akaike [1974](#)), the predicted squared error criterion (Barron [1984](#)), and the high order correlation tests (Billings and Voon [1986](#)). The validity of a neurofuzzy network model can be assessed using the following high order correlation tests (Billings and Voon [1986](#)).

$$\Phi_{ee}(\tau) = E[\varepsilon(t - \tau)\varepsilon(t)] = \delta(\tau) \quad \forall \tau \quad (21)$$

$$\Phi_{ue}(\tau) = E[u(t - \tau)\varepsilon(t)] = 0 \quad \forall \tau \quad (22)$$

$$\Phi_{u^2e}(\tau) = E\{[u^2(t - \tau) - \overline{u^2(t)}]\varepsilon(t)\} = 0 \quad \forall \tau \quad (23)$$

$$\Phi_{u^2\varepsilon^2}(\tau) = E\{[u^2(t - \tau) - \overline{u^2(t)}]\varepsilon^2(t)\} = 0 \quad \forall \tau \quad (24)$$

$$\Phi_{e(eu)}(\tau) = E[\varepsilon(t)\varepsilon(t - 1 - \tau)u(t - 1 - \tau)] = 0 \quad \forall \tau \quad (25)$$

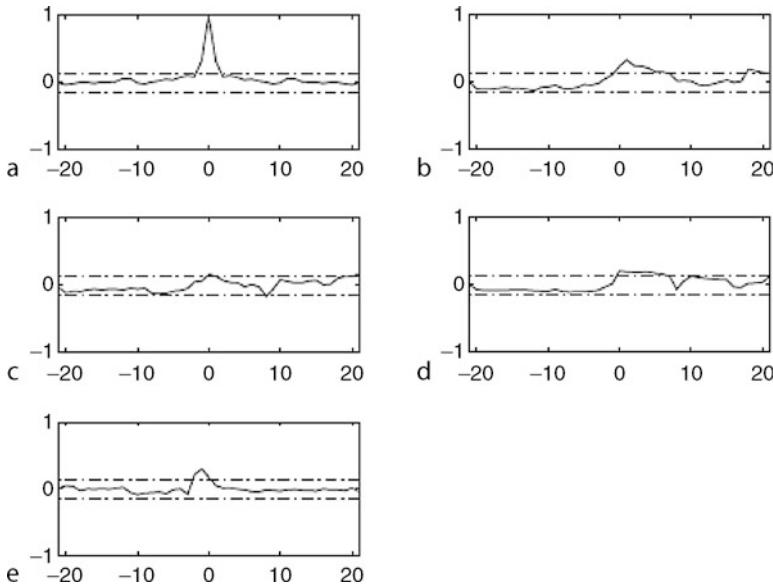
where ε is the model residual and $\overline{u^2}$ is the time average of u^2 . These tests look into the cross correlation amongst model residuals and inputs. Normalization to give all tests a range of plus and minus one and approximate the 95% confidence bounds at $1.96/\sqrt{N}$, N being the number of testing data, make the tests independent of signal amplitudes and easy to interpret. If these correlation tests are satisfied, then the model residuals are a random sequence and are not predictable from the model inputs. \blacktriangleright [Figure 14](#) shows the correlation-based model validation test (Billings and Voon [1986](#)) results for the recurrent neurofuzzy network-based multistep-ahead prediction model. In \blacktriangleright [Fig. 14](#), plots a–e are $\Phi_{ee}(\tau)$, $\Phi_{ue}(\tau)$, $\Phi_{u^2e}(\tau)$, $\Phi_{u^2\varepsilon^2}(\tau)$, and $\Phi_{e(eu)}(\tau)$, respectively. The dash-dot lines in each plot are the 95% confidence bounds. It can be seen that only a couple of points are slightly outside the 95% confidence bounds and, hence, the model can be regarded as being adequate for long-term predictions.

4 Nonlinear Model-Based Control Through Combination of Local Linear Controllers Based on Neurofuzzy Network Models

The neurofuzzy network topology proposed with its local linear models allow local linear model-based predictive controllers, such as generalized predictive controllers (GPC) (Clarke et al. [1987](#)), to be developed. These local controllers can then be combined in a similar method to that used in the heterogeneous control strategy proposed by Kuipers and Astrom (Kuipers and Astrom [1994](#)). The global controller output is obtained by combining the local model outputs in the format similar to the COG defuzzification. This is illustrated by letting

Fig. 14

Model validation tests for the recurrent neurofuzzy network.



μ_i and u_i be the membership function and output of the local controller for the i th operating region, respectively, and calculating the global controller output, u , as

$$u = \frac{\sum_{i=1}^{nr} \mu_i u_i}{\sum_{i=1}^{nr} \mu_i} \quad (26)$$

The advantage of using local linear models is that a nonlinear model predictive controller can be constructed through several local linear model predictive controllers which usually have analytical solutions. Hence, the control actions of this nonlinear model predictive controller can be obtained analytically avoiding the time-consuming numerical search procedures and the uncertainty in convergence to the global optimum which are typically seen in conventional nonlinear model-based predictive control strategies (Eaton and Rawlings 1990; Hernandez and Arkun 1993; Li and Biegler 1989; Maner and Doyle 1997; Saint-Donat et al. 1991; Sistu et al. 1993). Furthermore, control actions calculated from local linear models in incremental form contain integral actions which can naturally eliminate static control offsets.

Neurofuzzy network-based nonlinear model predictive control strategy is derived as follows. Consider the following local linear model

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) \quad (27)$$

where A and B are polynomials in the backward shift operator q^{-1} .

$$A(q^{-1}) = 1 + a_1 q^{-1} + \dots + a_{no} q^{-no} \quad (28)$$

$$B(q^{-1}) = 1 + b_1 q^{-1} + \dots + b_{ni} q^{-ni} \quad (29)$$

To derive a j -step ahead predictor of $y(t + j)$ based on [Eq. 27](#), consider the following Diophantine equation.

$$1 = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F_j(q^{-1}) \quad (30)$$

where $E_j(q^{-1})$ and $F_j(q^{-1})$ are polynomials in q^{-1} uniquely defined given $A(q^{-1})$ and the prediction interval j and Δ is the differencing operator, $1 - q^{-1}$. Multiplying [Eq. 9](#) by $E_j(q^{-1})\Delta q^j$ gives

$$E_j(q^{-1})A(q^{-1})\Delta y(t + j) = E_j(q^{-1})B(q^{-1})\Delta u(t + j - 1) \quad (31)$$

The substitution of $E_j(q^{-1})A(q^{-1})\Delta$ from [Eq. 12](#) gives

$$y(t + j) = E_j(q^{-1})B(q^{-1})\Delta u(t + j - 1) + F_j(q^{-1})y(t) \quad (32)$$

Therefore, the j -step ahead prediction of y , given measured output data up to time t and any given input $u(t + i)$ for $i > 1$, is

$$\hat{y}(t + j|t) = G_j(q^{-1})\Delta u(t + j - 1) + F_j(q^{-1})y(t) \quad (33)$$

where $G_j(q^{-1}) = E_j(q^{-1})B(q^{-1})$. The Diophantine equation can be recursively solved (Clarke et al. 1987). The following vector equation can be written for a prediction horizon of N .

$$\hat{Y} = GU + F \quad (34)$$

where

$$\hat{Y} = [\hat{y}(t + 1) \hat{y}(t + 2) \dots \hat{y}(t + N)]^T \quad (35)$$

$$U = [\Delta u(t) \Delta u(t + 1) \dots \Delta u(t + N - 1)]^T \quad (36)$$

$$F = [f(t + 1) f(t + 2) \dots f(t + N)]^T \quad (37)$$

The matrix G is lower-triangular of dimension $N \times N$:

$$G = \begin{bmatrix} g_0 & 0 & \dots & 0 \\ g_1 & g_0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ g_{N-1} & g_{N-2} & \dots & g_0 \end{bmatrix} \quad (38)$$

Suppose that a future set point or reference sequence $[w(t + j); j = 1, 2, \dots]$ is available. The control objective function can be written as

$$J = (\hat{Y} - W)^T(\hat{Y} - W) + \lambda U^T U \quad (39)$$

The unconstrained minimization of the above objective function leads to the following optimal control strategy.

$$U = (G^T G + \lambda I)^{-1} G^T (W - F) \quad (40)$$

Since the first element of U is $\Delta u(t)$, the current control action is given by

$$u(t) = u(t-1) + \bar{g}^T(W - F) \quad (41)$$

where \bar{g}^T is the first row of $(G^T G + \lambda I)^{-1} G^T$. The control action obtained in [Fig. 41](#) hence contains an integral action which provides zero static offset.

A neurofuzzy network-based heterogeneous long-range predictive controller was developed for the pH reactor. This consisted of three local linear GPC controllers based on the following three local linear incremental models transformed from the recurrent neurofuzzy network model:

IF pH low

THEN $\hat{y}(t) = 1.6367\hat{y}(t-1) - 0.4015\hat{y}(t-2) - 3.56\Delta u(t-1) - 1.81\Delta u(t-2)$

IF pH medium

THEN $\hat{y}(t) = 1.3309\hat{y}(t-1) - 0.0511\hat{y}(t-2) - 94.5\Delta u(t-1) - 22.53\Delta u(t-2)$

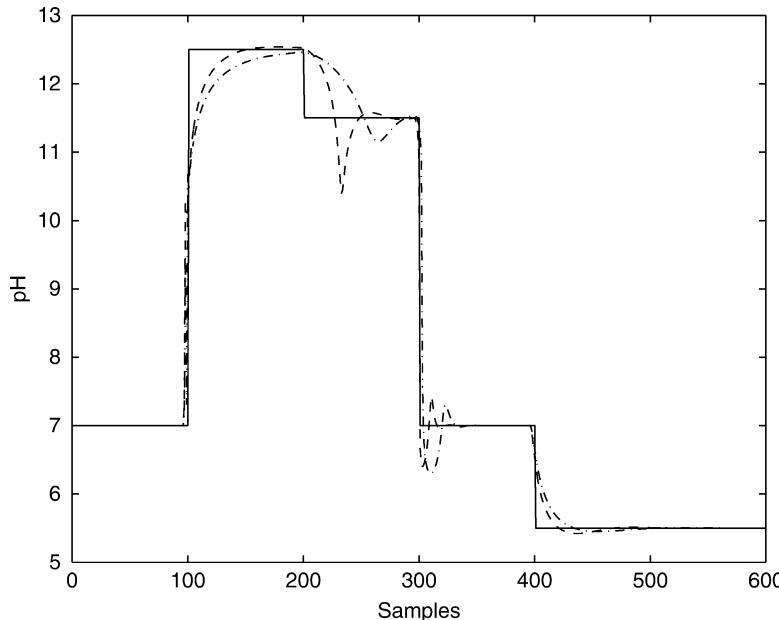
IF pH high

THEN $\hat{y}(t) = 1.6631\hat{y}(t-1) - 0.4036\hat{y}(t-2) - 6.16\Delta u(t-1) - 1.12\Delta u(t-2)$

The control horizons for the three local GPC controllers were all set to one and prediction horizons were all set to five. A linear GPC controller was also developed using a linear model identified from the data shown in [Figs. 8](#) and [9](#) again with a control horizon of one and a prediction horizon of five. [Figure 15](#) shows the performance of the neurofuzzy network-based heterogeneous long-range predictive controller and the linear GPC controller

Fig. 15

Control performance for set-point tracking.

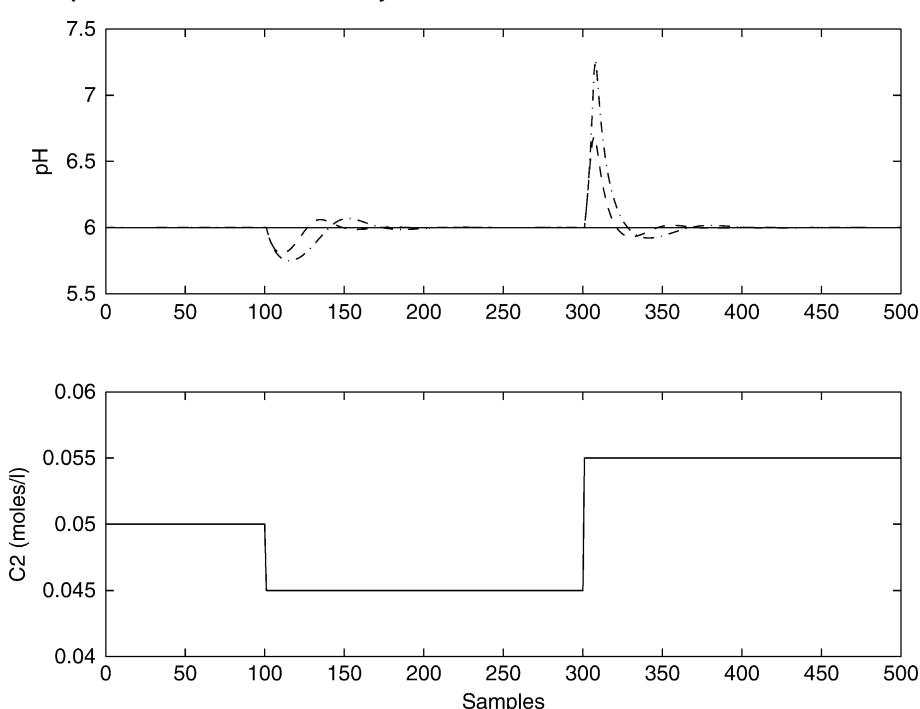


for set-point tracking over a very wide range of process operation. In [Fig. 15](#), the solid line represents the set points, the dashed line represents the responses under the neurofuzzy GPC controller, and the dash-dot line represents the responses under the linear GPC controller. It can be seen that the response under the linear GPC controller is very sluggish in the pH low and pH high regions. However, the response under the neurofuzzy GPC controller is satisfactory in all operating regions. This is due to the fact that the neurofuzzy GPC controller output in a particular operating region is mainly determined by the corresponding local controller which is based on a more appropriate local model for that region.

To examine the performance of the neurofuzzy network-based long-range predictive controller in rejecting disturbances, negative and positive step changes in base concentration (C_2) were added to the process. [Figure 16](#) shows the disturbances and the performance of the neurofuzzy network-based predictive controller and the linear GPC controller. In the upper plot of [Fig. 16](#), the solid line represents the set points, the dashed line represents the responses under the neurofuzzy GPC controller, and the dash-dot line represents the responses under the linear GPC controller. It can be seen that the neurofuzzy network-based predictive controller performs very well whereas the linear model-based controller performs very poorly.

Although a gain scheduled controller can also be developed to control the neutralization process (Astrom and Wittenmark 1989), the gain scheduling strategy is usually used on controllers with a simple form such as a PID controller where the controller gain is scheduled according to a monitored process variable. Traditional gain-scheduling techniques are not straightforward to be implemented on long-range model-based predictive controllers.

Fig. 16
Control performance for disturbance rejection.

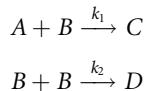


The recurrent neurofuzzy network-based long-range predictive control strategy presented in this section represents a model parameter scheduled long-range predictive control strategy.

5 Modeling and Optimal Control of Batch Processes Using Neurofuzzy Networks

5.1 A Fed-Batch Reactor

The fed-batch reactor is taken from Terwiesch et al. (1998). The following reaction system



is conducted in an isothermal semi-batch reactor. The objective in operating this reactor is, through the addition of reactant B , to convert as much as possible of reactant A to the desired product, C , in a specified time $t_f = 120$ min. It would not be optimal to add all B initially as the second-order side reaction yielding the undesired species D will be favored at high concentration of B . To keep this undesired species low, the reactor is operated in semi-batch mode where B is added in a feed stream with concentration $b_{\text{feed}} = 0.2$. Based on the reaction kinetics and material balances in the reactor, the following mechanistic model can be developed.

$$\frac{d[A]}{dt} = -k_1[A][B] - \frac{[A]}{V} u \quad (42)$$

$$\frac{d[B]}{dt} = -k_1[A][B] - 2k_2[B]^2 + \frac{b_{\text{feed}} - [B]}{V} u \quad (43)$$

$$\frac{d[C]}{dt} = k_1[A][B] - \frac{[C]}{V} u \quad (44)$$

$$\frac{d[D]}{dt} = 2k_2[B]^2 - \frac{[D]}{V} u \quad (45)$$

$$\frac{dV}{dt} = u \quad (46)$$

In the above equations, $[A]$, $[B]$, $[C]$, and $[D]$ denote, respectively, the concentrations of A , B , C , and D , V is the current reaction volume, u is the reactant feed rate, and the reaction rate constants have the nominal value $k_1 = 0.5$ and $k_2 = 0.5$. At the start of reaction, the reactor contains $[A](0) = 0.2$ moles/L of A , no B ($[B](0) = 0$) and is fed to 50% ($V(0) = 0.5$).

5.2 Recurrent Neurofuzzy Network Modeling

In this study, it is assumed that the above mechanistic model is not available (due to, e.g., unknown reaction mechanism or unknown parameters, which are commonly encountered in practice) so a data-based empirical model has to be utilized. Since the main interest in this process is on the end-of-batch product quality, the data-based empirical model should offer

accurate long-range predictions. A recurrent neurofuzzy network is used to model the process from the process operation data. \blacktriangleright Figure 17 shows the data from a typical batch run. It can be seen from \blacktriangleright Fig. 17 that the process behaves nonlinearly and the dynamic characteristics vary with the batch operating stages. \blacktriangleright Figure 17 indicates that the reaction volume could be used to divide the process operation into several stages, for example, labeled as “low volume,” “medium volume,” and “high volume.” Four batches of process operation under different feeding policies were simulated to produce the data for recurrent neurofuzzy network modeling. In agile responsive batch processing, it is highly desirable that a model can be developed using data from a limited number of process runs so that the model can be quickly developed and applied to the process. Based on this consideration, the data for developing recurrent neurofuzzy network model is limited to data from just four batch runs. It should be noted here that the model can be retrained when data from more batch runs are available.

After network training, the following model was identified.

IF $V(t)$ is low

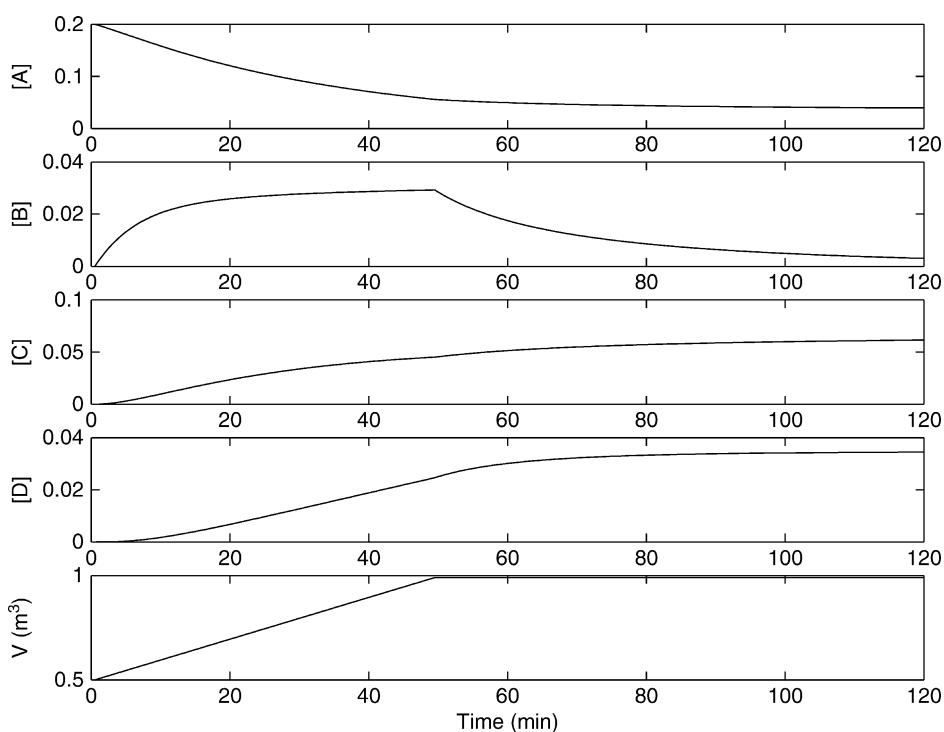
THEN

$$y_1(t) = 0.4577y_1(t-1) + 0.5449y_1(t-2) - 0.0339V(t-1) \\ + 0.0064V(t-2) + 0.0264V(t-3)$$

$$y_2(t) = 0.5678y_2(t-1) + 0.4340y_2(t-2) + 0.0066V(t-1) \\ + 0.0077V(t-2) - 0.0141V(t-3)$$

\blacksquare Fig. 17

Data from a typical batch run.



IF $V(t)$ is medium

THEN

$$y_1(t) = 0.8247y_1(t-1) + 0.0683y_1(t-2) - 0.0002V(t-1) \\ - 0.0002V(t-2) + 0.0079V(t-3)$$

$$y_2(t) = 0.2323y_2(t-1) + 0.7725y_2(t-2) - 0.0611V(t-1) \\ + 0.00002V(t-2) + 0.0608V(t-3)$$

IF $V(t)$ is high

THEN

$$y_1(t) = 0.8553y_1(t-1) + 0.1004y_1(t-2) - 0.0134V(t-1) \\ + 0.003V(t-2) + 0.0133V(t-3)$$

$$y_2(t) = -0.9829y_2(t-1) + 1.9827y_2(t-2) - 0.0143V(t-1) \\ + 0.0223V(t-2) - 0.008V(t-3)$$

In the above model, y_1 is $[C]$, y_2 is $[D]$, V is the reaction volume, and t is the discrete time. When performing long-range predictions using the above neurofuzzy network model, the future reaction volume $V(t)$ can be simply obtained by integrating the feeding rate (u). The identified membership functions for reaction volume “low,” “medium,” and “high” are shown in Fig. 18. The identified recurrent neurofuzzy network model was tested on an additional unseen testing batch, which is not used when developing the recurrent neurofuzzy network model. Fig. 19 shows the long-range predictions of $[C]$ as well as the model prediction

Fig. 18

Identified membership functions.

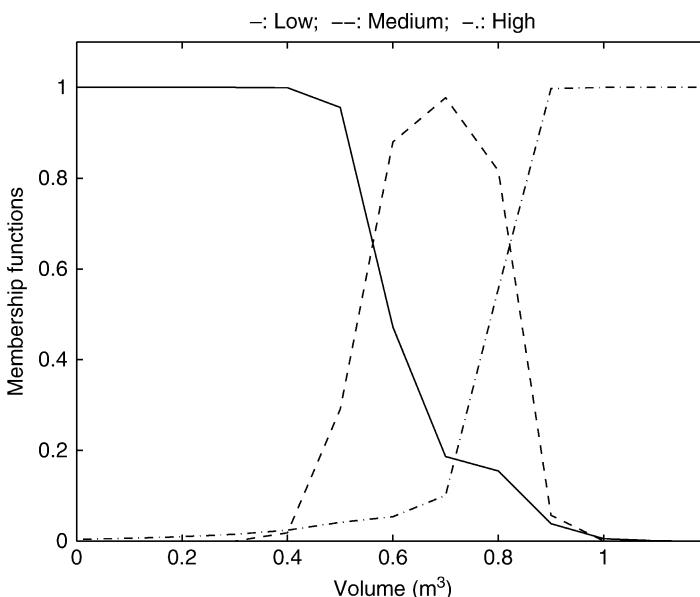
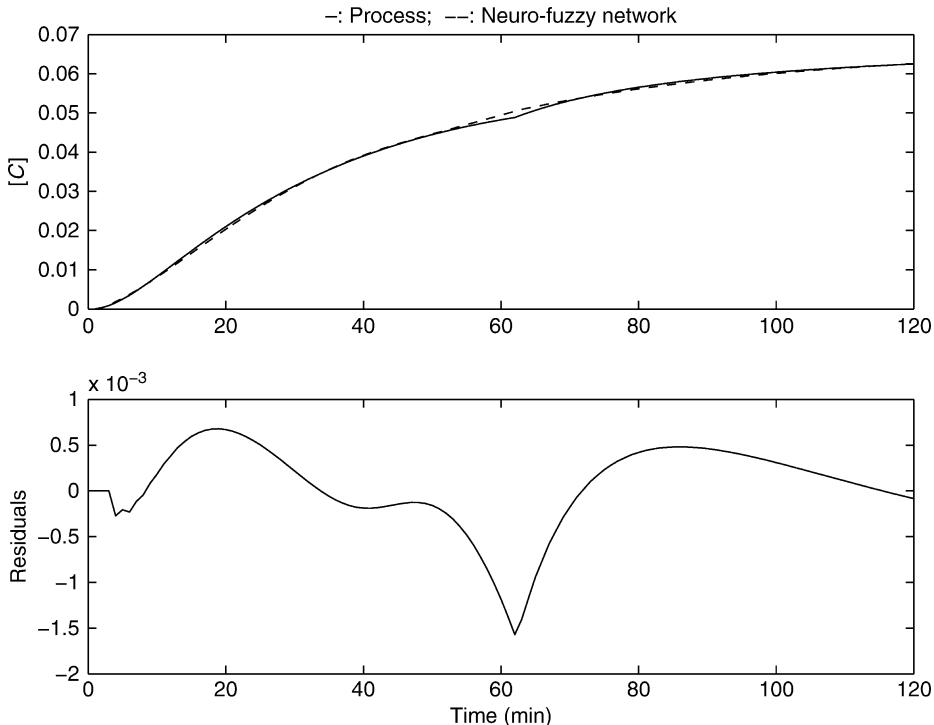
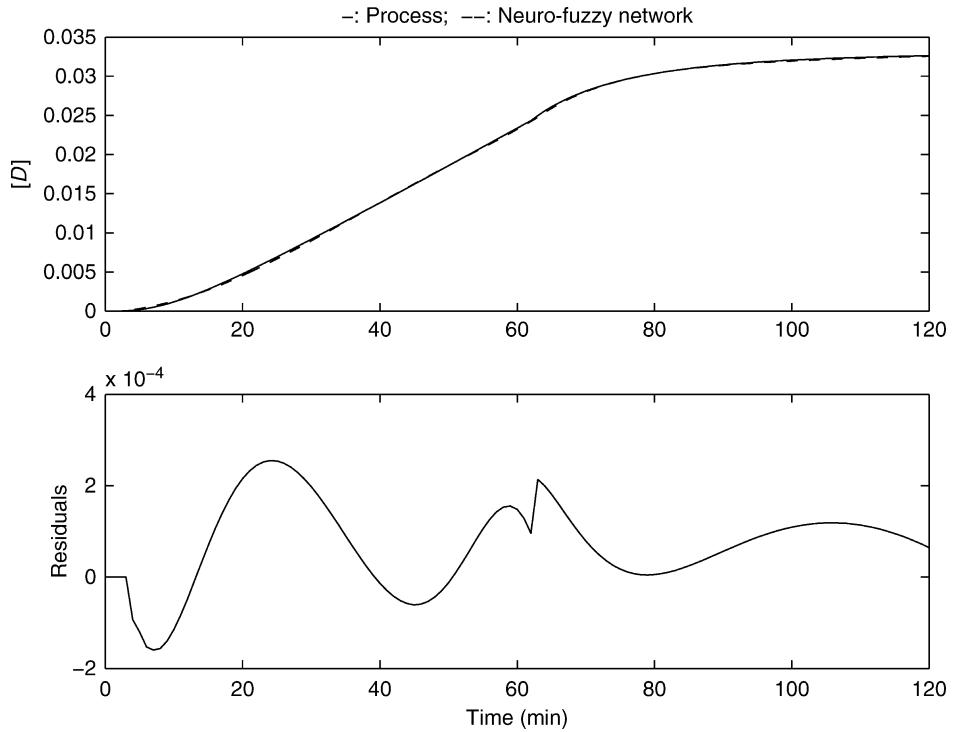


Fig. 19Long-range predictions of $[C]$.

residuals, on the unseen testing batch, whereas [Fig. 20](#) shows those for $[D]$. In the top plots of [Figs. 19](#) and [20](#), the solid lines represent the actual values whereas the dashed lines represent the model predictions. The bottom plots of [Figs. 19](#) and [20](#) show the recurrent neurofuzzy network model prediction residuals. It can be seen from [Figs. 19](#) and [20](#) that the model predictions can be considered as being accurate.

The above local linear models are easier to interpret than a pure black box model (e.g., a conventional neural network model). From the local model parameters, the local model characteristics, such as gain and time constant, can be calculated. From the identified membership functions, different process-operating regions can be visualized. [Table 3](#) shows the calculated static gain of different local models. It can be seen that $[C]$ has much higher gain in the “Volume low” region than other regions, whereas $[D]$ has high gain in the “Volume medium” region. Insights into these local model characteristics could help process operators in understanding the process operation and in judging the optimal control actions. It should be noticed that the static gains shown in [Table 3](#) can only give a rough guidance about the operational characteristics in the local regions since a batch process does not have a steady state. The local dynamic gains can be calculated from the local linear models. [Figure 21](#) shows the local model dynamic gains for $[C]$ while [Fig. 22](#) shows those for $[D]$.

Fig. 20Long-range predictions of $[D]$.

5.3 Multi-objective Optimal Control of the Fed-Batch Reactor

The objectives in operating this process are to maximize the amount of the final product $[C](t_f)V(t_f)$ and minimize the amount of the final undesired species $[D](t_f)V(t_f)$. This can be formulated as a multi-objective optimization problem with the goal-attainment method (Gembicki 1974):

$$\mathbf{F}(U) = \begin{bmatrix} -[C](t_f)V(t_f) \\ [D](t_f)V(t_f) \end{bmatrix} \quad (47)$$

$$\min_{U, \gamma} \gamma$$

$$\text{s.t. } \mathbf{F}(U) - \mathbf{W}\gamma \leq \mathbf{GOAL}$$

$$0 \leq u_i \leq 0.01 \quad (i = 1, 2, \dots, m)$$

$$V(t_f) \leq 1$$

where γ is a scalar variable, $\mathbf{W} = [w_1 \ w_2]^T$ are weighting parameters, $\mathbf{GOAL} = [g_1 \ g_2]^T$ are the goals that the objectives attain, $U = [u_1, u_2, \dots, u_m]$ is a sequence of the reactant-feeding rates and V is the reaction volume. In this study, the batch time (120 min) is divided into $m = 10$ segments of equal length and, within the i th segment, the reactant feeding rate is u_i .

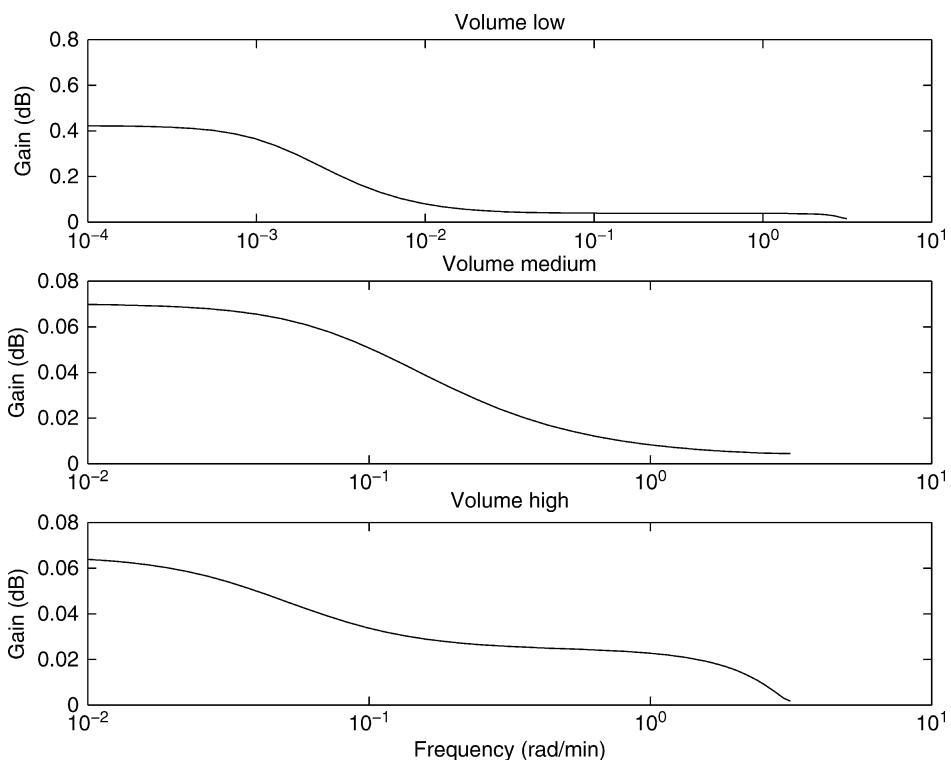
Table 3

Static gains calculated from the local models

	Volume low	Volume medium	Volume high
[C]	0.4231	0.0701	0.0655
[D]	-0.1111	0.0583	-0.0064

Fig. 21

The dynamic gains for the local models for [C].



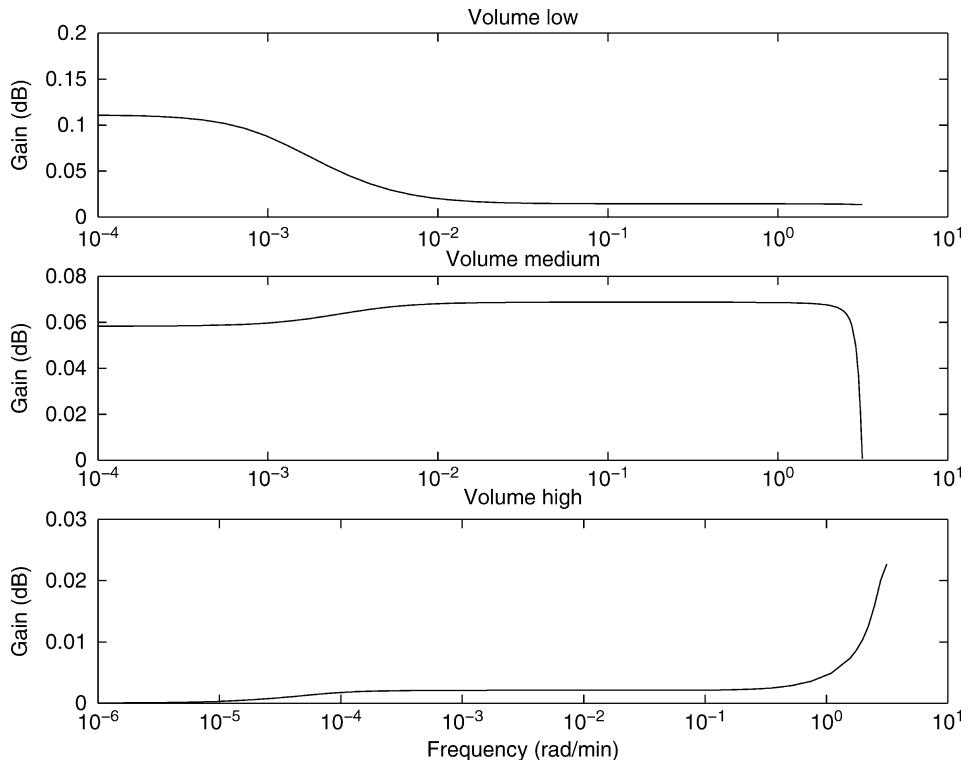
Two cases are studied here: Case 1 – $\mathbf{GOAL} = [-0.065 \ 0.01]^T$ and Case 2 – $\mathbf{GOAL} = [-0.1 \ 0.02]^T$. In both cases the weighting parameters w_1 and w_2 are selected as 1. Case 1 emphasizes on less by-product while Case 2 emphasizes on more product.

The optimization problem was solved using the MATLAB Optimization Toolbox function “`attgoal`.” The calculated optimal reactant feeding policies for both cases are plotted in [Fig. 23](#). It can be seen that the feeding policies are very different for the two cases.

Under these feeding policies, the trajectories of process variables in Cases 1 and 2 are respectively shown in [Figs. 24](#) and [25](#). In Case 1 the values of $[C](t_f)$ and $[D](t_f)$ are 0.0547 and 0.0202, respectively, whereas in Case 2 those are 0.0617 and 0.0349, respectively. It can be seen that these two different feeding policies lead to different operation objectives. Looking back at the local linear models and their gains given in [Table 3](#), the control

Fig. 22

The dynamic gains for the local models for $[D]$.

**Fig. 23**

Optimal reactant feeding policies.

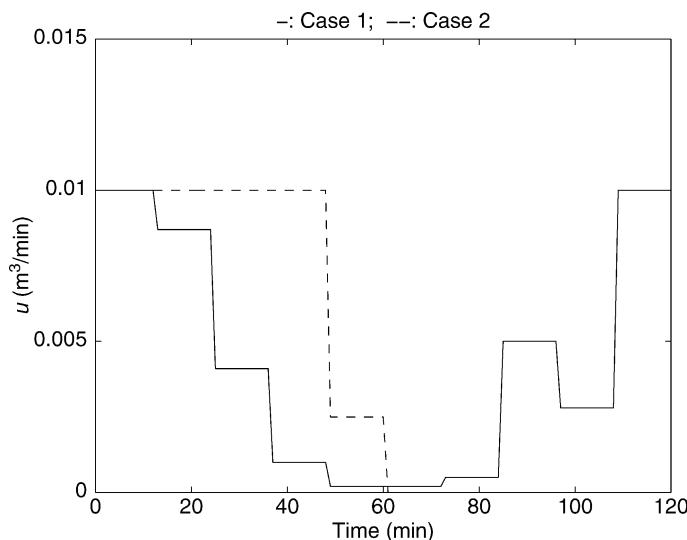
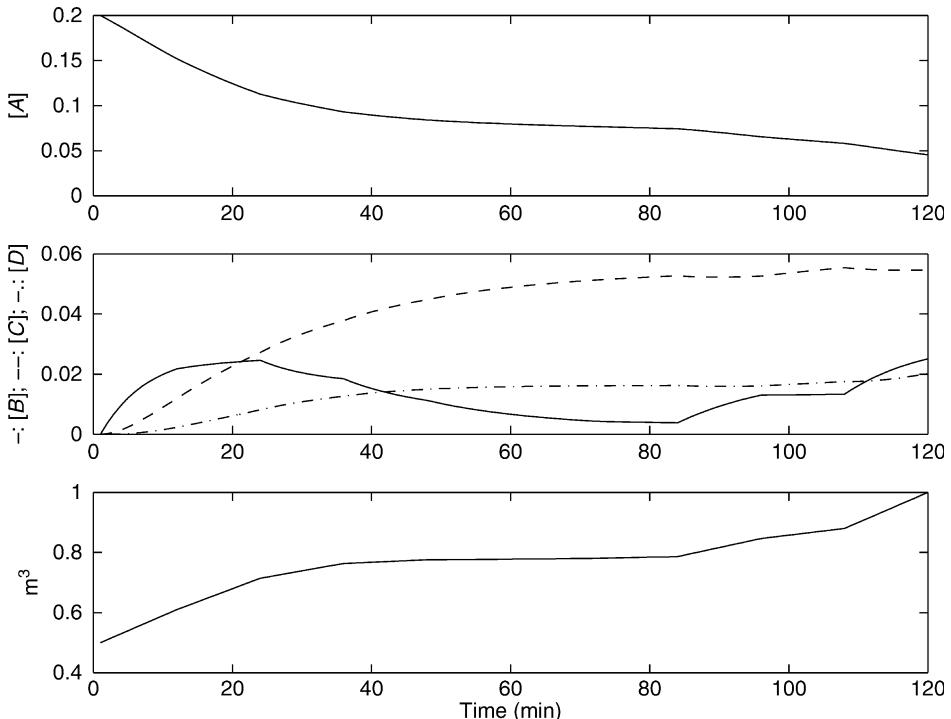


Fig. 24

Trajectories of process variables under optimal control in Case 1.

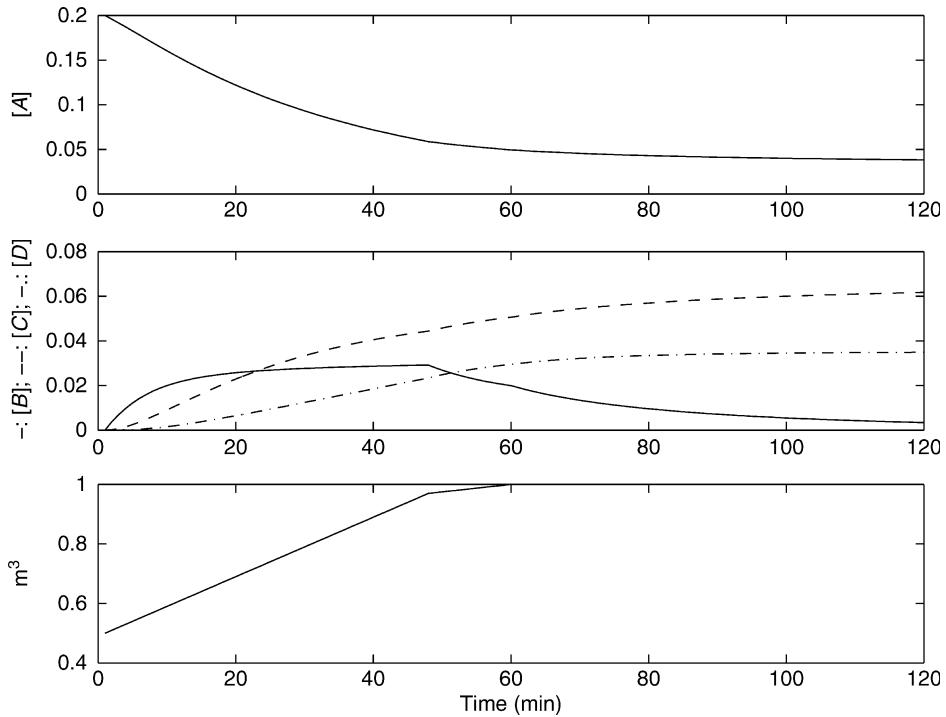


profiles given in [Fig. 23](#) are easy to comprehend. Case 1 emphasizes on less by-product and [Table 3](#) shows that the gain for by-product is high during the reaction volume medium stage, thus the control profile exhibits low feed rate during the reaction volume medium stage to limit the growth of by-product. Whereas Case 2 emphasizes on more product and [Table 3](#) indicates that the gain for product at low reaction volume is high, thus the control profile for Case 1 has maximum feed rate at the beginning to ensure a large amount of product being produced through fast growth of reaction volume during the reaction volume low stage as can be seen in [Fig. 25](#).

Terwiesch et al. (1998) reported a nominal control policy and a minimal risk control policy for this fed-batch process based on the mechanistic model. [Figure 26](#) shows these mechanistic model-based optimal control policies whereas [Figure 27](#) shows the trajectories of $[C]$ and $[D]$ under these control policies. [Table 4](#) gives the values of $[C](t_f)$ and $[D](t_f)$ under different control policies. It can be seen that the recurrent neurofuzzy network model-based optimal control policies give comparable performance as the mechanistic model-based optimal control policies. Results from the recurrent neurofuzzy network model-based optimal control policy (Case 2) is very close to those from mechanistic model-based optimal control policy (nominal). Although the final product concentration $[C](t_f)$ under the recurrent neurofuzzy network model-based optimal control policy (Case 1) is lower than those under the mechanistic model-based optimal control policies, the final concentration of the undesired species $[D](t_f)$ under the recurrent neurofuzzy network model-based optimal control policy (Case 1) is also lower than those under the mechanistic model-based optimal control policies.

Fig. 25

Trajectories of process variables under optimal control in Case 2.

**Fig. 26**

Mechanistic model-based nominal and minimal risk control policies.

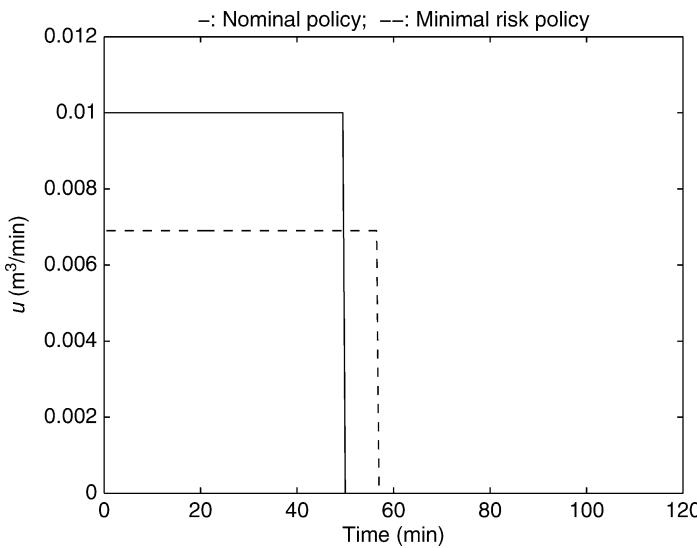
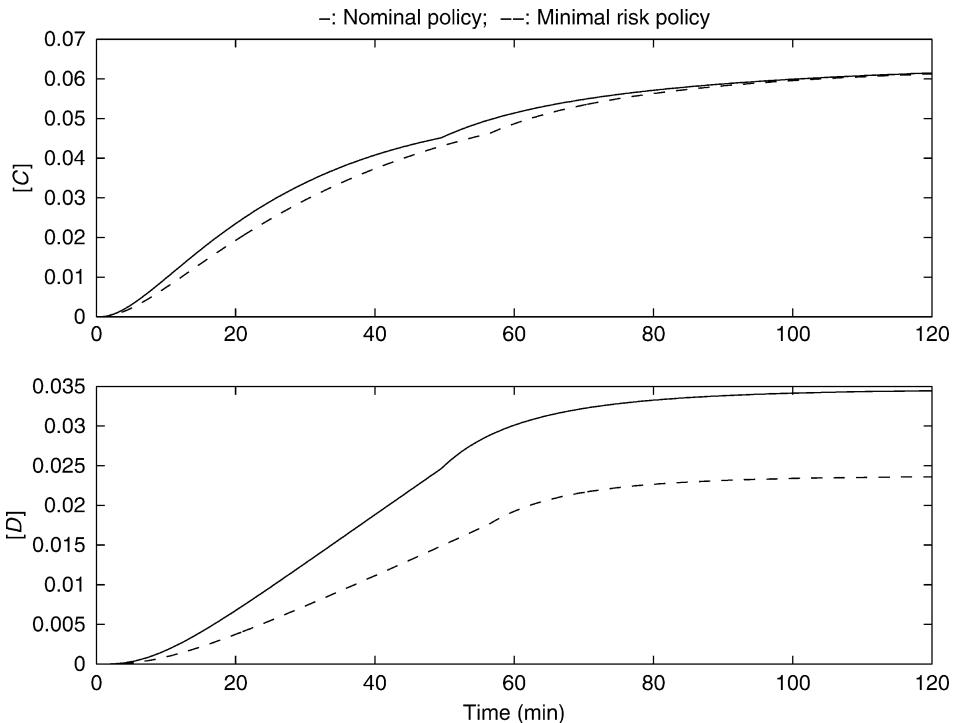


Fig. 27

Trajectories of $[C]$ and $[D]$ under the nominal and minimal risk control policies.

**Table 4**

Values of $[C](t_f)$ and $[D](t_f)$ under different control policies

Control policies	$[C](t_f)$	$[D](t_f)$
Recurrent neurofuzzy network – Case 1	0.0547	0.0202
Recurrent neurofuzzy network – Case 2	0.0617	0.0349
Nominal (Terwiesch et al. 1998)	0.0615	0.0345
Minimal risk (Terwiesch et al. 1998)	0.0612	0.0236

6 Conclusions

Recurrent neurofuzzy networks are very effective in building long-range prediction models for nonlinear processes from process operational data. A recurrent neurofuzzy network can offer accurate long-term predictions which are crucial to the success of many advanced process-control techniques such as nonlinear long-range predictive control and optimal batch process control. Neurofuzzy network models are easier to interpret than conventional neural network models. The weights in the fuzzification layer determine the membership functions of the fuzzy operating regions while the weights in the function layer determine the

local models in these operating regions. The neurofuzzy network-based approaches are therefore not purely "black box" approaches which are difficult to interpret.

Neurofuzzy network process models provide a basis for building a novel type of nonlinear controller which is composed of several local linear controllers. The global controller output is obtained by combining local controller outputs based upon their membership functions. A neurofuzzy network model-based predictive controller can be obtained by combining several local linear model-based predictive controllers which have analytical solutions. The advantage of this technique is that both the time-consuming numerical optimization methods and the uncertainty in convergence to the global optimum, which are typically seen in conventional nonlinear model-based predictive control, are avoided. Furthermore, control actions obtained, based on local incremental models, contain integration actions which can naturally eliminate static offsets. In most of the reported nonlinear model-based controllers, however, static offsets are usually eliminated using some ad hoc methods, such as by coupling a PI controller.

Applications to the modeling and control of a neutralization process and a fed-batch reactor demonstrate that the recurrent neurofuzzy networks are capable of providing accurate long-range predictions. The developed models are easy to comprehend and local dynamic characteristics of the modeled processes can easily be obtained.

References

- Akaike H (1974) A new look at the statistical model identification. *IEEE Trans Automatic Control* 19:716–723
- Astrom KJ, Wittenmark B (1989) Adaptive control, 2nd edn. Addison-Wesley, Reading, MA
- Barron AR (1984) Predicted squared error: a criterion for automatic model selection. In: Farlow SJ (ed) Self organising methods. Marcel Dekker, New York, pp 87–103
- Bhat NV, McAvoy TJ (1990) Use of neural nets for dynamical modelling and control of chemical process systems. *Comput Chem Eng* 14:573–583
- Billings SA, Voon WSF (1986) Correlation based model validity tests for non-linear models. *Int J Control* 44:235–244
- Blanco A, Delgado M, Pegalajar MC (2001a) A real-coded genetic algorithm for training recurrent neural networks. *Neural Netw* 14:93–105
- Blanco A, Delgado M, Pegalajar MC (2001b) Fuzzy automaton induction using neural networks. *Int J Approx Reasoning* 27:1–26
- Brown M, Harris CJ (1994) Neurofuzzy adaptive modeling and control. Prentice Hall, Hemel Hempstead
- Bulsari AB (ed) (1995) Computer-aided chemical engineering. Neural networks for chemical engineers, vol 6. Elsevier, Amsterdam
- Clarke DW, Mohtadi C, Tuffs PS (1987) Generalised predictive control, Parts 1 and 2. *Automatica* 23:859–875
- Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Math Control Signal Syst* 2:303–314
- Eaton JW, Rawlings JB (1990) Feedback control of chemical processes using on-line optimization techniques. *Comput Chem Eng* 14:469–479
- Elman JL (1990) Finding structures in time. *Cogn Sci* 14:179–211
- Frasconi P, Gori M, Soda G (1992) Local feedback multi-layered networks. *Neural Comput* 4:120–130
- Geladi P, Kowalski BR (1986) Partial least-squares regression: a tutorial. *Anal Chim Acta* 185:1–17
- Gembicki FW (1974) Vector optimisation for control with performance and parameter sensitivity indices. Ph.D. Thesis, Case Western Reserve University
- Girosi F, Poggio T (1990) Networks and the best approximation property. *Biol Cybern* 63:169–179
- Harris CJ, Brown M, Bossley KM, Mills DJ, Ming F (1996) Advances in neurofuzzy algorithms for real-time modelling and control. *Eng Appl Artif Intell* 9:1–16
- Hernandez E, Arkun Y (1993) Control of nonlinear systems using polynomial ARMA models. *AICHE J* 39:446–460
- Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* 12:55–67
- Horikawa S, Furuhashi T, Uchikawa Y (1992) On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Trans Neural Netw* 3:801–806
- Jang JSR (1992) Self-learning fuzzy controllers based on temporal back propagation. *IEEE Trans Neural Netw* 3:714–723

- Jang JSR, Sun CT (1995) Neuro-fuzzy modeling and control. *Proc IEEE* 83:378–406
- Jang JSR, Sun CT, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Englewood Cliffs, NJ
- Johansen TA, Foss BA (1993) Constructing NARMAX models using ARMAX models. *Int J Control* 58:1125–1153
- Kuipers B, Astrom K (1994) The composition and validation of heterogeneous control laws. *Automatica* 30:233–249
- Leonard JA, Kramer MA (1990) Improvement of the back-propagation algorithm for training neural networks. *Comput Chem Eng* 14:337–341
- Leontaris GJ, Billings SA (1987) Model selection and validation methods for nonlinear systems. *Int J Control* 45:311–341
- Li WC, Biegler LT (1989) Multistep, Newton-type control strategies for constrained, nonlinear processes. *Chem Eng Res Des* 67:562–577
- Mak MW, Ku KW, Lu YL (1999) On the improvement of the real time recurrent learning algorithm for recurrent neural networks. *Neurocomputing* 24:13–36
- Maner BR, Doyle FJ III (1997) Polymerization reactor control using autoregressive-plus Volterra-based MPC. *AICHE J* 43:1763–1784
- Marquardt D (1963) An algorithm for least squares estimation of nonlinear parameters. *SIAM J Appl Math* 11:431–441
- McAvoy TJ, Hsu E, Lowenthal S (1972) Dynamics of pH in controlled stirred tank reactor. *Ind Eng Chem Process Des Dev* 11:68–70
- Morris AJ, Montague GA, Willis MJ (1994) Artificial neural networks: studies in process modelling and control. *Chem Eng Res Des* 72:3–19
- Nie J, Linkens DA (1993) Learning control using fuzzified self-organising radial basis function networks. *IEEE Trans Fuzzy Syst* 1:280–287
- Omlin CW, Thorner KK, Giles CL (1998) Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks. *IEEE Trans Fuzzy Syst* 6:76–89
- Park J, Sandberg IW (1991) Universal approximation using radial basis function networks. *Neural Comput* 3:246–257
- Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing. MIT Press, Cambridge, MA
- Saint-Donat J, Bhat N, McAvoy TJ (1991) Neural net based model predictive control. *Int J Control* 54:1452–1468
- Scott GM, Ray WH (1993) Creating efficient nonlinear network process models that allow model interpretation. *J Process Control* 3:163–178
- Sistu PB, Gopinath RS, Bequette BW (1993) Computational issues in nonlinear predictive control. *Comput Chem Eng* 17:361–366
- Sjoberg J, Zhang Q, Ljung L, Benveniste A, Delyon B, Glrenne P, Hjalmarsson H, Juditsky A (1995) Nonlinear black-box modelling in system identification: a unified overview. *Automatica* 31: 1691–1724
- Su HT, McAvoy TJ, Werbos P (1992) Long term prediction of chemical processes using recurrent neural networks: a parallel training approach. *Ind Eng Chem Res* 31:1338–1352
- Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modelling and control. *IEEE Trans Syst Man Cybern* 15:116–132
- Terwiesch P, Ravemark D, Schenker B, Rippin DWT (1998) Semi-batch process optimization under uncertainty: theory and experiments. *Comput Chem Eng* 22:201–213
- Tian Y, Zhang J, Morris AJ (2002) Optimal control of a batch emulsion copolymerisation reactor based on recurrent neural network models. *Chem Eng Process* 41:531–538
- Tsoi AC, Back AD (1994) Locally recurrent globally feedforward networks: a critical review of architectures. *IEEE Trans Neural Netw* 5:229–239
- Wang LX (1994) Adaptive fuzzy systems and control: design and stability analysis. Prentice Hall, Englewood Cliffs, NJ
- Werbos PJ (1990) Backpropagation through time: what it does and how to do it. *Proc IEEE* 78:1550–1560
- Yager RR, Filev DP (1994) Essentials of fuzzy modelling and control. Wiley, New York
- Zhang J (2004) A reliable neural network model based optimal control strategy for a batch polymerisation reactor. *Ind Eng Chem Res* 43:1030–1038
- Zhang J (2005) Modelling and optimal control of batch processes using recurrent neuro-fuzzy networks. *IEEE Trans Fuzzy Syst* 13:417–427
- Zhang J (2006) Modelling and multi-objective optimal control of batch processes using recurrent neuro-fuzzy networks. *Int J Automation Comput* 3:1–7
- Zhang J, Morris AJ (1994) On-line process fault diagnosis using fuzzy neural networks. *Intell Syst Eng* 3:37–47
- Zhang J, Morris AJ (1995) Fuzzy neural networks for nonlinear systems modelling. *IEE Proc, Control Theory Appl* 142:551–561
- Zhang J, Morris AJ (1999) Recurrent neuro-fuzzy networks for nonlinear process modelling. *IEEE Trans Neural Netw* 10:313–326
- Zhang J, Morris AJ, Martin EB (1998) Long term prediction models based on mixed order locally recurrent neural networks. *Comput Chem Eng* 22: 1051–1063

13 Independent Component Analysis

Seungjin Choi

Pohang University of Science and Technology, Pohang, South Korea

seungjin@postech.ac.kr

1	<i>Introduction</i>	436
2	<i>Why Independent Components?</i>	437
3	<i>Principles</i>	439
4	<i>Natural Gradient Algorithm</i>	443
5	<i>Flexible ICA</i>	444
6	<i>Differential ICA</i>	447
7	<i>Nonstationary Source Separation</i>	449
8	<i>Spatial, Temporal, and Spatiotemporal ICA</i>	451
9	<i>Algebraic Methods for BSS</i>	452
10	<i>Software</i>	456
11	<i>Further Issues</i>	456
12	<i>Summary</i>	457

Abstract

Independent component analysis (ICA) is a statistical method, the goal of which is to decompose multivariate data into a linear sum of non-orthogonal basis vectors with coefficients (encoding variables, latent variables, and hidden variables) being statistically independent. ICA generalizes widely used subspace analysis methods such as principal component analysis (PCA) and factor analysis, allowing latent variables to be non-Gaussian and basis vectors to be non-orthogonal in general. ICA is a density-estimation method where a linear model is learned such that the probability distribution of the observed data is best captured, while factor analysis aims at best modeling the covariance structure of the observed data. We begin with a fundamental theory and present various principles and algorithms for ICA.

1 Introduction

Independent component analysis (ICA) is a widely used multivariate data analysis method that plays an important role in various applications such as pattern recognition, medical image analysis, bioinformatics, digital communications, computational neuroscience, and so on. ICA seeks a decomposition of multivariate data into a linear sum of non-orthogonal basis vectors with coefficients being statistically as independent as possible.

We consider a linear generative model where m -dimensional observed data $x \in \mathbb{R}^m$ is assumed to be generated by a linear combination of n basis vectors $\{a_i \in \mathbb{R}^m\}$,

$$x = a_1 s_1 + a_2 s_2 + \cdots + a_n s_n \quad (1)$$

where $\{s_i \in \mathbb{R}\}$ are *encoding variables*, representing the extent to which each basis vector is used to reconstruct the data vector. Given N samples, the model (Eq. 1) can be written in a compact form

$$X = AS \quad (2)$$

where $X = [x(1), \dots, x(N)] \in \mathbb{R}^{m \times N}$ is a data matrix, $A = [a_1, \dots, a_n] \in \mathbb{R}^{m \times n}$ is a basis matrix, and $S = [s(1), \dots, s(N)] \in \mathbb{R}^{n \times N}$ is an encoding matrix with $s(t) = [s_1(t), \dots, s_n(t)]^\top$.

Dual interpretation of basis encoding in the model (Eq. 2) is given as follows:

- When columns in X are treated as data points in m -dimensional space, columns in A are considered as *basis vectors* and each column in S is *encoding* that represents the extent to which each basis vector is used to reconstruct data vector.
- Alternatively, when rows in X are data points in N -dimensional space, rows in S correspond to basis vectors and each row in A represents encoding.

A strong application of ICA is a problem of *blind source separation* (BSS), the goal of which is to restore *sources* S (associated with encodings) without the knowledge of A , given the data matrix X . ICA and BSS have often been treated as an identical problem since they are closely related to each other. In BSS, the matrix A is referred to as a *mixing matrix*. In practice, we find a linear transformation W , referred to as a *demixing matrix*, such that the rows of the output matrix

$$Y = WX \quad (3)$$

are statistically independent. Assume that sources (rows of S) are statistically independent. In such a case, it is well known that WA becomes a *transparent transformation* when the rows of Y are statistically independent. The transparent transformation is given by $WA = PA$, where P is a permutation matrix and A is a nonsingular diagonal matrix involving scaling. This transparent transformation reflects two indeterminacies in ICA (Comon 1994): (1) scaling ambiguity; (2) permutation ambiguity. In other words, entries of Y correspond to scaled and permuted entries of S .

Since Jutten and Herault's first solution (Jutten and Herault 1991) to ICA, various methods have been developed so far, including a neural network approach (Cichocki and Unbehauen 1996), information maximization (Bell and Sejnowski 1995), natural gradient (or relative gradient) learning (Amari et al. 1996; Cardoso and Laheld 1996; Amari and Cichocki 1998), maximum likelihood estimation (Pham 1996; MacKay 1996; Pearlmutter and Parra 1997; Cardoso 1997), and nonlinear principal component analysis (PCA) (Karhunen 1996; Oja 1995; Hyvärinen and Oja 1997). Several books on ICA (Lee 1998; Hyvärinen et al. 2001; Haykin 2000; Cichocki and Amari 2002; Stone 1999) are available, serving as a good resource for a thorough review and tutorial on ICA. In addition, tutorial papers on ICA (Hyvärinen 1999; Choi et al. 2005) are useful resources.

This chapter begins with a fundamental idea, emphasizing why independent components are sought. Then, well-known principles to tackle ICA are introduced, leading to an objective function to be optimized. The natural gradient algorithm for ICA is explained. We also elucidate how we incorporate nonstationarity or temporal information into the standard ICA framework.

2 Why Independent Components?

PCA is a popular subspace analysis method that has been used for dimensionality reduction and feature extraction. Given a data matrix $X \in \mathbb{R}^{m \times N}$, the covariance matrix R_{xx} is computed by

$$R_{xx} = \frac{1}{N} X H X^\top$$

where $H = I_{N \times N} - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^\top$ is the *centering matrix*, where $I_{N \times N}$ is the $N \times N$ identity matrix and $\mathbf{1}_N = [1, \dots, 1]^\top \in \mathbb{R}^N$. The rank- n approximation of the covariance matrix R_{xx} is of the form

$$R_{xx} \approx U \Lambda U^\top$$

where $U \in \mathbb{R}^{m \times n}$ contains n eigenvectors associated with n largest eigenvalues of R_{xx} in its columns and the corresponding eigenvalues are in the diagonal entries of Λ (diagonal matrix). Then, principal components $z(t)$ are determined by projecting data points $x(t)$ onto these eigenvectors, leading to

$$z(t) = U^\top x(t)$$

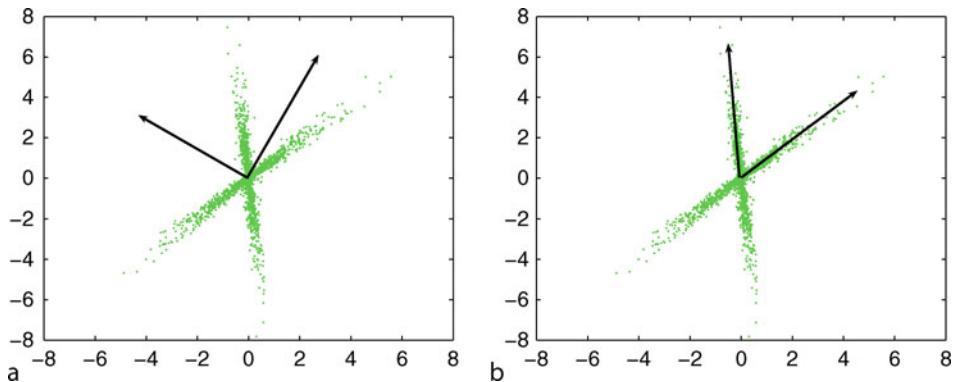
or in a compact form

$$Z = U^\top X$$

It is well known that rows of Z are uncorrelated with each other.

Fig. 1

Two-dimensional data with two main arms are fitted by two different basis vectors: (a) principal component analysis (PCA) makes the implicit assumption that the data have a Gaussian distribution and determines the optimal basis vectors that are orthogonal, which are not efficient at representing non-orthogonal distributions; (b) independent component analysis (ICA) does not require that the basis vectors be orthogonal and considers non-Gaussian distributions, which is more suitable in fitting more general types of distributions.



ICA generalizes PCA in the sense that latent variables (components) are non-Gaussian and A is allowed to be a non-orthogonal transformation, whereas PCA considers only orthogonal transformation and implicitly assumes Gaussian components. **Figure 1** shows a simple example, emphasizing the main difference between PCA and ICA.

A core theorem that plays an important role in ICA is presented. It provides a fundamental principle for various unsupervised learning algorithms for ICA and BSS.

Theorem 1 (Skitovich–Darmois) *Let $\{s_1, s_2, \dots, s_n\}$ be a set of independent random variables. Consider two random variables x_1 and x_2 which are linear combinations of $\{s_i\}$,*

$$\begin{aligned} y_1 &= \alpha_1 s_1 + \dots + \alpha_n s_n \\ y_2 &= \beta_1 s_1 + \dots + \beta_n s_n \end{aligned} \tag{4}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are real constants. If y_1 and y_2 are statistically independent, then each variable s_i for which $\alpha_i \beta_i \neq 0$ is Gaussian.

Consider the linear model (**Eq. 2**) for $m = n$. Throughout this chapter, the simplest case, where $m = n$ (square mixing), is considered. The global transformation can be defined as $G = WA$, where A is the mixing matrix and W is the demixing matrix. With this definition, the output $y(t)$ can be written as

$$y(t) = Wx(t) = Gs(t) \tag{5}$$

It is assumed that both A and W are nonsingular, hence G is nonsingular. Under this assumption, one can easily see that if $\{y_i(t)\}$ are mutually independent non-Gaussian signals, then invoking **Theorem 1**, G has the following decomposition

$$G = PA \tag{6}$$

This explains how ICA performs BSS.

3 Principles

The task of ICA is to estimate the mixing matrix A or its inverse $W = A^{-1}$ (referred to as the *demixing matrix*) such that elements of the estimate $y = A^{-1}x = Wx$ are as independent as possible. For the sake of simplicity, the index t is often left out if the time structure does not have to be considered. In this section, four different principles are reviewed: (1) maximum likelihood estimation; (2) mutual information minimization; (3) information maximization; and (4) negentropy maximization.

3.1 Maximum Likelihood Estimation

Suppose that sources s are independent with marginal distributions $q_i(s_i)$

$$q(s) = \prod_{i=1}^n q_i(s_i) \quad (7)$$

In the linear model, $x = As$, a single factor in the likelihood function is given by

$$\begin{aligned} p(x|A, q) &= \int p(x|s, A)q(s)ds \\ &= \int \prod_{j=1}^n \delta\left(x_j - \sum_{i=1}^n A_{ji}s_i\right) \prod_{i=1}^n q_i(s_i)ds \end{aligned} \quad (8)$$

$$= |\det A|^{-1} \prod_{i=1}^n q_i\left(\sum_{j=1}^n A_{ij}^{-1}x_j\right) \quad (9)$$

Then, we have

$$p(x|A, q) = |\det A|^{-1} r(A^{-1}x) \quad (10)$$

The log-likelihood is written as

$$\log p(x|A, q) = -\log |\det A| + \log q(A^{-1}x) \quad (11)$$

which can be also written as

$$\log p(x|W, q) = \log |\det W| + \log p(y) \quad (12)$$

where $W = A^{-1}$ and y is the estimate of s with the true distribution $q(\cdot)$ replaced by a hypothesized distribution $p(\cdot)$. Since sources are assumed to be statistically independent, (❷ Eq. 12) is written as

$$\log p(x|W, q) = \log |\det W| + \sum_{i=1}^n \log p_i(y_i) \quad (13)$$

The demixing matrix W is determined by

$$\widehat{W} = \arg \max_W \left\{ \log |\det W| + \sum_{i=1}^n \log p_i(y_i) \right\} \quad (14)$$

It is well known that maximum likelihood estimation is equivalent to Kullback matching where the optimal model is estimated by minimizing the Kullback–Leibler (KL) divergence between the empirical distribution and the model distribution. Consider KL divergence from the empirical distribution $\tilde{p}(x)$ to the model distribution $p_\theta(x) = p(x|A, q)$

$$KL[\tilde{p}(x)||p_\theta(x)] = \int \tilde{p}(x) \log \frac{\tilde{p}(x)}{p_\theta(x)} dx = -H(\tilde{p}) - \int \tilde{p}(x) \log p_\theta(x) dx \quad (15)$$

where $H(\tilde{p}) = -\int \tilde{p}(x) \log \tilde{p}(x) dx$ is the entropy of \tilde{p} . Given a set of data points, $\{x_1, \dots, x_N\}$ drawn from the underlying distribution $p(x)$, the empirical distribution $\tilde{p}(x)$ puts probability $\frac{1}{N}$ on each data point, leading to

$$\tilde{p}(x) = \frac{1}{N} \sum_{t=1}^N \delta(x - x_t) \quad (16)$$

It follows from (Eq. 15) that

$$\arg \min_{\theta} KL[\tilde{p}(x)||p_\theta(x)] \equiv \arg \max_{\theta} \langle \log p_\theta(x) \rangle_{\tilde{p}} \quad (17)$$

where $\langle \cdot \rangle_{\tilde{p}}$ represents the expectation with respect to the distribution \tilde{p} . Plugging (Eq. 16) into the right-hand side of (Eq. 15), leads to

$$\langle \log p_\theta(x) \rangle_{\tilde{p}} = \frac{1}{N} \int \sum_{t=1}^N N \delta(x - x_t) \log p_\theta(x) dx = \frac{1}{N} \sum_{t=1}^N \log p_\theta(x_t) \quad (18)$$

Apart from the scaling factor $\frac{1}{N}$, this is just the log-likelihood function. In other words, maximum likelihood estimation is obtained from the minimization of (Eq. 15).

3.2 Mutual Information Minimization

Mutual information is a measure for statistical independence. Demixing matrix W is learned such that the mutual information of $y = Wx$ is minimized, leading to the following objective function:

$$\begin{aligned} \mathcal{J}_{mi} &= \int p(y) \log \left[\frac{p(y)}{\prod_{i=1}^n p_i(y_i)} \right] dy \\ &= -H(y) - \left\langle \sum_{i=1}^n \log p_i(y_i) \right\rangle_y \end{aligned} \quad (19)$$

where $H(\cdot)$ represents the entropy, that is,

$$H(y) = - \int p(y) \log p(y) dy \quad (20)$$

and $\langle \cdot \rangle_y$ denotes the statistical average with respect to the distribution $p(y)$. Note that $p(y) = \frac{p(x)}{|\det W|}$. Thus, the objective function (Eq. 19) is given by

$$\mathcal{J}_{mi} = -\log |\det W| - \sum_{i=1}^n \langle \log p_i(y_i) \rangle \quad (21)$$

where $\langle \log p(x) \rangle$ is left out since it does not depend on parameters W . For online learning, only instantaneous value is taken into consideration, leading to

$$\mathcal{J}_{mi} = -\log |\det W| - \sum_{i=1}^n \log p_i(y_i) \quad (22)$$

3.3 Information Maximization

Infomax (Bell and Sejnowski 1995) involves the maximization of the output entropy $z = g(y)$ where $y = Wx$ and $g(\cdot)$ is a squashing function (e.g., $g_i(y_i) = \frac{1}{1+e^{-y_i}}$). It was shown that Infomax contrast maximization is equivalent to the minimization of KL divergence between the distribution of $y = Wx$ and the distribution $p(s) = \prod_{i=1}^n p_i(s_i)$. In fact, Infomax is nothing but mutual information minimization in the CA framework.

The Infomax contrast function is given by

$$\mathcal{J}_I(W) = H(g(Wx)) \quad (23)$$

where $g(y) = [g_1(y_1), \dots, g_n(y_n)]^\top$. If $g_i(\cdot)$ is differentiable, then it is the cumulative distribution function of some probability density function $q_i(\cdot)$,

$$g_i(y_i) = \int_{-\infty}^{y_i} q_i(s_i) ds_i$$

Let us choose a squashing function $g_i(y_i)$

$$g_i(y_i) = \frac{1}{1 + e^{-y_i}} \quad (24)$$

where $g_i(\cdot) : \mathbb{R} \rightarrow (0, 1)$ is a monotonically increasing function.

Let us consider an n -dimensional random vector \hat{s} , the joint distribution of which is factored into the product of marginal distributions:

$$q(\hat{s}) = \prod_{i=1}^n q_i(\hat{s}_i) \quad (25)$$

Then $g_i(\hat{s}_i)$ is distributed uniformly on $(0, 1)$, since $g_i(\cdot)$ is the cumulative distribution function of \hat{s}_i . Define $u = g(\hat{s}) = [g_1(\hat{s}_1), \dots, g_n(\hat{s}_n)]^\top$, which is distributed uniformly on $(0, 1)^n$.

Define $v = g(Wx)$. Then, the Infomax contrast function is rewritten as

$$\begin{aligned} \mathcal{J}_I(W) &= H(g(Wx)) \\ &= H(v) \\ &= - \int p(v) \log p(v) dv \\ &= - \int p(v) \log \left(\frac{p(v)}{\prod_{i=1}^n 1_{(0,1)}(v_i)} \right) dv \\ &= -KL[v \| u] \\ &= -KL[g(Wx) \| u] \end{aligned} \quad (26)$$

where $1_{(0,1)}(\cdot)$ denotes a uniform distribution on $(0,1)$. Note that KL divergence is invariant under an invertible transformation f ,

$$\begin{aligned} KL[f(u)\|f(v)] &= KL[u\|v] \\ &= KL[f^{-1}(u)\|f^{-1}(v)] \end{aligned}$$

Therefore, we have

$$\begin{aligned} \mathcal{J}_I(W) &= -KL[g(Wx)\|u] \\ &= -KL[Wx\|g^{-1}(u)] \\ &= -KL[Wx\|\hat{s}] \end{aligned} \tag{27}$$

It follows from (Eq. 27) that maximizing $\mathcal{J}_I(W)$ (Infomax principle) is identical to the minimization of the KL divergence between the distribution of the output vector $y = Wx$ and the distribution \hat{s} whose entries are statistically independent. In other words, Infomax is equivalent to mutual information minimization in the framework of ICA.

3.4 Negentropy Maximization

Negative entropy or negentropy is a measure of distance to Gaussianity, yielding a larger value for a random variable whose distribution is far from Gaussian. Negentropy is always nonnegative and vanishes if and only if the random variable is Gaussian. Negentropy is defined as

$$J(y) = H(y^G) - H(y) \tag{28}$$

where $H(y) = \mathbb{E}\{-\log p(y)\}$ represents the entropy and y^G is a Gaussian random vector whose mean vector and covariance matrix are the same as y . In fact, negentropy is the KL divergence of $p(y^G)$ from $p(y)$, that is

$$\begin{aligned} J(y) &= KL[p(y)\|p(y^G)] \\ &= \int p(y) \log \frac{p(y)}{p(y^G)} dy \end{aligned} \tag{29}$$

leading to (Eq. 28).

Let us discover a relation between negentropy and mutual information. To this end, we consider mutual information $I(y)$:

$$\begin{aligned} I(y) &= I(y_1, \dots, y_n) \\ &= \sum_{i=1}^n H(y_i) - H(y) \\ &= \sum_{i=1}^n H(y_i^G) - \sum_{i=1}^n J(y_i) + J(y) - H(y^G) \\ &= J(y) - \sum_{i=1}^n J(y_i) + \frac{1}{2} \log \left[\frac{\prod_{i=1}^n [R_{yy}]_{ii}}{\det R_{yy}} \right] \end{aligned} \tag{30}$$

where $R_{yy} = \mathbb{E}\{yy^\top\}$ and $[R_{yy}]_{ii}$ denotes the i th diagonal entry of R_{yy} .

Assume that y is already whitened (decorrelated), that is, $R_{yy} = I$. Then the sum of marginal negentropies is given by

$$\begin{aligned}\sum_{i=1}^n J(y_i) &= J(y) - I(y) + \underbrace{\frac{1}{2} \log \left[\frac{\prod_{i=1}^n [R_{yy}]_{ii}}{\det R_{yy}} \right]}_0 \\ &= -H(y) - \int p(y) \log p(y^G) dy - I(y) \\ &= -H(x) - \log |\det W| - I(y) - \int p(y) \log p(y^G) dy\end{aligned}\tag{31}$$

Invoking $R_{yy} = I$, (Eq. 31) becomes

$$\sum_{i=1}^n J(y_i) = -I(y) - H(x) - \log |\det W| + \frac{1}{2} \log |\det R_{yy}| \tag{32}$$

Note that

$$\frac{1}{2} \log |\det R_{yy}| = \frac{1}{2} \log |\det(WR_{xx}W^\top)| \tag{33}$$

Therefore, we have

$$\sum_{i=1}^n J(y_i) = -I(y) \tag{34}$$

where irrelevant terms are omitted. It follows from (Eq. 34) that maximizing the sum of marginal negentropies is equivalent to minimizing the mutual information.

4 Natural Gradient Algorithm

In Sect. 3, four different principles lead to the same objective function

$$\mathcal{J} = -\log |\det W| - \sum_{i=1}^n \log p_i(y_i) \tag{35}$$

That is, ICA boils down to learning W , which minimizes (Eq. 35),

$$\widehat{W} = \arg \min_W \left\{ -\log |\det W| - \sum_{i=1}^n \log p_i(y_i) \right\} \tag{36}$$

An easy way to solve (Eq. 36) is the gradient descent method, which gives a learning algorithm for W that has the form

$$\begin{aligned}\Delta W &= -\eta \frac{\partial \mathcal{J}}{\partial W} \\ &= -\eta \{ W^{-\top} - \varphi(y)x^\top \}\end{aligned}\tag{37}$$

where $\eta > 0$ is the learning rate and $\varphi(y) = [\varphi_1(y_1), \dots, \varphi_n(y_n)]^\top$ is the negative score function whose i th element $\varphi_i(y_i)$ is given by

$$\varphi_i(y_i) = -\frac{d \log p_i(y_i)}{dy_i} \tag{38}$$

A popular ICA algorithm is based on natural gradient (Amari 1998), which is known to be efficient since the steepest descent direction is used when the parameter space is on a Riemannian manifold. The natural gradient ICA algorithm can be derived (Amari et al. 1996).

Invoking (Eq. 38), we have

$$d \left\{ - \sum_{i=1}^n \log q_i(y_i) \right\} = \sum_{i=1}^n \varphi_i(y_i) dy_i \quad (39)$$

$$= \varphi^\top(y) dy \quad (40)$$

where $\varphi(y) = [\varphi_1(y_1) \dots \varphi_n(y_n)]^\top$ and dy is given in terms of dW as

$$dy = dWW^{-1}y \quad (41)$$

Define a modified coefficient differential dV as

$$dV = dWW^{-1} \quad (42)$$

With this definition, we have

$$d \left\{ - \sum_{i=1}^n \log q_i(y_i) \right\} = \varphi^\top(y) dV y \quad (43)$$

We calculate an infinitesimal increment of $\log |\det W|$, then we have

$$d\{\log |\det W|\} = \text{tr}\{dV\} \quad (44)$$

where $\text{tr}\{\cdot\}$ denotes the trace that adds up all diagonal elements.

Thus, combining (Eqs. 43 and 44) gives

$$d\mathcal{J} = \varphi^\top(y) dV y - \text{tr}\{dV\} \quad (45)$$

The differential in (Eq. 45) is in terms of the modified coefficient differential matrix dV . Note that dV is a linear combination of the coefficient differentials dW_{ij} . Thus, as long as dW is nonsingular, dV represents a valid search direction to minimize (Eq. 35), because dV spans the same tangent space of matrices as spanned by dW . This leads to a stochastic gradient learning algorithm for V given by

$$\begin{aligned} \Delta V &= -\eta \frac{d\mathcal{J}}{dV} \\ &= \eta \{I - \varphi(y)y^\top\} \end{aligned} \quad (46)$$

Thus the learning algorithm for updating W is described by

$$\begin{aligned} \Delta W &= \eta \Delta VW \\ &= \eta \{I - \varphi(y)y^\top\} W \end{aligned} \quad (47)$$

5 Flexible ICA

The optimal nonlinear function $\varphi_i(y_i)$ is given by (Eq. 38). However, it requires knowledge of the probability distributions of sources that are not available. A variety of hypothesized density models has been used. For example, for super-Gaussian source signals, the unimodal or hyperbolic-Cauchy distribution model (MacKay 1996) leads to the nonlinear function given by

$$\varphi_i(y_i) = \tanh(\beta y_i) \quad (48)$$

Such a sigmoid function was also used by Bell and Sejnowski (1995). For sub-Gaussian source signals, the cubic nonlinear function $\varphi_i(y_i) = y_i^3$ has been a favorite choice. For mixtures of sub- and super-Gaussian source signals, according to the estimated kurtosis of the extracted signals, the nonlinear function can be selected from two different choices (Lee et al. 1999).

The flexible ICA (Choi et al. 2000) incorporates the generalized Gaussian density model into the natural gradient ICA algorithm, so that the parameterized nonlinear function provides flexibility in learning. The *generalized Gaussian* probability distribution is a set of distributions parameterized by a positive real number α , which is usually referred to as the *Gaussian exponent* of the distribution. The Gaussian exponent α controls the “peakiness” of the distribution. The probability density function (PDF) for a generalized Gaussian is described by

$$p(y; \alpha) = \frac{\alpha}{2\lambda\Gamma(\frac{1}{\alpha})} e^{-|\frac{y}{\lambda}|^\alpha} \quad (49)$$

where $\Gamma(x)$ is the Gamma function given by

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \quad (50)$$

Note that if $\alpha = 1$, the distribution becomes the standard “Laplacian” distribution. If $\alpha = 2$, the distribution is a standard normal distribution (see Fig. 2).

For a generalized Gaussian distribution, the kurtosis can be expressed in terms of the Gaussian exponent, given by

$$\kappa_\alpha = \frac{\Gamma(\frac{5}{\alpha})\Gamma(\frac{1}{\alpha})}{\Gamma^2(\frac{3}{\alpha})} - 3 \quad (51)$$

Fig. 2

The generalized Gaussian distribution is plotted for several different values of Gaussian exponent, $\alpha = 0.8, 1, 2, 4$.

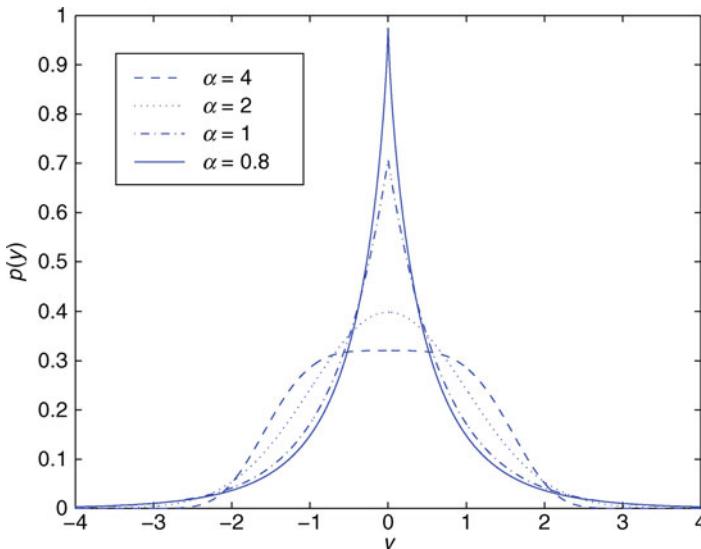
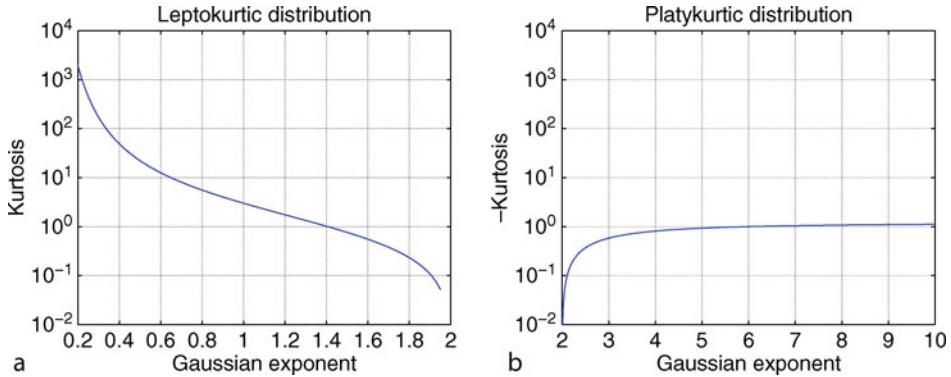


Fig. 3

The plot of kurtosis κ_α versus Gaussian exponent α : (a) for a leptokurtic signal; (b) for a platykurtic signal.



The plots of kurtosis κ_α versus the Gaussian exponent α for leptokurtic and platykurtic signals are shown in [Fig. 3](#).

From the parameterized generalized Gaussian density model, the nonlinear function in the algorithm ([Eq. 47](#)) is given by

$$\begin{aligned}\varphi_i(y_i) &= \frac{d \log p_i(y_i)}{dy_i} \\ &= |y_i|^{\alpha_i-1} \operatorname{sgn}(y_i)\end{aligned}\quad (52)$$

where $\operatorname{sgn}(y_i)$ is the signum function of y_i .

Note that for $\alpha_i = 1$, $\varphi_i(y_i)$ in [Eq. 38](#) becomes a signum function (which can also be derived from the Laplacian density model for sources). The signum nonlinear function is favorable for the separation of speech signals since natural speech is often modeled as a Laplacian distribution. Note also that for $\alpha_i = 4$, $\varphi_i(y_i)$ in [Eq. 38](#) becomes a cubic function, which is known to be a good choice for sub-Gaussian sources.

In order to select a proper value of the Gaussian exponent α_i , we estimate the kurtosis of the output signal y_i and select the corresponding α_i from the relationship in [Fig. 3](#). The kurtosis of y_i , κ_i , can be estimated via the following iterative algorithm:

$$\kappa_i(t+1) = \frac{M_{4i}(t+1)}{M_{2i}^2(t+1)} - 3 \quad (53)$$

where

$$M_{4i}(t+1) = (1 - \delta)M_{4i}(t) + \delta|y_i(t)|^4 \quad (54)$$

$$M_{2i}(t+1) = (1 - \delta)M_{2i}(t) + \delta|y_i(t)|^2 \quad (55)$$

where δ is a small constant, say 0.01.

In general, the estimated kurtosis of the demixing filter output does not exactly match the kurtosis of the original source. However, it provides an idea about whether the estimated source is a sub-Gaussian signal or a super-Gaussian signal. Moreover, it was shown (Cardoso

1997; Amari and Cardoso 1997) that the performance of the source separation is not degraded even if the hypothesized density does not match the true density. For these reasons, we suggest a practical method where only several different forms of nonlinear functions are used.

6 Differential ICA

In a wide sense, most ICA algorithms based on unsupervised learning belong to the Hebb-type rule or its generalization with adopting nonlinear functions. Motivated by the differential Hebb rule (Kosko 1986) and differential decorrelation (Choi 2002, 2003), we introduce an ICA algorithm employing differential learning and natural gradients, which leads to a differential ICA algorithm. A random walk model is first introduced for latent variables, in order to show that differential learning is interpreted as the maximum likelihood estimation of a linear generative model. Then the detailed derivation of the differential ICA algorithm is presented.

6.1 Random Walk Model for Latent Variables

Given a set of observation data, $\{\mathbf{x}(t)\}$, the task of learning the linear generative model (❷ Eq. 1) under the constraint of the latent variables being statistically independent, is a semiparametric estimation problem. The maximum likelihood estimation of basis vectors $\{a_i\}$ involves a probabilistic model for latent variables that are treated as nuisance parameters.

In order to show a link between the differential learning and the maximum likelihood estimation, a random walk model for latent variables $s_i(t)$ is considered, which is a simple Markov chain, that is,

$$s_i(t) = s_i(t - 1) + \varepsilon_i(t) \quad (56)$$

where the innovation $\varepsilon_i(t)$ is assumed to have zero mean with a density function $q_i(\varepsilon_i(t))$. In addition, innovation sequences $\{\varepsilon_i(t)\}$ are assumed to be mutually independent white sequences, that is, they are spatially independent and temporally white as well.

Let us consider latent variables $s_i(t)$ over an N -point time block. The vector \bar{s}_i can be defined as

$$\bar{s}_i = [s_i(0), \dots, s_i(N - 1)]^\top \quad (57)$$

Then the joint PDF of \bar{s}_i can be written as

$$\begin{aligned} p_i(\bar{s}_i) &= p_i(s_i(0), \dots, s_i(N - 1)) \\ &= \prod_{t=0}^{N-1} p_i(s_i(t)|s_i(t - 1)) \end{aligned} \quad (58)$$

where $s_i(t) = 0$ for $t < 0$ and the statistical independence of the innovation sequences is taken into account.

It follows from the random walk model (❷ Eq. 56) that the conditional probability density of $s_i(t)$ given its past samples can be written as

$$p_i(s_i(t)|s_i(t - 1)) = q_i(\varepsilon_i(t)) \quad (59)$$

Combining (Eq. 58) and (Eq. 59) leads to

$$\begin{aligned} p_i(\bar{s}_i) &= \prod_{t=0}^{N-1} q_i(\varepsilon_i(t)) \\ &= \prod_{t=0}^{N-1} q_i(s'_i(t)) \end{aligned} \quad (60)$$

where $s'_i(t) = s_i(t) - s_i(t-1)$, which is the first-order approximation of the differentiation.

Taking the statistical independence of the latent variables and (Eq. 60) into account, then the joint density $p(\bar{s}_1, \dots, \bar{s}_n)$ can be written as

$$\begin{aligned} p(\bar{s}_1, \dots, \bar{s}_n) &= \prod_{i=1}^n p_i(\bar{s}_i) \\ &= \prod_{t=0}^{N-1} \prod_{i=1}^n q_i(s'_i(t)) \end{aligned} \quad (61)$$

The factorial model given in (Eq. 61) will be used as an optimization criterion in deriving the differential ICA algorithm.

6.2 Algorithm

Denote a set of observation data by

$$\mathcal{X} = \{\bar{x}_1, \dots, \bar{x}_n\} \quad (62)$$

where

$$\bar{x}_i = [x_i(0), \dots, x_i(N-1)]^\top \quad (63)$$

Then the normalized log-likelihood is given by

$$\begin{aligned} \frac{1}{N} \log p(\mathcal{X}|A) &= -\log|\det A| + \frac{1}{N} \log p(\bar{s}_1, \dots, \bar{s}_n) \\ &= -\log|\det A| + \frac{1}{N} \sum_{t=0}^{N-1} \sum_{i=1}^n \log q_i(s'_i(t)) \end{aligned} \quad (64)$$

The inverse of A is denoted by $W = A^{-1}$. The estimate of latent variables is denoted by $y(t) = Wx(t)$. With these defined variables, the objective function (i.e., the negative normalized log-likelihood) is given by

$$\begin{aligned} \mathcal{J}_{di} &= -\frac{1}{N} \log p(\mathcal{X}|A) \\ &= -\log|\det W| - \frac{1}{N} \sum_{t=0}^{N-1} \sum_{i=1}^n \log q_i(y'_i(t)) \end{aligned} \quad (65)$$

where s_i is replaced by its estimate y_i and $y'_i(t) = y_i(t) - y_i(t-1)$ (the first-order approximation of the differentiation).

For online learning, the sample average is replaced by the instantaneous value. Hence the online version of the objective function (Eq. 65) is given by

$$\mathcal{J}_{di} = -\log|\det W| - \sum_{i=1}^n \log q_i(y'_i(t)) \quad (66)$$

Note that objective function (Eq. 66) is slightly different from (Eq. 35) used in the conventional ICA based on the minimization of mutual information or the maximum likelihood estimation.

We derive a natural gradient learning algorithm that finds a minimum of (Eq. 66). To this end, we follow the way that was discussed in (Amari et al. 1997; Amari 1998; Choi et al. 2000). The total differential $d\mathcal{J}_{di}(W)$ due to the change dW is calculated

$$\begin{aligned} d\mathcal{J}_{di} &= \mathcal{J}_{di}(W + dW) - \mathcal{J}_{di}(W) \\ &= d\{-\log|\det W|\} + d\left\{-\sum_{i=1}^n \log q_i(y'_i(t))\right\} \end{aligned} \quad (67)$$

Define

$$\varphi_i(y'_i) = -\frac{d \log q_i(y'_i)}{dy'_i} \quad (68)$$

and construct a vector $\varphi(y') = \varphi_1(y'_1), \dots, \varphi_n(y'_n)^\top$.

With this definition, we have

$$\begin{aligned} d\left\{-\sum_{i=1}^n \log q_i(y'_i(t))\right\} &= \sum_{i=1}^n \varphi_i(y'_i(t)) dy'_i(t) \\ &= \varphi^\top(y'(t)) dy'(t) \end{aligned} \quad (69)$$

One can easily see that

$$d\{-\log|\det W|\} = \text{tr}\{dWW^{-1}\} \quad (70)$$

Define a modified differential matrix dV by

$$dV = dWW^{-1} \quad (71)$$

Then, with this modified differential matrix, the total differential $d\mathcal{J}_{di}(W)$ is computed as

$$d\mathcal{J}_{di} = -\text{tr}\{dV\} + \varphi^\top(y'(t)) dV y'(t) \quad (72)$$

A gradient descent learning algorithm for updating V is given by

$$\begin{aligned} V(t+1) &= V(t) - \eta_t \frac{d\mathcal{J}_{di}}{dV} \\ &= \eta_t \{I - \varphi(y'(t)) y'^\top(t)\} \end{aligned} \quad (73)$$

Hence, it follows from the relation (Eq. 71) that the updating rule for W has the form

$$W(t+1) = W(t) + \eta_t \{I - \varphi(y'(t)) y'^\top(t)\} W(t) \quad (74)$$

7 Nonstationary Source Separation

So far, we assumed that sources are stationary random processes where the statistics do not vary over time. In this section, we show how the natural gradient ICA algorithm is modified to handle nonstationary sources. As in Matsuoka et al. (1995), the following assumptions are made in this section.

AS1 The mixing matrix A has full column rank.

AS2 Source signals $\{s_i(t)\}$ are statistically independent with zero mean. This implies that the covariance matrix of the source signal vector, $R_s(t) = E\{s(t)s^\top(t)\}$, is a diagonal matrix, that is,

$$R_s(t) = \text{diag}\{r_1(t), \dots, r_n(t)\} \quad (75)$$

where $r_i(t) = E[s_i^2(t)]$ and E denotes the statistical expectation operator.

AS3 $\frac{r_i(t)}{r_j(t)}$ ($i, j = 1, \dots, n$ and $i \neq j$) are not constant with time.

It should be pointed out that the first two assumptions (AS1, AS2) are common in most existing approaches to source separation, however the third assumption (AS3) is critical in this chapter. For nonstationary sources, the third assumption is satisfied and it allows one to separate linear mixtures of sources via second-order statistics (SOS).

For stationary source separation, the typical cost function is based on the mutual information, which requires knowledge of the underlying distributions of the sources. Since the probability distributions of the sources are not known in advance, most ICA algorithms rely on hypothesized distributions (e.g., see Choi et al. 2000 and references therein). Higher-order statistics (HOS) should be incorporated either explicitly or implicitly.

For nonstationary sources, Matsuoka et al. have shown that the decomposition (Eq. 6) is satisfied if cross-correlations $E\{y_i(t)y_j(t)\}$ ($i, j = 1, \dots, n$, $i \neq j$) are zeros at any time instant t , provided that the assumptions (AS1)–(AS3) are satisfied. To eliminate cross-correlations, the following cost function was proposed in Matsuoka et al. (1995),

$$\mathcal{J}(W) = \frac{1}{2} \left\{ \sum_{i=1}^n \log E\{y_i^2(t)\} - \log \det(E\{y(t)y^\top(t)\}) \right\} \quad (76)$$

where $\det(\cdot)$ denotes the determinant of a matrix. The cost function given in (Eq. 76) is a nonnegative function, which takes minima if and only if $E\{y_i(t)y_j(t)\} = 0$, for $i, j = 1, \dots, n$, $i \neq j$. This is the direct consequence of Hadamard's inequality, which is summarized below.

Theorem 2 (Hadamard's inequality) Suppose $K = [k_{ij}]$ is a nonnegative definite symmetric $n \times n$ matrix. Then,

$$\det(K) \leq \prod_{i=1}^n k_{ii} \quad (77)$$

with equality iff $k_{ij} = 0$, for $i \neq j$.

Take the logarithm on both sides of (Eq. 77) to obtain

$$\sum_{i=1}^n \log k_{ii} - \log \det(K) \geq 0 \quad (78)$$

Replacing the matrix K by $E\{y(t)y^\top(t)\}$, one can easily see that the cost function (Eq. 76) has the minima iff $E\{y_i(t)y_j(t)\} = 0$, for $i, j = 1, \dots, n$ and $i \neq j$.

We compute

$$\begin{aligned} d\{\log \det(E\{y(t)y^\top(t)\})\} &= 2d\{\log \det W\} + d\{\log \det C(t)\} \\ &= 2\text{tr}\{W^{-1}dW\} + d\{\log \det C(t)\} \end{aligned} \quad (79)$$

Define a modified differential matrix dV as

$$dV = W^{-1}dW \quad (80)$$

Then, we have

$$d \left\{ \sum_{i=1}^n \log E\{\gamma_i^2(t)\} \right\} = 2E\{y^\top(t) A^{-1}(t) dV y(t)\} \quad (81)$$

Similarly, we can derive the learning algorithm for W that has the form

$$\begin{aligned} \Delta W(t) &= \eta_t \{I - A^{-1}(t)y(t)y^\top(t)\} W(t) \\ &= \eta_t A^{-1}(t) \{A(t) - y(t)y^\top(t)\} W(t) \end{aligned} \quad (82)$$

8 Spatial, Temporal, and Spatiotemporal ICA

ICA decomposition, $X = AS$, inherently has duality. Considering the data matrix $X \in \mathbb{R}^{m \times N}$ where each of its rows is assumed to be a time course of an attribute, ICA decomposition produces n independent time courses. On the other hand, regarding the data matrix in the form of X^\top , ICA decomposition leads to n independent patterns (for instance, images in fMRI or arrays in DNA microarray data).

The standard ICA (where X is considered) is treated as *temporal ICA* (tICA). Its dual decomposition (regarding X^\top) is known as *spatial ICA* (sICA). Combining these two ideas leads to *spatiotemporal ICA* (stICA). These variations of ICA were first investigated in Stone et al. (2000). Spatial ICA or spatiotemporal ICA were shown to be useful in fMRI image analysis (Stone et al. 2000) and gene expression data analysis (Liebermeister 2002; Kim and Choi 2005).

Suppose that the singular value decomposition (SVD) of X is given by

$$X = U D V^\top = \left(U \ D^{1/2} \right) \left(V \ D^{1/2} \right)^\top = \tilde{U} \tilde{V}^\top \quad (83)$$

where $U \in \mathbb{R}^{m \times n}$, $D \in \mathbb{R}^{n \times n}$, and $V \in \mathbb{R}^{N \times n}$ for $n \leq \min(m, N)$.

8.1 Temporal ICA

Temporal ICA finds a set of independent time courses and a corresponding set of dual unconstrained spatial patterns. It embodies the assumption that each row vector in \tilde{V}^\top consists of a linear combination of n independent sequences, that is, $\tilde{V}^\top = \tilde{A}_T S_T$, where $S_T \in \mathbb{R}^{n \times N}$ has a set of n independent temporal sequences of length N , and $\tilde{A}_T \in \mathbb{R}^{n \times n}$ is an associated mixing matrix.

Unmixing by $Y_T = W_T \tilde{V}^\top$ where $W_T = P \tilde{A}_T^{-1}$, leads one to recover the n dual patterns A_T associated with the n independent time courses, by calculating $A_T = \tilde{U} W_T^{-1}$, which is a consequence of $\tilde{X} = A_T Y_T = \tilde{U} \tilde{V}^\top = \tilde{U} W_T^{-1} Y_T$.

8.2 Spatial ICA

Spatial ICA seeks a set of independent spatial patterns S_S and a corresponding set of dual unconstrained time courses A_S . It embodies the assumption that each row vector in \tilde{U}^\top is

composed of a linear combination of n independent spatial patterns, that is, $\tilde{X}^\top = \tilde{A}_S S_S$, where $S_S \in \mathbb{R}^{n \times m}$ contains a set of n independent m -dimensional patterns and $\tilde{A}_S \in \mathbb{R}^{n \times n}$ is an encoding variable matrix (mixing matrix).

Define $Y_S = W_S \tilde{U}^\top$ where W_S is a permuted version of \tilde{A}_S^{-1} . With this definition, the n dual time courses $A_S \in \mathbb{R}^{N \times n}$ associated with the n independent patterns are computed as $A_S = \tilde{V} W_S^{-1}$, since $\tilde{X}^\top = A_S Y_S = \tilde{U} \tilde{V}^\top = \tilde{V} W_S^{-1} Y_S$. Each column vector of A_S corresponds to a temporal mode.

8.3 Spatiotemporal ICA

In linear decomposition sICA enforces independence constraints over space to find a set of independent spatial patterns, whereas tICA embodies independence constraints over time to seek a set of independent time courses. Spatiotemporal ICA finds a linear decomposition by maximizing the degree of independence over space as well as over time, without necessarily producing independence in either space or time. In fact, it allows a trade-off between the independence of arrays and the independence of time courses.

Given $\tilde{X} = \tilde{U} \tilde{V}^\top$, stICA finds the following decomposition:

$$\tilde{X} = S_S^\top \Lambda S_T \quad (84)$$

where $S_S \in \mathbb{R}^{n \times m}$ contains a set of n independent m -dimensional patterns, $S_T \in \mathbb{R}^{n \times N}$ has a set of n independent temporal sequences of length N , and Λ is a diagonal scaling matrix. There exist two $n \times n$ mixing matrices, W_S and W_T , such that $S_S = W_S \tilde{U}^\top$ and $S_T = W_T \tilde{V}^\top$. The following relation

$$\begin{aligned} \tilde{X} &= S_S^\top \Lambda S_T \\ &= \tilde{U} W_S^\top \Lambda W_T \tilde{V}^\top \\ &= \tilde{U} \tilde{V}^\top \end{aligned} \quad (85)$$

implies that $W_S^\top \Lambda W_T = I$, which leads to

$$W_T = W_S^{-\top} \Lambda^{-1} \quad (86)$$

Linear transforms, W_S and W_T , are found by jointly optimizing objective functions associated with sICA and tICA. That is, the objective function for stICA has the form

$$\mathcal{J}_{\text{stICA}} = \alpha \mathcal{J}_{\text{sICA}} + (1 - \alpha) \mathcal{J}_{\text{tICA}} \quad (87)$$

where $\mathcal{J}_{\text{sICA}}$ and $\mathcal{J}_{\text{tICA}}$ could be Infomax criteria or log-likelihood functions and α defines the relative weighting for spatial independence and temporal independence. More details on stICA can be found in Stone et al. (2002).

9 Algebraic Methods for BSS

Up to now, online ICA algorithms in a framework of unsupervised learning have been introduced. In this section, several algebraic methods are explained for BSS where matrix decomposition plays a critical role.

9.1 Fundamental Principle of Algebraic BSS

Algebraic methods for BSS often make use of the eigen-decomposition of correlation matrices or cumulant matrices. Exemplary algebraic methods for BSS include FOBI (Cardoso 1989), AMUSE (Tong et al. 1990), JADE (Cardoso and Souloumiac 1993), SOBI (Belouchrani et al. 1997), and SEONS (Choi et al. 2002). Some of these methods (FOBI and AMUSE) are based on simultaneous diagonalization of two symmetric matrices. Methods such as JADE, SOBI, and SEONS make use of joint approximate diagonalization of multiple matrices (more than two). The following theorem provides a fundamental principle for algebraic BSS, justifying why simultaneous diagonalization of two symmetric data matrices (one of them is assumed to be positive definite) provides a solution to BSS.

Theorem 3 Let $\Lambda_1, D_1 \in \mathbb{R}^{n \times n}$ be diagonal matrices with positive diagonal entries and $\Lambda_2, D_2 \in \mathbb{R}^{n \times n}$ be diagonal matrices with nonzero diagonal entries. Suppose that $G \in \mathbb{R}^{n \times n}$ satisfies the following decompositions:

$$D_1 = G\Lambda_1 G^\top \quad (88)$$

$$D_2 = G\Lambda_2 G^\top \quad (89)$$

Then the matrix G is the generalized permutation matrix, i.e., $G = PA$ if $D_1^{-1}D_2$ and $\Lambda_1^{-1}\Lambda_2$ have distinct diagonal entries.

Proof It follows from (88) that there exists an orthogonal matrix Q such that

$$\left(G\Lambda_1^{\frac{1}{2}}\right) = \left(D_1^{\frac{1}{2}}\right) Q \quad (90)$$

Hence,

$$G = D_1^{\frac{1}{2}} Q \Lambda_1^{-\frac{1}{2}} \quad (91)$$

Substitute (91) into (89) to obtain

$$D_1^{-1} D_2 = Q \Lambda_1^{-1} \Lambda_2 Q^\top \quad (92)$$

Since the right-hand side of (92) is the eigen-decomposition of the left-hand side of (92), the diagonal elements of $D_1^{-1}D_2$ and $\Lambda_1^{-1}\Lambda_2$ are the same. From the assumption that the diagonal elements of $D_1^{-1}D_2$ and $\Lambda_1^{-1}\Lambda_2$ are distinct, the orthogonal matrix Q must have the form $Q = P\Psi$, where Ψ is a diagonal matrix whose diagonal elements are either +1 or -1. Hence, we have

$$\begin{aligned} G &= D_1^{\frac{1}{2}} P \Psi \Lambda_1^{-\frac{1}{2}} \\ &= P P^\top D_1^{\frac{1}{2}} P \Psi \Lambda_1^{-\frac{1}{2}} \\ &= P A \end{aligned} \quad (93)$$

where

$$A = P^\top D_1^{\frac{1}{2}} P \Psi \Lambda_1^{-\frac{1}{2}}$$

which completes the proof.

9.2 AMUSE

As an example of [Theorem 3](#), we briefly explain AMUSE (Tong et al. 1990), where a BSS solution is determined by simultaneously diagonalizing the equal-time correlation matrix of $x(t)$ and a time-delayed correlation matrix of $x(t)$.

It is assumed that sources $\{s_i(t)\}$ (entries of $s(t)$) are uncorrelated stochastic processes with zero mean, that is,

$$\mathbb{E}\{s_i(t)s_j(t - \tau)\} = \delta_{ij}\gamma_i(\tau) \quad (94)$$

where δ_{ij} is the Kronecker delta and $\gamma_i(\tau)$ are distinct for $i = 1, \dots, n$, given τ . In other words, the equal-time correlation matrix of the source, $R_{ss}(0) = \mathbb{E}\{s(t)s^\top(t)\}$, is a diagonal matrix with distinct diagonal entries. Moreover, a time-delayed correlation matrix of the source, $R_{ss}(\tau) = \mathbb{E}\{s(t)s^\top(t - \tau)\}$, is diagonal as well, with distinct nonzero diagonal entries.

It follows from [Eq. 2](#) that the correlation matrices of the observation vector $x(t)$ satisfy

$$R_{xx}(0) = AR_{ss}(0)A^\top \quad (95)$$

$$R_{xx}(\tau) = AR_{ss}(\tau)A^\top \quad (96)$$

for some nonzero time-lag τ and both $R_{ss}(0)$ and $R_{ss}(\tau)$ are diagonal matrices since sources are assumed to be spatially uncorrelated.

Invoking [Theorem 3](#), one can easily see that the inverse of the mixing matrix, A^{-1} , can be identified up to its rescaled and permuted version by the simultaneous diagonalization of $R_{xx}(0)$ and $R_{xx}(\tau)$, provided that $R_{ss}^{-1}(0)R_{ss}(\tau)$ has distinct diagonal elements. In other words, a linear transformation W is determined such that $R_{yy}(0)$ and $R_{yy}(\tau)$ of the output $y(t) = Wx(t)$ are simultaneously diagonalized:

$$\begin{aligned} R_{yy}(0) &= (WA)R_{ss}(0)(WA)^\top \\ R_{yy}(\tau) &= (WA)R_{ss}(\tau)(WA)^\top \end{aligned}$$

It follows from [Theorem 3](#) that WA becomes the transparent transformation.

9.3 Simultaneous Diagonalization

We explain how two symmetric matrices are simultaneously diagonalized by a linear transformation. More details on simultaneous diagonalization can be found in Fukunaga (1990). Simultaneous diagonalization consists of two steps (whitening followed by a unitary transformation):

1. First, the matrix $R_{xx}(0)$ is whitened by

$$z(t) = D_1^{-\frac{1}{2}}U_1^\top x(t) \quad (97)$$

where D_1 and U_1 are the eigenvalue and eigenvector matrices of $R_{xx}(0)$,

$$R_{xx}(0) = U_1 D_1 U_1^\top \quad (98)$$

Then, we have

$$\begin{aligned} R_{zz}(0) &= D_1^{-\frac{1}{2}}U_1^\top R_{xx}(0)U_1 D_1^{-\frac{1}{2}} = I_m \\ R_{zz}(\tau) &= D_1^{-\frac{1}{2}}U_1^\top R_{xx}(\tau)U_1 D_1^{-\frac{1}{2}} \end{aligned}$$

2. Second, a unitary transformation is applied to diagonalize the matrix $R_{zz}(\tau)$. The eigen-decomposition of $R_{zz}(\tau)$ has the form

$$R_{zz}(\tau) = U_2 D_2 U_2^\top \quad (99)$$

Then, $y(t) = U_2^\top z(t)$ satisfies

$$\begin{aligned} R_{yy}(0) &= U_2^\top R_{zz}(0) U_2 = I_m \\ R_{yy}(\tau) &= U_2^\top R_{zz}(\tau) U_2 = D_2 \end{aligned}$$

Thus, both matrices $R_{xx}(0)$ and $R_{xx}(\tau)$ are simultaneously diagonalized by a linear transform $W = U_2^\top D_1^{-\frac{1}{2}} U_1^\top$. It follows from [Theorem 3](#) that $W = U_2^\top D_1^{-\frac{1}{2}} U_1^\top$ is a valid demixing matrix if all the diagonal elements of D_2 are distinct.

9.4 Generalized Eigenvalue Problem

The simultaneous diagonalization of two symmetric matrices can be carried out without going through two-step procedures. From the discussion in [Sect. 9.3](#), we have

$$WR_{xx}(0)W^\top = I_n \quad (100)$$

$$WR_{xx}(\tau)W^\top = D_2 \quad (101)$$

The linear transformation W , which satisfies ([Eq. 100](#)) and ([Eq. 101](#)) is the eigenvector matrix of $R_{xx}^{-1}(0)R_{xx}(\tau)$ (Fukunaga 1990). In other words, the matrix W is the generalized eigenvector matrix of the pencil $R_{xx}(\tau) - \lambda R_{xx}(0)$ (Molgedey and Schuster 1994).

Recently Chang et al. (2000) proposed the matrix pencil method for BSS where they exploited $R_{xx}(\tau_1)$ and $R_{xx}(\tau_2)$ for $\tau_1 \neq \tau_2 \neq 0$. Since the noise vector was assumed to be temporally white, two matrices $R_{xx}(\tau_1)$ and $R_{xx}(\tau_2)$ are not theoretically affected by the noise vector, that is,

$$R_{xx}(\tau_1) = AR_{ss}(\tau_1)A^\top \quad (102)$$

$$R_{xx}(\tau_2) = AR_{ss}(\tau_2)A^\top \quad (103)$$

Thus, it is obvious that we can find an estimate of the demixing matrix that is not sensitive to the white noise. A similar idea was also exploited in Choi and Cichocki (2000a, b).

In general, the generalized eigenvalue decomposition requires the symmetric-definite pencil (one matrix is symmetric and the other is symmetric and positive definite). However, $R_{xx}(\tau_2) - \lambda R_{xx}(\tau_1)$ is not symmetric-definite, which might cause a numerical instability problem that results in complex-valued eigenvectors.

The set of all matrices of the form $R_1 - \lambda R_2$ with $\lambda \in \mathbb{R}$ is said to be a *pencil*. Frequently we encounter the case where R_1 is symmetric and R_2 is symmetric and positive definite. Pencils of this variety are referred to as *symmetric-definite pencils* (Golub and Loan 1993).

Theorem 4 (Golub and Loan 1993, p. 468) *If $R_1 - \lambda R_2$ is symmetric-definite, then there exists a nonsingular matrix $U = [u_1, \dots, u_n]$ such that*

$$U^\top R_1 U = \text{diag}\{\gamma_1(\tau_1), \dots, \gamma_n(\tau_1)\} \quad (104)$$

$$U^\top R_2 U = \text{diag}\{\gamma_1(\tau_2), \dots, \gamma_n(\tau_2)\} \quad (105)$$

Moreover $R_1 u_i = \lambda_i R_2 u_i$ for $i = 1, \dots, n$, and $\lambda_i = \frac{\gamma_i(\tau_1)}{\gamma_i(\tau_2)}$.

It is apparent from [Theorem 4](#) that R_1 should be symmetric and R_2 should be symmetric and positive definite so that the generalized eigenvector U can be a valid solution if $\{\lambda_i\}$ are distinct.

10 Software

A variety of ICA software is available. ICA Central (URL:<http://www.tsi.enst.fr/icacentral/>) was created in 1999 to promote research on ICA and BSS by means of public mailing lists, a repository of data sets, a repository of ICA/BSS algorithms, and so on. ICA Central might be the first place where you can find data sets and ICA algorithms. In addition, here are some widely used software packages.

- *ICALAB Toolboxes* (<http://www.bsp.brain.riken.go.jp/ICALAB/>): ICALAB is an ICA Matlab software toolbox developed in the Laboratory for Advanced Brain Signal Processing in the RIKEN Brain Science Institute, Japan. It consists of two independent packages, including ICALAB for signal processing and ICALAB for image processing, and each package contains a variety of algorithms.
- *FastICA* (<http://www.cis.hut.fi/projects/ica/fastica/>): This is the FastICA Matlab package that implements fast fixed-point algorithms for non-Gaussianity maximization (Hyvärinen et al. 2001). It was developed in the Helsinki University of Technology, Finland, and other environments (R, C++, Physon) are also available.
- *Infomax ICA* (http://www.cnl.salk.edu/~tewon/ica_cnl.html): Matlab and C codes for Bell and Sejnowski's Infomax algorithm (Bell and Sejnowski 1995) and extended Infomax (Lee 1998) where a parametric density model is incorporated into Infomax to handle both super-Gaussian and sub-Gaussian sources.
- *EEGLAB* (<http://sccn.ucsd.edu/eeglab/>): EEGLAB is an interactive Matlab toolbox for processing continuous and event-related EEG, MEG, and other electrophysiological data using ICA, time/frequency analysis, artifact rejection, and several modes of data visualization.
- *ICA: DTU Toolbox* (<http://isp.imm.dtu.dk/toolbox/ica/>): ICA: DTU Toolbox is a collection of ICA algorithms that includes: (1) icaML, which is an efficient implementation of Infomax; (2) icaMF, which is an iterative algorithm that offers a variety of possible source priors and mixing matrix constraints (e.g., positivity) and can also handle over- and under-complete mixing; and (3) icaMS, which is a “one shot” fast algorithm that requires time correlation between samples.

11 Further Issues

- *Overcomplete representation:* Overcomplete representation enforces the latent space dimension n to be greater than the data dimension m in the linear model ([Eq. 1](#)). Sparseness constraints on latent variables are necessary to learn fruitful representation (Lewicki and Sejnowski 2000).
- *Bayesian ICA:* Bayesian ICA incorporates uncertainty and prior distributions of latent variables in the model ([Eq. 1](#)). Independent factor analysis (Attias 1999) is pioneering work along this direction. The EM algorithm for ICA was developed in Welling and

Weber (2001) and a full Bayesian ICA (also known as ensemble learning) was developed in Miskin and MacKay (2001).

- *Kernel ICA*: Kernel methods were introduced to consider statistical independence in reproducing kernel Hilbert space (Bach and Jordan 2002), developing kernel ICA.
- *Nonnegative ICA*: Nonnegativity constraints were imposed on latent variables, yielding nonnegative ICA (Plumbley 2003). Rectified Gaussian priors can also be used in Bayesian ICA to handle nonnegative latent variables.
- *Sparseness*: Sparseness is another important characteristic of sources, besides independence. Sparse component analysis is studied in Lee et al. (2006).
- *Beyond ICA*: Independent subspace analysis (Hyvärinen and Hoyer 2000) and tree-dependent component analysis (Bach and Jordan 2003) generalizes ICA, allowing intra-dependence structures in feature subspaces or clusters.

12 Summary

ICA has been successfully applied to various applications of machine learning, pattern recognition, and signal processing. A brief overview of ICA has been presented, starting from fundamental principles on learning a linear latent variable model for parsimonious representation. Natural gradient ICA algorithms were derived in the framework of maximum likelihood estimation, mutual information minimization, Infomax, and negentropy maximization. We have explained flexible ICA where generalized Gaussian density was adopted such that a flexible nonlinear function was incorporated into the natural gradient ICA algorithm. Equivariant nonstationary source separation was presented in the framework of natural gradient as well. Differential learning was also adopted to incorporate a temporal structure of sources. We also presented a core idea and various methods for algebraic source separation. Various software packages for ICA were introduced, for easy application of ICA. Further issues were also briefly mentioned so that readers can follow the status of ICA.

References

- Amari S (1998) Natural gradient works efficiently in learning. *Neural Comput* 10(2):251–276
- Amari S, Cardoso JF (1997) Blind source separation: semiparametric statistical approach. *IEEE Trans Signal Process* 45:2692–2700
- Amari S, Chen TP, Cichocki A (1997) Stability analysis of learning algorithms for blind source separation. *Neural Networks* 10(8):1345–1351
- Amari S, Cichocki A (1998) Adaptive blind signal processing – neural network approaches. *Proc IEEE (Special Issue on Blind Identification and Estimation)* 86(10):2026–2048
- Amari S, Cichocki A, Yang HH (1996) A new learning algorithm for blind signal separation. In: Touretzky DS, Mozer MC, Hasselmo ME (eds) *Advances in neural information processing systems (NIPS)*, vol 8. MIT Press, Cambridge, pp 757–763
- Attias H (1999) Independent factor analysis. *Neural Comput* 11:803–851
- Bach F, Jordan MI (2002) Kernel independent component analysis. *JMLR* 3:1–48
- Bach FR, Jordan MI (2003) Beyond independent components: trees and clusters. *JMLR* 4:1205–1233
- Bell A, Sejnowski T (1995) An information maximisation approach to blind separation and blind deconvolution. *Neural Comput* 7:1129–1159
- Belouchrani A, Abed-Meraim K, Cardoso JF, Moulines E (1997) A blind source separation technique using second order statistics. *IEEE Trans Signal Process* 45:434–444

- Cardoso JF (1989) Source separation using higher-order moments. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP), Paris, France, 23–26 May 1989
- Cardoso JF (1997) Infomax and maximum likelihood for source separation. *IEEE Signal Process Lett* 4(4): 112–114
- Cardoso JF, Laheld BH (1996) Equivariant adaptive source separation. *IEEE Trans Signal Process* 44 (12):3017–3030
- Cardoso JF, Souloumiac A (1993) Blind beamforming for non Gaussian signals. *IEE Proc-F* 140(6):362–370
- Chang C, Ding Z, Yau SF, Chan FHY (2000) A matrix-pencil approach to blind separation of colored non-stationary signals. *IEEE Trans Signal Process* 48(3): 900–907
- Choi S (2002) Adaptive differential decorrelation: a natural gradient algorithm. In: Proceedings of the international conference on artificial neural networks (ICANN), Madrid, Spain. Lecture notes in computer science, vol 2415. Springer, Berlin, pp 1168–1173
- Choi S (2003) Differential learning and random walk model. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP), IEEE, Hong Kong, pp 724–727
- Choi S, Cichocki A (2000a) Blind separation of nonstationary and temporally correlated sources from noisy mixtures. In: Proceedings of IEEE workshop on neural networks for signal processing, IEEE, Sydney, Australia. pp 405–414
- Choi S, Cichocki A (2000b) Blind separation of nonstationary sources in noisy mixtures. *Electron Lett* 36(9):848–849
- Choi S, Cichocki A, Amari S (2000) Flexible independent component analysis. *J VLSI Signal Process* 26(1/2): 25–38
- Choi S, Cichocki A, Belouchrani A (2002) Second order nonstationary source separation. *J VLSI Signal Process* 32:93–104
- Choi S, Cichocki A, Park HM, Lee SY (2005) Blind source separation and independent component analysis: a review. *Neural Inf Process Lett Rev* 6(1): 1–57
- Cichocki A, Amari S (2002) Adaptive blind signal and image processing: learning algorithms and applications. Wiley, Chichester
- Cichocki A, Unbehauen R (1996) Robust neural networks with on-line learning for blind identification and blind separation of sources. *IEEE Trans Circ Syst Fund Theor Appl* 43:894–906
- Comon P (1994) Independent component analysis, a new concept? *Signal Process* 36(3):287–314
- Fukunaga K (1990) An introduction to statistical pattern recognition. Academic, New York
- Golub GH, Loan CFV (1993) Matrix computations, 2nd edn. Johns Hopkins, Baltimore
- Haykin S (2000) Unsupervised adaptive filtering: blind source separation. Prentice-Hall
- Hyvärinen A (1999) Survey on independent component analysis. *Neural Comput Surv* 2:94–128
- Hyvärinen A, Hoyer P (2000) Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Comput* 12(7):1705–1720
- Hyvärinen A, Karhunen J, Oja E (2001) Independent component analysis. Wiley, New York
- Hyvärinen A, Oja E (1997) A fast fixed-point algorithm for independent component analysis. *Neural Comput* 9:1483–1492
- Jutten C, Herault J (1991) Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture. *Signal Process* 24:1–10
- Karhunen J (1996) Neural approaches to independent component analysis and source separation. In: Proceedings of the European symposium on artificial neural networks (ESANN), Bruges, Belgium, pp 249–266
- Kim S, Choi S (2005) Independent arrays or independent time courses for gene expression data. In: Proceedings of the IEEE international symposium on circuits and systems (ISCAS), Kobe, Japan, 23–26 May 2005
- Kosko B (1986) Differential Hebbian learning. In: Proceedings of American Institute of Physics: neural networks for computing, Snowbird. American Institute of Physics, Woodbury, pp 277–282
- Lee TW (1998) Independent component analysis: theory and applications. Kluwer
- Lee TW, Girolami M, Sejnowski T (1999) Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. *Neural Comput* 11(2):609–633
- Lewicki MS, Sejnowski T (2000) Learning overcomplete representation. *Neural Comput* 12(2):337–365
- Li Y, Cichocki A, Amari S (2006) Blind estimation of channel parameters and source components for EEG signals: a sparse factorization approach. *IEEE Trans Neural Netw* 17(2):419–431
- Liebermeister W (2002) Linear modes of gene expression determined by independent component analysis. *Bioinformatics* 18(1):51–60
- MacKay DJC (1996) Maximum likelihood and covariant algorithms for independent component analysis. Technical Report Draft 3.7, University of Cambridge, Cavendish Laboratory
- Matsuoka K, Ohya M, Kawamoto M (1995) A neural net for blind separation of nonstationary signals. *Neural Networks* 8(3):411–419
- Miskin JW, MacKay DJC (2001) Ensemble learning for blind source separation. In: Roberts S, Everson R (eds) Independent component analysis: principles and practice. Cambridge University Press, Cambridge, UK, pp 209–233

- Molgedey L, Schuster HG (1994) Separation of a mixture of independent signals using time delayed correlations. *Phys Rev Lett* 72:3634–3637
- Oja E (1995) The nonlinear PCA learning rule and signal separation – mathematical analysis. Technical Report A26, Helsinki University of Technology, Laboratory of Computer and Information Science
- Pearlmutter B, Parra L (1997) Maximum likelihood blind source separation: a context-sensitive generalization of ICA. In: Mozer MC, Jordan MI, Petsche T (eds) *Advances in neural information processing systems (NIPS)*, vol 9. MIT Press, Cambridge, pp 613–619
- Pham DT (1996) Blind separation of instantaneous mixtures of sources via an independent component analysis. *IEEE Trans Signal Process* 44(11): 2768–2779
- Plumbley MD (2003) Algorithms for nonnegative independent component analysis. *IEEE Trans Neural Network* 14(3):534–543
- Stone JV (2004) *Independent component analysis: a tutorial introduction*. MIT Press, Cambridge
- Stone JV, Porrl J, Porter NR, Wilkinson IW (2002) Spatiotemporal independent component analysis of event-related fMRI data using skewed probability density functions. *NeuroImage* 15(2):407–421
- Tong L, Soon VC, Huang YF, Liu R (1990) AMUSE: a new blind identification algorithm. In: *Proceedings of the IEEE international symposium on circuits and systems (ISCAS)*, IEEE, New Orleans, pp 1784–1787
- Welling M, Weber M (2001) A constrained EM algorithm for independent component analysis. *Neural Comput* 13:677–689

14 Neural Networks for Time-Series Forecasting

G. Peter Zhang

Department of Managerial Sciences, Georgia State University,
Atlanta, GA, USA

gpzhang@gsu.edu

1	<i>Introduction</i>	462
2	<i>Neural Networks</i>	463
3	<i>Applications in Time Series Forecasting</i>	465
4	<i>Neural Network Modeling Issues</i>	466
5	<i>Methodological Issues</i>	470
6	<i>Conclusions</i>	473

Abstract

Neural networks has become an important method for time series forecasting. There is increasing interest in using neural networks to model and forecast time series. This chapter provides a review of some recent developments in time series forecasting with neural networks, a brief description of neural networks, their advantages over traditional forecasting models, and some recent applications. Several important data and modeling issues for time series forecasting are highlighted. In addition, recent developments in several methodological areas such as seasonal time series modeling, multi-period forecasting, and the ensemble method are reviewed.

1 Introduction

Time series forecasting is an active research area that has received a considerable amount of attention in the literature. Using the time series approach to forecasting, forecasters collect and analyze historical observations to determine a model to capture the underlying data-generating process. Then the model is extrapolated to forecast future values. This approach is useful for applications in many domains such as business, economics, industry, engineering, and science. Much effort has been devoted, over the past three decades, to the development and improvement of time series forecasting models.

There has been an increasing interest in using neural networks to model and forecast time series. Neural networks have been found to be a viable contender when compared to various traditional time series models (Zhang et al. 1998; Balkin and Ord 2000; Jain and Kumar 2007). Lapedes and Farber (1987) report the first attempt to model nonlinear time series with neural networks. De Groot and Wurtz (1991) present a detailed analysis of univariate time series forecasting using feedforward neural networks for two benchmark nonlinear time series. Chakraborty et al. (1992) conduct an empirical study on multivariate time series forecasting with neural networks. Atiya et al. (1999) present a case study of multistep river flow forecasting. Poli and Jones (1994) propose a stochastic neural net model based on the Kalman filter for nonlinear time series prediction. Weigend et al. (1990, 1992) and Cottrell et al. (1995) address the issue of network structure for forecasting real-world time series. Berardi and Zhang (2003) investigate the bias and variance issue in the time series forecasting context. Liang (2005) proposes a Bayesian neural network for time series analysis. In addition, results from several large forecasting competitions (Balkin and Ord 2000; Weigend and Gershenfeld 1994) suggest that neural networks can be a very useful addition to the time series forecasting toolbox.

Time series forecasting has been dominated by linear methods for decades. Linear methods are easy to develop and implement and they are also relatively simple to understand and interpret. However, it is important to understand the limitation of the linear models. They are not able to capture nonlinear relationships in the data. In addition, the approximation of linear models to complicated nonlinear relationships is not always satisfactory as evidenced by the well-known M-competition where a majority of commonly used linear methods were tested with more than 1,000 real time series data (Makridakis et al. 1982). The results clearly show that no single model is the best and the best performer is dependent on the data and other conditions. One explanation of the mixed findings is the failure of linear models to account for a varying degree of nonlinearity that is common in real-world problems.

That is, because of the inherent nonlinear characteristics in the data, no single linear model is able to approximate all types of nonlinear data structure equally well.

Neural networks provide a promising alternative tool for forecasters. The inherently nonlinear structure of neural networks is particularly useful for capturing the complex underlying relationship in many real-world problems. Neural networks are perhaps more versatile methods for forecasting applications in that, not only can they find nonlinear structures in a problem, they can also model linear processes. For example, the capability of neural networks in modeling linear time series has been studied and reported by several researchers (Hwang 2001; Medeiros and Pedreira 2001; Zhang 2001).

In addition to the nonlinear modeling capability, neural networks have several other features that make them valuable for time series forecasting. First, neural networks are data-driven nonparametric methods that do not require many restrictive assumptions on the underlying stochastic process from which data are generated. As such, they are less susceptible to the model misspecification problem than parametric methods. This “learning from data” feature is highly desirable in various forecasting situations where time series data are usually easy to collect but the underlying data-generating mechanism is not known. Second, neural networks have been shown to have universal functional approximating capability in that they can accurately approximate many types of complex functional relationships. This is an important and powerful characteristic as a time series model aims to accurately capture the functional relationship between the variable to be forecast and its historical observations. The combination of the above mentioned characteristics makes neural networks a quite general and flexible tool for forecasting.

Research efforts on neural networks for time series forecasting are considerable and numerous applications of neural networks for forecasting have been reported. Adya and Collopy (1998) and Zhang et al. (1998) reviewed the relevant literature in forecasting with neural networks. There has been a significant advance in research in this area since then. The purpose of this chapter is to summarize some of the important recent work with a focus on time series forecasting using neural networks.

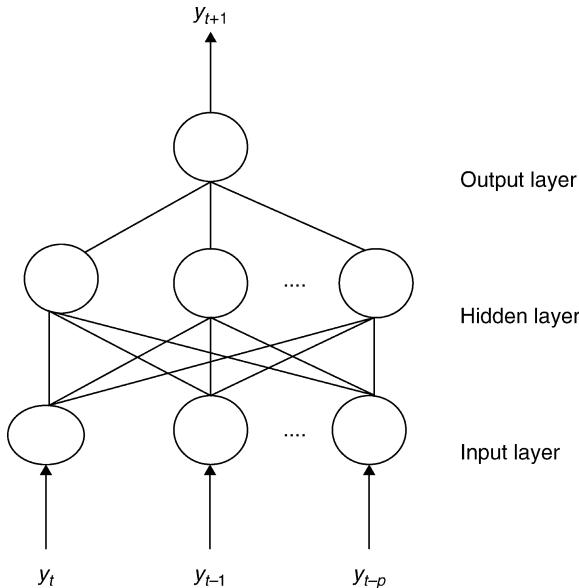
2 Neural Networks

Neural networks are computing models for information processing. They are useful for identifying the fundamental functional relationship or pattern in the data. Although many types of neural network models have been developed to solve different problems, the most widely used model by far for time series forecasting has been the feedforward neural network.

► *Figure 1* is a popular one-output feedforward neural network model. It is composed of several layers of basic processing units called neurons or nodes. Here, the network model has one input layer, one hidden layer, and one output layer. The nodes in the input layer are used to receive information from the data. For a time series forecasting problem, past lagged observations ($y_t, y_{t-1}, \dots, y_{t-p}$) are used as inputs. The hidden layer is composed of nodes that are connected to both the input and the output layer and is the most important part of a network. With nonlinear transfer functions, hidden nodes can process the information received by input nodes. The output from the network is used to predict the future value(s) of a time series. If the focus is on one-step-ahead forecasting, then only one output node is needed. If multistep-ahead forecasting is needed, then multiple nodes may be employed in the output layer. In a feedforward network, information is one directional. That is, it goes through the input nodes to hidden nodes and to output nodes, and there is no feedback from the

Fig. 1

A typical feedforward neural network for time series forecasting.



network output. The feedforward neural network illustrated in [Fig. 1](#) is functionally equivalent to a nonlinear autoregressive model

$$y_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-p}) + \varepsilon_{t+1}$$

where y_t is the observed time series value for variable y at time t and ε_{t+1} is the error term at time $t+1$. This model suggests that a future time series value, y_{t+1} , is an autoregressive function of its past observations, $y_t, y_{t-1}, \dots, y_{t-p}$ plus a random error. Because any time series forecasting model assumes that there is a functional relationship between the future value and the past observations, neural networks can be useful in identifying this relationship.

In developing a feedforward neural network model for forecasting tasks, specifying its architecture in terms of the number of input, hidden, and output neurons is an important yet nontrivial task. Most neural network applications use one output neuron for both one-step-ahead and multistep-ahead forecasting. However, as argued by Zhang et al. (1998), it may be beneficial to employ multiple output neurons for direct multistep-ahead forecasting. The input neurons or variables are very important in any modeling endeavor and especially important for neural network modeling because the success of a neural network depends, to a large extent, on the patterns represented by the input variables. For a time series forecasting problem, one needs to identify how many and what past lagged observations should be used as the inputs. Finally, the number of hidden nodes is usually unknown before building a neural network model and must be chosen during the model-building process. This parameter is useful for capturing the nonlinear relationship between input and output variables.

Before a neural network can be used for forecasting, it must be trained. Neural network training refers to the estimation of connection weights. Although the estimation process is similar to that in the regression modeling where one minimizes the sum of squared errors, the neural network training process is more difficult and complicated due to the nature of

nonlinear optimization process. There are many training algorithms developed in the literature and the most influential one is the backpropagation algorithm by Werbos (1974) and Rumelhart et al. (1986). The basic idea of backpropagation training is to use a gradient-descent approach to adjust and determine weights such that an overall error function such as the sum of squared errors can be minimized.

In addition to the most popular feedforward neural networks, other types of neural networks can also be used for time series forecasting purposes. For example, Barreto (2008) provides a review of time series forecasting using the self-organizing map. Recurrent neural networks (Connor et al. 1994; Kuan and Liu 1995; Kermanshahi 1998; Vermaak and Botha 1998; Parlous et al. 2000; Mandic and Chambers 2001; Huskent and Stagge 2003; Ghiassi et al. 2005; Cai et al. 2007; Jain and Kumar 2007; Menezes and Barreto 2008) that explicitly account for the dynamic nonlinear pattern are good alternatives to feedforward networks for certain time series forecasting problems. In a recurrent neural network, there are cycles or feedback connections among neurons. Outputs from a recurrent network can be directly fed back to inputs, generating dynamic feedbacks on errors of past patterns. In this sense, recurrent networks can model richer dynamics than feedforward networks just like linear autoregressive and moving average (ARMA) models that have certain advantages over autoregressive (AR) models. However, much less attention has been paid to the research and applications of recurrent networks, and the superiority of recurrent networks over feedforward networks has not been established. The practical difficulty of using recurrent neural networks may lie in the facts that (1) recurrent networks can assume very different architectures and it may be difficult to specify appropriate model structures to experiment with, and (2) it is more difficult to train recurrent networks due to the unstable nature of training algorithms.

For an in-depth coverage of many aspects of networks, readers are referred to a number of excellent books including Smith (1993), Bishop (1995), and Ripley (1996). For neural networks for forecasting research and applications, readers may consult Azoff (1994), Weigend and Gershenfeld (1994), Gately (1996), Zhang et al. (1998), and Zhang (2004).

3 Applications in Time Series Forecasting

Time series are data collected over time and this is one of the most commonly available forms of data in many forecasting applications. Therefore, it is not surprising that the use of neural networks for time series forecasting has received great attention in many different fields. Given that forecasting problems arise in so many different disciplines and the literature on forecasting with neural networks is scattered in so many diverse fields, it is difficult to cover all neural network applications in time series forecasting problems in this review.  [Table 1](#) provides a sample of recent time series forecasting applications reported in the literature since 2005. For other forecasting applications of neural networks, readers are referred to several survey articles such as Dougherty (1995) for transportation modeling and forecasting, Wong and Selvi (1998) and Fadlalla and Lin (2001) for financial applications, Krycha and Wagner (1999) for management science applications, Vellido et al. (1999) and Wong et al. (2000) for business applications, Maier and Dandy (2000) for water resource forecasting, and Hippert et al. (2001) for short-term load forecasting.

As can be seen from  [Table 1](#), a wide range of time series forecasting problems have been solved by neural networks. Some of these application areas include environment (air pollutant concentration, carbon monoxide concentration, drought, and ozone level), business and

Table 1

Some recent neural network applications in time series forecasting

Forecasting problems	Studies
Air pollutant concentration	Gautama et al. (2008)
Carbon monoxide concentration	Chelani and Devotta (2007)
Demand	Aburto and Weber (2007)
Drought	Mishra and Desai (2006)
Electrical consumption	Azadeh et al. (2007)
Electricity load	Hippert et al. (2005), Xiao et al. (2009)
Electricity price	Pino et al. (2008)
Energy demand	Abdel-Aal (2008)
Exchange rate	Zhang and Wan (2007)
Food grain price	Zou et al. (2007)
Food product sales	Doganis et al. (2006)
Gold price changes	Parisi et al. (2008)
Inflation	Nakamura (2005)
Inventory	Doganis et al. (2008)
Macroeconomic time series	Teräsvirta et al. (2005)
Stock index option price	Wang (2009)
Stock returns volatility	Bodyanskiy and Popov (2006)
Tourism demand	Palmer et al. (2006), Chu (2008)
Traffic flow	Jiang and Adeli (2005)
Ozone level	Coman et al. (2008)
River flow	Jain and Kumar (2007)
Wind speed	Cadenas and Rivera (2009)

finance (product sales, demand, inventory, stock market movement and risk, exchange rate, futures trading, commodity and option price), tourism and transportation (tourist volume and traffic flow), engineering (wind speed and river flow) and energy (electrical consumption and energy demand). Again this is only a relatively small sample of the application areas to which neural networks have been applied. One can find many more application areas of neural networks in time series analysis and forecasting.

4 Neural Network Modeling Issues

Developing a neural network model for a time series forecasting application is not a trivial task. Although many software packages exist to ease users' effort in building a neural network model, it is critical for forecasters to understand many important issues around the model-building process. It is important to point out that building a successful neural network is a combination of art and science, and software alone is not sufficient to solve all problems in the process. It is a pitfall to blindly throw data into a software package and then hope it will automatically give a satisfactory forecast.

Neural network modeling issues include the choice of network type and architecture, the training algorithm, as well as model validation, evaluation, and selection. Some of these can be solved during the model-building process while others must be carefully considered and planned before actual modeling starts.

4.1 Data Issues

Regarding the data issues, the major decisions a neural network forecaster must make include data preparation, data cleaning, data splitting, and input variable selection. Neural networks are data-driven techniques. Therefore, data preparation is a very critical step in building a successful neural network model. Without a good, adequate, and representative data set, it is impossible to develop a useful predictive model. The reliability of neural network models often depends, to a large degree, on the quality of data.

There are several practical issues around the data requirement for a neural network model. The first is the size of the sample used to build a neural network. While there is no specific rule that can be followed for all situations, the advantage of having a large sample should be clear because not only do neural networks typically have a large number of parameters to estimate, but also it is often necessary to split data into several portions to avoid overfitting, to select the model, and to perform model evaluation and comparison. A larger sample provides a better chance for neural networks to adequately capture the underlying data-generating process. Although large samples do not always give superior performance over small samples, forecasters should strive to get as large a size as they can. In time series forecasting problems, Box and Jenkins (1976) have suggested that at least 50, or better 100, observations are necessary to build linear autoregressive integrated moving average (ARIMA) models. Therefore, for non-linear modeling, a larger sample size should be more desirable. In fact, using the longest time series available for developing forecasting models is a time-tested principle in forecasting (Armstrong 2001). Of course, if observations in the time series are not homogeneous or the underlying data-generating process changes over time, then larger sample sizes may not help and can even hurt the performance of neural networks.

The second issue is the data splitting. Typically, for neural network applications, all available data are divided into an in-sample and an out-of-sample. The in-sample data are used for model fitting and selection, while the out-of-sample is used to evaluate the predictive ability of the model. The in-sample data sometimes are further split into a training sample and a validation sample. This division of data means that the true size of the sample used in model building is smaller than the initial sample size. Although there is no consensus on how to split the data, the general practice is to allocate more data for model building and selection. That is, most studies in the literature use convenient ratios of splitting for in-sample and out-of-sample, such as 70:30%, 80:20%, and 90:10%. It is important to note that in data splitting the issue is not about what proportion of data should be allocated in each sample but about sufficient data points in each sample to ensure adequate learning, validation, and testing. When the size of the available data set is large, different splitting strategies may not have a major impact. But it is quite different when the sample size is small. According to Chatfield (2001), forecasting analysts typically retain about 10% of the data as a hold-out sample. Granger (1993) suggests that for nonlinear modeling, at least 20% of the data should be held back for an out-of-sample evaluation. Hoptroff (1993) recommends that at least ten data points should be in the test sample, while Ashley (2003)

suggests that a much larger out-of-sample size is necessary in order to achieve statistically significant improvement for forecasting problems.

In addition, data splitting generally should be done randomly to make sure each subsample is representative of the population. However, time series data are difficult or impossible to split randomly because of the desire to keep the autocorrelation structure of the time series observations. For many time series problems, data splitting is typically done at researchers' discretion. However, it is important to make sure that each portion of the sample is characteristic of the true data-generating process. LeBaron and Weigend (1998) evaluate the effect of data splitting on time series forecasting and find that data splitting can cause more sample variation which in turn causes the variability of forecast performance. They caution the pitfall of ignoring variability across the splits and drawing too strong conclusions from such splits.

Data preprocessing is another issue that is often recommended to highlight important relationships or to create more uniform data to facilitate neural network learning, meet algorithm requirements, and avoid computation problems. Azoff (1994) summarizes four methods typically used for input data normalization. They are along-channel normalization, across-channel normalization, mixed-channel normalization, and external normalization. However, the necessity and effect of data normalization on network learning and forecasting are still not universally agreed upon. For example, in modeling and forecasting seasonal time series, some researchers (Gorr 1994) believe that data preprocessing is not necessary because the neural network is a universal approximator and is able to capture all of the underlying patterns well. Empirical studies (Nelson et al. 1999), however, find that pre-deseasonalization of the data is critical in improving forecasting performance. Zhang and Qi (2002, 2005) further demonstrate that for time series containing both trend and seasonal variations, preprocessing the data by both detrending and deseasonalization should be the most appropriate way to build neural networks for best forecasting performance.

4.2 Network Design

Neural network design and architecture selection are important yet difficult tasks. Not only are there many ways to build a neural network model and a large number of choices to be made during the model building and selection process, but also numerous parameters and issues have to be estimated and experimented with before a satisfactory model may emerge. Adding to the difficulty is the lack of standards in the process. Numerous rules of thumb are available but not all of them can be applied blindly to a new situation. In building an appropriate model for the forecasting task at hand, some experiments are usually necessary. Therefore, a good experimental design is needed. For discussions of many aspects of modeling issues, readers may consult Kaastra and Boyd (1996), Zhang et al. (1998), Coakley and Brown (1999), and Remus and O'Connor (2001).

A feedforward neural network is characterized by its architecture determined by the number of layers, the number of nodes in each layer, the transfer or activation function used in each layer, as well as how the nodes in each layer connect to nodes in adjacent layers. Although partial connections between nodes in adjacent layers and direct connection from input layer to output layer are possible, the most commonly used neural network is the fully connected one in which each node in one layer is fully connected only to all nodes in the adjacent layers.

The size of the output layer is usually determined by the nature of the problem. For example, in most time series forecasting problems, one output node is naturally used for

one-step-ahead forecasting, although one output node can also be employed for multistep-ahead forecasting, in which case iterative forecasting mode must be used. That is, forecasts for more than twosteps ahead in the time horizon must be based on earlier forecasts. This may not be effective for multistep forecasting as pointed out by Zhang et al. (1998) which is in line with Chatfield (2001) who discusses the potential benefits of using different forecasting models for different lead times. Therefore, for multistep forecasting, one may either use multiple output nodes or develop multiple neural networks each for one particular step forecasting.

The number of input nodes is perhaps the most important parameter for designing an effective neural network forecaster. For causal forecasting problems, it corresponds to the number of independent or predictor variables that forecasters believe are important in predicting the dependent variable. For univariate time series forecasting problems, it is the number of past lagged observations. Determining an appropriate set of input variables is vital for neural networks to capture the essential underlying relationship that can be used for successful forecasting. How many and what variables to use in the input layer will directly affect the performance of neural networks in both in-sample fitting and out-of-sample forecasting, resulting in the under-learning or over-fitting phenomenon. Empirical results (Lennon et al. 2001; Zhang et al. 2001; Zhang 2001) also suggest that the input layer is more important than the hidden layer in time series forecasting problems. Therefore, considerable attention should be given to input variable selection especially for time series forecasting. Balkin and Ord (2000) select the ordered variables (lags) sequentially by using a linear model and a forward stepwise regression procedure. Medeiros et al. (2006) also use a linear variable selection approach to choosing the input variables.

Although there is substantial flexibility in choosing the number of hidden layers and the number of hidden nodes in each layer, most forecasting applications use only one hidden layer and a small number of hidden nodes. In practice, the number of hidden nodes is often determined by experimenting with a number of choices and then selecting using the cross-validation approach or the performance on the validation set. Although the number of hidden nodes is an important factor, a number of studies have shown that the forecasting performance of neural networks is not very sensitive to this parameter (Bakirtzis et al. 1996; Khotanzad et al. 1997; Zhang et al. 2001). Medeiros et al. (2006) propose a statistical approach to selecting the number of hidden nodes by sequentially applying the Lagrange multiplier type tests.

Once a particular neural network architecture is determined, it must be trained so that the parameters of the network can be estimated from the data. To be effective in performing this task, a good training algorithm is needed. Training a neural network can be treated as a nonlinear mathematical optimization problem and different solution approaches or algorithms can have quite different effects on the training result. As a result, training with different algorithms and repeating with multiple random initial weights can be helpful in getting better solutions to the neural network training problem. In addition to the popular basic back-propagation training algorithm, users should be aware of many other (sometimes more effective) algorithms. These include the so-called second-order approaches such as conjugate gradient descent, quasi-Newton, and Levenberg–Marquardt (Bishop 1995).

4.3 Model Selection and Evaluation

Using the selection of a neural network model is typically done using the cross-validation process. That is, the in-sample data is split into a training set and a validation set. The network

parameters are estimated with the training sample, while the performance of the model is evaluated with the validation sample. The best model selected is the one that has the best performance on the validation sample. Of course, in choosing competing models, one must also apply the principle of parsimony. That is, a simpler model that has about the same performance as a more complex model should be preferred.

Model selection can also be done with solely the in-sample data. In this regard, several in-sample selection criteria are used to modify the total error function to include a penalty term that penalizes for the complexity of the model. In-sample model selection approaches are typically based on some information-based criteria such as Akaike's information criterion (AIC) and Bayesian (BIC) or Schwarz information criterion (SIC). However, it is important to note the limitation of these criteria as empirically demonstrated by Swanson and White (1995) and Qi and Zhang (2001). Egrioglu et al. (2008) propose a weighted information criterion to select the model. Other in-sample approaches are based on pruning methods such as node and weight pruning (Reed 1993) as well as constructive methods such as the upstart and cascade correlation approaches (Fahlman and Lebiere 1990; Frean 1990).

After the modeling process, the finally selected model must be evaluated using data not used in the model-building stage. In addition, as neural networks are often used as a nonlinear alternative to traditional statistical models, the performance of neural networks needs to be compared to that of statistical methods. As Adya and Collopy (1998) point out, "if such a comparison is not conducted it is difficult to argue that the study has taught one much about the value of neural networks." They further propose three evaluation criteria to objectively evaluate the performance of a neural network: (1) comparing it to well-accepted (traditional) models; (2) using true out-of-sample data and (3) ensuring enough sample size in the out-of-sample data (40 for classification problems and 75 for time series problems). It is important to note that the test sample served as the out-of-sample should not in any way be used in the model-building process. If the cross-validation is used for model selection and experimentation, the performance on the validation sample should not be treated as the true performance of the model.

5 Methodological Issues

5.1 Modeling Trend and Seasonal Time Series

Many business and economic time series exhibit both seasonal and trend variations. Seasonality is a periodic and recurrent pattern caused by factors such as weather, holidays, repeating promotions, as well as the behavior of economic agents (Hylleberg 1992). Because of the frequent occurrence of these time series in practice, how to model and forecast seasonal and trend time series has long been a major research topic that has significant practical implications.

Traditional analyses of time series are mainly concerned with modeling the autocorrelation structure of a time series, and typically require that the data under study be stationary. Trend and seasonality in time series violate the condition of stationarity. Thus, the removal of the trend and seasonality is often desired in time series analysis and forecasting. For example, the well-known Box-Jenkins approach to time series modeling relies entirely on the stationarity assumption. The classic decomposition technique decomposes a time series into trend, seasonal factor, and irregular components. The trend and seasonality are often estimated and removed from the data first before other components are estimated. Seasonal ARIMA models also require that the data be seasonally differenced to achieve stationarity condition (Box and Jenkins 1976).

However, seasonal adjustment is not without controversy. Ghysels et al. (1996) suggest that seasonal adjustment might lead to undesirable nonlinear properties in univariate time series. Ittig (1997) also questions the traditional method for generating seasonal indexes and proposed a nonlinear method to estimate the seasonal factors. More importantly, some empirical studies find that seasonal fluctuations are not always constant over time and at least in some time series, seasonal components and nonseasonal components are not independent, and thus not separable (Hylleberg 1994).

In the neural network literature, Gorr (1994) points out that neural networks should be able to simultaneously detect both the nonlinear trend and the seasonality in the data. Sharda and Patil (1992) examine 88 seasonal time series from the M-competition and find that neural networks can model seasonality effectively and pre-deseasonalizing the data is not necessary. Franses and Draisma (1997) find that neural networks can also detect possible changing seasonal patterns. Hamzacebi (2008) proposes a neural network with seasonal lag to directly model seasonality. Faraway and Chatfield (1995), however, find mixed results with the direct neural network approach. Kolarik and Rudorfer (1994) report similar findings. Based on a study of 68 time series from the M-competition, Nelson et al. (1999) find that neural networks trained on deseasonalized data forecast significantly better than those trained on seasonally non-adjusted data. Hansen and Nelson (2003) find that the combination of transformation, feature extraction, and neural networks through stacked generalization gives more accurate forecasts than classical decomposition or ARIMA models.

Due to the controversies around how to use neural networks to best model trend and/or seasonal time series, several researchers have systematically studied the issue recently. Qi and Zhang (2008) investigate the issue of how to best use neural networks to model trend time series. With a simulation study, they address the question: what is the most effective way to model and forecast trend time series with neural networks? A variety of different underlying data-generating processes are considered that have different trend mechanisms. Results show that directly modeling the trend component is not a good strategy and differencing the data first is the overall most effective approach in modeling trend time series. Zhang and Qi (2005) further look into the issue of how to best model time series with both trend and seasonal components. Using both simulated and real time series, they find that preprocessing the data by both detrending and deseasonalization is the most appropriate way to build neural networks for best forecasting performance. This finding is supported by Zhang and Kline (2007) who empirically examine the issue of data preprocessing and model selection using a large data set of 756 quarterly time series from the M3 forecasting competition.

5.2 Multi-period Forecasting

One of the methodological issues that has received limited attention in the time series forecasting as well as the neural network literature is multi-period (or multistep) forecasting. A forecaster facing a multiple-period forecasting problem typically has a choice between the iterated method – using a general single-step model to iteratively generate forecasts, and the direct method – using a tailored model that directly forecasts the future value for each forecast horizon. Which method the forecaster should use is an important research and practical question.

Theoretically, the direct method should be more appealing because it is less sensitive to model misspecification (Chevillon and Hendry 2005). Several empirical studies on the relative performance of iterated vs. direct forecasts, however, yield mixed findings. Findley (1985) find

some improvement in forecasting accuracy using the direct method. Based on several simulated autoregressive and moving average time series, Stoica and Nehorai (1989) find no significant differences between the two methods. Bhansali (1997), however, shows that the iterated method has a clear advantage over the direct method for finite autoregressive processes. Kang (2003) uses univariate autoregressive models to forecast nine US economic time series and found inconclusive results regarding which multistep method is preferred. Ang et al. (2006) find that iterated forecasts of US GDP growth outperformed direct forecasts at least during the 1990s. The most comprehensive empirical study to date was undertaken by Marcellino et al. (2006) who compared the relative performance of the iterated vs. direct forecasts with several univariate and multivariate autoregressive (AR) models. Using 170 US monthly macroeconomic time series spanning 1959 to 2002, they find that iterated forecasts are generally better than direct forecasts under several different scenarios. In addition, they showed that direct forecasts are increasingly less accurate as the forecast horizon increases.

For nonlinear models, multi-period forecasting receives little attention in the literature because of the analytical challenges and the computational difficulties (De Gooijer and Kumar 1992). Lin and Granger (1994) recommend the strategy of fitting a new model for each forecast horizon in nonlinear forecasting. Tiao and Tsay (1994) find that the direct forecasting method can dramatically outperform the iterated method, especially for long-term forecasts. In the neural network literature, mixed findings have been reported (Zhang et al. 1998). Although Zhang et al. (1998) argue for using the direct method, Weigend et al. (1992) and Hill et al. (1996) find that the direct method performed much worse than the iterated method. Kline (2004) specifically addresses the issue of the relative performance of iterated vs. direct methods. He proposes three methods – iterated, direct, and joint (using one neural network model to predict all forecast horizons simultaneously) – and compares them using a subset of quarterly series from the M3 competition. He finds that the direct method significantly outperformed the iterated method. In a recent study, Hamzacebi et al. (2009) also reports better results achieved by using the direct method.

5.3 Ensemble Models

One of the major developments in neural network time series forecasting is model combining or ensemble modeling. The basic idea of this multi-model approach is the use of each component model's unique capability to better capture different patterns in the data. Both theoretical and empirical findings have suggested that combining different models can be an effective way to improve the predictive performance of each individual model, especially when the models in the ensemble are quite different. Although a majority of the neural ensemble literature is focused on pattern classification problems, a number of combining schemes have been proposed for time series forecasting problems. For example, Pelikan et al. (1992) and Ginzburg and Horn (1994) combine several feedforward neural networks for time series forecasting. Wedding and Cios (1996) describe a combining methodology using radial basis function networks and the Box-Jenkins models. Goh et al. (2003) use an ensemble of boosted Elman networks for predicting drug dissolution profiles. Medeiros and Veiga (2000) consider a hybrid time series forecasting system with neural networks used to control the time-varying parameters of a smooth transition autoregressive model. Armano et al. (2005) use a combined genetic-neural model to forecast stock indexes. Zhang (2003) proposes a hybrid neural-ARIMA model for time series forecasting. Aslanargun et al. (2007) use a

similar hybrid model to forecast tourist arrivals and find improved results. Liu and Yao (1999) develop a simultaneous training system for negatively correlated networks to overcome the limitation of sequential or independent training methods. Khashei et al. (2008) propose a hybrid artificial neural network and fuzzy regression model for financial time series forecasting. Freitas and Rodrigues (2006) discuss the different ways of combining Gaussian radial basis function networks. They also propose a prefiltering methodology to address the problem caused by nonstationary time series. Wichard and Ogorzalek (2007) use several different model architectures with an iterated prediction procedure to select the final ensemble members.

An ensemble can be formed by multiple network architectures, the same architecture trained with different algorithms, different initial random weights, or even different methods. The component networks can also be developed by training with different data such as the resampling data or with different inputs. In general, as discussed in Sharkey (1996) and Sharkey and Sharkey (1997), the neural ensemble formed by varying the training data typically has more component diversity than that trained on different starting points, with a different number of hidden nodes, or using different algorithms. Thus, this approach is the most commonly used in the literature. There are many different ways to alter training data including cross-validation, bootstrapping, using different data sources or different preprocessing techniques, as well as a combination of the above techniques (Sharkey 1996). Zhang and Berardi (2001) propose two data-splitting schemes to form multiple sub-time-series upon which ensemble networks are built. They find that the ensemble achieves significant improvement in forecasting performance. Zhang (2007b) proposes a neural ensemble model based on the idea of adding noises to the input data and forming different training sets with the jittered input data.

6 Conclusions

Neural networks have become an important tool for time series forecasting. They have many desired features that are quite suitable for practical applications. This chapter provides a general overview of the neural networks for time series forecasting problems. Successful application areas of neural networks as well as critical modeling issues are reviewed. It should be emphasized that each time series forecasting situation requires a careful study of the problem characteristics, careful examination of the data characteristics, prudent design of the modeling strategy, and full consideration of modeling issues. Many rules of thumb in neural networks may not be useful for a new application although good forecasting principles and established guidelines should be followed. Researchers need to be aware of many pitfalls that could arise in using neural networks in their research and applications (Zhang 2007a).

It is important to recognize that although some of the modeling issues are unique to neural networks, some are general issues for any forecasting method. Therefore, good forecasting practice and principles should be followed. It is beneficial to consult Armstrong (2001) which provides a good source of information on useful principles for forecasting model building, evaluation, and use.

Neural networks have achieved great successes in the field of time series forecasting. It is, however, important to note that they may not always yield better results than traditional methods for every forecasting task under all circumstances. Therefore, researchers should not focus only on neural networks and completely ignore the traditional methods in their forecasting applications. A number of forecasting competitions suggest that no single method

including neural networks is universally the best for all types of problem in every situation. Thus it may be beneficial to combine different models (e.g., combine neural networks and statistical models) in improving forecasting performance. Indeed, efforts to find better ways to use neural networks for time series forecasting should continue.

References

- Abdel-Aal RE (2008) Univariate modeling and forecasting of monthly energy demand time series using abductive and neural networks. *Comput Ind Eng* 54:903–917
- Aburto L, Weber R (2007) Improved supply chain management based on hybrid demand forecasts. *Appl Soft Comput* 7(1):136–144
- Adya M, Collopy F (1998) How effective are neural networks at forecasting and prediction? A review and evaluation. *J Forecasting* 17:481–495
- Ang A, Piazzesi M, Wei M (2006) What does the yield curve tell us about GDP growth? *J Econometrics* 131:359–403
- Armano G, Marchesi M, Murru A (2005) A hybrid genetic-neural architecture for stock indexes forecasting. *Info Sci* 170(1):3–33
- Armstrong JS (2001) Principles of forecasting: A handbook for researchers and practitioners. Kluwer, Boston, MA
- Ashley R (2003) Statistically significant forecasting improvements: how much out-of-sample data is likely necessary? *Int J Forecasting* 19(2):229–239
- Aslanargun A, Mammadov M, Yazici B, Yolacan S (2007) Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting. *J Stat Comput Simulation* 77(1):29–53
- Atiya AF, El-Shoura SM, Shaheen SI, El-Sherif MS (1999) A comparison between neural-network forecasting techniques-case study: river flow forecasting. *IEEE Trans Neural Netw* 10(2):402–409
- Azadeh A, Ghaderi SF, Sohrabkhani S (2007) Forecasting electrical consumption by integration of neural network, time series and ANOVA. *Appl Math Comput* 186:1753–1761
- Azoff EM (1994) Neural network time series forecasting of financial markets. Wiley, Chichester, UK
- Bakirtzis AG, Petridis V, Kiartzis SJ, Alexiadis MC, Maassis AH (1996) A neural network short term load forecasting model for the Greek power system. *IEEE Trans Power Syst* 11(2):858–863
- Balkin DS, Ord KJ (2000) Automatic neural network modeling for univariate time series. *Int J Forecasting* 16:509–515
- Barreto GA (2008) Time series prediction with the self-organizing map: a review. *Stud Comput Intell* 77:135–158
- Berardi LV, Zhang PG (2003) An empirical investigation of bias and variance in time series forecasting: modeling considerations and error evaluation. *IEEE Trans Neural Netw* 14(3):668–679
- Bhansali RJ (1997) Direct autoregressive predictions for multistep prediction: order selection and performance relative to the plug in predictors. *Stat Sin* 7:425–449
- Bishop M (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
- Bodyanskiy Y, Popov S (2006) Neural network approach to forecasting of quasiperiodic financial time series. *Eur J Oper Res* 175:1357–1366
- Box GEP, Jenkins G (1976) Time series analysis: forecasting and control. Holden-Day, San Francisco, CA
- Cadenas E, Rivera W (2009) Short term wind speed forecasting in La Venta, Oaxaca, México, using artificial neural networks. *Renewable Energy* 34(1):274–278
- Cai X, Zhang N, Venayagamoorthy GK, Wunsch DC (2007) Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm. *Neurocomputing* 70:2342–2353
- Chakraborty K, Mehrotra K, Mohan KC, Ranka S (1992) Forecasting the behavior of multivariate time series using neural networks. *Neural Netw* 5:961–970
- Chatfield C (2001) Time-series forecasting. Chapman & Hall/CRC, Boca Raton, FL
- Chelani AB, Devotta S (2007) Prediction of ambient carbon monoxide concentration using nonlinear time series analysis technique. *Transportation Res Part D* 12:596–600
- Chevillon G, Hendry DF (2005) Non-parametric direct multi-step estimation for forecasting economic processes. *Int J Forecasting* 21:201–218
- Chu FL (2008) Analyzing and forecasting tourism demand with ARAR algorithm. *Tourism Manag* 29(6):1185–1196
- Coakley JR, Brown CE (1999) Artificial neural networks in accounting and finance: modeling issues. *Int J Intell Syst Acc Finance Manag* 9:119–144
- Coman A, Ionescu A, Candau Y (2008) Hourly ozone prediction for a 24-h horizon using neural networks. *Environ Model Software* 23(12):1407–1421
- Connor JT, Martin RD, Atlas LE (1994) Recurrent neural networks and robust time series prediction. *IEEE Trans Neural Netw* 51(2):240–254

- Cottrell M, Girard B, Girard Y, Mangeas M, Muller C (1995) Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Trans Neural Netw* 6(6):1355–1364
- De Gooijer JG, Kumar K (1992) Some recent developments in non-linear time series modeling, testing, and forecasting. *Int J Forecasting* 8:135–156
- De Groot C, Wurtz D (1991) Analysis of univariate time series with connectionist nets: a case study of two classical examples. *Neurocomputing* 3:177–192
- Doganis P, Aggelogiannaki E, Patrinos P, Sarimveis H (2006) Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *J Food Eng* 75:196–204
- Doganis P, Aggelogiannaki E, Sarimveis H (2008) A combined model predictive control and time series forecasting framework for production-inventory systems. *Int J Prod Res* 46(24):6841–6853
- Dougherty M (1995) A review of neural networks applied to transport. *Transportation Res Part C* 3(4): 247–260
- Egrioglu E, Aladag CAH, Gunay S (2008) A new model selection strategy in artificial neural networks. *Appl Math Comput* 195:591–597
- Fadlalla A, Lin CH (2001) An analysis of the applications of neural networks in finance. *Interfaces* 31(4): 112–122
- Fahlman S, Lebiere C (1990) The cascade-correlation learning architecture. In: Touretzky D (ed) *Advances in neural information processing systems*, vol 2. Morgan Kaufmann, Los Altos, CA, pp 524–532
- Faraway J, Chatfield C (1995) Time series forecasting with neural networks: a comparative study using the airline data. *Appl Stat* 47:231–250
- Findley DF (1985) Model selection for multi-step-ahead forecasting. In: Baker HA, Young PC (eds) *Proceedings of the seventh symposium on identification and system parameter estimation*. Pergamon, Oxford, New York, pp 1039–1044
- Franses PH, Draisma G (1997) Recognizing changing seasonal patterns using artificial neural networks. *J Econometrics* 81:273–280
- Frean M (1990) The Upstart algorithm: A method for constructing and training feed-forward networks. *Neural Comput* 2:198–209
- Freitas and Rodrigues (2006) Model combination in neural-based forecasting. *Eur J Oper Res* 173: 801–814
- Gately E (1996) Neural networks for financial forecasting. Wiley, New York
- Gautama AK, Chelanib AB, Jaina VK, Devotta S (2008) A new scheme to predict chaotic time series of air pollutant concentrations using artificial neural network and nearest neighbor searching. *Atmospheric Environ* 42:4409–4417
- Ghiassi M, Saidane H, Zimbra DK (2005) A dynamic artificial neural network model for forecasting time series events. *Int J Forecasting* 21(2):341–362
- Ghysels E, Granger CWJ, Siklos PL (1996) Is seasonal adjustment a linear or nonlinear data filtering process? *J Bus Econ Stat* 14:374–386
- Ginzburg I, Horn D (1994) Combined neural networks for time series analysis. *Adv Neural Info Process Syst* 6:224–231
- Goh YW, Lim PC, Peh KK (2003) Predicting drug dissolution profiles with an ensemble of boosted neural networks: A time series approach. *IEEE Trans Neural Netw* 14(2):459–463
- Gorr L (1994) Research prospective on neural network forecasting. *Int J Forecasting* 10:1–4
- Granger CWJ (1993) Strategies for modelling nonlinear time-series relationships. *Econ Rec* 69(206):233–238
- Hansen JV, Nelson RD (2003) Forecasting and recombining time-series components by using neural networks. *J Oper Res Soc* 54(3):307–317
- Hamzacebi C (2008) Improving artificial neural networks' performance in seasonal time series forecasting. *Inf Sci* 178:4550–4559
- Hamzacebi C, Akay D, Kutay F (2009) Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Syst Appl* 36(2):3839–3844
- Hill T, O'Connor M, Remus W (1996) Neural network models for time series forecasts. *Manag Sci* 42: 1082–1092
- Hippert HS, Pedreira CE, Souza RC (2001) Neural networks for short-term load forecasting: a review and evaluation. *IEEE Trans Power Syst* 16(1):44–55
- Hippert HS, Bunn DW, Souza RC (2005) Large neural networks for electricity load forecasting: are they overfitted? *Int J Forecasting* 21(3):425–434
- Hoptroff RG (1993) The principles and practice of time series forecasting and business modeling using neural networks. *Neural Comput Appl* 1:59–66
- Huskent M, Stagge P (2003) Recurrent neural networks for time series classification. *Neurocomputing* 50:223–235
- Hwang HB (2001) Insights into neural-network forecasting of time series corresponding to ARMA (p,q) structures. *Omega* 29:273–289
- Hylleberg S (1992) General introduction. In: Hylleberg S (ed) *Modelling seasonality*. Oxford University Press, Oxford, pp 3–14
- Hylleberg S (1994) Modelling seasonal variation. In: Hargreaves CP (ed) *Nonstationary time series analysis and cointegration*. Oxford University Press, Oxford, pp 153–178
- Ittig PT (1997) A seasonal index for business. *Decis Sci* 28(2):335–355
- Jain A, Kumar AM (2007) Hybrid neural network models for hydrologic time series forecasting. *Appl Soft Comput* 7:585–592

- Jiang X, Adeli H (2005) Dynamic wavelet neural network model for traffic flow forecasting. *J Transportation Eng* 131(10):771–779
- Kang I-B (2003) Multi-period forecasting using different models for different horizons: an application to U.S. economic time series data. *Int J Forecasting* 19:387–400
- Kaastra I, Boyd M (1996) Designing a neural network for forecasting financial and economic time series. *Neurocomputing* 10:215–236
- Kermanshahi B (1998) Recurrent neural network for forecasting next 10 years loads of nine Japanese utilities. *Neurocomputing* 23:125–133
- Khashei M, Hejazi SR, Bijari M (2008) A new hybrid artificial neural networks and fuzzy regression model for time series forecasting. *Fuzzy Sets Syst* 159:769–786
- Khotanzad A, Afkhami-Rohani R, Lu TL, Abaye A, Davis M, Maratukulam DJ (1997) ANNSTLF—A neural-network-based electric load forecasting system. *IEEE Trans Neural Netw* 8(4):835–846
- Kline DM (2004) Methods for multi-step time series forecasting with neural networks. In: Zhang GP (ed) *Neural networks for business forecasting*. Idea Group, Hershey, PA, pp 226–250
- Kolarik T, Rudorfer G (1994) Time series forecasting using neural networks. *APL Quote Quad* 25:86–94
- Krycha KA, Wagner U (1999) Applications of artificial neural networks in management science: a survey. *J Retailing Consum Serv* 6:185–203
- Kuan C-M, Liu T (1995) Forecasting exchange rates using feedforward and recurrent neural networks. *J Appl Economet* 10:347–364
- Lapedes A, Farber R (1987) Nonlinear signal processing using neural networks: Prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM
- LeBaron B, Weigend AS (1998) A bootstrap evaluation of the effect of data splitting on financial time series. *IEEE Trans Neural Netw* 9(1):213–220
- Lennon B, Montague GA, Frith AM, Gent C, Bevan V (2001) Industrial applications of neural networks—An investigation. *J Process Control* 11:497–507
- Liang F (2005) Bayesian neural networks for nonlinear time series forecasting. *Stat Comput* 15:13–29
- Lin J-L, Granger CWJ (1994) Forecasting from non-linear models in practice. *J Forecasting* 13:1–9
- Liu Y, Yao X (1999) Ensemble learning via negative correlation. *Neural Netw* 12:1399–1404
- Maier HR, Dandy GC (2000) Neural networks for the prediction and forecasting of water resource variables: a review of modeling issues and applications. *Environ Model Software* 15:101–124
- Makridakis S, Anderson A, Carbone R, Fildes R, Hibdon M, Lewandowski R, Newton J, Parzen E, Winkler R (1982) The accuracy of extrapolation (time series) methods: results of a forecasting competition. *J Forecasting* 1(2):111–153
- Mandic D, Chambers J (2001) *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, Chichester, UK
- Marcellino M, Stock JH, Watson MW (2006) A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series. *J Economet* 135:499–526
- Medeiros MC, Pedreira CE (2001) What are the effects of forecasting linear time series with neural networks? *Eng Intell Syst* 4:237–424
- Medeiros MC, Veiga A (2000) A hybrid linear-neural model for time series forecasting. *IEEE Trans Neural Netw* 11(6):1402–1412
- Medeiros MC, Terasvirta T, Rech G (2006) Building neural network models for time series: a statistical approach. *J Forecasting* 25:49–75
- Menezes JMP, Barreto GA (2008) Long-term time series prediction with the NARX network: an empirical evaluation. *Neurocomputing* 71:3335–3343
- Mishra AK, Desai VR (2006) Drought forecasting using feed-forward recursive neural network. *Ecol Model* 198:127–138
- Nakamura E (2005) Inflation forecasting using a neural network. *Econ Lett* 86:373–378
- Nelson M, Hill T, Remus T, O'Connor M (1999) Time series forecasting using neural networks: should the data be deseasonalized first? *J Forecasting* 18: 359–367
- Palmer A, Montano JJ, Sese A (2006) Designing an artificial neural network for forecasting tourism time series. *Tourism Manag* 27:781–790
- Parisi A, Parisi F, Diaz D (2008) Forecasting gold price changes: rolling and recursive neural network models. *J Multinational Financial Manag* 18(5):477–487
- Parlos AG, Rais OT, Atiya AF (2000) Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural Netw* 13:765–786
- Pelikan E, de Groot C, Wurtz D (1992) Power consumption in West-Bohemia: improved forecasts with decorrelating connectionist networks. *Neural Netw World* 2(6):701–712
- Pino P, Parreno J, Gomez A, Priore P (2008) Forecasting next-day price of electricity in the Spanish energy market using artificial neural networks. *Eng Appl Artif Intell* 21:53–62
- Poli I, Jones DR (1994) A neural net model for prediction. *J Am Stat Assoc* 89:117–121
- Qi M, Zhang GP (2001) An investigation of model selection criteria for neural network time series forecasting. *Eur J Oper Res* 132:666–680
- Qi M, Zhang GP (2008) Trend time-series modeling and forecasting with neural networks. *IEEE Trans Neural Netw* 19(5):808–816

- Reed R (1993) Pruning algorithms—A survey. *IEEE Trans Neural Netw* 4(5):740–747
- Remus W, O'Connor M (2001) Neural networks for time series forecasting. In: Armstrong JS (ed) *Principles of forecasting: a handbook for researchers and practitioners*. Kluwer, Norwell, MA, pp 245–256
- Ripley BD (1996) *Pattern recognition and neural networks*. Cambridge University Press, Cambridge
- Rumelhart DE, McClelland JL, PDP Research Group (1986) Parallel distributed processing: explorations in the microstructure of cognition, *Foundations*, vol 1. MIT Press, Cambridge, MA
- Sharda R, Patil RB (1992) Connectionist approach to time series prediction: an empirical test. *J Intell Manufacturing* 3:317–323
- Sharkey CJ (1996) On combining artificial neural nets. *Connect Sci* 8:299–314
- Sharkey CJ, Sharkey EN (1997) Combining diverse neural nets. *Knowledge Eng Rev* 12(3):231–247
- Smith M (1993) *Neural networks for statistical modeling*. Van Nostrand Reinhold, New York
- Stoica P, Nehorai A (1989) On multi-step prediction errors methods for time series models. *J Forecasting* 13:109–131
- Swanson NR, White H (1995) A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks. *J Bus Econ Stat* 13:265–275
- Tiao GC, Tsay RS (1994) Some advances in non-linear and adaptive modeling in time-series. *J Forecasting* 13:109–131
- Teräsvirta T, van Dijk D, Medeiros MC (2005) Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: a re-examination. *Int J Forecasting* 21(4):755–774
- Vellido A, Lisboa PJG, Vaughan J (1999) Neural networks in business: a survey of applications (1992–1998). *Expert Syst Appl* 17:51–70
- Vermaak J, Botha EC (1998) Recurrent neural networks for short-term load forecasting. *IEEE Trans Power Syst* 13(1):126–132
- Wang Y-H (2009) Nonlinear neural network forecasting model for stock index option price: hybrid GJR-GARCH approach. *Expert Syst Appl* 36(1):564–570
- Wedding K II, Cios JK (1996) Time series forecasting by combining RBF networks, certainty factors, and the Box-Jenkins model. *Neurocomputing* 10:149–168
- Weigend AS, Gershenfeld NA (1994) *Time series prediction: forecasting the future and understanding the past*. Addison-Wesley, Reading, MA
- Weigend AS, Huberman BA, Rumelhart DE (1990) Predicting the future: a connectionist approach. *Int J Neural Syst* 1:193–209
- Weigend AS, Huberman BA, Rumelhart DE (1992) Predicting sunspots and exchange rates with connectionist networks. In: Casdagli M, Eubank S (eds) *Nonlinear modeling and forecasting*. Addison-Wesley, Redwood City, CA, pp 395–432
- Werbos P (1974) Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University
- Wichard J, Ogorzalek M (2007) Time series prediction with ensemble models applied to the CATS benchmark. *Neurocomputing* 70:2371–2378
- Wong BK, Selvi Y (1998) Neural network applications in finance: a review and analysis of literature (1990–1996). *Inf Manag* 34:129–139
- Wong BK, Lai VS, Lam J (2000) A bibliography of neural network business applications research: 1994–1998. *Comput Oper Res* 27:1045–1076
- Xiao Z, Ye SJ, Zhong B, Sun CX (2009) BP neural network with rough set for short term load forecasting. *Expert Syst Appl* 36(1):273–279
- Zhang G, Patuwo EP, Hu MY (1998) Forecasting with artificial neural networks: the state of the art. *Int J Forecasting* 14:35–62
- Zhang GP (2001) An investigation of neural networks for linear time-series forecasting. *Comput Oper Res* 28:1183–1202
- Zhang GP (2003) Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50:159–175
- Zhang GP (2004) Neural networks in business forecasting. Idea Group, Hershey, PA
- Zhang GP (2007a) Avoiding pitfalls in neural network research. *IEEE Trans Syst Man and Cybern* 37:3–16
- Zhang GP (2007b) A neural network ensemble method with jittered training data for time series forecasting. *Inf Sci* 177:5329–5346
- Zhang GP, Berardi LV (2001) Time series forecasting with neural network ensembles: An application for exchange rate prediction. *J Oper Res Soc* 52(6):652–664
- Zhang GP, Kline D (2007) Quarterly time-series forecasting with neural networks. *IEEE Trans Neural Netw* 18(6):1800–1814
- Zhang GP, Qi M (2002) Predicting consumer retail sales using neural networks. In: Smith K, Gupta J (eds) *Neural networks in business: techniques and applications*. Idea Group, Hershey, PA, pp 26–40
- Zhang GP, Qi M (2005) Neural network forecasting for seasonal and trend time series. *Eur J Oper Res* 160(2):501–514
- Zhang GP, Patuwo EP, Hu MY (2001) A simulation study of artificial neural networks for nonlinear time series forecasting. *Comput Oper Res* 28:381–396
- Zhang YQ, Wan X (2007) Statistical fuzzy interval neural networks for currency exchange rate time series prediction. *Appl Soft Comput* 7:1149–1156
- Zou HF, Xia GP, Yang FT, Wang HY (2007) An investigation and comparison of artificial neural network and time series models for Chinese food grain price forecasting. *Neurocomputing* 70:2913–2923

15 SVM Tutorial — Classification, Regression and Ranking

Hwanjo Yu¹ · Sungchul Kim²

¹Data Mining Lab, Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, South Korea
hwanjoyu@postech.ac.kr

²Data Mining Lab, Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, South Korea
subright@postech.ac.kr

1	<i>Introduction</i>	480
2	<i>SVM Classification</i>	480
3	<i>SVM Regression</i>	491
4	<i>SVM Ranking</i>	493
5	<i>Ranking Vector Machine: An Efficient Method for Learning the 1-Norm Ranking SVM</i>	496

Abstract

Support vector machines (SVMs) have been extensively researched in the data mining and machine learning communities for the last decade, and applied in various domains. They represent a set of supervised learning techniques that create a function from training data, which usually consists of pairs of an input object, typically vectors, and a desired output. SVMs learn a function that generates the desired output given the input, and the learned function can be used to predict the output of a new object. They belong to a family of generalized linear classifier where the classification (or boundary) function is a hyperplane in the feature space. This chapter introduces the basic concepts and techniques of SVMs for learning classification, regression, and ranking functions.

1 Introduction

Support vector machines are typically used for learning classification, regression, or ranking functions, for which they are called classifying SVM, support vector regression (SVR), or ranking SVM (RankSVM), respectively. Two special properties of SVMs are that they achieve high generalization by maximizing the margin, and they support an efficient learning of nonlinear functions using the kernel trick. This chapter introduces these general concepts and techniques of SVMs for learning classification, regression, and ranking functions. In particular, the SVMs for binary classification are first presented in [Sect. 2](#), SVR in [Sect. 3](#), ranking SVM in [Sect. 4](#), and another recently developed method for learning ranking SVM called Ranking Vector Machine (RVM) in [Sect. 5](#).

2 SVM Classification

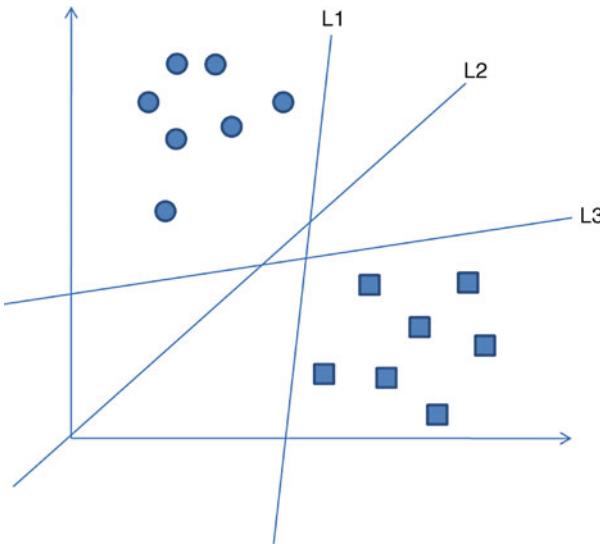
SVMs were initially developed for classification (Burges 1998) and have been extended for regression (Smola and Schölkopf 1998) and preference (or rank) learning (Herbrich et al. 2000; Yu 2005). The initial form of SVMs is a binary classifier where the output of the learned function is either positive or negative. A multiclass classification can be implemented by combining multiple binary classifiers using the pairwise coupling method (Hastie and Tibshirani 1998; Friedman 1998). This section explains the motivation and formalization of SVM as a binary classifier, and the two key properties – margin maximization and the kernel trick.

Binary SVMs are classifiers that discriminate data points of two categories. Each data object (or data point) is represented by an n -dimensional vector. Each of these data points belongs to only one of two classes. A *linear* classifier separates them with a hyperplane. For example, [Fig. 1](#) shows two groups of data and separating hyperplanes that are lines in a two-dimensional space. There are many linear classifiers that correctly classify (or divide) the two groups of data such as L1, L2, and L3 in [Fig. 1](#). In order to achieve maximum separation between the two classes, the SVM picks the hyperplane that has the largest margin. The margin is the summation of the shortest distance from the separating hyperplane to the nearest data point of both categories. Such a hyperplane is likely to generalize better, meaning that the hyperplane can correctly classify “unseen” or testing data points.

SVMs do the mapping from the input space to the feature space to support nonlinear classification problems. The kernel trick is helpful for doing this by allowing the absence of the

Fig. 1

Linear classifiers (hyperplane) in two-dimensional spaces.



exact formulation of the mapping function which could introduce a case of the curse of dimensionality problem. This makes a linear classification in the new space (or the feature space) equivalent to nonlinear classification in the original space (or the input space). SVMs do this by mapping input vectors to a higher dimensional space (or feature space) where a maximal separating hyperplane is constructed.

2.1 Hard-Margin SVM Classification

To understand how SVMs compute the hyperplane of maximal margin and support nonlinear classification, we first explain the hard-margin SVM where the training data is free of noise and can be correctly classified by a linear function.

The data points D in Fig. 1 (or training set) can be expressed mathematically as follows:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\} \quad (1)$$

where \mathbf{x}_i is an n -dimensional real vector, y_i is either 1 or -1 denoting the class to which the point \mathbf{x}_i belongs. The SVM classification function $F(\mathbf{x})$ takes the form

$$F(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - b \quad (2)$$

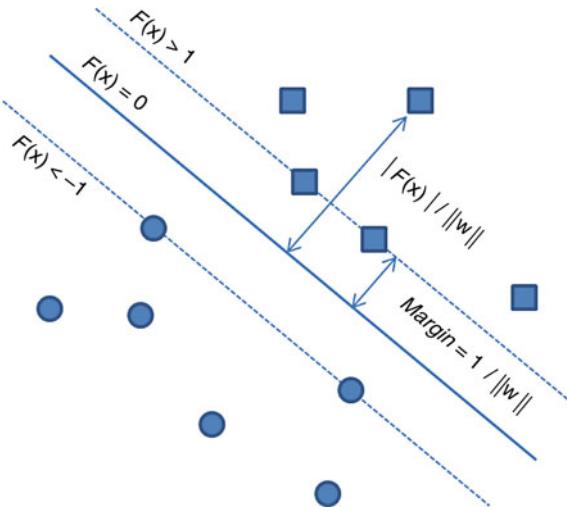
\mathbf{w} is the weight vector and b is the bias, which will be computed by the SVM in the training process.

First, to correctly classify the training set, $F(\cdot)$ (or \mathbf{w} and b) must return positive numbers for positive data points and negative numbers otherwise, that is, for every point \mathbf{x}_i in D ,

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i - b &> 0 & \text{if } y_i = 1, \text{ and} \\ \mathbf{w} \cdot \mathbf{x}_i - b &< 0 & \text{if } y_i = -1 \end{aligned}$$

Fig. 2

SVM classification function: the hyperplane maximizing the margin in a two-dimensional space.



These conditions can be revised into

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) > 0, \forall (\mathbf{x}_i, y_i) \in D \quad (3)$$

If there exists such a linear function F that correctly classifies every point in D or satisfies **Eq. 3**, D is called *linearly separable*.

Second, F (or the hyperplane) needs to maximize the *margin*. The margin is the distance from the hyperplane to the closest data points. An example of such a hyperplane is illustrated in **Fig. 2**. To achieve this, **Eq. 3** is revised into the following:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall (\mathbf{x}_i, y_i) \in D \quad (4)$$

Note that **Eq. 4** includes the equality sign, and the right side becomes 1 instead of 0. If D is linearly separable, or every point in D satisfies **Eq. 3**, then there exists such an F that satisfies **Eq. 4**. This is because, if there exist such \mathbf{w} and b that satisfy **Eq. 3**, they can always be rescaled to satisfy **Eq. 4**.

The distance from the hyperplane to a vector \mathbf{x}_i is formulated as $\frac{|F(\mathbf{x}_i)|}{\|\mathbf{w}\|}$. Thus, the margin becomes

$$\text{margin} = \frac{1}{\|\mathbf{w}\|} \quad (5)$$

because when \mathbf{x}_i are the closest vectors, $F(\mathbf{x})$ will return to 1 according to **Eq. 4**. The closest vectors that satisfy **Eq. 4** with the equality sign are called *support vectors*.

Maximizing the margin becomes minimizing $\|\mathbf{w}\|$. Thus, the training problem in an SVM becomes a constrained optimization problem as follows:

$$\text{minimize: } Q(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \quad (6)$$

$$\text{subject to: } y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall (\mathbf{x}_i, y_i) \in D \quad (7)$$

The factor of $\frac{1}{2}$ is used for mathematical convenience.

2.1.1 Solving the Constrained Optimization Problem

The constrained optimization problem (Eq. 6) and (Eq. 7) is called a *primal problem*. It is characterized as follows:

- The objective function (Eq. 6) is a *convex* function of \mathbf{w} .
- The constraints are *linear* in \mathbf{w} .

Accordingly, we may solve the constrained optimization problem using the method of Lagrange multipliers (Bertsekas 1995). First, we construct the Lagrange function

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^m \alpha_i \{y_i(\mathbf{w} \cdot \mathbf{x}_i - b) - 1\} \quad (8)$$

where the auxiliary nonnegative variables α are called Lagrange multipliers. The solution to the constrained optimization problem is determined by the saddle point of the Lagrange function $J(\mathbf{w}, b, \alpha)$, which has to be minimized with respect to \mathbf{w} and b ; it also has to be maximized with respect to α . Thus, differentiating $J(\mathbf{w}, b, \alpha)$ with respect to \mathbf{w} and b and setting the results equal to zero, we get the following two conditions of optimality:

$$\text{Condition 1: } \frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \quad (9)$$

$$\text{Condition 2: } \frac{\partial J(\mathbf{w}, b, \alpha)}{\partial b} = 0 \quad (10)$$

After rearrangement of terms, Condition 1 yields

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \quad (11)$$

and Condition 2 yields

$$\sum_{i=1}^m \alpha_i y_i = 0 \quad (12)$$

The solution vector \mathbf{w} is defined in terms of an expansion that involves the m training examples.

As noted earlier, the primal problem deals with a convex cost function and linear constraints. Given such a constrained optimization problem, it is possible to construct another problem called the *dual problem*. The dual problem has the same optimal value as the primal problem, but with the Lagrange multipliers providing the optimal solution.

To postulate the dual problem for the primal problem, Eq. 8 is first expanded term by term, as follows:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - b \sum_{i=1}^m \alpha_i y_i + \sum_{i=1}^m \alpha_i \quad (13)$$

The third term on the right-hand side of Eq. 13 is zero by virtue of the optimality condition of Eq. 12. Furthermore, from Eq. 11, we have

$$\mathbf{w} \cdot \mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (14)$$

Accordingly, setting the objective function $J(\mathbf{w}, b, \alpha) = Q(\alpha)$, [Eq. 13](#) can be formulated as

$$Q(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (15)$$

where the α_i are nonnegative.

The dual problem can be now stated as follows:

$$\text{maximize: } Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (16)$$

$$\text{subject to: } \sum_i \alpha_i y_i = 0 \quad (17)$$

$$\alpha \geq 0 \quad (18)$$

Note that the dual problem is cast entirely in terms of the training data. Moreover, the function $Q(\alpha)$ to be maximized depends only on the input patterns in the form of a set of dot products $\{\mathbf{x}_i \cdot \mathbf{x}_j\}_{(i,j)=1}^m$.

Having determined the optimum Lagrange multipliers, denoted by α_i^* , the optimum weight vector \mathbf{w}^* may be computed using [Eq. 11](#) and so can be written as

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \quad (19)$$

Note that according to the property of Kuhn–Tucker conditions of optimization theory, the solution of the dual problem α_i^* must satisfy the following condition:

$$\alpha_i^* \{y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) - 1\} = 0 \quad \text{for } i = 1, 2, \dots, m \quad (20)$$

and either α_i^* or its corresponding constraint $\{y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) - 1\}$ must be nonzero. This condition implies that only when \mathbf{x}_i is a support vector or $y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) = 1$, its corresponding coefficient α_i will be nonzero (or nonnegative from [Eq. 18](#)). In other words, the \mathbf{x}_i whose corresponding coefficients α_i are zero will not affect the optimum weight vector \mathbf{w}^* due to [Eq. 19](#). Thus, the optimum weight vector \mathbf{w}^* will only depend on the support vectors whose coefficients are nonnegative.

Once the nonnegative α_i^* and their corresponding support vectors are computed, we can compute the bias b using a positive support vector \mathbf{x}_i from the following equation:

$$b^* = 1 - \mathbf{w}^* \cdot \mathbf{x}_i \quad (21)$$

The classification of [Eq. 2](#) now becomes

$$F(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} - b \quad (22)$$

2.2 Soft-Margin SVM Classification

The discussion so far has focused on linearly separable cases. However, the optimization problem [Eqs. 6](#) and [7](#) will not have a solution if D is not linearly separable. To deal with such cases, a *soft margin* SVM allows mislabeled data points while still maximizing the margin. The method introduces slack variables, ξ_p , which measure the degree of misclassification. The following is the optimization problem for a soft margin SVM.

$$\text{minimize: } Q_1(w, b, \xi_i) = \frac{1}{2} ||w||^2 + C \sum_i \xi_i \quad (23)$$

$$\text{subject to: } y_i(w \cdot \mathbf{x}_i - b) \geq 1 - \xi_i, \quad \forall (\mathbf{x}_i, y_i) \in D \quad (24)$$

$$\xi_i \geq 0 \quad (25)$$

Due to the ξ_i in [Eq. 24](#), data points are allowed to be misclassified, and the amount of misclassification will be minimized while maximizing the margin according to the objective function ([Eq. 23](#)). C is a parameter that determines the trade-off between the margin size and the amount of error in training.

Similarly to the case of hard-margin SVM, this primal form can be transformed to the following dual form using the Lagrange multipliers:

$$\text{maximize: } Q_2(\alpha) = \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (26)$$

$$\text{subject to: } \sum_i \alpha_i y_i = 0 \quad (27)$$

$$C \geq \alpha \geq 0 \quad (28)$$

Note that neither the slack variables ξ_i nor their Lagrange multipliers appear in the dual problem. The dual problem for the case of nonseparable patterns is thus similar to that for the simple case of linearly separable patterns except for a minor but important difference. The objective function $Q(\alpha)$ to be maximized is the same in both cases. The nonseparable case differs from the separable case in that the constraint $\alpha_i \geq 0$ is replaced with the more stringent constraint $C \geq \alpha_i \geq 0$. Except for this modification, the constrained optimization for the nonseparable case and computations of the optimum values of the weight vector w and bias b proceed in the same way as in the linearly separable case.

Just as for the hard-margin SVM, α constitutes a dual representation for the weight vector such that

$$\mathbf{w}^* = \sum_{i=1}^{m_s} \alpha_i^* y_i \mathbf{x}_i \quad (29)$$

where m_s is the number of support vectors whose corresponding coefficient $\alpha_i > 0$. The determination of the optimum values of the bias also follows a procedure similar to that described before. Once α and b are computed, the function [Eq. 22](#) is used to classify new objects.

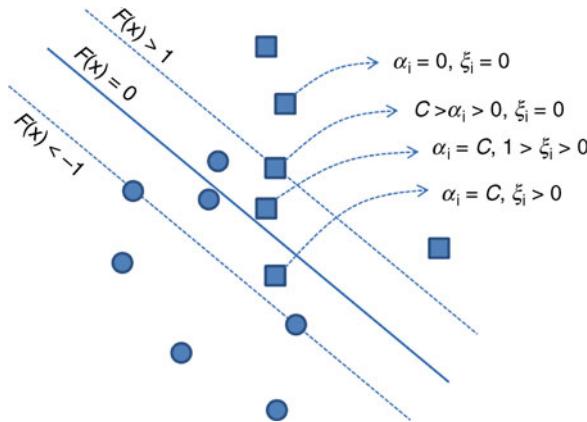
Relationships among α , ξ , and C can be further disclosed using the Kuhn–Tucker conditions that are defined by

$$\alpha_i \{y_i(w \cdot \mathbf{x}_i - b) - 1 + \xi_i\} = 0, \quad i = 1, 2, \dots, m \quad (30)$$

and

$$\mu_i \xi_i = 0, \quad i = 1, 2, \dots, m \quad (31)$$

[Eq. 30](#) is a rewrite of [Eq. 20](#) except for the replacement of the unity term $(1 - \xi_i)$. As for [Eq. 31](#), the μ_i are Lagrange multipliers that have been introduced to enforce the nonnegativity of the slack variables ξ_i for all i . At the saddle point, the derivative of the

Fig. 3Graphical relationships among α_i , ξ_i , and C .

Lagrange function for the primal problem with respect to the slack variable ξ_i is zero, the evaluation of which yields

$$\alpha_i + \mu_i = C \quad (32)$$

By combining [Eqs. 31](#) and [32](#), we see that

$$\xi_i = 0 \quad \text{if } \alpha_i < C, \text{ and} \quad (33)$$

$$\xi_i \geq 0 \quad \text{if } \alpha_i = C \quad (34)$$

We can graphically display the relationships among α_i , ξ_i , and C in [Fig. 3](#).

Data points outside the margin will have $\alpha = 0$ and $\xi = 0$ and those on the margin line will have $C > \alpha > 0$ and still $\xi = 0$. Data points within the margin will have $\alpha = C$. Among them, those correctly classified will have $1 > \xi > 0$ and misclassified points will have $\xi > 1$.

2.3 Kernel Trick for Nonlinear Classification

If the training data is not linearly separable, there is no straight hyperplane that can separate the classes. In order to learn a nonlinear function in that case, linear SVMs must be extended to nonlinear SVMs for the classification of nonlinearly separable data. The process of finding classification functions using nonlinear SVMs consists of two steps. First, the input vectors are transformed into high-dimensional feature vectors where the training data can be linearly separated. Then, SVMs are used to find the hyperplane of maximal margin in the new feature space. The separating hyperplane becomes a linear function in the transformed feature space but a nonlinear function in the original input space.

Let \mathbf{x} be a vector in the n -dimensional input space and $\varphi(\cdot)$ be a nonlinear mapping function from the input space to the high-dimensional feature space. The hyperplane representing the decision boundary in the feature space is defined as follows:

$$\mathbf{w} \cdot \varphi(\mathbf{x}) - b = 0 \quad (35)$$

where \mathbf{w} denotes a weight vector that can map the training data in the high-dimensional feature space to the output space, and b is the bias. Using the $\varphi(\cdot)$ function, the weight becomes

$$\mathbf{w} = \sum \alpha_i y_i \varphi(\mathbf{x}_i) \quad (36)$$

The decision function of [Eq. 22](#) becomes

$$F(\mathbf{x}) = \sum_i^m \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) - b \quad (37)$$

Furthermore, the dual problem of the soft-margin SVM ([Eq. 26](#)) can be rewritten using the mapping function on the data vectors as follows:

$$Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) \quad (38)$$

holding the same constraints.

Note that the feature mapping functions in the optimization problem and also in the classifying function always appear as dot products, for example, $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$. $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is the inner product between pairs of vectors in the transformed feature space. Computing the inner product in the transformed feature space seems to be quite complex and suffers from the curse of dimensionality problem. To avoid this problem, the kernel trick is used. The kernel trick replaces the inner product in the feature space with a kernel function K in the original input space as follows.

$$K(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u}) \cdot \varphi(\mathbf{v}) \quad (39)$$

Mercer's theorem proves that a kernel function K is valid if and only if the following conditions are satisfied, for any function $\psi(\mathbf{x})$ (refer to Christianini and Shawe-Taylor ([2000](#)) for the proof in detail):

$$\int K(\mathbf{u}, \mathbf{v}) \psi(\mathbf{u}) \psi(\mathbf{v}) d\mathbf{x} d\mathbf{y} \leq 0 \quad (40)$$

where $\int \psi(x)^2 dx \leq 0$

Mercer's theorem ensures that the kernel function can be always expressed as the inner product between pairs of input vectors in some high-dimensional space, thus the inner product can be calculated using the kernel function only with input vectors in the original space without transforming the input vectors into the high-dimensional feature vectors.

The dual problem is now defined using the kernel function as follows:

$$\text{maximize: } Q_2(\alpha) = \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (41)$$

$$\text{subject to: } \sum_i \alpha_i y_i = 0 \quad (42)$$

$$C \geq \alpha \geq 0 \quad (43)$$

The classification function becomes

$$F(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - b \quad (44)$$

Since $K(\cdot)$ is computed in the input space, no feature transformation will actually be done or no $\varphi(\cdot)$ will be computed, and thus the weight vector $\mathbf{w} = \sum \alpha_i y_i \varphi(\mathbf{x})$ will not be computed either in nonlinear SVMs.

The following are popularly used kernel functions:

- Polynomial: $K(a, b) = (a \cdot b + 1)^d$
- Radial Basis Function (RBF): $K(a, b) = \exp(-\gamma \|a - b\|^2)$
- Sigmoid: $K(a, b) = \tanh(\kappa a \cdot b + c)$

Note that the kernel function is a kind of similarity function between two vectors where the function output is maximized when the two vectors become equivalent. Because of this, SVM can learn a function from any shapes of data beyond vectors (such as trees or graphs) as long as we can compute a similarity function between any pair of data objects. Further discussions on the properties of these kernel functions are out of the scope of this chapter. Instead, we will give an example of using the polynomial kernel for learning an XOR function in the following section.

2.3.1 Example: XOR Problem

To illustrate the procedure of training a nonlinear SVM function, assume that the training set of  [Table 1](#) is given.

 [Figure 4](#) plots the training points in the 2D input space. There is no linear function that can separate the training points.

To proceed let

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x} \cdot \mathbf{x}_i)^2 \quad (45)$$

If we denote $\mathbf{x} = (x_1, x_2)$ and $\mathbf{x}_i = (x_{i1}, x_{i2})$, the kernel function is expressed in terms of monomials of various orders as follows.

$$K(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} \quad (46)$$

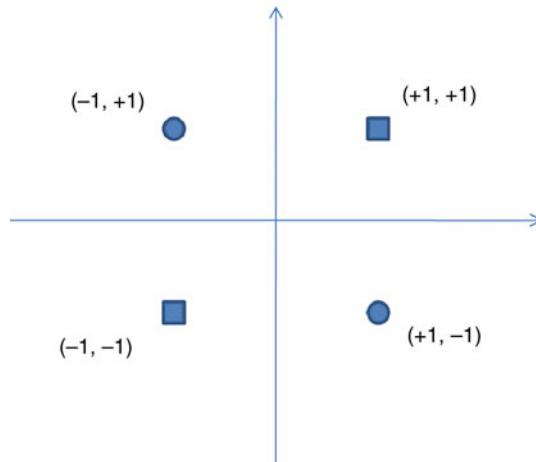
The image of the input vector \mathbf{x} induced in the feature space is therefore deduced to be

$$\varphi(\mathbf{x}) = (1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2) \quad (47)$$

 **Table 1**
XOR problem

Input vector \mathbf{x}	Desired output y
(-1, -1)	-1
(-1, +1)	+1
(+1, -1)	+1
(+1, +1)	-1

Fig. 4
XOR problem.



Based on this mapping function, the objective function for the dual form can be derived from [Eq. 41](#) as follows.

$$\begin{aligned} Q(\alpha) &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ &\quad - \frac{1}{2} (9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ &\quad + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + \alpha_4^2) \end{aligned} \tag{48}$$

Optimizing $Q(\alpha)$ with respect to the Lagrange multipliers yields the following set of simultaneous equations:

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

Hence, the optimal values of the Lagrange multipliers are

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$$

This result denotes that all four input vectors are support vectors and the optimum value of $Q(\alpha)$ is

$$Q(\alpha) = \frac{1}{4}$$

and

$$\frac{1}{2} \|w\|^2 = \frac{1}{4}, \quad \text{or} \quad \|w\| = \frac{1}{\sqrt{2}}$$

From [Eq. 36](#), we find that the optimum weight vector is

$$\begin{aligned} w &= \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)] \\ &= \frac{1}{8} \left[- \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (49)$$

The bias b is 0 because the first element of w is 0. The optimal hyperplane becomes

$$w \cdot \varphi(\mathbf{x}) = [0 \quad 0 \quad \frac{-1}{\sqrt{2}} \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0 \quad (50)$$

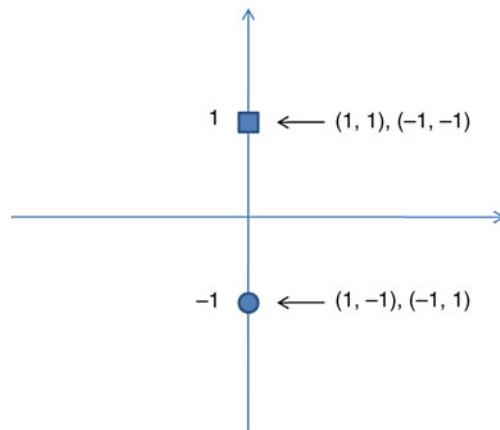
which reduces to

$$-x_1x_2 = 0 \quad (51)$$

this is the optimal hyperplane, the solution of the XOR problem. It makes the output $y = 1$ for both input points $x_1 = x_2 = 1$ and $x_1 = x_2 = -1$, and $y = -1$ for both input points $x_1 = 1, x_2 = -1$ or $x_1 = -1, x_2 = 1$. [Figure 5](#) represents the four points in the transformed feature space.

Fig. 5

The four data points of the XOR problem in the transformed feature space.



3 SVM Regression

SVM regression (SVR) is a method to estimate a function that maps from an input object to a real number based on training data. Similarly to the classifying SVM, SVR has the same properties of the margin maximization and kernel trick for nonlinear mapping.

A training set for regression is represented as follows.

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\} \quad (52)$$

where \mathbf{x}_i is a n -dimensional vector, y is the real number for each \mathbf{x}_i . The SVR function $F(\mathbf{x}_i)$ makes a mapping from an input vector \mathbf{x}_i to the target y_i and takes the form:

$$F(\mathbf{x}) \Rightarrow \mathbf{w} \cdot \mathbf{x} - b \quad (53)$$

where \mathbf{w} is the weight vector and b is the bias. The goal is to estimate the parameters (\mathbf{w} and b) of the function that give the best fit of the data. An SVR function $F(\mathbf{x})$ approximates all pairs (\mathbf{x}_i, y_i) while maintaining the differences between estimated values and real values under ε precision. That is, for every input vector \mathbf{x} in D ,

$$y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leq \varepsilon \quad (54)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b - y_i \leq \varepsilon \quad (55)$$

The margin is

$$\text{margin} = \frac{1}{\|\mathbf{w}\|} \quad (56)$$

By minimizing $\|\mathbf{w}\|^2$ to maximize the margin, the training in SVR becomes a constrained optimization problem, as follows:

$$\text{minimize: } L(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \quad (57)$$

$$\text{subject to: } y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leq \varepsilon \quad (58)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b - y_i \leq \varepsilon \quad (59)$$

The solution of this problem does not allow any errors. To allow some errors to deal with noise in the training data, the soft margin SVR uses slack variables ξ and $\hat{\xi}$. Then the optimization problem can be revised as follows:

$$\text{minimize: } L(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i (\xi_i^2, \hat{\xi}_i^2), C > 0 \quad (60)$$

$$\text{subject to: } y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leq \varepsilon + \xi_i, \quad \forall (\mathbf{x}_i, y_i) \in D \quad (61)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b - y_i \leq \varepsilon + \hat{\xi}_i, \quad \forall (\mathbf{x}_i, y_i) \in D \quad (62)$$

$$\xi, \hat{\xi} \geq 0 \quad (63)$$

The constant $C > 0$ is the trade-off parameter between the margin size and the number of errors. The slack variables ξ and $\hat{\xi}$ deal with infeasible constraints of the optimization problem by imposing a penalty on excess deviations that are larger than ε .

To solve the optimization problem [Eq. 60](#), we can construct a Lagrange function from the objective function with Lagrange multipliers as follows:

$$\begin{aligned} \text{minimize: } L = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i (\xi_i + \hat{\xi}_i) - \sum_i (\eta_i \xi_i + \hat{\eta}_i \hat{\xi}_i) \\ & - \sum_i \alpha_i (\varepsilon + \eta_i - \gamma_i + \mathbf{w} \cdot \mathbf{x}_i + b) \\ & - \sum_i \hat{\alpha}_i (\varepsilon + \hat{\eta}_i + \gamma_i - \mathbf{w} \cdot \mathbf{x}_i - b) \end{aligned} \quad (64)$$

$$\text{subject to: } \eta, \hat{\eta}_i \geq 0 \quad (65)$$

$$\alpha, \hat{\alpha}_i \geq 0 \quad (66)$$

where η_i , $\hat{\eta}_i$, α , and $\hat{\alpha}_i$ are the Lagrange multipliers that satisfy positive constraints. The following is the process to find the saddle point by using the partial derivatives of L with respect to each Lagrangian multiplier for minimizing the function L :

$$\frac{\partial L}{\partial b} = \sum_i (\alpha_i - \hat{\alpha}_i) = 0 \quad (67)$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i (\alpha_i - \hat{\alpha}_i) \mathbf{x}_i = 0, \mathbf{w} = \sum_i (\alpha_i - \hat{\alpha}_i) \mathbf{x}_i \quad (68)$$

$$\frac{\partial L}{\partial \xi_i} = C - \hat{\alpha}_i - \hat{\eta}_i = 0, \hat{\eta}_i = C - \hat{\alpha}_i \quad (69)$$

The optimization problem with inequality constraints can be changed to the following dual optimization problem by substituting [Eqs. 67](#), [68](#), and [69](#) into [64](#).

$$\begin{aligned} \text{maximize: } L(\alpha) = & \sum_i \gamma_i (\alpha_i - \hat{\alpha}_i) - \varepsilon \sum_i (\alpha_i + \hat{\alpha}_i) \\ & - \frac{1}{2} \sum_i \sum_j (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \mathbf{x}_i \mathbf{x}_j \end{aligned} \quad (70)$$

$$\text{subject to: } \sum_i (\alpha_i - \hat{\alpha}_i) = 0 \quad (72)$$

$$0 \leq \alpha, \hat{\alpha} \leq C \quad (73)$$

The dual variables $\eta, \hat{\eta}_i$ are eliminated in revising [Eq. 64](#) into [Eq. 70](#). [Eqs. 68](#) and [69](#) can be rewritten as follows:

$$\mathbf{w} = \sum_i (\alpha_i - \hat{\alpha}_i) \mathbf{x}_i \quad (74)$$

$$\eta_i = C - \alpha_i \quad (75)$$

$$\hat{\eta}_i = C - \hat{\alpha}_i \quad (76)$$

where w is represented by a linear combination of the training vectors \mathbf{x}_i . Accordingly, the SVR function $F(\mathbf{x})$ becomes the following function:

$$F(\mathbf{x}) \Rightarrow \sum_i (\alpha_i - \hat{\alpha}_i) \mathbf{x}_i \cdot \mathbf{x} + b \quad (77)$$

❷ Eq. 77 can map the training vectors to target real values which allow some errors but it cannot handle the nonlinear SVR case. The same kernel trick can be applied by replacing the inner product of two vectors $\mathbf{x}_i, \mathbf{x}_j$ with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. The transformed feature space is usually high dimensional, and the SVR function in this space becomes nonlinear in the original input space. Using the kernel function K , the inner product in the transformed feature space can be computed as fast as the inner product $\mathbf{x}_i \cdot \mathbf{x}_j$ in the original input space. The same kernel functions introduced in ❷ Sect. 2.3 can be applied here.

Once we replace the original inner product with a kernel function K , the remaining process for solving the optimization problem is very similar to that for the linear SVR. The linear optimization function can be changed by using the kernel function as follows:

$$\begin{aligned} \text{maximize: } L(\alpha) = & \sum_i y_i (\alpha_i - \hat{\alpha}_i) - \varepsilon \sum_i (\alpha_i + \hat{\alpha}_i) \\ & - \frac{1}{2} \sum_i \sum_j (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \quad (78)$$

$$\text{subject to: } \sum_i (\alpha_i - \hat{\alpha}_i) = 0 \quad (79)$$

$$\hat{\alpha}_i \geq 0, \alpha_i \geq 0 \quad (80)$$

$$0 \leq \alpha, \hat{\alpha} \leq C \quad (81)$$

Finally, the SVR function $F(\mathbf{x})$ becomes the following using the kernel function:

$$F(\mathbf{x}) \Rightarrow \sum_i (\hat{\alpha}_i - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b \quad (82)$$

4 SVM Ranking

Ranking SVM, learning a ranking (or preference) function, has resulted in various applications in information retrieval (Herbrich et al. 2000; Joachims 2002; Yu et al. 2007). The task of learning *ranking* functions is distinguished from that of learning *classification* functions as follows:

1. While a training set in a classification is a set of data objects and their class labels, in *ranking* a training set is an ordering of data. Let “A is preferred to B” be specified as “ $A \succ B$.” A training set for ranking SVM is denoted as $R = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ where y_i is the ranking of \mathbf{x}_i , that is, $y_i < y_j$ if $\mathbf{x}_i \succ \mathbf{x}_j$.
2. Unlike a classification function, which outputs a distinct class for a data object, a ranking function outputs a *score* for each data object, from which a *global ordering* of data is constructed. That is, the target function $F(\mathbf{x}_i)$ outputs a score such that $F(\mathbf{x}_i) > F(\mathbf{x}_j)$ for any $\mathbf{x}_i \succ \mathbf{x}_j$.

If not stated, R is assumed to be strict ordering, which means that for all pairs \mathbf{x}_i and \mathbf{x}_j in a set D , either $\mathbf{x}_i \succ_R \mathbf{x}_j$ or $\mathbf{x}_i \prec_R \mathbf{x}_j$. However, it can be straightforwardly generalized to weak orderings. Let R^* be the optimal ranking of data in which the data is ordered perfectly according to the user's preference. A ranking function F is typically evaluated by how closely its ordering R^F approximates R^* .

Using the techniques of SVM, a global ranking function F can be learned from an ordering R . For now, assume F is a *linear* ranking function such that

$$\forall \{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\} : F(\mathbf{x}_i) > F(\mathbf{x}_j) \iff \mathbf{w} \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_j \quad (83)$$

A weight vector \mathbf{w} is adjusted by a learning algorithm. We say that an ordering R is *linearly rankable* if there exists a function F (represented by a weight vector \mathbf{w}) that satisfies [Eq. 83](#) for all $\{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\}$.

The goal is to learn F that is concordant with the ordering R and also generalize well beyond R . That is to find the weight vector \mathbf{w} such that $\mathbf{w} \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_j$ for most data pairs $\{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\}$.

Though this problem is known to be NP-hard (Cohen et al. 1998), the solution can be approximated using SVM techniques by introducing (nonnegative) slack variables ξ_{ij} and minimizing the upper bound $\sum \xi_{ij}$ as follows (Herbrich et al. 2000):

$$\text{minimize: } L_1(\mathbf{w}, \xi_{ij}) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum \xi_{ij} \quad (84)$$

$$\text{subject to: } \forall \{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\} : \mathbf{w} \cdot \mathbf{x}_i \geq \mathbf{w} \cdot \mathbf{x}_j + 1 - \xi_{ij} \quad (85)$$

$$\forall (i, j) : \xi_{ij} \geq 0 \quad (86)$$

By the constraint ([Eq. 85](#)) and by minimizing the upper bound $\sum \xi_{ij}$ in ([Eq. 84](#)), the above optimization problem satisfies orderings on the training set R with minimal error. By minimizing $\mathbf{w} \cdot \mathbf{w}$ or by maximizing the margin ($= \frac{1}{\|\mathbf{w}\|}$), it tries to maximize the generalization of the ranking function. We will explain how maximizing the margin corresponds to increasing the generalization of *ranking* in [Sect. 4.1](#). C is the soft margin parameter that controls the trade-off between the margin size and the training error.

By rearranging the constraint ([Eq. 85](#)) we get

$$\mathbf{w}(\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{ij} \quad (87)$$

The optimization problem becomes equivalent to that of *classifying* SVM on *pairwise difference vectors* $(\mathbf{x}_i - \mathbf{x}_j)$. Thus, we can extend an existing SVM implementation to solve the problem.

Note that the support vectors are the data pairs $(\mathbf{x}_i^s, \mathbf{x}_j^s)$ such that constraint ([Eq. 87](#)) is satisfied with the *equality* sign, that is, $\mathbf{w}(\mathbf{x}_i^s - \mathbf{x}_j^s) = 1 - \xi_{ij}$. Unbounded support vectors are the ones on the margin (i.e., their slack variables $\xi_{ij} = 0$), and bounded support vectors are the ones within the margin (i.e., $1 > \xi_{ij} > 0$) or misranked (i.e., $\xi_{ij} > 1$). As done in the classifying SVM, a function F in ranking SVM is also expressed only by the support vectors.

Similarly to the classifying SVM, the primal problem of ranking SVM can be transformed to the following dual problem using the Lagrange multipliers:

$$\text{maximize: } L_2(\alpha) = \sum_{ij} \alpha_{ij} - \sum_{ij} \sum_{uv} \alpha_{ij} \alpha_{uv} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_u - \mathbf{x}_v) \quad (88)$$

$$\text{subject to: } C \geq \alpha \geq 0 \quad (89)$$

Once transformed to the dual, the kernel trick can be applied to support the nonlinear ranking function. $K(\cdot)$ is a kernel function. α_{ij} is a coefficient for pairwise difference vectors $(\mathbf{x}_i - \mathbf{x}_j)$. Note that the kernel function is computed for $P^2 (\sim m^4)$ times where P is the number of data pairs and m is the number of data points in the training set, thus solving the ranking SVM takes $O(m^4)$ at least. Fast training algorithms for ranking SVM have been proposed (Joachims 2006) but they are limited to linear kernels.

Once α is computed, \mathbf{w} can be written in terms of the pairwise difference vectors and their coefficients such that

$$\mathbf{w} = \sum_{ij} \alpha_{ij} (\mathbf{x}_i - \mathbf{x}_j) \quad (90)$$

The ranking function F on a new vector \mathbf{z} can be computed using the kernel function replacing the dot product as follows:

$$F(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} = \sum_{ij} \alpha_{ij} (\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{z} = \sum_{ij} \alpha_{ij} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{z}). \quad (91)$$

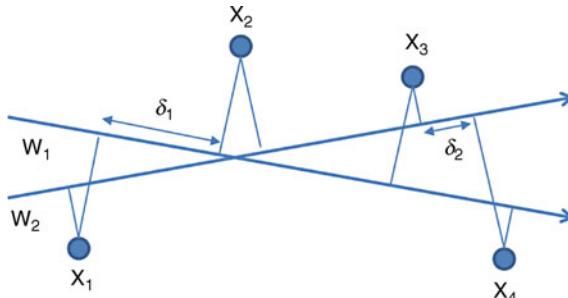
4.1 Margin-Maximization in Ranking SVM

We now explain the margin-maximization of the ranking SVM, to reason about how the ranking SVM generates a ranking function of *high generalization*. Some essential properties of ranking SVM are first established. For convenience of explanation, it is assumed that a training set R is linearly rankable and thus we use hard-margin SVM, that is, $\xi_{ij} = 0$ for all (i, j) in the objective (☞ Eq. 84) and the constraints (☞ Eq. 85).

In the ranking formulation, from (☞ Eq. 83), the linear ranking function F_w projects data vectors onto a weight vector \mathbf{w} . For instance, (☞ Fig. 6) illustrates linear projections of four vectors $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ onto two different weight vectors \mathbf{w}_1 and \mathbf{w}_2 , respectively, in a two-dimensional space. Both F_{x_1} and F_{x_2} make the same ordering R for the four vectors, that is, $x_1 >_R x_2 >_R x_3 >_R x_4$. The ranking difference of two vectors $(\mathbf{x}_i, \mathbf{x}_j)$ according to a ranking

Fig. 6

Linear projection of four data points.



function F_w is denoted by the geometric distance of the two vectors projected onto w , that is, formulated as $\frac{w(x_i - x_j)}{\|w\|}$.

Corollary 1 Suppose F_w is a ranking function computed by the hard-margin ranking SVM on an ordering R . Then, the support vectors of F_w represent the data pairs that are closest to each other when projected to w , thus closest in ranking.

Proof The support vectors are the data pairs (x_i^s, x_j^s) such that $w(x_i^s - x_j^s) = 1$ in constraint (Eq. 87), which is the smallest possible value for all data pairs $\forall (x_i, x_j) \in R$. Thus, its ranking difference according to $F_w\left(\frac{w(x_i^s - x_j^s)}{\|w\|}\right)$ is also the smallest among them (Vapnik 1998).

Corollary 2 The ranking function F , generated by the hard-margin ranking SVM, maximizes the minimal difference of any data pairs in ranking.

Proof By minimizing $w \cdot w$, the ranking SVM maximizes the margin $\delta = \frac{1}{\|w\|} = \frac{w(x_i^s - x_j^s)}{\|w\|}$ where (x_i^s, x_j^s) are the support vectors that denote, from the proof of Corollary 1, the minimal difference of any data pairs in the ranking.

The soft margin SVM allows bounded support vectors whose $\xi_{ij} > 0$ as well as unbounded support vectors whose $\xi_{ij} = 0$, in order to deal with noise and allow small errors for the R which is not completely linearly rankable. However, the objective function in (Eq. 84) also minimizes the number of the slacks and thus the amount of error, and the support vectors are the close data pairs in the ranking. Thus, maximizing the margin generates the effect of maximizing the differences of close data pairs in the ranking.

From Corollaries 1 and 2, we observe that ranking SVM improves the generalization performance by maximizing the minimal ranking difference. For example, consider the two linear ranking functions F_{w_1} and F_{w_2} in Fig. 6. Although the two weight vectors w_1 and w_2 make the same ordering, intuitively w_1 generalizes better than w_2 because the distance between the closest vectors on w_1 (i.e., δ_1) is larger than that on w_2 (i.e., δ_2). The SVM computes the weight vector w that maximizes the differences of close data pairs in the ranking. Ranking SVMs find a ranking function of high generalization in this way.

5 Ranking Vector Machine: An Efficient Method for Learning the 1-Norm Ranking SVM

This section presents another rank learning method, *RVM*, a revised 1-norm ranking SVM that is better for feature selection and more scalable to large datasets than the standard ranking SVM.

We first develop a 1-norm ranking SVM, a ranking SVM that is based on the *1-norm* objective function. (The standard ranking SVM is based on the 2-norm objective function.) The 1-norm ranking SVM learns a function with much fewer support vectors than the standard SVM. Thereby, its testing time is much faster than 2-norm SVMs and provides better feature selection properties. (The function of the 1-norm SVM is likely to utilize fewer features by using fewer support vectors (Fung and Mangasarian 2004).) Feature selection is also important in ranking. Ranking functions are relevance or preference functions in

document or data retrieval. Identifying key features increases the interpretability of the function. Feature selection for nonlinear kernels is especially challenging, and the fewer the number of support vectors, the more efficiently feature selection can be done (Guyon and Elisseeff 2003; Mangasarian and Wild 1998; Cao et al. 2007; Yu et al. 2003; Cho et al. 2008).

We next present an RVM that revises the 1-norm ranking SVM for fast training. The RVM trains much faster than standard SVMs while not compromising the accuracy when the training set is relatively large. *The key idea of a RVM is to express the ranking function with “ranking vectors” instead of support vectors.* Support vectors in ranking SVMs are pairwise difference vectors of the closest pairs as discussed in [Sect. 4](#). Thus, the training requires investigating every *data pair* as potential candidates of support vectors, and the number of data pairs are quadratic to the size of training set. On the other hand, the ranking function of the RVM utilizes each training data object instead of data pairs. Thus, the number of variables for optimization is substantially reduced in the RVM.

5.1 1-Norm Ranking SVM

The goal of 1-norm ranking SVM is the same as that of the standard ranking SVM, that is, to learn F that satisfies [Eq. 83](#) for most $\{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\}$ and generalize well beyond the training set. In the 1-norm ranking SVM, [Eq. 83](#) can be expressed using the F of [Eq. 91](#) as follows:

$$F(\mathbf{x}_u) > F(\mathbf{x}_v) \implies \sum_{ij}^P \alpha_{ij} (\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{x}_u > \sum_{ij}^P \alpha_{ij} (\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{x}_v \quad (92)$$

$$\implies \sum_{ij}^P \alpha_{ij} (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_u - \mathbf{x}_v) > 0 \quad (93)$$

Then, replacing the inner product with a kernel function, the 1-norm ranking SVM is formulated as

$$\text{minimize: } L(\alpha, \xi) = \sum_{ij}^P \alpha_{ij} + C \sum_{ij}^P \xi_{ij} \quad (94)$$

$$\text{s.t.: } \sum_{ij}^P \alpha_{ij} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_u - \mathbf{x}_v) \geq 1 - \xi_{uv}, \forall \{(u, v) : y_u < y_v \in R\} \quad (95)$$

$$\alpha \geq 0, \xi \geq 0 \quad (96)$$

While the standard ranking SVM suppresses the weight \mathbf{w} to improve the generalization performance, the 1-norm ranking suppresses α in the objective function. Since the weight is expressed by the sum of the coefficient times pairwise ranking difference vectors, suppressing the coefficient α corresponds to suppressing the weight \mathbf{w} in the standard SVM. (Mangasarian proves it in [Mangasarian \(2000\)](#).) C is a user parameter controlling the trade-off between the margin size and the amount of the error, ξ , and K is the kernel function. P is the number of pairwise difference vectors ($\sim m^2$).

The training of the 1-norm ranking SVM becomes a linear programming (LP) problem, thus solvable by LP algorithms such as the Simplex and Interior Point method ([Mangasarian](#)

2000, 2006; Fung and Mangasarian 2004). Just as for the standard ranking SVM, K needs to be computed P^2 ($\sim m^4$) times, and there are P number of constraints (Eq. 95) and α to compute. Once α is computed, F is computed using the same ranking function as the standard ranking SVM, that is, Eq. 91.

The accuracies of the 1-norm ranking SVM and standard ranking SVM are comparable, and both methods need to compute the kernel function $O(m^4)$ times. In practice, the training of the standard SVM is more efficient because fast decomposition algorithms have been developed such as sequential minimal optimization (SMO) (Platt 1998), while the 1-norm ranking SVM uses common LP solvers.

It is shown that *1-norm SVMs use much fewer support vectors than standard 2-norm SVMs*, that is, the number of positive coefficients (i.e., $\alpha > 0$) after training is much fewer in the 1-norm SVMs than in the standard 2-norm SVMs (Mangasarian 2006; Fung and Mangasarian 2004). This is because, unlike the standard 2-norm SVM, the support vectors in the 1-norm SVM are not bounded to those close to the boundary in classification or the minimal ranking difference vectors in ranking. Thus, the testing involves much fewer kernel evaluations, and it is more robust when the training set contains noisy features (Zhu et al. 2003).

5.2 Ranking Vector Machine

Although the 1-norm ranking SVM has merits over the standard ranking SVM in terms of the testing efficiency and feature selection, its training complexity is very high with respect to the number of data points. In this section, we present an RVM that revises the 1-norm ranking SVM to reduce the training time substantially. The RVM significantly reduces the number of variables in the optimization problem while not compromising the accuracy. *The key idea of RVM is to express the ranking function with “ranking vectors” instead of support vectors.* The support vectors in ranking SVMs are chosen from pairwise difference vectors, and the number of pairwise difference vectors are quadratic to the size of the training set. On the other hand, the ranking vectors are chosen from the training vectors, thus the number of variables to optimize is substantially reduced.

To theoretically justify this approach, we first present the representer theorem.

Theorem 1 (Representer Theorem (Schölkopf et al. 2001)).

Denote by $\Omega: [0, \infty) \rightarrow \mathcal{R}$ a strictly monotonic increasing function, by \mathcal{X} a set, and by $c: (\mathcal{X} \times \mathcal{R}^2)^m \rightarrow \mathcal{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $F \in \mathcal{H}$ of the regularized risk

$$c((x_1, y_1, F(x_1)), \dots, (x_m, y_m, F(x_m))) + \Omega(\|F\|_{\mathcal{H}}) \quad (97)$$

admits a representation of the form

$$F(x) = \sum_{i=1}^m \alpha_i K(x_i, x) \quad (98)$$

The proof of the theorem is presented in Schölkopf et al. (2001).

Note that, in the theorem, the loss function c is *arbitrary* allowing *coupling* between data points (x_i, y_i) , and the regularizer Ω has to be monotonic.

Given such a loss function and regularizer, the representer theorem states that although we might be trying to solve the optimization problem in an infinite-dimensional space \mathcal{H} ,

containing linear combinations of kernels centered on *arbitrary* points of \mathcal{X} , the solution lies in the span of m particular kernels – those centered on the *training* points (Schölkopf et al. 2001).

Based on the theorem, our ranking function F can be defined as (Eq. 98), which is based on the *training points* rather than arbitrary points (or pairwise difference vectors). Function (Eq. 98) is similar to function (Eq. 91) except that, unlike the latter using pairwise difference vectors ($\mathbf{x}_i - \mathbf{x}_j$) and their coefficients (α_{ij}), the former utilizes the training vectors (\mathbf{x}_i) and their coefficients (α_i). With this function, (Eq. 92) becomes the following.

$$F(\mathbf{x}_u) > F(\mathbf{x}_v) \implies \sum_i^m \alpha_i K(\mathbf{x}_i, \mathbf{x}_u) > \sum_i^m \alpha_i K(\mathbf{x}_i, \mathbf{x}_v) \quad (99)$$

$$\implies \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v)) > 0. \quad (100)$$

Thus, we set the loss function c as follows.

$$c = \sum_{\{(u,v): y_u < y_v \in R\}} \left(1 - \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v)) \right) \quad (101)$$

The loss function utilizes couples of data points penalizing misranked pairs, that is, it returns higher values as the number of misranked pairs increases. Thus, the loss function is order sensitive, and it is an instance of the function class c in (Eq. 97).

We set the regularizer $\Omega(||f||_{\mathcal{H}}) = \sum_i^m \alpha_i$ ($\alpha_i \geq 0$), which is strictly monotonically increasing. Let P be the number of pairs $(u, v) \in R$ such that $y_u < y_v$ and let $\xi_{uv} = 1 - \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v))$. Then, the RVM is formulated as follows.

$$\text{minimize: } L(\alpha, \xi) = \sum_i^m \alpha_i + C \sum_{ij}^P \xi_{ij} \quad (102)$$

$$\text{s.t.: } \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v)) \geq 1 - \xi_{uv}, \forall \{(u, v) : y_u < y_v \in R\} \quad (103)$$

$$\alpha, \xi \geq 0 \quad (104)$$

The solution of the optimization problem lies in the span of kernels centered on the training points (i.e., Eq. 98) as suggested in the representer theorem. Just as the 1-norm ranking SVM, the RVM suppresses α to improve the generalization, and forces (Eq. 100) by constraint (Eq. 103). Note that there are only m number of α_i in the RVM. Thus, the kernel function is evaluated $O(m^3)$ times while the standard ranking SVM computes it $O(m^4)$ times.

Another rationale of RVM or a rationale for using training vectors instead of pairwise difference vectors in the ranking function is that the support vectors in the 1-norm ranking SVM are not the closest pairwise difference vectors, thus expressing the ranking function with pairwise difference vectors becomes not as beneficial in the 1-norm ranking SVM. To explain this further, consider *classifying* SVMs. Unlike the 2-norm (classifying) SVM, the support

vectors in the 1-norm (classifying) SVM are *not* limited to those close to the decision boundary. This makes it possible that the 1-norm (classifying) SVM can express the similar boundary function with fewer support vectors. Directly extended from the 2-norm (classifying) SVM, the 2-norm *ranking* SVM improves the generalization by maximizing the closest pairwise ranking difference that corresponds to the margin in the 2-norm (classifying) SVM as discussed in [Sect. 4](#). Thus, the 2-norm ranking SVM expresses the function with the closest pairwise difference vectors (i.e., the support vectors). However, the 1-norm ranking SVM improves the generalization by suppressing the coefficients α just as the 1-norm (classifying) SVM. Thus, the support vectors in the 1-norm ranking SVM are not the closest pairwise difference vectors any more, and thus expressing the ranking function with pairwise difference vectors becomes not as beneficial in the 1-norm ranking SVM.

5.3 Experiment

This section evaluates the RVM on synthetic datasets ([Sect. 5.3.1](#)) and a real-world dataset ([Sect. 5.3.2](#)). The RVM is compared with the state-of-the-art ranking SVM provided in SVM-light. Experiment results show that the RVM trains substantially faster than the SVM-light for nonlinear kernels while their accuracies are comparable. More importantly, the number of ranking vectors in the RVM is multiple orders of magnitude smaller than the number of support vectors in the SVM-light. Experiments are performed on a Windows XP Professional machine with a Pentium IV 2.8 GHz and 1 GB of RAM. We implemented the RVM using C and used CPLEX (<http://www.ilog.com/products/cplex/>) for the LP solver. The source codes are freely available at <http://iis.postech.ac.kr/rvm> (Yu et al. 2008).

Evaluation Metric: MAP (mean average precision) is used to measure the ranking quality when there are only two classes of ranking (Yan et al. 2003), and NDCG is used to evaluate ranking performance for IR applications when there are multiple levels of ranking (Baeza-Yates and Ribeiro-Neto 1999; Burges et al. 2004; Cao et al. 2006; Xu and Li 2007). Kendall's τ is used when there is a global ordering of data and the training data is a subset of it. Ranking SVMs as well as the RVM minimize the amount of error or mis-ranking, which corresponds to optimizing the Kendall's τ (Joachims 2002; Yu 2005). Thus, we use the Kendall's τ to compare their accuracy.

Kendall's τ computes the overall accuracy by comparing the similarity of two orderings R^* and R^F . (R^F is the ordering of D according to the learned function F .) The Kendall's τ is defined based on the number of concordant pairs and discordant pairs. If R^* and R^F agree on how they order a pair, \mathbf{x}_i and \mathbf{x}_j , the pair is concordant, otherwise it is discordant. The accuracy of function F is defined as the number of concordant pairs between R^* and R^F per the total number of pairs in D as follows.

$$F(R^*, R^F) = \frac{\# \text{ of concordant pairs}}{\binom{|R|}{2}}$$

For example, suppose R^* and R^F order five points $\mathbf{x}_1, \dots, \mathbf{x}_5$ as follow:

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) \in R^*$$

$$(\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5) \in R^F$$

Then, the accuracy of F is 0.7, as the number of discordant pairs is 3, i.e., $\{\mathbf{x}_1, \mathbf{x}_2\}$, $\{\mathbf{x}_1, \mathbf{x}_3\}$, $\{\mathbf{x}_2, \mathbf{x}_3\}$ while all remaining seven pairs are concordant.

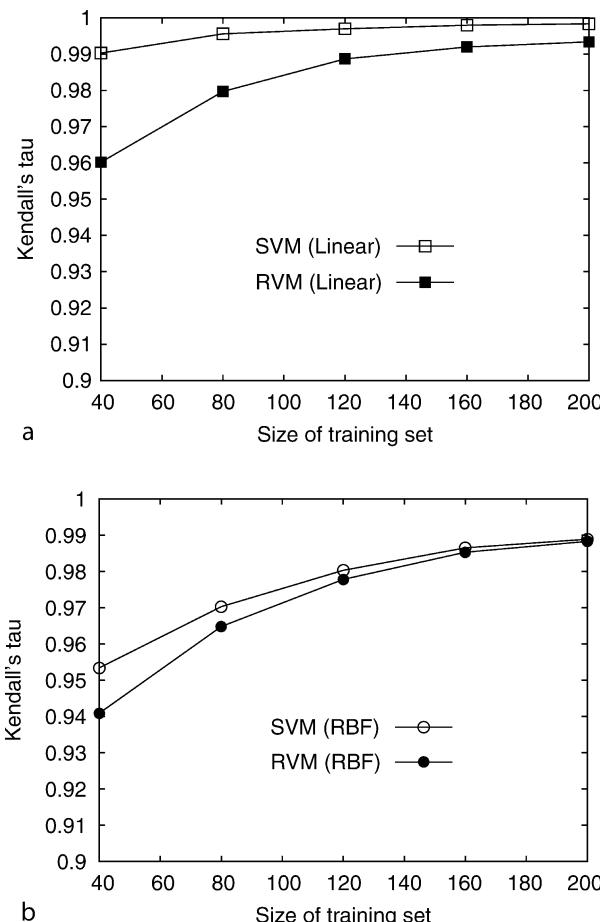
5.3.1 Experiments on Synthetic Datasets

Below is the description of the experiments on synthetic datasets.

1. We randomly generated a training and a testing dataset D_{train} and D_{test} , respectively, where D_{train} contains m_{train} ($= 40, 80, 120, 160, 200$) data points of n (e.g., 5) dimensions (i.e., m_{train} -by- n matrix), and D_{test} contains m_{test} ($= 50$) data points of n dimensions (i.e., m_{test} -by- n matrix). Each element in the matrices is a random number between zero and one. (We only did experiments on the dataset of up to 200 objects for performance reasons. Ranking SVMs run intolerably slow on datasets larger than 200.)
2. A global ranking function F^* is randomly generated, by randomly generating the weight vector \mathbf{w} in $F^*(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ for linear, and in $F^*(\mathbf{x}) = \exp(-\|\mathbf{w} - \mathbf{x}\|)^2$ for RBF function.

Fig. 7

Accuracy: (a) Linear (b) RBF.



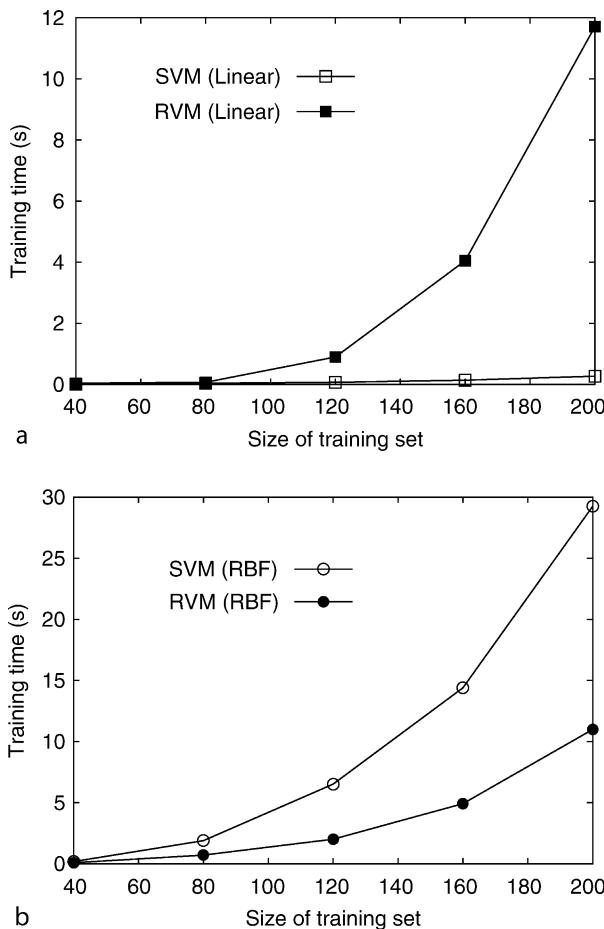
3. D_{train} and D_{test} are ranked according to F^* , which forms the global ordering R_{train}^* and R_{test}^* on the training and testing data.
4. We train a function F from R_{train}^* , and test the accuracy of F on R_{test}^* .

We tuned the soft margin parameter C by trying $C = 10^{-5}, 10^{-4}, \dots, 10^5$, and used the highest accuracy for comparison. For the linear and RBF functions, we used linear and RBF kernels accordingly. This entire process is repeated 30 times to get the mean accuracy.

Accuracy:  [Figure 7](#) compares the accuracies of the RVM and the ranking SVM from the SVM-light. The ranking SVM outperforms RVM when the size of the dataset is small, but their difference becomes trivial as the size of the dataset increases. This phenomenon can be explained by the fact that when the training size is too small, the number of potential ranking vectors becomes too small to draw an accurate ranking function whereas the number of potential support vectors is still large. However, as the size of the training set increases, RVM

 **Fig. 8**

Training time: (a) Linear Kernel (b) RBF Kernel.



becomes as accurate as the ranking SVM because the number of potential ranking vectors becomes large as well.

Training Time:  [Figure 8](#) compares the training time of the RVM and the SVM-light. While the SVM-light trains much faster than RVM for linear kernel (SVM-light is specially optimized for linear kernels), *the RVM trains significantly faster than the SVM-light for RBF kernels*.

Number of Support (or Ranking) Vectors:  [Figure 9](#) compares the number of support (or ranking) vectors used in the function of the RVM and the SVM-light. The RVM's model uses a significantly smaller number of support vectors than the SVM-light.

Sensitivity to Noise: In this experiment, the sensitivity of each method is compared to noise. Noise is inserted by switching the orders of some data pairs in R_{train}^* . We set the size of the training set $m_{\text{train}} = 100$ and the dimension $n = 5$. After R_{train}^* is made from a random function F^* , k vectors are randomly picked from the R_{train}^* and switched with their adjacent vectors in

 **Fig. 9**

Number of support (or ranking) vectors: (a) Linear Kernel (b) RBF Kernel.

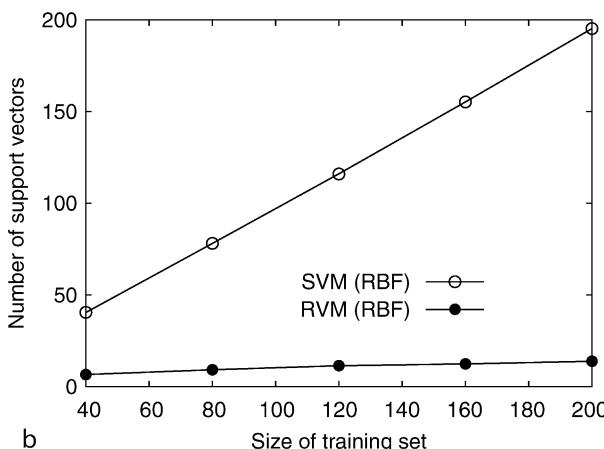
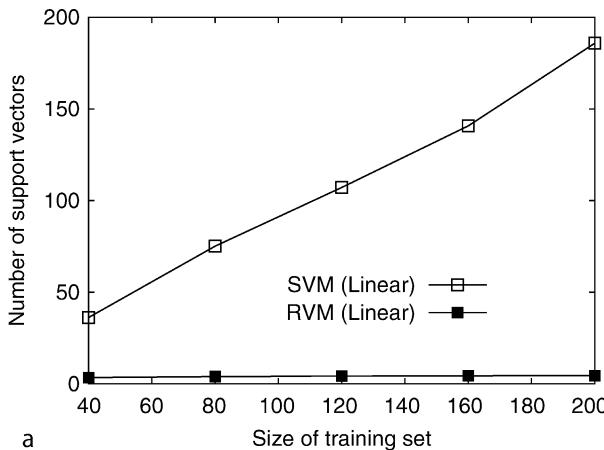
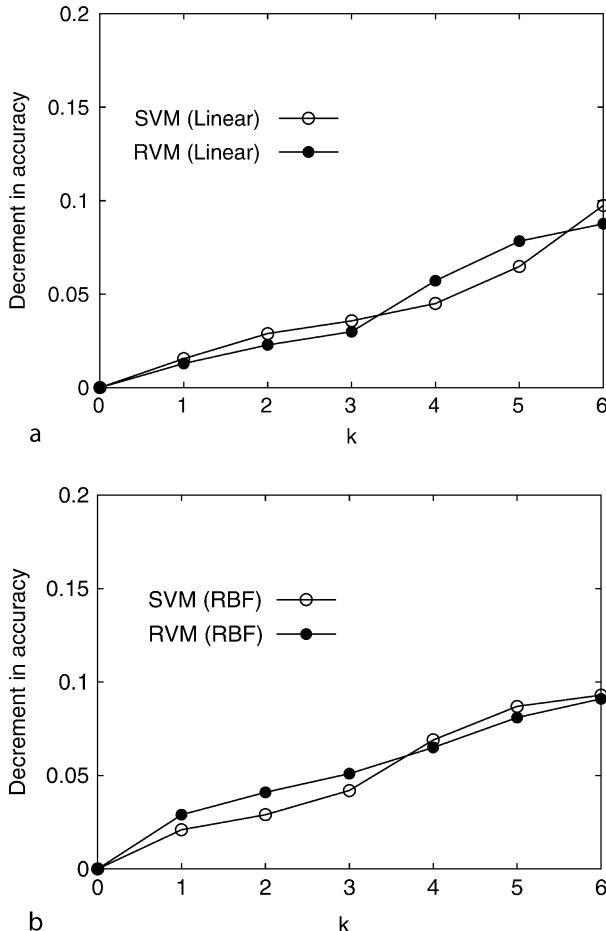


Fig. 10
Sensitivity to noise ($m_{\text{train}} = 100$): (a) Linear (b) RBF.



the ordering to implant noise in the training set. [Figure 10](#) shows the decrements of the accuracies as the number of misorderings increases in the training set. Their accuracies are moderately decreasing as the noise increases in the training set, and their sensitivities to noise are comparable.

5.3.2 Experiment on Real Dataset

In this section, we experiment using the OHSUMED dataset obtained from LETOR, the site containing benchmark datasets for ranking (LETOR). OHSUMED is a collection of documents and queries on medicine, consisting of 348,566 references and 106 queries. There are in total 16,140 query-document pairs upon which relevance judgments are made. In this dataset, the relevance judgments have three levels: “definitely relevant,” “partially relevant,” and

Table 2

Experiment results: accuracy (Acc), training time (Time), and number of support or ranking vectors (#SV or #RV)

		Query 1			Query 2			Query 3		
		D = 134		D = 128			D = 182			
				Acc	Time	#SV or #RV	Acc	Time	#SV or #RV	
RVM	Linear	0.5484	0.23	1.4	0.6730	0.41	3.83	0.6611	1.94	1.99
	RBF	0.5055	0.85	4.3	0.6637	0.41	2.83	0.6723	4.71	1
SVM	Linear	0.5634	1.83	92	0.6723	1.03	101.66	0.6588	4.24	156.55
	RBF	0.5490	3.05	92	0.6762	3.50	102	0.6710	55.08	156.66

“irrelevant.” The OHSUMED dataset in LETOR extracts 25 features. We report our experiments on the first three queries and their documents. We compare the performance of RVM and SVM-light on them. We tuned the parameters threefold using cross-validation by trying C and $\gamma = 10^{-6}, 10^{-5}, \dots, 10^6$ for the linear and RBF kernels and compared the highest performances. The training time is measured for training the model with the tuned parameters. The whole process was repeated three times and the mean values reported.

➤ *Table 2* shows the results. The accuracies of the SVM and RVM are comparable overall; SVM shows a little higher accuracy than RVM for query 1, but for the other queries their accuracy differences are not statistically significant. More importantly, *the number of ranking vectors in RVM is significantly smaller than that of support vectors in SVM*. For example, for query 3, an RVM having just one ranking vector outperformed an SVM with over 150 support vectors. *The training time of the RVM is significantly shorter than that of SVM-light.*

References

- Baeza-Yates R, Ribeiro-Neto B (eds) (1999) Modern information retrieval. ACM Press, New York
- Bertsekas DP (1995) Nonlinear programming. Athena Scientific, Belmont, MA
- Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2004) Learning to rank using gradient descent. In: Proceedings of the international conference on machine learning (ICML'04), Oregon State University, Corvallis, OR, USA
- Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Mining Knowl Discov 2:121–167
- Cao B, Shen D, Sun JT, Yang Q, Chen Z (2007) Feature selection in a kernel space. In: Proceedings of the international conference on machine learning (ICML'07), Oregon State University, Corvallis, OR, USA
- Cao Y, Xu J, Liu TY, Li H, Huang Y, Hon HW (2006) Adapting ranking SVM to document retrieval. In: Proceedings of the ACM SIGIR international conference on information retrieval (SIGIR'06), New York
- Cho B, Yu H, Lee J, Chee Y, Kim I (2008) Nonlinear support vector machine visualization for risk factor analysis using nomograms and localized radial basis function kernels. IEEE Trans Inf Technol Biomed 12(2)
- Christianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK
- Cohen WW, Schapire RE, Singer Y (1998) Learning to order things. In: Proceedings of the advances in neural information processing systems (NIPS'98), Cambridge, MA

- Friedman H (1998) Another approach to polychotomous classification. Tech. rep., Stanford University, Department of Statistics, Stanford, CA 10:1895–1924
- Fung G, Mangasarian OL (2004) A feature selection Newton method for support vector machine classification. *Comput Optim Appl* 28:185–202
- Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. *J Mach Learn Res* 3:1157–1182
- Hastie T, Tibshirani R (1998) Classification by pairwise coupling. In: *Advances in neural information processing systems*. MIT Press, Cambridge, MA
- Herbrich R, Graepel T, Obermayer K (eds) (2000) Large margin rank boundaries for ordinal regression. MIT Press, Cambridge, MA
- Joachims T (2002) Optimizing search engines using clickthrough data. In: *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD'02)*, Paris, France
- Joachims T (2006) Training linear SVMs in linear time. In: *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06)* Philadelphia, PA, USA
- Liu T-Y (2009) Learning to rank for information retrieval. *Found Trends Inf Retr* 3(3):225–331
- Mangasarian OL (2000) Generalized support vector machines. MIT Press, Cambridge, MA
- Mangasarian OL (2006) Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *J Mach Learn Res* 7:1517–1530
- Mangasarian OL, Wild EW (1998) Feature selection for nonlinear kernel support vector machines. Tech. rep., University of Wisconsin, Madison
- Platt J (1998) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges CJC (eds) *Advances in kernel methods: support vector machines*, MIT Press, Cambridge, MA
- Schölkopf B, Herbrich R, Smola AJ, Williamson RC (2001) A generalized representer theorem. In: *Proceedings of COLT*, Amsterdam, The Netherlands
- Smola AJ, Schölkopf B (1998) A tutorial on support vector regression. Tech. rep., NeuroCOLT2 Technical Report NC2-TR-1998-030
- Vapnik V (1998) *Statistical learning theory*. John Wiley and Sons, New York
- Xu J, Li H (2007) Ada Rank: a boosting algorithm for information retrieval. In: *Proceedings of the ACM SIGIR international conference on information retrieval (SIGIR'07)*, New York
- Yan L, Dodier R, Mozer MC, Wolniewicz R (2003) Optimizing classifier performance via the Wilcoxon-Mann-Whitney statistics. In: *Proceedings of the international conference on machine learning (ICML'03)*, Washington, DC
- Yu H (2005) SVM selective sampling for ranking with application to data retrieval. In: *Proceedings of the international conference on knowledge discovery and data mining (KDD'05)*, Chicago, IL
- Yu H, Hwang SW, Chang KCC (2007) Enabling soft queries for data retrieval. *Inf Syst* 32:560–574
- Yu H, Kim Y, Hwang SW (2008) RVM: An efficient method for learning ranking SVM. Tech. rep., Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea, <http://iis.hwanjoyu.org/rvm>
- Yu H, Yang J, Wang W, Han J (2003) Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In: *IEEE computer society bioinformatics conference (CSB'03)*, Stanford, CA, pp 220–228
- Zhu J, Rosset S, Hastie T, Tibshirani R (2003) 1-norm support vector machines. In: *Proceedings of the advances in neural information processing systems (NIPS'00)* Berlin, Germany

16 Fast Construction of Single-Hidden-Layer Feedforward Networks

Kang Li¹ · Guang-Bin Huang² · Shuzhi Sam Ge³

¹School of Electronics, Electrical Engineering and Computer Science,
Queen's University, Belfast, UK
k.li@ee.qub.ac.uk

²School of Electrical and Electronic Engineering, Nanyang Technological
University, Singapore
egbhuang@ntu.edu.sg

³Social Robotics Lab, Interactive Digital Media Institute, The National
University of Singapore, Singapore
elegesz@nus.edu.sg

1	<i>Introduction</i>	508
2	<i>Review of SLFNs with Random Nodes</i>	510
3	<i>Problem Formulation for Fast Construction of SLFNs</i>	511
4	<i>Fast Network Construction Approach for SLFNs – Phase 1: Subset Selection</i>	512
5	<i>Fast Construction of SLFNs – Phase 2: Fine Tuning</i>	517
6	<i>Performance Evaluation</i>	521
7	<i>Conclusion</i>	529

Abstract

In this chapter, two major issues are addressed: (i) how to obtain a more compact network architecture and (ii) how to reduce the overall computational complexity. An integrated analytic framework is introduced for the fast construction of single-hidden-layer feedforward networks (SLFNs) with two sequential phases. The first phase of the algorithm focuses on the computational efficiency for fast computation of the unknown parameters and fast selection of the hidden nodes. The second phase focuses on improving the performance of the network obtained in the first phase. The proposed algorithm is evaluated on several benchmark problems.

1 Introduction

The single-hidden-layer feedforward neural network (SLFN) represents a large class of flexible and efficient structures in the neural network literature. Due to their excellent capabilities, SLFNs have been widely used in many areas such as pattern recognition, bioinformatics, signal processing, time series prediction, nonlinear system modeling and control, etc. (Hong et al. 2008).

An SLFN is a linear combination of some basis functions that are usually nonlinear functions of the inputs. Many SLFNs with different types of basis functions have been proposed and intensively researched in the last three decades. Generally speaking, there are two mainstream architectures for SLFNs (Huang et al. 2006a, b; Peng and Irwin 2008): (i) SLFNs with additive nodes, and (ii) SLFNs with radial basis function (RBF) nodes.

An SLFN with n additive nodes can be formulated by

$$y(\mathbf{x}) = \sum_{i=1}^n \theta_i g(\mathbf{w}_i^T \mathbf{x} + b_i) \quad (1)$$

where $\mathbf{x} \in \Re^d$ is the input vector, $\mathbf{w}_i \in \Re^d$ is the input weight vector connecting the input layer to the i th hidden node, $b_i \in \Re$ is the bias of the i th hidden node, θ_i is the output weight connecting the i th hidden node to the output node, and g is the hidden node activation function.

The RBF networks belong to a specific class of SLFNs that use RBF nodes in the hidden layer, and each RBF node has its own center and impact factor with its output being some radially symmetric function of the distance between the input and the center (Peng et al. 2007; Li et al. 2009):

$$y(\mathbf{x}) = \sum_{i=1}^n \theta_i g(\gamma_i \|\mathbf{x} - \mathbf{c}_i\|) \quad (2)$$

where $\mathbf{x} \in \Re^d$ is again the neural input vector, $\mathbf{c}_i \in \Re^d$ and $\gamma_i \in \Re$ are the center and impact factors of the i th RBF node, and θ_i is the weight connecting the i th RBF hidden node to the output node.

In neural network applications, learning is a critical phase. For SLFNs with additive nodes, the early training algorithms make use of the first derivative information, that is, backpropagation with adaptive learning rate and momentum (BPAM), conjugate gradient, QuickProp, etc. (Bishop 1995). Advanced training algorithms like the Levenberg–Marquardt (LM) algorithm, which make use of the second derivative information, have proven to be more efficient and have been widely used in applications (Marquardt 1963; Hagan and Menhaj 1994).

For the training of SLFNs with RBF kernels, the conventional approach takes a two-stage procedure, that is, unsupervised learning of both the centers and impact factors for the RBF nodes and supervised learning of the linear output weights. With respect to the center location, clustering techniques have been proposed (Sutanto et al. 1997; Musavi et al. 1992). Once the centers and the impact factors are determined, the linear output weights can be obtained using Cholesky decomposition, orthogonal least squares or singular value decomposition (Press et al. 1992; Lawson and Hanson 1974; Serre 2002; Chen et al. 1991).

In contrast to the conventional two-stage learning procedure, supervised learning methods aim to optimize all the network parameters. To improve the convergence, various techniques have been introduced. For example, hybrid algorithms combine the gradient-based search for the nonlinear parameters (the impact factors and centers) of the RBF nodes and the least-squares estimation of the linear output weights (McLoone et al. 1998; Peng et al. 2003; Panchapakesan et al. 2002). Second-order algorithms have also been proposed, which use an additional adaptive momentum term to the Levenberg–Marquardt (LM) algorithm in order to maintain the conjugacy between successive minimization directions, resulting in good convergence for some well-known hard problems (Ampazis and Perantonis 2002).

In addition to the network learning, to control the network complexity of SLFNs is another important issue, and a number of additive (or constructive, or growing) methods, and subtractive (or destructive, or pruning) methods have been proposed (Mao and Huang 2005; Chng et al. 1996; Miller 1990; Platt 1991; Kadirkamanathan and Niranjan 1993; Yingwei et al. 1997; Huang et al. 2005). In particular, it has been shown that one can add nodes to the network one by one until a stopping criterion is satisfied, thus obtaining more compact networks (Chen et al. 1991; Chng et al. 1996). The techniques they used are subset selection algorithms, where a small number of nodes are selected from a large pool of candidates based on the orthogonal least squares (OLS) and its variants or the fast recursive algorithm (Chen et al. 1991; Mao and Huang 2005; Korenberg 1998; Zhang and Billings 1996; Chen and Wigger 1995; Li et al. 2005). The elegance of the method is that the net contribution of the newly added node can be explicitly identified without solving the whole least-square problem, leading to significantly reduced computational complexity.

Generally speaking, conventional network training and selection methods for SLFNs can be extremely time consuming if both issues have to be considered simultaneously (Peng et al. 2006). Unlike the conventional learning schemes, a more general network learning concept, namely, the extreme learning machine (ELM), was proposed for SLFNs (Huang et al. 2006b, c; Liang et al. 2006a). For ELM, all the nonlinear parameters in the hidden nodes, including both the input weights and hidden nodes' biases for SLFNs with additive nodes or both the centers and impact factors for SLFNs with RBF kernels, are randomly chosen independently of the training data. Thus the training is transformed to a standard least-squares problem, leading to a significant improvement of the generalization performance at a fast learning speed. The output weights in the SLFNs are linearly analytically determined (Huang et al. 2006b, c; Liang et al. 2006a). It has been proven in an incremental method that the SLFNs with randomly generated hidden nodes, independent of the training data, according to any continuous sampling distribution, can approximate any continuous target functions (Huang et al. 2006a). The activation functions for additive nodes in such SLFNs can be any bounded piecewise continuous function and the activation functions for RBF nodes can be any integrable piecewise continuous function. Further, they extended ELM to a much wider type of hidden nodes, including fuzzy rules as well as additive nodes, etc. (Huang and Chen 2007). The ELM has also been applied in some other areas like biomedical engineering

(Xu et al. 2005), bioinformatics (Handoko et al. 2006; Zhang et al. 2007; Wang and Huang 2005) human-computer interfaces (HCI) (Liang et al. 2006b), text classification (Liu et al. 2005), terrain reconstruction (Yeu et al. 2006) and communication channel equalization (Li et al. 2005).

This chapter addresses two critical issues arising from a wide range of applications for SLFNs with random hidden nodes: (i) how to obtain a more compact network architecture; (ii) how to reduce the overall computational complexity. These are the two bottlenecks restricting the application of the SLFNs to some large databases or complex problems. In this chapter, it will be shown that the above two objectives can be achieved using an integrated framework. A detailed performance evaluation will be performed on the benchmark problems in regression, time series prediction, and nonlinear system modeling and identification.

2 Review of SLFNs with Random Nodes

In this section, SLFNs with fully randomly generated hidden nodes are briefly reviewed. In particular, the ELM concept (Huang et al. 2006b, c; Huang and Chen 2007) is discussed. To begin with, a unified description is introduced.

The output of a generalized SLFN with n hidden nodes can be represented by

$$y(\mathbf{x}) = \sum_{i=1}^n \theta_i g(\omega_i; \mathbf{x}) \quad (3)$$

where ω is the vector of nonlinear parameters of the hidden nodes and θ_i is the weight connecting the i th hidden node to the output node. $g(\omega_i; \mathbf{x})$ is the output of the i th hidden node with respect to the input vector \mathbf{x} . Obviously, according to formulas (❶ 1), (❶ 2), and (❶ 3), for the SLFN with additive hidden nodes, $g(\omega_i; \mathbf{x}) = g(\mathbf{w}_i^T \mathbf{x} + b_i)$, and for the SLFN with RBF hidden nodes, $g(\omega_i; \mathbf{x}) = g(y_i \|\mathbf{x} - \mathbf{c}_i\|)$.

Now suppose N samples $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$ are used to train the network (❶ Eq. 3). Here, $\mathbf{x}_i \in \mathbb{R}^d$ is the input vector and $t_i \in \mathbb{R}$ is a target (it should be noted that all the analysis in this chapter can be linearly extended to SLFNs with multi-output nodes). If an SLFN with n hidden nodes can approximate these N samples with zero error, it implies that there exist θ_i and ω_i such that

$$y(\mathbf{x}_j) = \sum_{i=1}^n \theta_i g(\omega_i; \mathbf{x}_j) = t_j, j = 1, \dots, N \quad (4)$$

❷ Equation 4 can be rewritten compactly as:

$$\mathbf{y} = \mathbf{P}\Theta = \mathbf{t} \quad (5)$$

where

$$\mathbf{P}(\omega_1, \dots, \omega_n, \mathbf{x}_1, \dots, \mathbf{x}_n) = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n] = \begin{bmatrix} g(\omega_1, \mathbf{x}_1) & \cdots & g(\omega_n, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ g(\omega_1, \mathbf{x}_N) & \cdots & g(\omega_n, \mathbf{x}_N) \end{bmatrix} \quad (6)$$

$$\Theta = [\theta_1, \dots, \theta_n]^T, \mathbf{t} = [t_1, \dots, t_N]^T, \mathbf{y} = [y(\mathbf{x}_1), \dots, y(\mathbf{x}_N)]^T$$

\mathbf{P} is the output matrix of the hidden layer; the i th column of \mathbf{P} is the i th hidden node's output vector corresponding to inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and the j th row of \mathbf{P} is the output vector of the hidden layer with respect to input \mathbf{x}_j . In the regression context, \mathbf{P} is also called the regression matrix, and the i th column of \mathbf{P} corresponds to the i th regressor term (hidden node).

It has been proven that the universal approximation capability of SLFNs with random hidden nodes generated according to any continuous sampling distribution can approximate any continuous target function (Huang et al. 2006a; Huang and Chen 2007). Therefore, the hidden node parameters ω_i of SLFNs *need not be tuned during training and may simply be assigned with random values* (Huang et al. 2006a, b; Huang and Chen 2007). ➤ [Equation 5](#) then becomes a linear system and the output weights Θ are estimated as:

$$\Theta = \mathbf{P}^\dagger \mathbf{P} \quad (7)$$

where \mathbf{P}^\dagger is the Moore–Penrose generalized inverse (Rao and Mitra 1971) of the hidden layer output matrix \mathbf{P} .

The above introduction reveals the three issues in the ELM that have to be dealt with in applications: (i) the ELM usually requires a larger number of hidden nodes, thus a compact network implementation is demanded; (ii) a large network may lead to the singularity problem in the matrix computation, and controlling the network complexity can also reduce or eliminate this problem; and (iii) a large network generates a high computational overhead in deriving the output weights and so a more efficient algorithm is needed. In fact, these three problems are also closely coupled. If the performance of the network can be improved, the number of required hidden nodes and the corresponding number of output weights can be significantly reduced, leading to an overall reduction of the computational complexity. Moreover, the improved network performance will eventually reduce or eliminate the singularity problem. Once $\mathbf{P}^T \mathbf{P}$ becomes nonsingular, a number of efficient algorithms can be used for fast computation of the output weights. The basic idea in this chapter is to introduce an integrated framework, to further select the hidden nodes from the randomly generated hidden node pools in ELM, in order to further improve the network performance and to efficiently calculate the output weights as well. To propose this framework, the problem of fast construction of SLFNs will be formulated first.

3 Problem Formulation for Fast Construction of SLFNs

In this chapter, the network construction is carried out in two phases. The first phase is carried out by selecting the hidden nodes one by one from a pool of candidates, which are randomly generated at the beginning of the construction process, and then compute the linear output weights. Two scenarios are considered:

1. **Case 1:** The number of candidates exactly matches the desired number of hidden nodes for the SLFNs. In this case, the issue in this chapter is to efficiently compute the output weights and to increase the numerical stability. In the original ELM algorithm (Huang et al. 2006b), this is achieved via ➤ [Eq. 7](#) which is practically implemented using singular value decomposition. In this chapter, in order to further reduce the computational complexity and to increase the numerical stability, the selection process is still embedded in the algorithm, that is, the output weights are calculated only after all the hidden nodes are ordered according to their contribution to the cost function.
2. **Case 2:** The number of candidates is much bigger than required, and only a small subset of hidden nodes is selected. This is a standard subset selection problem, and the issue is to both reduce the computational complexity in the selection process and in the computation of output weights. For this subset selection problem, the OLS method and its variants (Chen et al. 1991; Chng et al. 1996; Korenberg 1988; Zhang and Billings 1996; Chen and

Wigger 1995; Zhu and Billings 1996; Gomm and Yu 2000) are perhaps among the most well-known approaches for the model selection.

Therefore, a selection process needs to be implemented in the above two scenarios. This selection problem can be described as follows.

Suppose that a set of M randomly generated nodes serves as a pool of candidates from which n hidden nodes will be selected, and a set of N samples (\mathbf{x}_i, t_i) is used to train the network. This leads to the following regression matrix of candidate nodes:

$$\left. \begin{aligned} \Phi &= [\phi_1, \phi_2, \dots, \phi_M] \\ \phi_i &= [\phi_i(1), \phi_i(2), \dots, \phi_i(N)]^T, i = 1, 2, \dots, M \end{aligned} \right\} \quad (8)$$

where ϕ_i is the output vector of the i th candidate node corresponding to the N training samples.

Now, the problem is to select, say, n hidden nodes, the corresponding regressor terms are denoted as $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$, and $\mathbf{p}_i \in \{\phi_1, \dots, \phi_M\}, i = 1, \dots, n$ forming the regression matrix of the SLFN hidden layer:

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n] \quad (9)$$

producing the following SLFN outputs:

$$\mathbf{y} = \mathbf{P}\Theta + \mathbf{e} = \mathbf{t} \quad (10)$$

where $\mathbf{e} = [e_1, \dots, e_N]^T \in \Re^N$ is the modeling error vector.

Now, the objective of network training is to minimize the error with respect to the cost function

$$J = \|\mathbf{e}\| = \|\mathbf{y} - \mathbf{P}\theta\| \quad (11)$$

where $\|\cdot\|$ denotes the two-norm of a vector.

If \mathbf{P} is of full column rank, the optimum estimation of the output weights is

$$\Theta = [\mathbf{P}^T \mathbf{P}]^{-1} \mathbf{P}^T \mathbf{y} \quad (12)$$

Theoretically, there are $M!/n!/(M-n)!$ possible combinations of n hidden nodes out of the total M candidates. Obviously, obtaining the optimal subset can be very expensive or impossible if M is a very large number, and part of this is also referred to as the curse of dimensionality in the literature. Practically, this problem can be partially solved using fast forward network growing or backward pruning methods. In the following section, an efficient approach will be employed.

4 Fast Network Construction Approach for SLFNs – Phase 1: Subset Selection

Before introducing the detailed scheme a core algorithm will be presented.

4.1 Fast Selection of One Hidden Node

The core idea of the fast network construction is to select hidden nodes for SLFNs one by one from a pool of candidates, each time the reduction of the cost function is maximized.

This procedure is iterated until n hidden nodes are selected. Therefore, the major objective in this section is to propose a recursive algorithm to select the hidden nodes.

To begin with, suppose that k hidden nodes have been selected, producing the following regression matrix

$$\mathbf{P}_k = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k] \quad (13)$$

The corresponding cost function is given by

$$J(\mathbf{P}_k) = \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{P}_k (\mathbf{P}_k^T \mathbf{P}_k)^{-1} \mathbf{P}_k^T \mathbf{y} \quad (14)$$

If \mathbf{P}_k is of full column rank, $(\mathbf{P}_k^T \mathbf{P}_k)$ in [Eq. 14](#) is symmetric and positive definite.

Define

$$\mathbf{W} = \mathbf{P}_k^T \mathbf{P}_k = [w_{i,j}]_{k \times k} \quad (15)$$

then \mathbf{W} can be decomposed as

$$\mathbf{W} = \mathbf{P}_k^T \mathbf{P}_k = \tilde{\mathbf{A}}^T \mathbf{D} \tilde{\mathbf{A}} \quad (16)$$

where $\mathbf{D} = \text{diag}(d_1, \dots, d_k)$ is a diagonal matrix and $\tilde{\mathbf{A}} = [\tilde{a}_{i,j}]_{k \times k}$ is a unity upper triangular matrix.

Define

$$\mathbf{A} = \mathbf{D} \tilde{\mathbf{A}} = [a_{i,j}]_{k \times k}, a_{i,j} = \begin{cases} 0, & j < i \\ d_i \tilde{a}_{i,j}, & j \geq i \end{cases} \quad (17)$$

Obviously, $\tilde{a}_{i,i} = 1$, therefore $d_i = a_{i,i}$, $i = 1, \dots, k$.

According to [Eq. 16](#), it can be derived that

$$a_{i,j} = w_{i,j} - \sum_{s=1}^{j-1} a_{s,i} a_{s,j} / a_{s,s}, i = 1, \dots, k, j = i, \dots, k \quad (18)$$

Define

$$\mathbf{a}_y = \mathbf{A} \boldsymbol{\theta} = \mathbf{D} \tilde{\mathbf{A}} \boldsymbol{\theta} = [a_{1,y}, \dots, a_{k,y}]^T \quad (19)$$

and

$$\mathbf{w}_y = \mathbf{P}_k^T \mathbf{y} = [w_{1,y}, \dots, w_{k,y}]^T \quad (20)$$

Left-multiplying both sides of [Eq. 12](#) with \mathbf{W} for $\mathbf{P} = \mathbf{P}_k$ and substituting [Eq. 16](#), we have

$$\tilde{\mathbf{A}}^T \mathbf{D} \tilde{\mathbf{A}} \boldsymbol{\theta} = \mathbf{P}_k^T \mathbf{y} \text{ or } \tilde{\mathbf{A}}^T \mathbf{a}_y = \mathbf{w}_y \quad (21)$$

Noting that $\tilde{\mathbf{A}}$ is a unity upper triangular matrix, and noting the relationship of $\tilde{\mathbf{A}}$ with \mathbf{A} defined in [Eq. 17](#), \mathbf{a}_y in [Eq. 21](#) could be computed as

$$a_{i,y} = w_{i,y} - \sum_{s=1}^{k-1} a_{s,i} a_{s,y} / a_{s,s}, i = 1, \dots, k \quad (22)$$

Substituting ⚡ Eq. 16 into ⚡ Eq. 14 and noting ⚡ Eq. 21 or

$$\mathbf{a}_y = (\tilde{\mathbf{A}}^T)^{-1} \mathbf{w}_y = (\tilde{\mathbf{A}}^T)^{-1} \mathbf{P}_k^T \mathbf{y} \quad (23)$$

we have

$$J(\mathbf{P}_k) = \mathbf{y}^T \mathbf{y} - \mathbf{a}_y^T \mathbf{D}^{-1} \mathbf{a}_y = \mathbf{y}^T \mathbf{y} - \sum_{i=1}^k a_{i,y}^2 / a_{i,i} \quad (24)$$

Now, suppose that one more hidden node is selected with the corresponding regressor term \mathbf{p}_{k+1} , the cost function becomes

$$J(\mathbf{P}_{k+1}) = \mathbf{y}^T \mathbf{y} - \sum_{i=1}^{k+1} a_{i,y}^2 / a_{i,i} \quad (25)$$

where $\mathbf{P}_{k+1} = [\mathbf{P}_k \ \mathbf{p}_{k+1}]$.

The net reduction of the cost function due to adding one more hidden node is given by

$$\Delta J_{k+1}(\mathbf{p}_{k+1}) = J(\mathbf{P}_k) - J(\mathbf{P}_{k+1}) = a_{k+1,y}^2 / a_{k+1,k+1} \quad (26)$$

where $a_{k+1,y}, a_{k+1,k+1}$ are computed using ⚡ Eqs. 18 and ⚡ 22 as k increases by 1.

According to ⚡ Eqs. 18 and ⚡ 22, it can be seen that elements $a_{i,j}$ and $a_{i,y}$ for $i = 1, \dots, k$ and $j = i, \dots, k$, which correspond to the previously selected hidden nodes $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m$, do not change as a new hidden node is added into the SLFN. Therefore, the selection of the next hidden node is formulated as the following optimization problem according to ⚡ Eq. 26

$$\begin{aligned} \min\{J([\mathbf{P}_k, \phi])\} &= J(\mathbf{P}_k) - \max\{\Delta J_{k+1}(\phi)\} \\ \text{s.t. } \phi &\in \{\phi_1, \dots, \phi_M\}, \phi \notin \{\mathbf{p}_1, \dots, \mathbf{p}_k\} \end{aligned} \quad (27)$$

where $\{\phi_1, \dots, \phi_M\}$ is the candidate node pool.

According to ⚡ Eq. 27, the contributions of all remaining candidate nodes in $\Phi = \{\phi_1, \dots, \phi_M\}$ need to be calculated using ⚡ Eq. 26. To achieve this, the dimension of \mathbf{A} , \mathbf{a}_y defined above will be augmented to store the information of all remaining candidate nodes in Φ . To achieve this, redefine

$$\begin{aligned} \Phi &= [\mathbf{P}_k, \mathbf{C}_{M-k}] \\ \mathbf{C}_{M-k} &= [\phi_{k+1}, \dots, \phi_M] \end{aligned} \quad (28)$$

where the first k regressor terms in Φ (i.e., \mathbf{P}_k) correspond to the selected k nodes, the remaining $M - k$ terms $\mathbf{C}_{M-k} = [\phi_{k+1}, \dots, \phi_M]$ are candidates, forming the candidate pool \mathbf{C}_{M-k} .

Now, for an SLFN with k hidden nodes, augment the $k \times k$ matrices $\tilde{\mathbf{A}}$ and \mathbf{A} , and the $k \times 1$ vector \mathbf{a}_y to $k \times M$ and $M \times 1$, respectively, in order to store information about all the remaining candidates in \mathbf{C}_{M-k} for selection of the next hidden node.

Based on ⚡ Eq. 18, \mathbf{A} is redefined as

$$\begin{aligned} \mathbf{A} &= [a_{i,j}]_{k \times M} \\ a_{i,j} &= \begin{cases} 0, & j < i \\ w_{i,j} - \sum_{s=1}^{i-1} \frac{a_{s,i} a_{s,j}}{a_{s,s}}, & i \leq j \leq M \end{cases} \end{aligned} \quad (29)$$

where

$$w_{i,j} = \begin{cases} \mathbf{p}_i^T \mathbf{p}_j, & j \leq k \\ \mathbf{p}_i^T \phi_j, & j > k \end{cases} \quad (30)$$

Based on [Eq. 17](#), $\tilde{\mathbf{A}}$ is redefined as

$$\tilde{\mathbf{A}} = [\tilde{a}_{i,j}]_{k \times M}, \tilde{a}_{i,j} = a_{i,j}/a_{i,i} \quad (31)$$

and based on [Eq. 23](#), vector \mathbf{a}_y is redefined as

$$\mathbf{a}_y = [a_{i,y}]_{M \times 1}$$

$$a_{i,y} = \begin{cases} \mathbf{y}^T \mathbf{p}_i - \sum_{s=1}^{i-1} a_{s,i} a_{s,y} / a_{s,s}, & i \leq k \\ \mathbf{y}^T \phi_i - \sum_{s=1}^k a_{s,i} a_{s,y} / a_{s,s}, & i > k \end{cases} \quad (32)$$

In addition, one more $M \times 1$ vector \mathbf{b} is defined as

$$\mathbf{b} = [b_i]_{M \times 1}$$

$$b_i = \begin{cases} \mathbf{p}_i^T \mathbf{p}_i - \sum_{s=1}^{i-1} a_{s,i} a_{s,i} / a_{s,s}, & i \leq k \\ \phi_i^T \phi_i - \sum_{s=1}^k a_{s,i} a_{s,i} / a_{s,s}, & i > k \end{cases} \quad (33)$$

Then

$$b_i = a_{i,i}, i = 1, \dots, k \quad (34)$$

Thus, based on [Eq. 26](#), the contribution of each of the candidates in \mathbf{C}_{M-k} to the cost function can be computed as follows:

$$\Delta J_{k+1}(\phi_i) = a_{i,y}^2 / b_i, i = k + 1, \dots, M \quad (35)$$

and that from \mathbf{C}_{M-k} which gives the maximum contribution is then selected as the $(k+1)$ th hidden node.

For example, assume

$$j = \arg \max_{k < i \leq M} \{\Delta J_{k+1}(\phi_i)\} \quad (36)$$

then ϕ_j is selected as the $(k+1)$ th hidden node, and re-denoted as $\mathbf{p}_{k+1} = \phi_j$, and the selected hidden node regression matrix becomes $\mathbf{P}_{k+1} = [\mathbf{P}_k \mathbf{p}_{k+1}]$, while the candidate pool is reduced in size and becomes \mathbf{C}_{M-k-1} , and the remaining candidates in \mathbf{C}_{M-k-1} are re-indexed as $\phi_{k+2}, \dots, \phi_M$. Thus, the full regression matrix Φ becomes $\Phi = [\mathbf{P}_{k+1} \mathbf{C}_{M-k-1}]$. This change leads to some changes in matrices \mathbf{A} and $\tilde{\mathbf{A}}$.

In detail, due to the interchange of ϕ_{k+1} and ϕ_j in Φ , columns $k+1$ and j of \mathbf{A} and $\tilde{\mathbf{A}}$ should also be interchanged, that is

$$\begin{cases} \bar{a}_{i,k+1} = a_{i,j}, \bar{a}_{i,j} = a_{i,k+1}, i = 1, \dots, k \\ \bar{\tilde{a}}_{i,k+1} = \tilde{a}_{i,j}, \bar{\tilde{a}}_{i,j} = \tilde{a}_{i,k+1} \end{cases} \quad (37)$$

where $\bar{a}_{i,k+1}$ and $\bar{a}_{i,j}$ denote the i th elements in columns $k+1$ and j of the updated matrices \mathbf{A} and $\tilde{\mathbf{A}}$, respectively. To denote an updated element in these matrices, a bar is applied to the element hereafter.

Similarly, the $(k+1)$ th and j th elements in vectors \mathbf{a}_y and \mathbf{b} should also be interchanged, that is,

$$\begin{cases} \bar{a}_{k+1,y} = a_{j,y}, \bar{a}_{j,y} = a_{k+1,y} \\ \bar{b}_{k+1} = b_j, \bar{b}_j = b_{k+1} \end{cases} \quad (38)$$

In addition, as the $(k+1)$ th hidden node is now selected, a new row (the $(k+1)$ th row) will be appended to matrices \mathbf{A} and $\tilde{\mathbf{A}}$ (☞ Eqs. 29–31), that is

$$\begin{cases} a_{k+1,i} = \mathbf{p}_{k+1}^T \phi_i - \sum_{s=1}^k \tilde{a}_{s,k+1} a_{s,i}, i = k+1, \dots, M \\ \tilde{a}_{k+1,i} = a_{k+1,i} / a_{k+1,k+1} \end{cases} \quad (39)$$

Finally, the $(k+2)$ th to the last elements of both vectors \mathbf{a}_y and \mathbf{b} need to be updated if the $(k+2)$ th hidden node will be selected from the remaining candidates in \mathbf{C}_{M-k-1} . This can be done as follows, according to their definitions (☞ 32) and (☞ 33), respectively

$$\begin{cases} \bar{a}_{i,y} = a_{i,y} - \tilde{a}_{k+1,i} a_{k+1,y} \\ \bar{b}_i = b_i - \tilde{a}_{k+1,i} a_{k+1,i} \end{cases}, i = k+1, \dots, M \quad (40)$$

This section has provided a framework to iteratively select the $(k+1)$ th hidden node for a pool of candidates for $k = 0, 1, \dots, (n-1)$. In the following sections, an integrated framework will be proposed for both the selection of hidden nodes and the computation of output weights.

4.2 Fast Construction of SLFNs – Phase 1

4.2.1 Algorithm: Fast construction algorithm for SLFNs – Phase 1 (FCA-I)

1. Initialization phase:

Collect N data samples, randomly generate M candidate hidden nodes, producing the full regression matrix $\Phi = [\phi_1, \phi_2, \dots, \phi_M]$. Let the cost function $J = \mathbf{y}^T \mathbf{y}$, and calculate $a_{i,y} = \mathbf{y}^T \phi_i$, $b_i = \phi_i^T \phi_i$ for $i = 1, \dots, M$. Finally, calculate \mathbf{A} and $\tilde{\mathbf{A}}$ according to (☞ Eqs. 29 and (☞ 31). Let $k = 0$.

2. Construction phase:

- (a) Calculate $\Delta J_{k+1}(\phi_i)$ using (☞ Eq. 35 for $i = k+1, \dots, M$.
- (b) Based on the values of $\Delta J_{k+1}(\phi_i)$, $i = k+1, \dots, M$, search for the hidden node which gives the maximum contribution to the cost function, assuming $\phi_j = \arg \max \{\Delta J_{k+1}(\phi_i), k < i \leq M\}$, then update the cost function as $J = J - \Delta J_{k+1}(\phi_j)$.
- (c) Reorder columns in \mathbf{A} and $\tilde{\mathbf{A}}$ correspondingly, according to (☞ Eq. 37 and also update these two matrices according to (☞ Eq. 39. Finally, rearrange and update elements of \mathbf{a}_y and \mathbf{b} according to (☞ Eq. 38 and (☞ Eq. 40.

- (d) If k is less than the desired number (say n) of hidden nodes or the cost function (the training error) is higher than expected, let $k \leftarrow k + 1$ and go to **step 2a** for the next round of hidden node selection.
3. Once the selection procedure is terminated, the output weights can be immediately computed from the upper triangular equation (**19**) using backward substitution:

$$\theta_k = a_{k,y}/a_{k,k} - \sum_{i=k+1}^n \tilde{a}_{k,i}\theta_i, \quad k = n, \dots, 1 \quad (41)$$

where $\theta_n = a_{n,y}/a_{n,n}$.

Remark Some other stopping criteria can also be used in **step 2d**, for example, the construction phase may stop when Akaike's information criterion (AIC) begins to increase (Akaike 1974).

5 Fast Construction of SLFNs – Phase 2: Fine Tuning

The above phase 1 fast construction of SLFNs is by nature a variant of the forward subset select algorithm, which selects one hidden node at a time by maximizing the reduction of error. The main advantage is its computational efficiency; however, the disadvantage is that the resultant network is not necessarily compact due to the fact that the later introduced regressor terms (neurons) may affect the contribution of previously selected regressor terms (nodes). Then, the previously selected regressor terms may become insignificant due to the late introduced regressor terms (Peng et al. 2006; Li et al. 2006).

To improve the performance of the network obtained in phase 1, this section will introduce a method for fine-tuning the network structure. This is done by reassessing all selected hidden nodes within the same analytic framework developed in phase 1, and any insignificant regressor terms will be removed and replaced within the same analytic framework, leading to improved network performance without introducing a significant amount of computation effort.

5.1 Reevaluation of a Selected Hidden Node

The core issue in the phase 2 network construction is to reassess the contribution of all selected hidden nodes and all candidate nodes. In this section, algorithms will be introduced to reassess any of the hidden nodes selected in the phase 1 fast SLFN construction procedure.

Suppose a hidden node (out of all selected n nodes), say \mathbf{p}_i , $1 \leq i < n$, is to be reevaluated. Its contribution $\Delta J_n(\mathbf{p}_i)$ to the cost function needs to be recalculated and then compared with the candidate hidden nodes. Denote the maximum candidate contribution as $\Delta J_n(\phi_j)$. If $\Delta J_n(\mathbf{p}_i) < \Delta J_n(\phi_j)$, \mathbf{p}_i is said to be insignificant and will be replaced with ϕ_j , and \mathbf{p}_i will be put back into the candidate pool. This procedure will further reduce the cost function by $\Delta J_n(\phi_j) - \Delta J_n(\mathbf{p}_i)$, thus the network performance is further improved.

To reevaluate a selected hidden node, say \mathbf{p}_i , its contribution to the cost function and that of the candidates $\phi_{n+1}, \dots, \phi_M$ needs to be recalculated. To achieve efficient computation, the regression context has to be reconstructed. From the phase 1 fast SLFN construction, it can be

seen that the proper regression context should include \mathbf{A} , $\tilde{\mathbf{A}}$, \mathbf{a}_y , and \mathbf{b} , which are used to compute the contribution of a hidden node. The algorithm to reconstruct the regression context is as follows.

The first step in the regression context reconstruction is to move the selected hidden node of interest, say \mathbf{p}_p , $i \in \{1, \dots, n-1\}$, to the n th position in the selected regression matrix \mathbf{P}_n , as if it was the last selected hidden node. This can be achieved by a series of interchanges between two adjacent regressor terms \mathbf{p}_k and \mathbf{p}_{k+1} for $k = i, \dots, n-1$. And each time two adjacent regressor terms are interchanged in their position, the regression context, including \mathbf{A} , $\tilde{\mathbf{A}}$, \mathbf{a}_y , and \mathbf{b} , will be reconstructed once.

To begin with, consider two adjacent regressor terms \mathbf{p}_k and \mathbf{p}_{k+1} for $k = i, i+1, \dots, n-1$ are interchanged. Then the n regressor terms in the new selected order becomes $\mathbf{p}_1, \dots, \mathbf{p}_{k-1}, \hat{\mathbf{p}}_k, \hat{\mathbf{p}}_{k+1}, \mathbf{p}_{k+1}, \dots, \mathbf{p}_n$, where $\hat{\mathbf{p}}_k = \mathbf{p}_{k+1}$ and $\hat{\mathbf{p}}_{k+1} = \mathbf{p}_k$. This shift of positions for the two adjacent regressor terms leads to the changes in \mathbf{A} , $\tilde{\mathbf{A}}$, \mathbf{a}_y , and \mathbf{b} .

For \mathbf{A} , according to $\textcircled{2}$ Eq. 17, from rows 1 to $k-1$, we have

$$\left. \begin{aligned} \hat{w}_{i,k} &= \mathbf{p}_i^T \hat{\mathbf{p}}_k = \mathbf{p}_i^T \mathbf{p}_{k+1} = w_{i,k+1} \\ \hat{w}_{i,k+1} &= \mathbf{p}_i^T \hat{\mathbf{p}}_{k+1} = \mathbf{p}_i^T \mathbf{p}_k = w_{i,k} \\ \hat{w}_{i,j} &= w_{i,j}, i = 1, \dots, k-1; j = i, \dots, k-1, k+2, \dots, M \end{aligned} \right\} \quad (42)$$

therefore

$$\left. \begin{aligned} \hat{a}_{i,k} &= a_{i,k+1}, \hat{a}_{i,k+1} = a_{i,k} \\ \hat{a}_{i,j} &= a_{i,j}, i = 1, \dots, k-1; j = i, \dots, k-1, k+2, \dots, M \end{aligned} \right\} \quad (43)$$

According to $\textcircled{2}$ Eqs. 42 and $\textcircled{2}$ 43, it can be found that only columns k and $k+1$ are interchanged in \mathbf{A} for rows 1 to $k-1$, and the rest are unchanged.

Noting that $\hat{w}_{k,k+1} = \hat{\mathbf{p}}_k^T \hat{\mathbf{p}}_{k+1} = \hat{\mathbf{p}}_{k+1}^T \mathbf{p}_k = w_{k+1,k}$, for the k th row of \mathbf{A} , it holds that

$$\hat{a}_{k,k+1} = a_{k,k+1} \quad (44)$$

Noting that $\hat{w}_{k,k} = \hat{\mathbf{p}}_k^T \hat{\mathbf{p}}_k = \mathbf{p}_{k+1}^T \mathbf{p}_{k+1} = w_{k+1,k+1}$,

$$\hat{a}_{k,k} = \hat{w}_{k,k} - \sum_{s=1}^{k-1} \frac{\hat{a}_{s,k} \hat{a}_{s,k}}{\hat{a}_{s,s}} = a_{k+1,k+1} + a_{k,k+1} a_{k,k+1} / a_{k,k} \quad (45)$$

While for other elements of the k th row in \mathbf{A} it holds that

$$\hat{a}_{k,j} = \hat{w}_{k,j} - \sum_{s=1}^{k-1} \frac{\hat{a}_{s,k} \hat{a}_{s,j}}{\hat{a}_{s,s}} = a_{k,j} + a_{k,k+1} a_{k,j} / a_{k,k}, j = k+2, \dots, n \quad (46)$$

Also, because $\hat{w}_{k+1,k+1} = \hat{\mathbf{p}}_{k+1}^T \hat{\mathbf{p}}_{k+1} = \mathbf{p}_k^T \mathbf{p}_k = w_{k,k}$, for the $(k+1)$ th row of \mathbf{A} , it holds that

$$\hat{a}_{k+1,k+1} = \hat{w}_{k+1,k+1} - \sum_{s=1}^k \hat{a}_{s,k+1} \hat{a}_{s,k+1} / \hat{a}_{s,s} = a_{k,k} - (a_{k,k+1})^2 / \hat{a}_{k,k} \quad (47)$$

and for $j = k+2, \dots, M$ it holds that

$$\begin{aligned} \hat{a}_{k+1,j} &= \hat{w}_{k+1,j} - \sum_{s=1}^k \hat{a}_{s,k+1} \hat{a}_{s,j} / \hat{a}_{s,s} \\ &= a_{k,j} - a_{k,k+1} \hat{a}_{k,j} / \hat{a}_{k,k}, j = k+2, \dots, M \end{aligned} \quad (48)$$

Since $\hat{w}_{i,j} = \hat{\mathbf{p}}_i^T \hat{\mathbf{p}}_j = \mathbf{p}_i^T \mathbf{p}_j = w_{i,j}$, $i, j > k + 1$, for the elements of the $(k + 2)$ th row in \mathbf{A} , it holds that

$$\begin{aligned}\hat{a}_{k+2,j} &= \hat{w}_{k+2,j} - \sum_{s=1}^{k+1} \hat{a}_{s,k+2} \hat{a}_{s,j} / \hat{a}_{s,s} \\ &= w_{k+2,j} - \sum_{s=1}^{k-1} \frac{a_{s,k+2} a_{s,j}}{a_{s,s}} - \frac{\hat{a}_{k,k+2} \hat{a}_{k,j}}{\hat{a}_{k,k}} - \frac{\hat{a}_{k+1,k+2} \hat{a}_{k+1,j}}{\hat{a}_{k+1,k+1}}\end{aligned}\quad (49)$$

From the k th and $(k + 1)$ th row of \mathbf{A} , it could be derived that

$$\frac{\hat{a}_{k,k+2} \hat{a}_{k,j}}{\hat{a}_{k,k}} - \frac{\hat{a}_{k+1,k+2} \hat{a}_{k+1,j}}{\hat{a}_{k+1,k+1}} = \frac{a_{k,k+2} a_{k,j}}{a_{k,k}} - \frac{a_{k+1,k+2} a_{k+1,j}}{a_{k+1,k+1}} \quad (50)$$

which implies that the $(k + 2)$ th row of \mathbf{A} has no change. Furthermore, it could be derived that rows $(x + 2)$ to n of \mathbf{A} have no change.

Similarly, it could be derived that for vector \mathbf{a}_y only two elements are changed as follows

$$\left. \begin{array}{l} \hat{a}_{k,y} = a_{k+1,y} + a_{k,k+1} a_{k,y} / a_{k,k} \\ \hat{a}_{k+1,y} = a_{k,y} - a_{k,k+1} \hat{a}_{k,j} / \hat{a}_{k,k} \end{array} \right\} \quad (51)$$

For vector \mathbf{b} , only the k th and the $(k + 1)$ th elements are changed, that is

$$\hat{b}_k = \hat{a}_{k,k}, \hat{b}_{k+1} = \hat{a}_{k+1,k+1} \quad (52)$$

To summarize, if two adjacent regressor terms \mathbf{p}_k and \mathbf{p}_{k+1} are interchanged in position, the regression context of \mathbf{A} , \mathbf{a}_y and \mathbf{b} can be reconstructed as shown in [Eq. 53](#), where the changed elements (with hat) are highlighted.

$$\mathbf{A} \Rightarrow \begin{bmatrix} a_{1,1} & \cdots & \hat{a}_{1,k} & \hat{a}_{1,k+1} & \cdots & a_{1,k} & a_{1,k+1} & \cdots & a_{1,M} \\ \vdots & \vdots \\ \hat{a}_{k,k} & \hat{a}_{k,k+1} & \cdots & \hat{a}_{k,k} & \hat{a}_{k,k+1} & \cdots & \hat{a}_{k,M} \\ \hat{a}_{k+1,k+1} & \cdots & \hat{a}_{k+1,k} & \hat{a}_{k+1,k+1} & \cdots & \hat{a}_{k+1,M} \\ \mathbf{0} & & \ddots & \vdots & \vdots & & \vdots \\ & & & a_{k,k} & a_{k,k+1} & \cdots & a_{k,M} \end{bmatrix}, \quad (53)$$

$$\mathbf{a}_y \Rightarrow \begin{bmatrix} a_{1,y} \\ \vdots \\ \hat{a}_{k,y} \\ \hat{a}_{k+1,y} \\ \vdots \\ a_{M,y} \end{bmatrix}, \mathbf{b} \Rightarrow \begin{bmatrix} b_1 \\ \vdots \\ \hat{a}_k \\ \hat{a}_{k+1} \\ \vdots \\ b_M \end{bmatrix}$$

The above analysis shows how the regression context is changed as two adjacent regressor terms interchange their positions in the selected regression matrix \mathbf{P}_n . Through a series of

such interchanges between two adjacent regressor terms, \mathbf{p}_i can be moved to the n th position in the selected regression matrix \mathbf{P}_n . Then the contribution of \mathbf{p}_i of interest can be computed as

$$\Delta J_n(\mathbf{p}_i) = \Delta J_n(\hat{\mathbf{p}}_n) = a_{n,y}^2 / a_{n,n} \quad (54)$$

where $\mathbf{p}_i = \hat{\mathbf{p}}_n$.

5.2 Reevaluation of a Candidate Hidden Node

The main objective in the phase 2 fast SLFN construction is to reevaluate the contribution of a selected hidden node, and this contribution is then compared with that of the candidate nodes. If its contribution is less than a candidate node, it will be replaced. The contribution of a candidate node to the cost function is calculated as if the last selected hidden node is pruned, and the following quantities will have to be computed:

$$\left. \begin{aligned} a_{j,y}^{(-i)} &= a_{j,y} + a_{n,j}a_{n,y}/a_{n,n} \\ b_j^{(-i)} &= b_j + (a_{n,n+1})^2/a_{n,n} \end{aligned} \right\}, j = n+1, \dots, M \quad (55)$$

These quantities are in fact the values of the corresponding elements in vector \mathbf{a}_y and \mathbf{b} when the regressor term \mathbf{p}_i is removed from the model. Then, the contribution of all candidates can be computed as

$$\Delta J_n(\phi_j) = (a_{j,y}^{(-i)})^2 / b_j^{(-i)}, j = n+1, \dots, M \quad (56)$$

Now, the next step is to identify the maximum contribution among all the candidates

$$\Delta J_n(\phi_s) = \max_{j=n+1, \dots, M} \{\Delta J_n(\phi_j)\} \quad (57)$$

If $\Delta J_n(\phi_s) > \Delta J_n(\mathbf{p}_i)$, then \mathbf{p}_i should be replaced by ϕ_s , while \mathbf{p}_i should be put back into the candidate pool, taking the position of ϕ_s in the candidate pool. In this case, the regression context needs to be updated again to reflect the interchange of ϕ_s and \mathbf{p}_i .

For matrix \mathbf{A} , only the n th and s th column starting from row 1 to row $n-1$ need to be interchanged, noting that \mathbf{p}_i has already been moved to the n th position in the regression matrix \mathbf{P}_n . For the n th row of \mathbf{A} , the elements are calculated as

$$\left. \begin{aligned} \hat{a}_{n,n} &= b_s^{(-i)}, \hat{a}_{n,s} = a_{n,s}, w_{n,k} = \hat{\mathbf{p}}_n^T \phi_k \\ \hat{a}_{n,k} &= w_{n,k} - \sum_{j=1}^{n-1} \frac{a_{j,n}a_{j,k}}{a_{j,j}} \\ k &= n+1, \dots, M, k \neq s \end{aligned} \right\} \quad (58)$$

where $\hat{\mathbf{p}}_n$ is the \mathbf{p}_i which has now been moved to the n th position. In the meantime, the n th to M th elements of vector \mathbf{a}_y and \mathbf{b} are updated, respectively, as

$$\left. \begin{aligned} \hat{a}_{n,y} &= a_{n,y}^{(-i)}, a_{s,y} = a_{n,y} - \hat{a}_{n,s}\hat{a}_{n,y}/\hat{a}_{n,n}, \\ a_{j,y} &= a_{j,y}^{(-i)} - \frac{\hat{a}_{n,j}\hat{a}_{n,y}}{\hat{a}_{n,n}}, j = n+1, \dots, M, j \neq s \end{aligned} \right\} \quad (59)$$

and

$$\left. \begin{aligned} \hat{b}_n &= b_s^{(-i)}, \hat{b}_s = a_{n,n} - (\hat{a}_{n,n+1})^2 / \hat{a}_{n,n} \\ \hat{b}_j &= b_j^{(-i)} - (\hat{a}_{n,n+1})^2 / \hat{a}_{n,n}, j = n + 1, \dots, M, j \neq s \end{aligned} \right\} \quad (60)$$

It should be noted that if \mathbf{A} is updated, the corresponding element of $\tilde{\mathbf{A}}$ should be recalculated as well according to its definition.

5.3 Fast Construction of SLFNs: Phase 2

5.3.1 Algorithm: Fast Construction Algorithm for SLFNs – Phase 2 (FCA-II)

1. Let $i = n - 1$, go to [step 2](#) to perform a check loop.
2. For $j = i, \dots, n - 1$, exchange regressor terms \mathbf{p}_j and \mathbf{p}_{j+1} to move the x th regressor term to the n th position. Elements of \mathbf{A} , \mathbf{a}_y , and \mathbf{b} are updated correspondingly for each regressor term interchange as [Eqs. 42–53](#).
3. Compute the contribution of \mathbf{p}_i once it has been moved to the n th position in the regression matrix \mathbf{P}_n according to [Eq. 54](#) and those of all the $M - n$ candidates according to [Eq. 56](#).
4. Identify the candidate ϕ_s that has the maximum contributions. If $\Delta J_n(\mathbf{p}_i) \geq \Delta J_n(\phi_s)$ let $i \leftarrow i - 1$; otherwise select ϕ_s as the new n th regressor by interchanging \mathbf{p}_i with ϕ_s and modify the regression context as [Eqs. 58–60](#), and then let $i = k - 1$.
5. If $i = 0$, all the selected regressors are reevaluated (in this case, the cost function converges to a minimum), and are terminated from this reviewing procedure. Otherwise, continue the reevaluation procedure from [step 2](#). Note that as each insignificant regressor is being replaced, the cost function is further reduced by $\Delta J_k(\phi_s) - \Delta J_k(\mathbf{p}_i)$. The monotonic decrease will make the cost function converge to a minimum, hence improving the model performance.

6 Performance Evaluation

6.1 Benchmark Problems

The performance of the proposed fast construction algorithm (FCA) for SLFNs was evaluated using the benchmark problems described in [Table 1](#), which includes two regression applications (California Housing and Abalone) (Blake and Merz 1998), one time series prediction (Mackey and Glass 1977), and one nonlinear system modeling (Piroddi and Spinelli 2003).

1. **California Housing:** California Housing is a dataset obtained from the StatLib repository. There are 20,640 observations for predicting the price of houses in California. Information on the variables was collected using all the block groups in California from the 1990 census.
2. **Abalone:** The abalone problem is to estimate the age of abalone from physical measurements (Blake and Merz 1998).

Table 1**Specification of benchmark datasets**

Dataset	# Attributes	# Training data	# Testing data
CalHousing	8	8,000	12,640
Abalone	8	3,000	1,177
Mackey–Glass	4	20,000	500
Ident	6	20,000	500

3. **Mackey–Glass:** The need for time series prediction arises in many real-world problems such as detecting arrhythmia in heartbeats, stock market indices, etc. One of the classical benchmark problems in the literature is the chaotic Mackey–Glass differential delay equation given by Mackey and Glass (1977):

$$\frac{dx(t)}{dt} = \frac{ax(t - \tau)}{1 + x^{10}(t - \tau)} - bx(t) \quad (61)$$

for $a = 0.2$, $b = 0.1$, and $\tau = 17$. Integrating the equation over the time interval $[t, t + \Delta t]$ by the trapezoidal rule yields:

$$x(t + \Delta t) = \frac{2 - b\Delta t}{2 + \Delta t} x(t) + \frac{a\Delta t}{2 + b\Delta t} \left[\frac{x(t + \Delta t - \tau)}{1 + x^{10}(t + \Delta t - \tau)} + \frac{x(t - \tau)}{1 + x^{10}(t - \tau)} \right] \quad (62)$$

The time series is generated under the condition $x(t - \tau) = 0.3$ for $0 \leq t \leq \tau$ and predicted with $v = 50$ sample steps ahead using the four past samples: s_{n-v} , s_{n-v-6} , s_{n-v-12} , and s_{n-v-18} . Hence, the n th input–output instance is:

$$\begin{aligned} \mathbf{x}_n &= [s_{n-v}, s_{n-v-6}, s_{n-v-12}, s_{n-v-18}]^T \\ y_n &= s_n \end{aligned} \quad (63)$$

In this simulation, $\Delta t = 1$, and the training samples were from $t = 1$ to 20,000 and the testing samples were from $t = 20,001$ to $t = 20,500$.

4. **Ident:** Modeling and identification of nonlinear dynamic systems has been intensively researched in recent decades due to its wide applications in almost all engineering sectors. This chapter considers the following artificial nonlinear discrete system (Piroddi and Spinelli 2003):

$$\begin{aligned} y(t) &= 0.8y(t - 1) + u(t - 1) - 0.3u(t - 2) \\ &\quad - 0.4u(t - 3) + 0.25u(t - 1)u(t - 2) \\ &\quad - 0.2u(t - 2)u(t - 3) - 0.3u(t - 1)^3 \\ &\quad + 0.24u(t - 2)^3 + \epsilon(t) \end{aligned} \quad (64)$$

where $\epsilon(t)$ is a random noise within the range $[0, 0.02]$.

Data points numbering 20,500 were generated in this case where $u(t)$ was uniformly distributed within $[-1, 1]$. The first 20,000 samples were used for network training and the other 500 points for testing.

In this example, the inputs of the SLFN are $y(t - 1), y(t - 2), y(t - 3), u(t - 1), u(t - 2), u(t - 3)$, and the output is $y(t)$. In this chapter, a slightly

large number of training samples were generated in the last two examples, with the aim to test the computational demand of the algorithms when the size of the problem becomes large.

The first three problems were used in the literature on the performance evaluation of ELM (see Huang et al. 2006a, b); therefore, the results published in these papers can be directly used for the comparison purpose.

Following the same procedures as in Huang et al. (2006a, b), both the Gaussian RBF activation function and the sigmoidal additive activation function were used in the simulations. In the simulations, all the inputs and outputs were normalized so that they fall within the range [0, 1]. For SLFNs with additive hidden nodes, the input weights and biases were randomly chosen from the range [-1, 1]. For SLFNs with RBF hidden nodes, the centers were

Table 2
Comparison for FCA-I and ELM on the regression problems

Data sets	Algorithms	Training time (s)	RMSE		# Nodes
			Training	Testing	
CalHousing	FCA-I(Sigmoid)	0.0094	0.1452	0.1457	10
	ELM(Sigmoid)	0.0361	0.1449	0.1452	
	FCA-I(RBF)	0.0124	0.1528	0.1521	
	ELM(RBF)	0.0343	0.1567	0.1566	
	FCA-I(Sigmoid)	0.0500	0.1399	0.1410	20
	ELM(Sigmoid)	0.0984	0.1403	0.1413	
	FCA-I(RBF)	0.0500	0.1402	0.1411	
	ELM(RBF)	0.0950	0.1404	0.1415	
Abalone	FCA-I(Sigmoid)	0.1189	0.1357	0.1371	30
	ELM(Sigmoid)	0.2015	0.1355	0.1373	
	FCA-I(RBF)	0.1219	0.1351	0.1364	
	ELM(RBF)	0.2002	0.1366	0.1377	
	FCA-I(Sigmoid)	0.0000	0.0903	0.0907	5
	ELM(Sigmoid)	0.0016	0.0903	0.0890	
	FCA-I(RBF)	0.0000	0.1052	0.1031	
	ELM(RBF)	0.0032	0.0956	0.0935	
	FCA-I(Sigmoid)	0.0064	0.0793	0.0818	10
	ELM(Sigmoid)	0.0154	0.0788	0.0811	
	FCA-I(RBF)	0.0031	0.0841	0.0858	
	ELM(RBF)	0.0063	0.0855	0.0857	
	FCA-I(Sigmoid)	0.0125	0.0757	0.0796	20
	ELM(Sigmoid)	0.0157	0.0758	0.0801	
	FCA-I(RBF)	0.0124	0.0766	0.0797	
	ELM(RBF)	0.0158	0.0765	0.0798	

randomly chosen from the range $[-1, 1]$, and the impact factors were randomly chosen from the range $(0, 0.5)$. Each of the test cases reported in this chapter were performed 10 times, and the results reported are the mean values. All the simulations in this chapter were performed in the MATLAB 6.0 environment running on an ordinary laptop with 1.66 GHZ CPU.

6.2 Performance Evaluation of FCA-I with ELM

The ELM has been compared with those methods in previous works (Huang et al. 2006a, b), and it has been shown that the ELM outperforms those methods in terms of computational complexity and network performance, for details please refer to Huang et al. (2006a, b).

■ Table 3

Comparison for FCA-I and ELM on the time-series and system identification problems

Data sets	Algorithms	Training time (s)	RMSE		# Nodes
			Training	Testing	
Ident	FCA-I(Sigmoid)	0.0157	0.1058	0.1117	10
	ELM(Sigmoid)	0.0469	0.1258	0.1332	
	FCA-I(RBF)	0.0188	0.2697	0.2765	
	ELM(RBF)	0.0437	0.2642	0.2695	
	FCA-I(Sigmoid)	0.0638	0.0675	0.0701	20
	ELM(Sigmoid)	0.1250	0.0718	0.0743	
	FCA-I(RBF)	0.0626	0.1455	0.1512	
	ELM(RBF)	0.1250	0.1400	0.1455	
Mackey-Glass	FCA-I(Sigmoid)	0.1438	0.0553	0.0561	30
	ELM(Sigmoid)	0.2626	0.0561	0.0571	
	FCA-I(RBF)	0.1440	0.1114	0.1152	
	ELM(RBF)	0.2625	0.0989	0.1020	
	FCA-I(Sigmoid)	0.0155	0.0092	0.0089	10
	ELM(Sigmoid)	0.0406	0.0100	0.0097	
	FCA-I(RBF)	0.0187	0.0690	0.0656	
	ELM(RBF)	0.0437	0.0501	0.0478	
	FCA-I(Sigmoid)	0.0653	0.0045	0.0042	20
	ELM(Sigmoid)	0.1265	0.0046	0.0043	
	FCA-I(RBF)	0.0626	0.0118	0.0111	
	ELM(RBF)	0.1250	0.0104	0.0099	
	FCA-I(Sigmoid)	0.1451	0.0025	0.0024	30
	ELM(Sigmoid)	0.2564	0.0025	0.0023	
	FCA-I(RBF)	0.1423	0.0035	0.0033	
	ELM(RBF)	0.2532	0.0039	0.0038	

Therefore, here the performance evaluation was only made between FCA-I and ELM on the four benchmark problems. For all the problems studied here, 10 trials were performed for each algorithm, and in each trial the hidden nodes were randomly generated. It should be noted that in order to compare the performance of FCA-I with ELM, the number of candidates for SLFNs should be equal to the desired number of hidden nodes, that is $M = n$. This implies that the FCA-I proposed in the previous section will not select the hidden nodes from a large pool of candidates. In this case, FCA-I will perform the same task as ELM, that is to compute the linear output weights for fixed SLFNs with random nodes.

⌚ *Table 2* summarizes the results (mean values over the 10 trials) for the first two regression problems. ⌚ *Table 2* reveals that: (1) the training and testing performance of both the FCA-I and ELM are similar in almost all cases for the two regression problems; (2) the FCA-I performed consistently faster than ELM in all cases; (3) for these two regression

⌚ **Table 4**
Comparison of FCA-I and OLS on the regression problems

Data sets	Algorithms	Training time (s)	RMSE		# Nodes out of 100
			Training	Testing	
CalHousing	FCA-I(Sigmoid)	0.2376	0.1401	0.1413	10
	OLS(Sigmoid)	1.8467	0.1411	0.1423	
	FCA-I(RBF)	0.2364	0.1401	0.1411	
	OLS(RBF)	1.8781	0.1396	0.1403	
	FCA-I(Sigmoid)	0.4515	0.1334	0.1350	20
	OLS(Sigmoid)	3.49840	0.13438	0.13619	
	FCA-I(RBF)	0.4469	0.1332	0.1341	
	OLS(RBF)	3.5391	0.1332	0.1345	
Abalone	FCA-I(Sigmoid)	0.6425	0.1305	0.1322	30
	OLS(Sigmoid)	5.00010	0.13084	0.13271	
	FCA-I(RBF)	0.6407	0.1304	0.1322	
	OLS(RBF)	5.0046	0.1295	0.1307	
	FCA-I(Sigmoid)	0.0316	0.0794	0.0820	5
	OLS(Sigmoid)	0.2002	0.0798	0.0819	
	FCA-I(RBF)	0.0313	0.0823	0.0822	
	OLS(RBF)	0.1970	0.0805	0.0827	
	FCA-I(Sigmoid)	0.0689	0.0764	0.0797	10
	OLS(Sigmoid)	0.3797	0.0765	0.0794	
	FCA-I(RBF)	0.0659	0.0765	0.0789	
	OLS(RBF)	0.3876	0.0760	0.0778	
	FCA-I(Sigmoid)	0.1250	0.0745	0.0813	20
	OLS(Sigmoid)	0.7313	0.0745	0.0798	
	FCA-I(RBF)	0.1234	0.0745	0.0786	
	OLS(RBF)	0.7330	0.0745	0.0783	

problems, the network size can be quite small, and further increases of the number of hidden nodes did not have significant impact on the network performance (training and testing errors).

► [Table 3](#) lists the results for the last two problems, that is, the time series prediction and nonlinear system modeling. Several observations can be identified from ► [Table 3](#): (1) SLFNs with additive hidden nodes perform much better than SLFNs with RBF nodes in these two problems; (2) the training and testing performance of both the FCA-I and ELM again are quite similar; (3) the FCA-I performs consistently faster than the ELM in all cases, as observed in the first two regression problems; (4) the number of hidden nodes for SLFNs does have significant impact on the network performance. It implies that these two problems are more complex than the previous two cases, and therefore require SLFNs of bigger number of random hidden nodes.

■ **Table 5**
Comparison of FCA-I and OLS on the time-series and system identification problems

Data sets	Algorithms	Training time (s)	RMSE		# Nodes out of 100
			Training	Testing	
Ident	FCA-I(Sigmoid)	0.2986	0.0659	0.0645	10
	OLS(Sigmoid)	2.3312	0.0610	0.0621	
	FCA-I(RBF)	0.2984	0.1128	0.1159	
	OLS(RBF)	2.3453	0.1155	0.1200	
	FCA-I(Sigmoid)	0.5659	0.0478	0.0477	20
	OLS(Sigmoid)	4.4079	0.0477	0.0484	
	FCA-I(RBF)	0.5655	0.0781	0.0803	
	OLS(RBF)	4.4327	0.0746	0.0759	
Mackey-Glass	FCA-I(Sigmoid)	0.8063	0.0410	0.0414	30
	OLS(Sigmoid)	6.2485	0.0409	0.0412	
	FCA-I(RBF)	0.8062	0.0582	0.0588	
	OLS(RBF)	6.2847	0.0594	0.0600	
	FCA-I(Sigmoid)	0.2985	0.0070	0.0066	10
	OLS(Sigmoid)	2.3470	0.0066	0.0064	
	FCA-I(RBF)	0.2969	0.0111	0.0105	
	OLS(RBF)	2.3580	0.0138	0.0132	
	FCA-I(Sigmoid)	0.5704	0.0030	0.0028	20
	OLS(Sigmoid)	4.4406	0.0028	0.0026	
	FCA-I(RBF)	0.5703	0.0042	0.0039	
	OLS(RBF)	4.4546	0.0050	0.0048	
	FCA-I(Sigmoid)	0.8079	0.0018	0.0018	30
	OLS(Sigmoid)	6.2860	0.0019	0.0018	
	FCA-I(RBF)	0.8091	0.0025	0.0025	
	OLS(RBF)	6.2891	0.0026	0.0025	

To conclude, both [Tables 2](#) and [3](#) have shown that the FCA-I performs consistently faster than the ELM for fixed SLFNs with random nodes.

6.3 Performance Evaluation of FCA-I with OLS

As discussed in previous sections, a compact network is always desirable for many real-life applications. Therefore, to select a subset of nodes from a large pool of randomly generated candidates is one of the most popular approaches. For this subset selection problem, OLS (Chen and Wigger 1995; Chen et al. 1989) has been intensively used in the literature. In this study, both FCA-I and OLS are applied to the selection of hidden nodes for the four problems.

Table 6
Comparison of FCA-(I+II) and FCA-I only on the regression problems

Data sets	Algorithms	RMSE		# Nodes out of 100
		Training	Testing	
CalHousing	FCA-I(Sigmoid)	0.1401	0.1413	10
	FCA-(I+II)(Sigmoid)	0.1380	0.1393	
	FCA-I(RBF)	0.1401	0.1411	
	FCA-(I+II)(RBF)	0.1379	0.1390	
	FCA-I(Sigmoid)	0.1334	0.1350	20
	FCA-(I+II)(Sigmoid)	0.1313	0.1330	
	FCA-I(RBF)	0.1332	0.1341	
	FCA-(I+II)(RBF)	0.1315	0.1329	
Abalone	FCA-I(Sigmoid)	0.1305	0.1322	30
	FCA-(I+II)(Sigmoid)	0.1285	0.1308	
	FCA-I(RBF)	0.1304	0.1322	
	FCA-(I+II)(RBF)	0.1280	0.1295	
	FCA-I(Sigmoid)	0.0794	0.0820	5
	FCA-(I+II)(Sigmoid)	0.0781	0.0815	
	FCA-I(RBF)	0.0823	0.0822	
	FCA-(I+II)(RBF)	0.0794	0.0809	
Boston	FCA-I(Sigmoid)	0.0764	0.0797	10
	FCA-(I+II)(Sigmoid)	0.0757	0.0791	
	FCA-I(RBF)	0.0765	0.0789	
	FCA-(I+II)(RBF)	0.0756	0.0783	
	FCA-I(Sigmoid)	0.0745	0.0813	20
	FCA-(I+II)(Sigmoid)	0.0738	0.0865	
	FCA-I(RBF)	0.0745	0.0786	
	FCA-(I+II)(RBF)	0.0738	0.0781	

For all four problems, the number of candidate nodes is set to be 100, that is, the hidden nodes are selected from a pool of 100 randomly generated candidates.

➲ *Table 4* summarizes the results for the first two regression problems. ➲ *Table 4* shows that: (1) the training and testing performance of both the FCA-I and OLS are similar in all cases for the two regression problems; (2) the FCA-I performs significantly faster than OLS in all cases; (3) the network performance is not significantly improved as the number of hidden nodes increases.

➲ *Table 5* lists the results for the last two problems, that is, the time series prediction and nonlinear system modeling. It is observed that: (1) SLFNs with additive nodes perform much better than SLFNs with RBF nodes in these two cases; (2) the training and testing performance of both the FCA-I and OLS again are quite similar; (3) FCA-I performs significantly faster than OLS in all cases, this observation agrees with the previous findings for the first two regression problems; (4) the network performance improves as the number of hidden nodes increases.

➲ **Table 7**

Comparison of FCA-(I+II) and FCA-I only on the time-series and system identification problems

Data sets	Algorithms	RMSE		# Nodes out of 100
		Training	Testing	
Ident	FCA-I(Sigmoid)	0.0645	0.0659	10
	FCA-(I+II)(Sigmoid)	0.0534	0.0539	
	FCA-I(RBF)	0.1128	0.1159	
	FCA-(I+II)(RBF)	0.0973	0.0994	
	FCA-I(Sigmoid)	0.0478	0.0477	20
	FCA-(I+II)(Sigmoid)	0.0428	0.0429	
	FCA-I(RBF)	0.0781	0.0803	
	FCA-(I+II)(RBF)	0.0623	0.0641	
Mackey-Glass	FCA-I(Sigmoid)	0.0410	0.0414	30
	FCA-(I+II)(Sigmoid)	0.0330	0.0335	
	FCA-I(RBF)	0.0582	0.0588	
	FCA-(I+II)(RBF)	0.0499	0.0513	
	FCA-I(Sigmoid)	0.0070	0.0066	10
	FCA-(I+II)(Sigmoid)	0.0047	0.0044	
	FCA-I(RBF)	0.0111	0.0105	
	FCA-(I+II)(RBF)	0.0096	0.0091	
	FCA-I(Sigmoid)	0.0030	0.0028	20
	FCA-(I+II)(Sigmoid)	0.0023	0.0022	
	FCA-I(RBF)	0.0042	0.0039	
	FCA-(I+II)(RBF)	0.0032	0.0030	
	FCA-I(Sigmoid)	0.0018	0.0018	30
	FCA-(I+II)(Sigmoid)	0.0015	0.0014	
	FCA-I(RBF)	0.0025	0.0025	
	FCA-(I+II)(RBF)	0.0020	0.0020	

To conclude, both [Tables 4](#) and [5](#) have shown that the FCA-I performs consistently faster than the OLS for subset selection problems.

6.4 Performance Evaluation of FCA-II with FCA-I

While the previous sections on performance evaluation focus on the computational complexity of FCA-I with OLS and ELM, this section focuses on the efficacy of FCA-II on improving the network performance of the SLFNs with random nodes. It has been shown in the previous sections that FCA-I, OLS, and ELM produce the network of similar generalization performance, except for their difference in the computational complexity. This section will compare the network compactness produced by FCA (FCA-I + FCA-II) with FCA-I only. [Tables 6](#) and [7](#) show the performance of the two algorithms. Again, each test case in the tables was performed 10 times, and these values are the mean values of the 10 tests. It reveals that phase II fast network construction (FCA-II) can improve the compactness of the network produced in phase I when the network size is small. Among these four benchmark problems, the improvements on regression problems are less significant while more significant on the time series and nonlinear dynamic systems modeling.

7 Conclusion

In this chapter, an FCA has been proposed for a class of SLFNs with fully random hidden nodes. An integrated framework has been developed through a two-phase network construction procedure with improved computational efficiency and network performance. Detailed performance evaluation has been performed on four benchmark problems ranging from regression, to time series prediction, to nonlinear system identification. For the first phase fast construction algorithm, FCA-I, the results show that it can consistently achieve faster computation of the linear weights than the ELM algorithm for fixed SLFNs, and consistently perform faster selection of hidden nodes than the OLS method. It has also been shown that for SLFNs with random hidden nodes, hidden node selection can significantly improve the performance of compact networks. Finally, it has been shown that the second phase (fine-tuning phase) fast construction algorithm, FCA-II, can significantly improve the network performance on time series and nonlinear system modeling problems. We believe that the fast construction approach proposed in this chapter can be used in generalized SLFNs with non-neural-alike hidden nodes proposed in Huang and Chen ([2007](#)), which is worth investigating in the future.

Acknowledgment

K. Li would like to acknowledge the helpful comments from Lei Chen of the National University of Singapore. He would also like to acknowledge the support of the International Exchange program of Queen's University Belfast.

References

- Akaike H (1974) New look at the statistical model identification. *IEEE Trans Automat Cont* AC-19(1): 716–723
- Ampazis N, Perantonis SJ (2002) Two highly efficient second-order algorithms for training feedforward networks. *IEEE Trans Neural Netw* 13(3): 1064–1074

- Bishop CM (1995) Neural networks for pattern recognition. Clarendon Press, Oxford
- Blake C, Merz C (1998) UCI repository of machine learning databases. In: Department of Information and Computer Sciences, University of California, Irvine, CA. <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- Chen S, Wigger J (1995) Fast orthogonal least squares algorithm for efficient subset model selection. *IEEE Trans Signal Process* 43(7):1713–1715
- Chen S, Billings SA, Luo W (1989) Orthogonal least squares methods and their application to non-linear system identification. *Int J Control* 50(5):1873–1896
- Chen S, Cowan CFN, Grant PM (1991) Orthogonal least squares learning algorithm for radial basis functions. *IEEE Trans Neural Netw* 2:302–309
- Chng ES, Chen S, Mulgrew B (1996) Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *IEEE Trans Neural Netw* 7(1):190–194
- Gomm JB, Yu DL (March 2000) Selecting radial basis function network centers with recursive orthogonal least squares training. *IEEE Trans Neural Netw* 11(2):306–314
- Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. *IEEE Trans Neural Netw* 5(6):989–993
- Handoko SD, Keong KC, Soon OY, Zhang GL, Brusic V (2006) Extreme learning machine for predicting HLA-peptide binding. *Lect Notes Comput Sci* 3973:716–721
- Hong X, Mitchell RJ, Chen S, Harris CJ, Li K, Irwin G (2008) Model selection approaches for nonlinear system identification: a review. *Int J Syst Sci* 39(10):925–946
- Huang G-B, Chen L (2007) Convex incremental extreme learning machine. *Neurocomputing* 70 (16–18):3056–3062
- Huang G-B, Saratchandran P, Sundararajan N (2005) A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Trans Neural Netw* 16(1):57–67
- Huang G-B, Chen L, Siew C-K (2006a) Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans Neural Netw* 17(4):879–892
- Huang G-B, Zhu Q-Y, Siew C-K (2006b) Extreme learning machine: theory and applications. *Neurocomputing* 70:489–501
- Huang G-B, Zhu Q-Y, Mao KZ, Siew C-K, Saratchandran P, Sundararajan N (2006c) Can threshold networks be trained directly? *IEEE Trans Circuits Syst II* 53(3):187–191
- Kadirkamanathan V, Niranjani M (1993) A function estimation approach to sequential learning with neural networks. *Neural Comput* 5:954–975
- Korenberg MJ (1988) Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm. *Ann Biomed Eng* 16:123–142
- Lawson L, Hanson RJ (1974) Solving least squares problem. Prentice-Hall, Englewood Cliffs, NJ
- Li K, Peng J, Irwin GW (2005) A fast nonlinear model identification method. *IEEE Trans Automat Cont* 50(8):1211–1216
- Li K, Peng J, Bai EW (2006) A two-stage algorithm for identification of nonlinear dynamic systems. *Automatica* 42(7):1189–1197
- Li K, Peng J, Bai E (2009) Two-stage mixed discrete-continuous identification of radial basis function (RBF) neural models for nonlinear systems. *IEEE Trans Circuits Syst I Regular Pap* 56(3):630–643
- Li M-B, Huang G-B, Saratchandran P, Sundararajan N (2005) Fully complex extreme learning machine. *Neurocomputing* 68:306–314
- Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006a) A fast and accurate on-line sequential learning algorithm for feedforward networks. *IEEE Trans Neural Netw* 17(6):1411–1423
- Liang N-Y, Saratchandran P, Huang G-B, Sundararajan N (2006b) Classification of mental tasks from EEG signals using extreme learning machine. *Int J Neural Syst* 16(1):29–38
- Liu Y, Loh HT, Tor SB (2005) Comparison of extreme learning machine with support vector machine for text classification. *Lect Notes Comput Sci* 3533:390–399
- Mackey MC, Glass L (1977) Oscillation and chaos in physiological control systems. *Science* 197:287–289
- Mao KZ, Huang G-B (2005) Neuron selection for RBF neural network classifier based on data structure preserving criterion. *IEEE Trans Neural Netw* 16(6):1531–1540
- Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. *SIAM J Appl Math* 11:431–441
- McLoone S, Brown MD, Irwin GW, Lightbody G (1998) A hybrid linear/nonlinear training algorithm for feedforward neural networks. *IEEE Trans Neural Netw* 9:669–684
- Miller AJ (1990) Subset selection in regression. Chapman & Hall, London
- Musavi M, Ahmed W, Chan K, Faris K, Hummels D (1992) On training of radial basis function classifiers. *Neural Netw* 5:595–603
- Panchapakesan C, Palaniswami M, Ralph D, Manzie C (2002) Effects of moving the centers in an RBF network. *IEEE Trans Neural Netw* 13:1299–1307
- Peng H, Ozaki T, Haggan-Ozaki V, Toyoda Y (2003) A parameter optimization method for radial basis function type models. *IEEE Trans Neural Netw* 14:432–438

- Peng J, Li K, Huang DS (2006) A hybrid forward algorithm for RBF neural network construction. *IEEE Trans Neural Netw* 17(11):1439–1451
- Peng J, Li K, Irwin GW (2007) A novel continuous forward algorithm for RBF neural modelling. *IEEE Trans Automat Cont* 52(1):117–122
- Peng J, Li K, Irwin GW (2008) A new Jacobian matrix for optimal learning of single-layer neural nets. *IEEE Trans Neural Netw* 19(1):119–129
- Piroddi L, Spinelli W (2003) An identification algorithm for polynomial NARX models based on simulation error minimization. *Int J Cont* 76:1767–1781
- Platt J (1991) A resource-allocating network for function interpolation. *Neural Comput* 3:213–225
- Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C: the art of scientific computing. Cambridge University Press, Cambridge
- Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
- Serre D (2002) Matrices: theory and applications. Springer, New York
- Sutanto EL, Mason JD, Warwick K (1997) Mean-tracking clustering algorithm for radial basis function centre selection. *Int J Control* 67:961–977
- Wang D, Huang G-B (2005) Protein sequence classification using extreme learning machine. In: Proceedings of international joint conference on neural networks (IJCNN2005), Montreal, Canada, 31 July – 4 August 2005
- Xu J-X, Wang W, Goh JCH, Lee G (2005) Internal model approach for gait modeling and classification. In: the 27th annual international conference of the IEEE, Engineering in Medicine and Biology Society (EMBS), Shanghai, China, 1–4 September 2005
- Yeu C-WT, Lim M-H, Huang G-B, Agarwal A, Ong Y-S (2006) A new machine learning paradigm for terrain reconstruction. *IEEE Geosci Rem Sens Lett* 3(3):382–386
- Yingwei L, Sundararajan N, Saratchandran P (1997) A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks. *Neural Comput* 9:461–478
- Zhang GL, Billings SA (1996) Radial basis function network configuration using mutual information and the orthogonal least squares algorithm. *Neural Netw* 9:1619–1637
- Zhang R, Huang G-B, Sundararajan N, Saratchandran P (2007) Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis. *IEEE/ACM Trans Comput Biol Bioinform* 4(3):485–495
- Zhu QM, Billings SA (1996) Fast orthogonal identification of nonlinear stochastic models and radial basis function neural networks. *Int J Control* 64(5): 871–886

17 Modeling Biological Neural Networks

Joaquin J. Torres¹ · Pablo Varona²

¹Institute “Carlos I” for Theoretical and Computational Physics and Department of Electromagnetism and Matter Physics, Facultad de Ciencias, Universidad de Granada, Spain

jtorres@ugr.es

²Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain

pablo.varona@uam.es

1	<i>Introduction</i>	534
2	<i>The Neuron Level</i>	534
3	<i>The Synapse Level</i>	540
4	<i>The Network Level</i>	548
5	<i>Summary</i>	561

Abstract

In recent years, many new experimental studies on emerging phenomena in neural systems have been reported. The high efficiency of living neural systems to encode, process, and learn information has stimulated an increased interest among theoreticians in developing mathematical approaches and tools to model biological neural networks. In this chapter we review some of the most popular models of neurons and neural networks that help us understand how living systems perform information processing. Beyond the fundamental goal of understanding the function of the nervous system, the lessons learned from these models can also be used to build bio-inspired paradigms of artificial intelligence and robotics.

1 Introduction

Neural systems, including those of small insects and invertebrates, are the most efficient and adaptable information-processing devices even when compared to state-of-the-art human technology. There are many computing strategies that can be learned from living neural networks. Computational neuroscience studies the nervous system from the point of view of its functionality and relies both on experimental data and on theoretical models of individual neurons and networks. Experimental data alone is not enough to address information processing because neural systems are only partially observable with the current electrophysiological and imaging techniques. Models provide a mechanism to address the role of all variables that are thought to be involved in a particular aspect of the system under study. In many cases, the models can provide new insight and hypotheses that can be tested experimentally. Computational neuroscience involves multidisciplinary research and contributes to a better understanding of the nervous system (including several brain diseases), to the development of prosthetic devices and brain-machine interfaces, and to the design of novel paradigms of artificial intelligence and robotics.

When modeling biological neural media, one must take into account different components that extend over several temporal and spatial scales and involve individual cells, synapses, and structural topology. This chapter is organized as follows. In the first section we review the theoretical descriptions of single neurons, from detailed biophysical paradigms to simplified models. In the second section we review how single neurons can receive external signals through different kinds of synapses. In the third section we describe small circuit paradigms. Finally in the fourth section we describe models of large neural networks.

2 The Neuron Level

Informational processes in the nervous system take place at subcellular, cellular, network, and system levels. The timescales of these processes can be very different, and there is a cumulative interaction between all temporal and spatial scales. In this context it is not straightforward to select a single unit of information processing to describe neural activity. A widespread view, which is followed here, considers neurons as the basic units of information processing in the brain.

During the last six decades, phenomenological models have been developed to understand individual neuron behavior and their network dynamics. How detailed does the description of a neuron have to be to build a model that is biologically realistic and computationally tractable?

In this section we will try to answer this question by reviewing some of the most popular neuronal models. For a more detailed description of these and other models see, for example, Koch and Segev (1998), Koch (1999), Gerstner and Kistler (2002), and Izhikevich (2004).

Neuron models can be assigned into classes depending on the general goal of the modeling. If one wishes to understand, for example, how information is generated and propagated in a single neuron due to a complex synaptic bombardment, a detailed conductance-based model may be required. This model will have to describe the distribution of active properties in the membrane throughout the morphology and the propagation of the electrical activity along different compartments. On the other hand, if one wants to describe emerging phenomena in a network due to the synchronization of many units, it may be reasonable enough to use a simplified model that reproduces the main features of neuronal activity related to this overall goal. Here, modeling approaches are divided into two classes: biophysical models and simplified models. In many cases, neural models built on simplified paradigms can lead to more detailed biophysical models based on the same dynamical principles but implemented with more biophysically realistic mechanisms. By the same token, a realistic model can also be simplified by keeping only those features that allow us to reproduce the behavior that is important for the modeling goal. A good indication that the level of the description was chosen wisely comes if the model can reproduce with the same parameters the main bifurcations observed in the experiments.

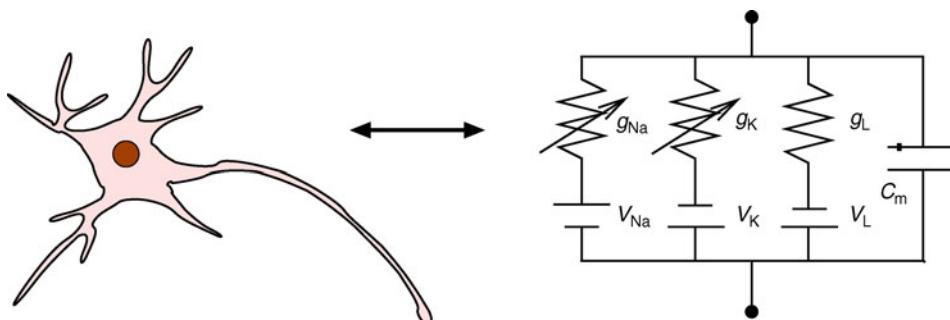
2.1 Biophysical Models

2.1.1 Hodgkin–Huxley Model

Neurons talk to each other in the language of action potentials. An action potential, also called a spike, is a rapid change in voltage across the membrane. Biophysical models describe the generation of action potentials in terms of the electrical properties of the membranes. Detailed conductance-based neuron models take into account the ionic currents flowing across the membrane (Koch 1999) and describe their active (voltage dependent) and passive (capacitive and voltage independent) electrical properties (see Fig. 1). Neural membranes contain

Fig. 1

Electrical circuit that represents a single neuron as an isopotential compartment with active and passive electrical properties. The circuit on the right includes the description of two active ionic channels (Na^+ and K^+) and one passive leakage current.



several types of voltage-dependent sodium, potassium, and calcium channels. The dynamics of some of these channels can also depend on the concentration of specific ions. In addition, there is a leakage current that is often considered to be independent of the voltage level. The flow of all these currents results in changes in the voltage across the membrane. The probability that a type of ionic channel is open depends nonlinearly on the membrane voltage and on the current state of the channel. This was first modeled by Hodgkin and Huxley (H–H) (1952).

Hodgkin and Huxley expressed the nonlinear conductance dependencies of the ionic channels in a set of coupled nonlinear differential equations that describe the electrical activity of the cell as

$$\begin{aligned} C_m \frac{dV(t)}{dt} &= g_L[V_L - V(t)] + g_{Na}m(t)^3 h(t)[V_{Na} - V(t)] + g_K n(t)^4 [V_K - V(t)] + I \\ \frac{dm(t)}{dt} &= \frac{m_\infty(V(t)) - m(t)}{\tau_m(V(t))} \\ \frac{dh(t)}{dt} &= \frac{h_\infty(V(t)) - h(t)}{\tau_h(V(t))} \\ \frac{dn(t)}{dt} &= \frac{n_\infty(V(t)) - n(t)}{\tau_n(V(t))} \end{aligned} \quad (1)$$

where $V(t)$ is the membrane potential, and $m(t)$, $h(t)$, and $n(t)$ represent empirical variables describing the activation and inactivation of the ionic conductances. In this model, there are two active channels, a Na^+ channel and a K^+ channel, and a passive leakage channel L . I is an external current that acts as a stimulus to the model; C_m is the capacitance of the membrane; g_L , g_{Na} , and g_K are the maximum conductances of the ionic channels; and V_L , V_{Na} , V_K are the reversal potentials for the different ionic channels. The steady-state values of the active conductance variables m_∞ , h_∞ , n_∞ and the time constants τ_m , τ_h , τ_n have a nonlinear voltage dependence, typically through sigmoidal or exponential functions, whose parameters can be estimated from experimental data.

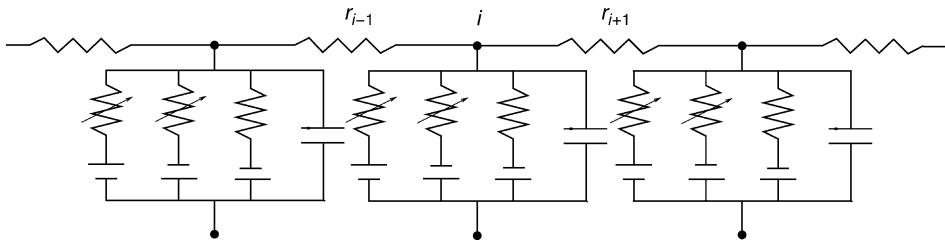
The Hodgkin–Huxley model is one of the most successful phenomenological descriptions of neuronal activity, and can be enhanced by the addition of other voltage and ionic-concentration currents. These active membrane conductances can enable neurons to generate different spike patterns that are commonly observed in a variety of motor neural circuits and brain regions. The biophysical mechanisms of spike generation enable individual neurons to encode different stimulus features into distinct spike patterns. Spikes and bursts of spikes of different durations code for different stimulus features, which can be quantified without *a priori* assumptions about those features (Kepecs and Lisman 2003). The different timescales and the nonlinear interaction between the different variables can also account for different coding mechanisms, preferred input/output relations, and even the creation of new information (Latorre et al. 2006; Rabinovich et al. 2006).

2.1.2 Multicompartmental Models

The H–H ODEs (Eq. 1) represent point neurons. As mentioned above, some modeling goals may require additional descriptions of the morphology, and thus a multicompartmental description of the neuron is needed. In this case, two adjacent isopotential compartments

Fig. 2

Multicompartment models are built out of several isopotential compartments linked by axial resistances between them.



i and j can be linked through axial resistances $r_{i,j}$ and the membrane potential of the i th compartment can be described with the following equation:

$$C_i \frac{dV_i}{dt} = g_L(V_{L_i} - V_i) + \frac{(V_{i+1} - V_i)}{r_{i,i+1}} + \frac{(V_{i-1} - V_i)}{r_{i,i-1}} + I_{act,i}$$

where $I_{act,i}$ is the sum of all active currents in the i th compartment and $r_{i,j}$ is the axial resistance between compartments i and j (see Fig. 2). Following this strategy, one can build complex morphology patterns for single neuron models (Koch and Segev 1998; Varona et al. 2000).

2.2 Simplified Models

There is a large list of models derived from the H–H model, which in general try to simplify the complexity of the H–H equations by reducing the number of variables or their associated nonlinearities. In many cases, these simplifications allow the use of theoretical analysis that adds to the insight provided by the numerical simulations of the models. The most abstract models do not keep any biophysical parameter and just consider neurons as basic integrators or pure linear threshold gates. One of the main motivations to simplify the H–H model is to reduce the underlying computations and, as a result, to allow for large-scale network simulations.

2.2.1 FitzHugh–Nagumo Model

The FitzHugh–Nagumo model describes the excitation properties of a neuron without a detailed description of the nonlinearities associated with the electrochemical properties of the ionic channels. This model is often written as

$$\begin{aligned} \frac{dV(t)}{dt} &= V(t) - cV^3(t) - W(t) + I \\ \frac{dW(t)}{dt} &= V(t) + bW(t) - a \end{aligned}$$

where $V(t)$ is the membrane potential, $W(t)$ describes the dynamics of fast currents, and I is an external current. The parameter values a , b , and c are constants chosen to generate spiking

activity with different features. This reduced model with only polynomial nonlinearities accounts for the description of oscillatory spiking neural dynamics including bistability (Nagumo et al. 1962; FitzHugh 1961).

2.2.2 Morris–Lecar Model

The Morris–Lecar model (1981) maintains the description of the nonlinearities associated with two ionic channels while reducing the number of dimensions by considering an instantaneously responding voltage-sensitive Ca^{2+} conductance:

$$\begin{aligned}\frac{dV(t)}{dt} &= g_L[V_L - V(t)] + n(t)g_n[V_n - V(t)] + g_m m_\infty(V(t))[V_m - V(t)] + I \\ \frac{dn(t)}{dt} &= \lambda(V(t))[n_\infty(V(t)) - n(t)] \\ m_\infty(V) &= \frac{1}{2} \left(1 + \tanh \frac{V - V_m}{V_m^0} \right) \\ n_\infty(V) &= \frac{1}{2} \left(1 + \tanh \frac{V - V_n}{V_n^0} \right) \\ \lambda(V) &= \phi_n \cosh \frac{V - V_n}{2V_n^0}\end{aligned}$$

where $V(t)$ is the membrane potential, $n(t)$ describes the recovery activity of the K^+ current, and I is an external current. This simplified model, which reduces the number of dynamical variables of the H–H model, allows the use of phase plane methods for bifurcation analysis. The Morris–Lecar type models are widely used to study spike rate and precise timing codes (Gutkin 1998).

2.2.3 Hindmarsh–Rose Model

The H–R model is a simplified three-dimensional model that uses a polynomial approximation to the right-hand side of a Hodgkin–Huxley model (Hindmarsh and Rose 1984)

$$\begin{aligned}\frac{dV(t)}{dt} &= W(t) + aV(t)^2 - bV(t)^3 - Z(t) + I \\ \frac{dW(t)}{dt} &= C - dV(t)^2 - W(t) \\ \frac{dZ(t)}{dt} &= r[s(V(t) - V_0) - Z(t)]\end{aligned}$$

where $V(t)$ is the membrane potential, $W(t)$ describes fast currents, $Z(t)$ describes slow currents, and I is an external current. This simple model can display complex spiking and spiking-bursting regimes. H–R type models have been successfully used to build artificial electronic neurons that can interact bidirectionally with living neurons to recover a central pattern generator rhythm (Szucs et al. 2000; Pinto et al. 2000).

2.2.4 Integrate and Fire Models

Integrate and fire models reduce the computational cost of calculating the nonlinearities associated with the action potential by describing the dynamics of subthreshold activity and

imposing the spike when a threshold is reached. The first integrate and fire model was introduced by Lapicque in 1907 (Lapicque 1907), before the mechanisms responsible for action potential generation were known. Many of these types of models can be described as

$$\frac{dV(t)}{dt} = \begin{cases} \frac{-V(t) + R_m I}{\tau} & 0 < V(t) < \theta \\ \frac{V(t_0^+)}{\tau} = 0 & V(t_0^-) = \theta \end{cases}$$

where $V(t)$ is the neuron membrane potential, θ is the threshold for spike generation, R_m is the membrane resistance, $\tau \equiv R_m \times C_m$, and I is an external stimulus current. A spike occurs when the neuron reaches the threshold θ in $V(t)$ after which the cell is reset to the resting state.

2.2.5 Map Models

Discrete time maps are simple phenomenological models of spiking and bursting neurons in the form of difference equations that are computationally very fast, and thus adequate to model large populations of neurons (Cazelles et al. 2001; Rulkov 2001). An example of such a map is

$$\begin{aligned} V(t+1) &= \frac{\alpha}{1 + V(t)^2} + W(t) + I \\ W(t+1) &= W(t) - \sigma V(t) - \beta \end{aligned}$$

where $V(t)$ represents the spiking activity and $W(t)$ represents a slow variable. The parameters of the map allow us to tune the type of spiking activity that this model can produce (Rulkov 2002).

2.2.6 McCulloch and Pitts Model

The McCulloch–Pitts paradigm was the first computational model for an artificial neuron (McCulloch and Pitts 1943). This model neglects the relative timing of neural spikes and is also known as a linear threshold device model that can be described as

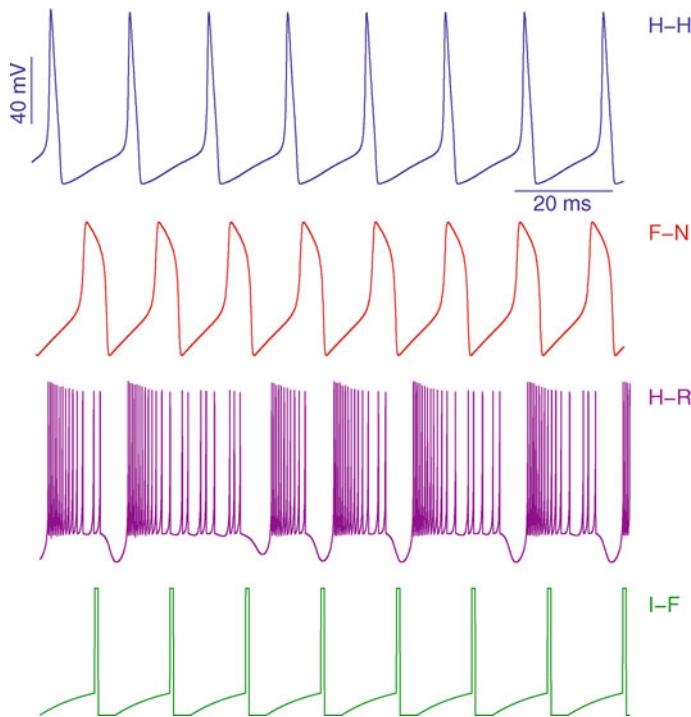
$$\begin{aligned} V(t+1) &= \Theta(I(t) - \theta) \\ \Theta(X) &= 1, \quad X > 0 \\ &= 0, \quad X \leq 0 \end{aligned}$$

where $V(t)$ is the variable that represents the activity of a neuron that receives a total input $I(t)$ from other neurons, and θ is the neuron threshold for firing. This model is also used in computational neuroscience and has proved to be very adequate to address the role of connection topologies in different neural systems (Nowotny and Huerta 2003; Garcia-Sanchez and Huerta 2003).

❸ *Figure 3* represents several time series of spiking activity from some of the models described in this section under the stimulus of a constant current injection. Of course, constant currents are not a usual stimulus to a neuron. Stimuli arrive to these cells in the form of synapses, which will be the subject of the next section.

Fig. 3

Time series of spiking activity from several widely used neuronal models. From top to bottom: Hodgkin–Huxley model (H–H), FitzHugh–Nagumo model (F–N), Hindmarsh–Rose (H–R) model (in a chaotic spiking-bursting regime), and integrate and fire model (I–F).



3 The Synapse Level

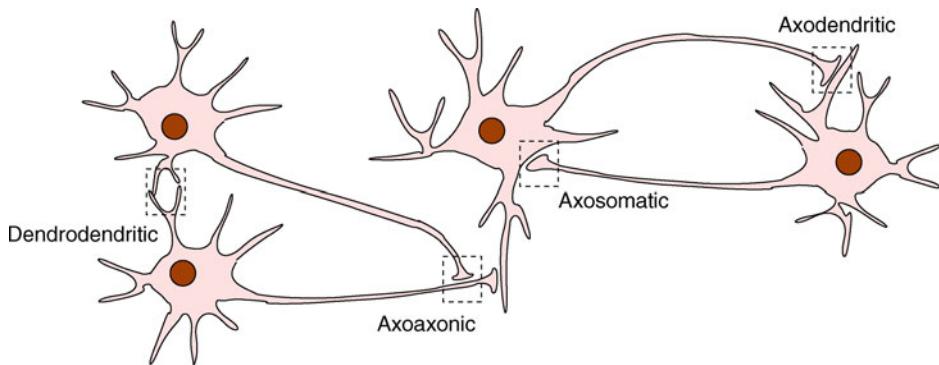
In the previous section we briefly reviewed several mathematical techniques used by neuroscientists to model the electrical activity of neurons. These techniques model, at different levels of description, the so-called neuronal signal, action potential, or neuronal *spike*. In an actual neuron, an action potential is generated when the sum of all involved membrane ionic currents is enough to overcome an excitation threshold. When this occurs, it is said that the cell is excitable. Once the spike has been generated, it propagates through the axon. At the end of the axon, the signal can be transmitted to other neurons through complex inter-neuron structures called synapses.

In order to build biologically inspired neural network models, it is fundamental to understand how neurons communicate through the synapses. To go further, let us first briefly review the different types of synaptic connections that can be found in living neural systems.

Traditionally and depending on the position at which the neural signal generated in the *presynaptic* neuron is transmitted to the *postsynaptic* neuron, the synapses have been denoted as (see [Fig. 4](#) for an illustration) *axodendritic* (when the transmission occurs from the axon of the presynaptic neuron to the dendrites of the postsynaptic neuron), *axoaxonic* (between the two axons, as in several types of “inhibitory” synapses), and *axosomatic* (between the axon

Fig. 4

Classification of synapses depending on their relative location along the cell body of both presynaptic and postsynaptic neurons.



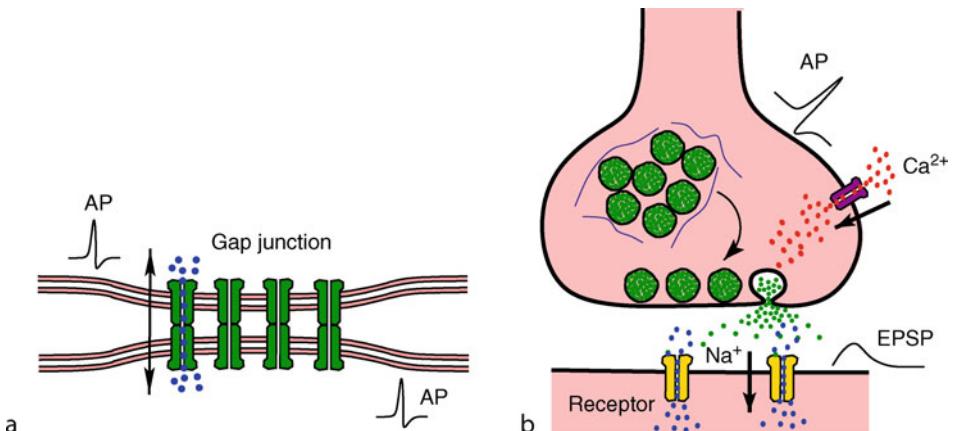
of the presynaptic neuron and the soma of the postsynaptic neuron). In recent years, many other types of connections between different parts of the neurons involved in the synapse have been discovered, including *dendrodendritic*, *dendro-axonic*, and *dendro-somatic synapses* (see for instance Pinault et al. 1997). In many models of neural networks built with point neurons, all these types of synapses play the same role, and a frequent assumption is that neurons i and j are connected through a synapse w_{ij} without the specification of the precise spatial location at which it occurs.

3.1 Electrical Versus Chemical Synapses

More relevant for a realistic modeling of neural networks is the classification of synapses depending on the mechanism that the cell uses for the transmission of the signal. The synapse can be either *electrical* or *chemical*. In an electrical synapse, the separation between the presynaptic and postsynaptic membranes is about 3.5 nm (Kandel et al. 2000) and they are joined by specific protein structures called *gap junctions* (see ▷ Fig. 5), which are specialized ionic channels that connect the cytoplasm of both cells (Kandel et al. 2000). When a neural signal (action potential) comes to the gap junction, it depolarizes (or hyperpolarizes) the membrane, which induces the opening of the channels and the diffusion of ions through them from one neuron to another. In a chemical synapse, however, the separation is larger than in the electrical synapses, namely, 20–40 nm (Kandel et al. 2000), so there is no physical contact among the cells. In this case the transmission of the signal is actively mediated by the release, in this space, of some chemical messengers called *neurotransmitters*, which are generated at the end of the axon of the presynaptic neuron (see ▷ Fig. 5). Signal transmission in a chemical synapse occurs as follows: when a presynaptic action potential arrives at the synapse, it produces the opening of some neurotransmitter vesicles near the membrane, the so-called *ready releasable pool* of vesicles. This causes the release and diffusion of a large number of neurotransmitters in the inter-synaptic space. The neurotransmitters have affinity for certain molecular receptors in the postsynaptic membrane and, after binding to them, they induce the opening of specific ionic channels. This produces the depolarization (or hyperpolarization) of the postsynaptic

Fig. 5

Scheme showing the differences during signal transmission between electrical synapses (panel a) and chemical synapses (panel b). Signal transmission in electrical synapses is a passive mechanism in which ions can quickly diffuse between presynaptic and postsynaptic neurons. In chemical synapses, signal transmission occurs as the result of different biophysical mechanisms, which occur sequentially in different timescales, and which generate an excitatory (or inhibitory) postsynaptic potential, namely, EPSP (or IPSP for inhibitory synapses), during a time delay of τ_{in} .



membrane. Finally, the released vesicles are replaced from others that are in a *reserve pool* of vesicles, located relatively far from the membrane (see [Fig. 5](#)). In both cases, the communication among neurons is the result of a flux of ions through the postsynaptic membrane constituting a synaptic current, namely I_{syn} . Theoretically, one can see the effect of such a current on the excitability of the membrane of the postsynaptic neuron i using, for instance, a standard Hodgkin–Huxley formalism (see [Sect. 1](#)):

$$C_m \frac{dV_i(t)}{dt} = - \left[\sum_{\kappa} I_{i,\kappa}(t) + I_i^{syn}(t) \right] \quad (2)$$

where C_m is the capacitance of the membrane, I_{κ} represents the different ionic currents defining the excitability of the neuron membrane (see [Sect. 1](#)), and $I_i^{syn}(t) = \sum_j I_{ij}^{syn}(t)$ is the total synaptic current received by the postsynaptic neuron i . Using Ohm's Law, one has for an electrical synapse

$$I_{ij}^{syn}(t) = \bar{G}_{ij} \times [V_i(t) - V_j(t)] \quad (3)$$

where $\bar{G}_{ij} = \text{constant}$ is the maximum synaptic conductance. That is, [Eq. 3](#) takes into account the passive diffusion of ions through the synapse. For a chemical synapse, however, one has

$$I_{ij}^{syn} = G_{ij}(t) \times (V_i - E_j^{syn}) \quad (4)$$

with E_j^{syn} being the synaptic reversal potential, that is, the voltage value at which the synaptic current cancels. Moreover, $G_{ij}(t)$ now depends on time following a complex dynamics that considers all the biophysical processes affecting the release and trafficking of neurotransmitters, their posterior binding to the synaptic receptors, and the permeability of ionic channels,

in the postsynaptic membrane, to the passage of ions through them, as explained above. The shape of $G_{ij}(t)$ is usually modeled using an α function, namely,

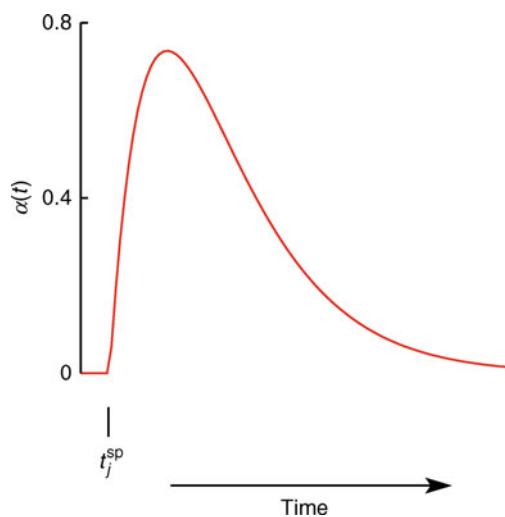
$$G_{ij}(t) = \bar{G}_{ij} \times \Theta(t - t_j^{\text{sp}}) \times \alpha(t - t_j^{\text{sp}}) \quad (5)$$

with $\alpha(t) = t \times e^{-t/\tau_{\text{in}}}$ (see Fig. 6), where t_j^{sp} is the time at which the presynaptic spike occurs, τ_{in} is the time constant for the synaptic transmission, and $\Theta(x)$ is the Heaviside step function (Also the α -function can be modeled as the difference of two exponentials, namely, $\alpha(t) = e^{-t/\tau_1} - e^{-t/\tau_2}$ with $\tau_1 > \tau_2$). A computationally efficient alternative to the α -function is discussed in Destexhe et al. (1994).

These two different processes for electrical and chemical synapses make their functionality strongly different, in such a way that electrical synapses are faster (a small fraction of a millisecond) and the signal can be transmitted in both directions. In the chemical synapses, however, the transmission of the signal is slower (about 1–5 ms) and they are intrinsically asymmetric because information is transmitted from the presynaptic neuron to the postsynaptic neuron. (This fact has strong implications under a theoretical point of view, if one wants to build and study neural networks models that include symmetric or asymmetric synaptic connections. Large neural networks with symmetric connections can be studied using standard equilibrium statistical physics techniques, which is not possible for asymmetric

Fig. 6

Temporal behavior of an α function $\alpha(t)$ (as defined in the text), which traditionally has been used to model an evoked postsynaptic response in a single chemical synapse after the arrival of a presynaptic action potential. The function has an initially fast increase, which accounts for the effect of the neurotransmitter binding to the postsynaptic receptors, and an exponential decay with a relatively slow time constant τ_{in} , which considers the typical time delay observed in chemical synapses. Note that $\alpha(t)$ can be used to model both excitatory and inhibitory synapses, the main functional differences being due to the particular value of the synaptic reversal potential E_j^{syn} , which is around zero for excitatory synapses and strongly hyperpolarized for inhibitory synapses (see main text).



connections; see, for instance, Peretto (1992)). Concerning their functionality, electrical synapses are important to enhance fast synchronization of a large group of electrically connected neurons or to transmit a metabolic signal among neurons (because different types of ions or other molecules can diffuse through gap-junctions). The transmission of signals on chemical synapses, however, involves more complex nonlinear processes that act like gain functions, which amplify the effect of the weak electrical presynaptic currents, depolarizing relatively large extensions of the postsynaptic membrane.

3.2 Excitatory and Inhibitory Synapses

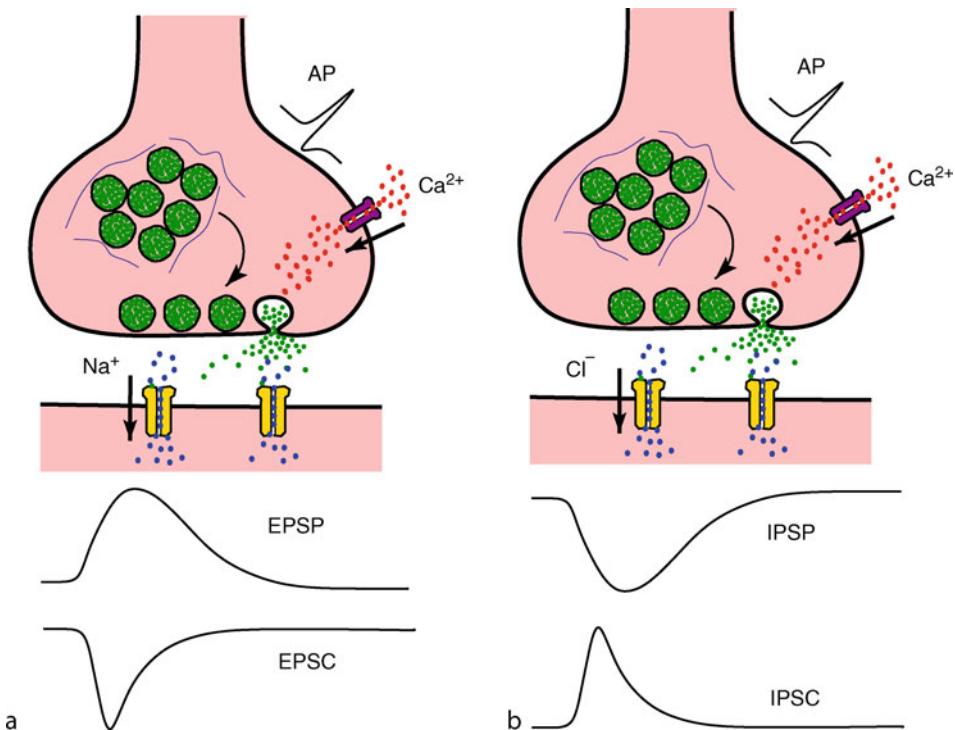
The synapses can also be classified into *excitatory*, when the release of neurotransmitters produces a depolarization of the postsynaptic membrane potential (typically due to the influx of Na^+ or Ca^{2+} ions through the postsynaptic ionic channels) or *inhibitory*, when there is a hyperpolarization of the postsynaptic membrane potential (i.e., there is a flux of K^+ ions outside the cell, or an influx of Cl^- ions from the extracellular medium) (see Fig. 7). For these different effects, not only the type of neurotransmitter but also the type of the postsynaptic receptor is important. The most conventional neurotransmitter associated with an excitatory postsynaptic response is glutamate, which activates both the *fast* AMPA/kainate and the relatively *slow* NMDA excitatory receptors (Koch and Segev 1998). The resulting generated synaptic current is produced by the influx of Na^+ or Ca^{2+} ions (i.e., is an inward current) and it has associated an approximate reversal potential of $E^{\text{syn}} = 0$ mV. (Note that the synaptic reversal potential is smaller than, for instance, the Na^+ equilibrium potential, which is about +55 mV. This is because, in excitatory synapses, the synaptic channels are permeable for both Na^+ and K^+ ions, which induces a flux of both ions in opposite directions (inward for Na^+ and outward for K^+) resulting in a net synaptic current that cancels at the reversal potential of $E^{\text{syn}} = 0$ mV (Kandel et al. 2000).) Other neurotransmitters associated with excitatory responses are aspartate, acetylcholine, and dopamine. On the other hand, gamma-aminobutyric acid (GABA) is the most common neurotransmitter involved in inhibitory synapses in the central nervous system and, similarly to glutamate, it activates a *fast* inhibitory receptor, or GABAa, and a relatively *slow* inhibitory receptor named GABAb (Koch and Segev 1998; Kandel et al. 2000). In the spinal cord, however, the most common neurotransmitter associated with inhibitory responses is glycine. The activation of the GABAa receptor produces a hyperpolarized synaptic current due to the influx of Cl^- ions (inward current) and has $E^{\text{syn}} \approx E_{\text{Cl}} = -70$ mV. The activation of the GABAb receptor, however, produces the efflux of K^+ ions from the cell to the extracellular medium (outward current) and has a reversal potential of approximately $E^{\text{syn}} = E_K = -95$ mV.

3.3 Dynamic Synapses

When modeling neural systems, synapses have been traditionally considered as *static* identities with the only possible modification of their maximum conductances (\bar{G}_{ij}) due to learning (Hopfield 1982), this being a *slow* process that takes place in a temporal scale of seconds, minutes, or years. Recently, it has been reported in actual neural media that the synaptic strength (measured as the amplitude of the generated postsynaptic response) also varies in short timescales producing a fluctuating postsynaptic response (Dobrunz and Stevens 1997),

Fig. 7

Differences between excitatory (panel a) and inhibitory (panel b) chemical synapses. In excitatory synapses, an excitatory postsynaptic ionic current (EPSC) is generated, which can be inward (due to the influx of Na^+ ions as plotted in the panel) or outward (due to the efflux of K^+ ions through the channel) and which induces an excitatory or depolarized postsynaptic membrane potential (EPSP). In inhibitory synapses, an inhibitory postsynaptic current (IPSC) is generated, usually due to the influx of Cl^- ions through the channel (i.e., it is an inward current) and it produces a hyperpolarization of the postsynaptic membrane potential (IPSP).



which can be depressed (synaptic depression) or enhanced (synaptic facilitation) depending on the presynaptic activity (Abbott et al. 1997; Markram et al. 1998; Tsodyks et al. 1998). This defines the so-called *dynamic synapses*. In recent years, several studies have shown that activity-dependent processes such as synaptic depression and facilitation have a key role in information processing in living neural systems. For instance, it is well known that short-term depression plays an important role in several emerging phenomena in the brain, such as selective attention (Buia and Tiesinga 2005; McAdams and Maunsell 1999) and cortical gain control (Abbott et al. 1997), and it is responsible for the complex switching behavior between activity patterns observed in neural network models with depressing and facilitating synapses (Pantic et al. 2002; Cortes et al. 2006).

From a biophysical point of view, synaptic depression is a consequence of the fact that the number of available neurotransmitters in the synaptic buttons is limited in the ready releasable pool. Therefore, the neuron needs some time to recover these synaptic resources in order to transmit the next incoming spike. As a consequence, the dynamics of the synapse is affected by

an activity-dependent mechanism that produces nonlinear effects in the postsynaptic response (see [Fig. 8](#)).

On the other hand, synaptic facilitation is produced by the influx of calcium ions into the neuron near the synapse through voltage-sensitive channels after the arrival of each presynaptic spike. These ions bind to some acceptor that gates and facilitates the neurotransmitter vesicle depletion in such a way that the postsynaptic response increases for successive spikes (Kamiya and Zucker 1994) due mainly to residual cytosolic calcium (see [Fig. 9](#)). Facilitation, therefore, increases the synaptic strength for high-frequency presynaptic stimuli. However, given a train of presynaptic spikes, a strong facilitating effect for the first incoming spikes produces an even stronger depressing effect for the last spikes in the train. The phenomenon of synaptic facilitation has been reported to be relevant for synchrony and selective attention (Buia and Tiesinga 2005), and in the detection of bursts of action potentials (Matveev and Wang 2000; Destexhe and Marder 2004). More recently, it has been proved that synaptic facilitation had a positive effect, for instance, in the efficient transmission of temporal correlations between spike trains arriving from different synapses (Mejias and Torres 2008). This feature, known as spike coincidence detection, has been measured *in vivo* in cortical neurons and is related to some dynamical processes that affect neuron firing thresholds (Azouz and Gray 2000), so that it seems to be an important mechanism for efficient transmission of information in actual neural media.

Fig. 8

Simulated train of EPSPs generated in a single synapse after the arrival of a train of presynaptic spikes at frequency of 5 Hz (top panels) and 15 Hz (bottom panels), when the synapse is static (a), and dynamic with only a depressing mechanism (b) and with depressing and facilitating mechanisms (c). The synapse parameters were $U_{SE} = 0.1$ and $\tau_{rec} = \tau_{fac} = 0$ for panel (a), $U_{SE} = 0.3$, $\tau_{rec} = 600$ ms, $\tau_{fac} = 0$ for panel (b), and $U_{SE} = 0.02$, $\tau_{rec} = 600$ ms, $\tau_{fac} = 6$ s for panel (c).

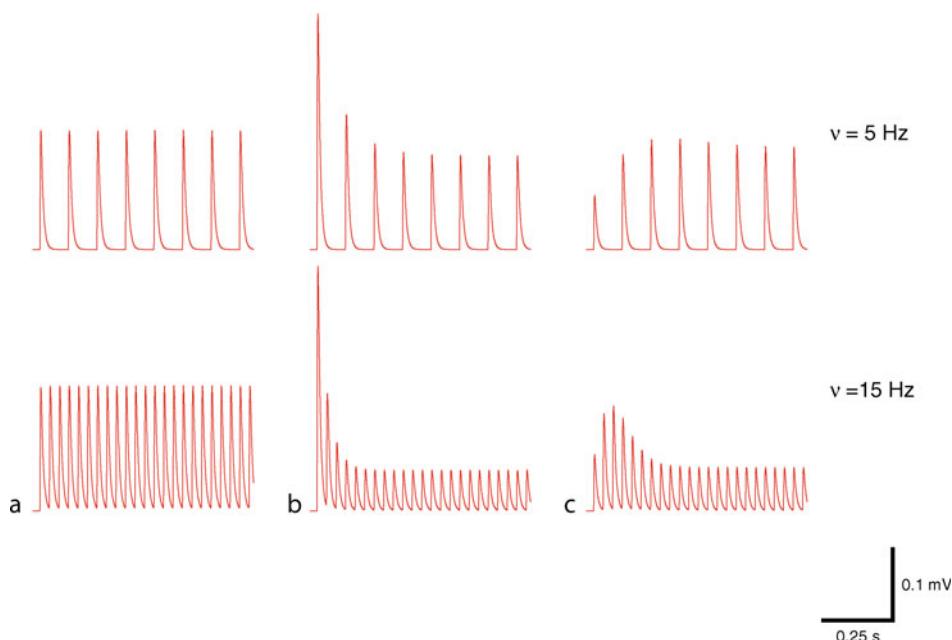
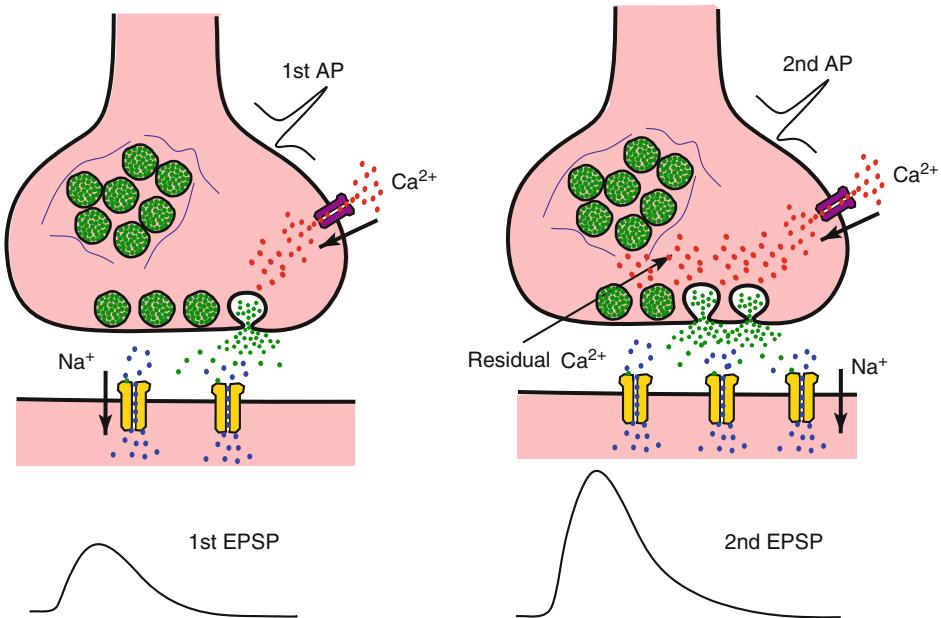


Fig. 9

Biophysical mechanism for facilitation: After a first presynaptic spike that induces a given EPSP, the existence of a certain amount of residual-free cytosolic calcium increases the neurotransmitter probability release for a second spike that enlarges the secondary EPSP.



One of the most important theoretical approaches to modeling dynamic synapses is the phenomenological model presented in Tsodyks and Markram (1997), which considers that the state of a synapse between a presynaptic neuron j and a postsynaptic neuron i is governed by the system of equations

$$\begin{aligned}\frac{dx_j(t)}{dt} &= \frac{z_j(t)}{\tau_{\text{rec}}} - U_j(t)x_j(t)\delta(t - t_j^{\text{sp}}) \\ \frac{dy_j(t)}{dt} &= -\frac{y_j(t)}{\tau_{\text{in}}} + U_j(t)x_j(t)\delta(t - t_j^{\text{sp}}) \\ \frac{dz_j(t)}{dt} &= \frac{y_j(t)}{\tau_{\text{in}}} - \frac{z_j(t)}{\tau_{\text{rec}}}\end{aligned}\quad (6)$$

where $x_j(t)$, $y_j(t)$, $z_j(t)$ are the fractions of neurotransmitters in recovered, active, and inactive states, respectively. (Neurotransmitters in a recovered state are those that are in vesicles refilling the ready releasable pool, active neurotransmitters are those that have been released and can produce a postsynaptic response, and inactive neurotransmitters are the remaining.) Here, τ_{in} and τ_{rec} are the inactivation and recovery time constants, respectively. Depressing synapses are obtained, for instance, for $U_j(t) = U_{\text{SE}}$ constant, which represents the maximum number of neurotransmitters that can be released (activated) after the arrival of each presynaptic spike. The delta functions appearing in Eq. 6 consider that a spike arrives to the synapse at a fixed time $t = t_j^{\text{sp}}$. Typical values of these parameters in cortical depressing synapses are $\tau_{\text{in}} = 3$ ms, $\tau_{\text{rec}} = 800$ ms, and $U_{\text{SE}} = 0.5$ (Tsodyks and Markram 1997). The

synaptic facilitation mechanism can be introduced assuming that $U_j(t)$ has its own dynamics related with the release of calcium from intracellular stores and the influx of calcium from the extracellular medium each time a spike arrives (see for instance [Fig. 5](#)), which in this phenomenological model is $U_j(t) \equiv u_j(t)(1 - U_{\text{SE}}) + U_{\text{SE}}$ with

$$\frac{du_j(t)}{dt} = \frac{u_j(t)}{\tau_{\text{fac}}} + U_{\text{SE}}[1 - u_j(t)]\delta(t - t_j^{\text{sp}}) \quad (7)$$

Here, $u_j(t)$ is a dynamical variable that takes into account the influx of calcium ions into the presynaptic neuron near the synapse through voltage-sensitive ion channels (Bertram et al. 1996). As we described previously, these ions can usually bind to some acceptor that gates and facilitates the release of neurotransmitters. A typical value for the facilitation time constant is $\tau_{\text{fac}} = 530$ ms (Markram et al. 1998). The variable $U_j(t)$ represents then the maximum fraction of neurotransmitters that can be activated, both by the arrival of a presynaptic spike (U_{SE}) and by means of facilitating mechanisms (i.e., $u_j(t)(1 - U_{\text{SE}})$).

One can use then $y_j(t)$ instead of the “static” $\alpha(t)$ function in [Eqs. 4](#) and [5](#) to model the corresponding synaptic current I_{ij}^{syn} . However, one can also assume that, for instance, the excitatory postsynaptic current (EPSC) generated in a particular synapse after the arrival of a presynaptic spike is proportional to the fraction of neurotransmitters that are in the active state, that is, $I_{ij}^{\text{syn}} = -A_{\text{SE}} \times y_j(t)$ (the minus indicates that the current is inward in this particular case), where A_{SE} is the maximum postsynaptic current that can be generated and, typically, it takes a value $A_{\text{SE}} \approx 42.5$ pA for cortical neurons. The previous assumption for I_{ij}^{syn} remains valid as long as the corresponding generated postsynaptic potential is below the threshold for the generation of a postsynaptic spike, which is about $V_{\text{th}} = 13$ mV. In this situation, the postsynaptic potential varies slowly compared to the variation in the synaptic conductance (which is determined by $y_j(t)$) and, therefore, it can be considered almost constant during the generation of the synaptic current.

4 The Network Level

4.1 Modeling Small Networks

Once one knows how to model individual neurons and synapses one is ready to build models of neural networks. The relationship between the dynamics of individual neurons and the activity of a large network in the nervous system is complex and difficult to assess experimentally. In most cases, there is a lack of knowledge about the details of the connections and the properties of the individual cells. In large-scale models, most of the relevant parameters can only be estimated. By studying small circuits of invertebrates first, one can have a closer relationship between a model and the living neural network, and thus a more detailed understanding of the basic principles underlying neural dynamics. Another advantage in this case is that the predictions of the models can be more easily tested in systems where neurons and connections are accessible and identifiable.

4.1.1 Central Pattern Generators

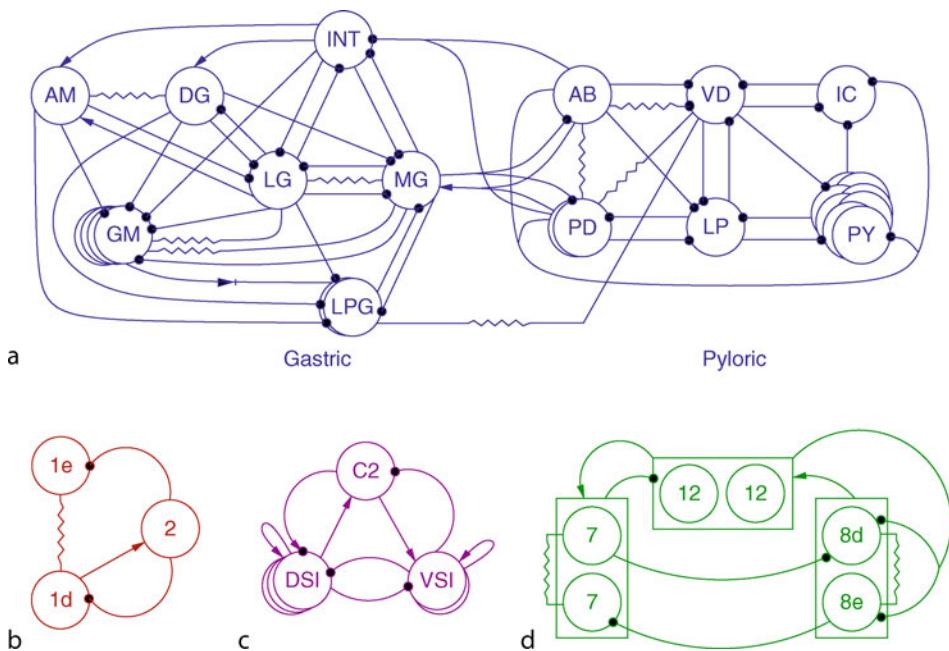
Central pattern generators (CPGs) are small neural circuits that produce stereotyped cyclic outputs without rhythmic sensory or central input (Marder and Calabrese 1996; Stein et al.

1997; Selverston 2005). CPGs are oscillators that underlie the production of motor commands for muscles that execute rhythmic motion involved in locomotion (Getting et al. 1989), respiration (Ramirez et al. 2004), heart beat (Cymbalyuk et al. 2002), etc. CPG circuits produce signals in the form of spatiotemporal patterns with specific phase lags between the activity of different motoneurons that innervate muscles.

Because of their relative simplicity, the network architecture and the main features of CPG neurons and synapses are known much better than any other brain circuits. Examples of typical invertebrate CPG networks are shown in Fig. 10. Common to many CPG circuits are network topologies built with electrical and mutual inhibitory connections. The cells of these circuits typically display spiking–bursting activity. The characteristics of the spatiotemporal patterns generated by the CPG, such as burst frequency, phase, duration of the slow wave, etc., are determined by the intrinsic properties of each individual neuron, the properties of the synapses, and the architecture of the circuit. Regulation mechanisms at all levels contribute to network stability and the generation of robust and flexible motor patterns.

Fig. 10

Examples of CPG topologies in invertebrates: (a) the gastric and pyloric CPGs in crustacea (modified from Selverston and Moulins (1987)), (b) the feeding CPG in *Planorbid* (modified from Arshavsky et al. (1985)), (c) the swimming CPG in *Tritonia* (modified from Getting (1989)), (d) the swimming CPG in *Clione* (modified from Arshavsky et al. (1991)). Dots represent inhibitory synapses while arrows represent excitatory synapses. Electrotonic gap junctions are represented by resistors. Note that all topologies are non-open, that is, each neuron receives feedback from at least one other neuron of the CPG (Huerta et al. 2001). As individual neurons are identifiable from one preparation to another in these small circuits, the connectivity diagrams can be fully established.



Although CPGs autonomously establish rhythmic firing patterns, they also receive supervision signals from higher centers and from local reflex pathways. These inputs allow the animal to constantly adapt its behavior to the environment by producing different stable spatiotemporal patterns (Selverston and Moulins 1987; Rabinovich et al. 2006). Thus, CPGs produce robust rhythms but at the same time are flexible to incoming signals. There are many experimental results that show that CPGs can be reconfigured by neuromodulatory substances in the blood or released synaptically. These substances induce changes both in the connectivity (by altering synaptic strength) and in the intrinsic properties of the neurons (by affecting the voltage-dependent conductances) (Simmers and Moulins 1988; Marder and Bucher 2007).

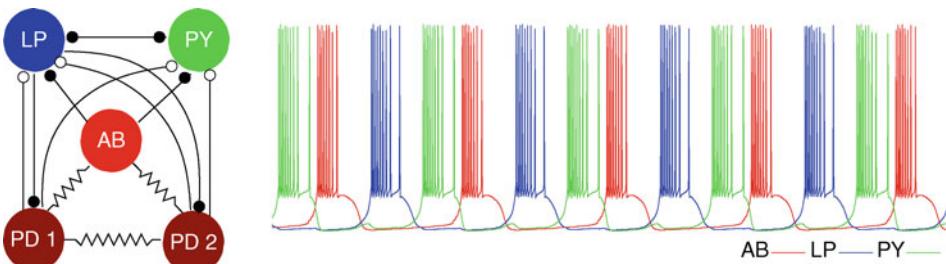
Many modeling studies have contributed to understanding the interaction between intrinsic and network properties in CPGs as illustrated here with a few examples. Many of these modeling efforts use the pyloric CPG of crustaceans (Prinz 2006), one of the best known neural circuits. This CPG is composed of 14 neurons (see [Fig. 10a](#)) that produce a spiking-bursting activity that keeps a characteristic phase relationship between the bursts of other members due, mainly, to the inhibitory connection topology of the CPG (Selverston and Moulins 1987). The electrical activity of the neurons in this CPG is propagated to the muscles where the spikes are integrated to exert force on different places of the pylorus. The main task of the pyloric CPG is to transport shredded food from the stomach to the digestive system.

A remarkable property of the motoneurons of the pyloric CPG is that, when they are isolated from other cells in the circuit, they display highly irregular, and, in fact, chaotic spiking-bursting activity (Abarbanel et al. 1996; Elson et al. 1998, 1999; Selverston et al. 2000). Models of the pyloric CPG have been able to explain how the mutual inhibitory connections and the slow subcellular dynamics are responsible for the regularity of the rhythms (Varona et al. 2001a, b) (see [Fig. 11](#)). The dynamical richness of the individual neurons contributes to a fast response in the form of a new CPG rhythm when incoming signals or neuromodulatory input arrives to the circuit. Models have also suggested that non-open connection topologies are widespread in many different CPGs to produce regular rhythms while preserving the effect of individual rich dynamics to maximize the gain of information (Huerta et al. 2001; Stiesberg et al. 2007).

Recent experimental findings have revealed important new phenomena in the dynamics of CPG circuits. One of them is related to the presence of robust and cell-type-specific intra-burst firing patterns in the bursting neurons of the pyloric CPG (Szucs et al. 2003).

Fig. 11

Reduced pyloric CPG model and the regular triphasic rhythm produced by this circuit. Individual neurons are modeled with H-H chaotic dynamics. Neurons within this circuit produce neural signatures in the form of cell-specific interspike intervals (Szucs et al. 2003; Latorre et al. 2006).



These patterns, termed as interspike interval signatures or neural signatures, depend on the intrinsic biophysical properties of the bursting neurons as well as on the synaptic connectivity of the surrounding network, and can influence the overall activity of the system. The existence of these precise and reproducible intra-burst spiking patterns might indicate that they contribute to specific computational tasks. Modeling studies have shown that H–H models can react distinctly to the neural signatures and point them out as elements of a multicoding mechanism for spiking–bursting neurons (Latorre et al. 2006). Neural signatures coexist with the generation of robust rhythms (see Fig. 11) and thus the bursting signal can contain information about the who and the what of the information. Beyond the context of the CPGs, and taking into account that neural signatures have also been found in vertebrate neurons (Zeck and Masland 2007), these results suggest the design of artificial neural network paradigms with units that use multicoding strategies and local information discrimination for information processing.

Another phenomenon that can provide inspiration for bio-inspired CPG control in robotics is related to the existence of dynamical invariants (such as the maintenance of the ratio between the change in burst periods and the change in phase lags) when the CPG suffers perturbations (Reyes et al. 2003). These dynamical invariants are built through the cellular and synaptic contributions to network properties, and can account for a wide range of efficient rhythms in response to external stimuli for many types of motor tasks.

4.1.2 Small Sensory Networks

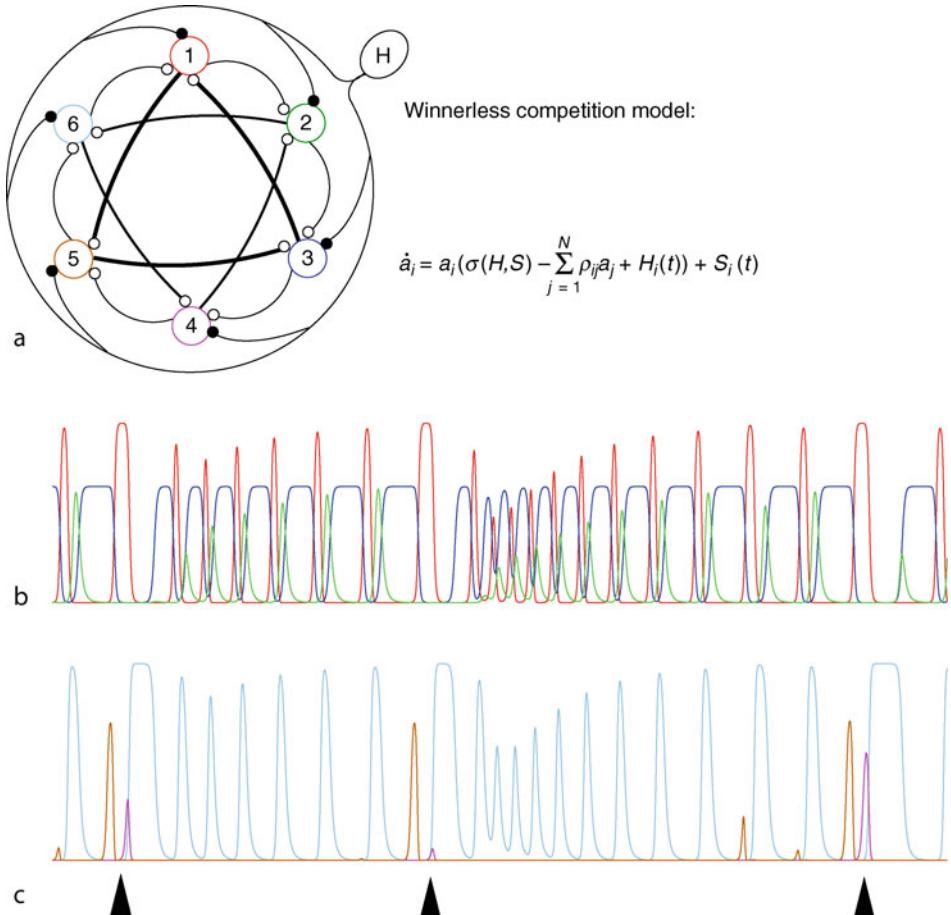
Simple circuits of sensory systems have also contributed to a better understanding of intrinsic sensory network dynamics and its role in the sensory–motor transformation. Some of the clearest examples are found in invertebrate systems in which the different processing stages are more directly accessible (Lewis and Kristan 1998; Levi and Camhi 2000; Laurent et al. 2001; Beenhakker and Nusbaum 2004). In some cases, modeling results have provided striking hypotheses about small sensory networks that then have been tested experimentally. One of these examples is related to the dual role of the gravimetric sensory network of the mollusk *Clione*. This model will also serve to illustrate the use of rate models as another type of simplified paradigms for network activity.

Clione is a marine mollusk that has to maintain a continuous motor activity in order to keep its preferred head-up orientation. The motor activity is controlled by the wing central pattern generator and the tail motoneurons that use the signals from its gravity sensory organs, the statocysts (Panchin et al. 1995). A statocyst is a small sphere in which the statolith, a stone-like structure, moves according to the gravitational field. The statolith exerts pressure on the statocyst receptor neurons (SRCs) that are located in the statocyst internal wall. When excited, these mechanoreceptors send signals to the neural systems responsible for wing beating and tail orientation. The SRCs also receive input from a pair of cerebral hunting interneurons (H) (Panchin et al. 1995), which are activated in the presence of the prey by signals from the chemoreceptors. *Clione* does not have a visual system and although its chemosensors can detect the presence of prey, they are not direction sensitive. When hunting behavior is triggered, *Clione* loops and turns in a complex trajectory trying to locate its prey.

A six-receptor neural network model with synaptic inhibitions was built to describe a single statocyst (Varona et al. 2002) (see Fig. 12). The model of the statocyst sensory network suggested the hypothesis that the SRC sensory network has a dual role and participates not

Fig. 12

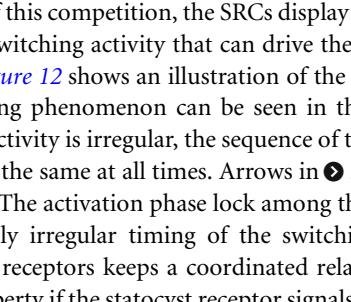
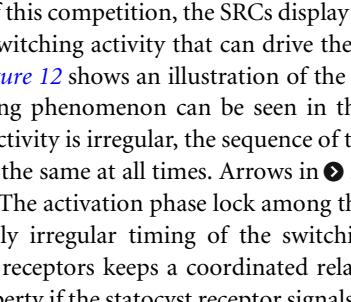
Panel a: Lotka–Volterra rate model of the gravimetric sensory network of the mollusk *Clione*. **Panels b and c:** Time series of rate activity for each neuron (colors correspond to the different neurons in panel a, units are dimensionless). This network displays irregular winnerless competition (WLC) dynamics with activation phase locks (black arrows) for those neurons that are active in a particular time window.



only in keeping a head-up position during routine swimming, but also in the organization of the complex hunting behavior of *Clione* (Varona et al. 2002).

The model of a statocyst receptor network consists of six neurons (see Fig. 12, panel a). Each receptor neuron follows a Lotka–Volterra type dynamics given by the equation shown in panel a, where $a_i > 0$ represents the instantaneous spiking rate of the neurons, $H_i(t)$ represents the excitatory stimulus from the cerebral hunting interneuron to neuron i , and $S_i(t)$ represents the action of the statolith on the receptor that is pressing. When there is no stimulus from the hunting neuron ($H_i = 0, \forall i$) or the statolith ($S_i = 0, \forall i$), then $\sigma(H, S) = -1$ and all neurons are silent; $\sigma(H, S) = 1$ when the hunting neuron is active and/or the statolith is pressing one of the receptors. $\rho_{i,j}$ is the connection matrix that follows the diagram illustrated in the figure.

When there is no activation of the sensory neurons by the hunting neuron, the effect of the statolith ($S_i \neq 0$) in the model is to induce a higher rate of activity on one of the neurons (the neuron i where it rests for a big enough S_i value). We assume that this higher rate of activity affects the behavior of the motoneurons to organize the head-up position. The other neurons are either silent or have a lower rate of activity and we can suppose that they do not influence the posture of *Clione*. The model displays a winner-take-all situation for normal swimming.

When the hunting neuron is active, a completely different behavior arises. We assume that the action of the hunting neuron overrides the effect of the statolith and thus $S_i \approx 0, \forall i$. The network with a stimuli from the hunting neuron establishes a winnerless competition (WLC) among the SRCs. As a result of this competition, the SRCs display a highly irregular, and in fact chaotic (Varona et al. 2002) switching activity that can drive the complex loops and turns of *Clione*'s hunting search.  [Figure 12](#) shows an illustration of the nonsteady switching activity of the receptors. An interesting phenomenon can be seen in this multifunctional network. Although the timing of each activity is irregular, the sequence of the switching among the SRC activity (when it is present) is the same at all times. Arrows in  point out this fact for neuron 4 (cf. panels b and c). The activation phase lock among the statocyst receptor neurons emerges in spite of the highly irregular timing of the switching dynamics. The complex switching activity among the receptors keeps a coordinated relation in the activation of the signals. This is a desirable property if the statocyst receptor signals are active participants in the control of *Clione*'s tail and wing motion during hunting behavior.

The Lotka–Volterra rate formalism allows us to establish the conditions for stability of the sequential switching. The predictions of this model have been demonstrated experimentally (Levi et al. 2004, 2005) and indeed the gravimetric network has a distinct role during routine swimming and during hunting. During normal swimming, only the neurons that are excited by the statolith are active, which leads to a winner-take-all mode as a result of inhibitory connections in the network. However, during hunting search behavior, the hunting neuron triggers WLC between all statocyst neurons whose signals participate in the generation of a complex motion that the animal uses to scan the space until it finds its prey.

Beyond this simple network, WLC dynamics has been proposed as a general paradigm to represent transient dynamics in many types of neural functions (Rabinovich et al. 2006): from sensory encoding to decision making in cognitive tasks (Rabinovich et al. 2008). The mathematical image of WLC is a stable heteroclinic channel, that is a set of trajectories in the vicinity of a heteroclinic skeleton that consists of saddles and unstable separatrices that connect their surroundings (Rabinovich et al. 2008). This image satisfies the competing requirements of stability and flexibility, which is required in most descriptions of neural activity.

4.2 Modeling Large Networks

Since the pioneering work of McCulloch and Pitts (1943), neural networks models have been widely used to artificially mimic (with more or less success) some features of the complex behavior of different neural systems. This is due to the fact that, as a first approximation, neural systems and even the whole brain can be viewed as large networks constituted by nodes connected by edges or *synapses*, where the nodes are excitable. This means that each element has a threshold and a refractory time between consecutive responses, a behavior that in some cases impedes thermal equilibrium and induces complex emergent phenomenology

including, for instance, learning and memory recall, pattern formation, and chaotic neural activity. In recent decades, many different theoretical approaches and computational techniques (with different levels of description for neuron, synapses, and network topology) have been developed to study extended neural systems (see for instance, Brette et al. (2007) for a recent review about the modeling of networks of spiking neurons and Swiercz et al. (2006) for a recent model of synaptic plasticity).

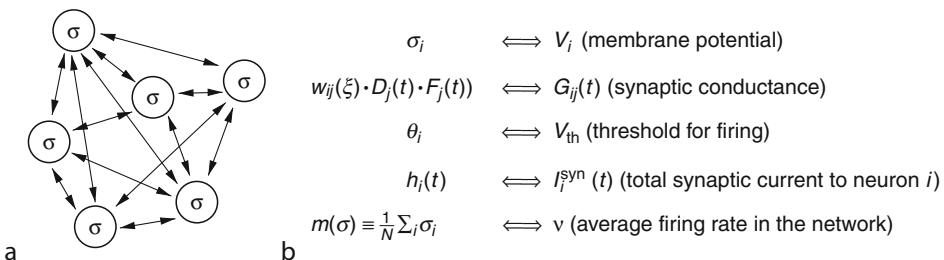
One of the most successful models of large neural networks was due to Amari (1972) and Hopfield (1982) who separately defined the so-called Amari–Hopfield attractor neural network (ANN). This is a neural network where the edges are fixed in time and weighted according to a prescription (Hebb 1949), which in a sense stores or memorizes information (during a slow learning process) from a set of given patterns of activity. As a consequence, these patterns become attractors of the phase space dynamics and one may interpret that the system shows retrieval of the stored patterns. This process, known as *associative memory*, can be considered as a suitable representation of the memory and recall processes observed in the brain. Actual neural systems do much more than just recalling a memory and staying there, however. That is, one should expect dynamic instabilities or other destabilizing mechanisms on the attractor that allow for recall of new memories, for instance. This expectation is reinforced by recent experiments suggesting that synapses can also undergo rapid changes with time, as we explained in a previous section, which may both determine brain complex tasks (Pantic et al. 2002; Abbott et al. 1997; Tsodyks et al. 1998) and induce irregular and perhaps chaotic activity (Barrie et al. 1996; Korn and Faure 2003). In order to include these synaptic features in neural networks models, we first mathematically define a large neural network model.

4.2.1 Mathematical Definition of a Neural Network

In a general form, a large neural network can be defined by a set of N nodes or *neurons* (placed in some specific positions x_i of a d -dimensional vector space) whose state is represented by some variable s_i . Neurons are connected to each other by edges or *synapses* whose state, *strength*, or *weight* is represented by a real variable w_{ij} (see Fig. 13). The map of connections defines the *topological structure* of the network. Following the standard ANN model introduced above, we assume that w_{ij} fixed and containing information from a set of M

Fig. 13

(a) Scheme of an artificial neural network with arbitrary network topology and symmetric connections $w_{ij} = w_{ji}$ and (b) analogy between the variables defining the dynamic of a neural network and those related with the dynamics of actual neural systems.



patterns of activity $\xi^v = \{\xi_i^v, i = 1, \dots, N\}, v = 1, \dots, M$ are previously stored in a slow learning process, that is, $w_{ij} = w_{ij}(\xi)$.

In many neural networks models in the literature, neurons have been considered as binary elements with their state being $s_i = +1, -1$ (in the so-called $-1, +1$ code) or $\sigma_i = \frac{1+s_i}{2} = 0, 1$ (in the $0, 1$ code). In biologically motivated neural networks models, each one of these two possible values represents a neuron in a silent state (below the threshold for firing) or a neuron that fires an action potential. Several recent studies have already shown that binary neurons may capture the essence of cooperation in many more complex settings (see, for instance, Pantic et al. (2002) in the case of integrate and fire neuron models of pyramidal cells).

In the following, the more “realistic” $0, 1$ code shall be taken into consideration. Each node σ_i then receives information from its neighbors (the neurons that are directly connected to it), which is measured in terms of the so-called *local field*

$$h_i(\sigma, t) = \sum_{j \neq i} w_{ij}(\xi) \sigma_j(t) \quad (8)$$

In the absence of other external factors, when this field is larger than some threshold θ_i the neuron i fires, so that one can update the current state of a particular neuron (assuming discrete time update) as

$$\sigma_i(t+1) = \Theta[h_i(\sigma, t) - \theta_i] \quad (9)$$

where $\Theta(X)$ is the Heaviside step function. If the network is in the presence of different sources of noise (such as the stochastic neurotransmitter release or the stochasticity associated with the opening and closing of different ionic channels responsible for the excitability of the neuron), which can affect the dynamics of the neurons, one can substitute the dynamics (► Eq. 9) by the most realistic stochastic dynamics

$$\text{Prob}[\sigma_i(t+1) = +1] = \mathcal{S}[2\beta\{h_i(\sigma, t) - \theta_i\}] \quad (10)$$

where $\mathcal{S}(X)$ is a nonlinear sigmoidal function, such as, for instance, $\mathcal{S}(X) = \frac{1}{2}[1 + \tanh(X)]$. The dynamical rule (► 10) can be updated in parallel and synchronously for all neurons in the network, or sequentially at random, one neuron per time step (although one can also think of other types of updating). In the last case and for fixed symmetric weights $w_{ij} = w_{ji}$, the steady state is characterized by a Gibbs equilibrium probability distribution $P(\sigma) \propto e^{-\beta \mathcal{H}(\sigma)}$ with $\mathcal{H}(\sigma) = -\frac{1}{2} \sum_i h_i(\sigma)$.

From this very general and simple picture, one can easily find a simple analogy between the variables defining the dynamics of artificial neural networks and the variables involved in the dynamics of actual extended neural systems, as shown in ► Fig. 13b. Note that in this scheme, extra variables $D_j(t)$ and $F_j(t)$ modulating the maximum conductances are introduced due to learning $w_{ij}(\xi)$, and these could define, for instance, the dynamics of the depressing and facilitating synaptic mechanisms introduced in ► Sect. 3.

4.2.2 Neural Networks with Dynamic Synapses

The stochastic dynamics (► 10) can be derived from a more general mathematical formalism, which assumes that both neurons and synapses can suffer stochastic changes in different timescales due to the presence of one (or some times more) thermal bath at a given temperature $T \equiv \beta^{-1}$. The dynamics of the system is then represented by a general *master equation*

(for more details, see, for instance, Torres et al. (1997, 2007a) and Cortes et al. (2006)). Some particular cases of this general model, with a biological motivation, account for fast synaptic changes coupled with neuron activity (one can think, for instance, of the stochasticity of the opening and closing of the neurotransmitter vesicles, the stochasticity of the postsynaptic receptor, which in turn has several sources, for example, variations of the glutamate concentration in the synaptic cleft, and differences in the power released from different locations on the active zone of the synapses (Franks et al. 2003)). As a consequence, the resulting model exhibits much more varied and intriguing behavior than the standard static Amari–Hopfield model. For example, the network exhibits high sensitivity to external stimuli and, in some conditions, chaotic jumping among the stored memories, which allows for better exploring of the stored information.

For a better understanding of how these instabilities can emerge in the memory attractors and the resulting network behavior, an example will be described. In particular, we will study (in a biologically motivated neural network) the consequences of introducing a realistic dynamics for synaptic transmission, such as that described by the set of equations (6 and 7) in the stochastic dynamics of neurons described by Eq. 10.

To go further it is necessary to propose some simplifications. First, the dynamics (Eq. 10) assumes a discrete time evolution of neuron states, so one can consider that the typical temporal scale for neuron dynamics in Eq. 10 is larger than the absolute refractory period in actual neurons, which is about (5 ms). On the other hand, one has that the synaptic delay $\tau_{\text{in}} = 2 \text{ ms} \ll \tau_{\text{rec}}, \tau_{\text{fac}}$, so $y(t) = y(t^{\text{sp}})e^{-(t-t^{\text{sp}})/\tau_{\text{in}}} \approx y(t^{\text{sp}}) = U(t^{\text{sp}})x(t^{\text{sp}})$. This allows us to consider a discrete dynamics for synapse states and to reduce the dimensionality of the system Eqs. 6 and 7 from four to two equations (note that $x + y + z = 1$, because they are molar fractions), which is important when one has to simulate large networks. One then has that the generated synaptic current in a single synapse is $I_{ij}^{\text{syn}}(t) \propto U_j(t)x_j(t)$. Therefore, the total synaptic current that arrives at a particular neuron i (using the analogy described in Fig. 13) can be written as

$$I_i^{\text{syn}}(t) = h_i(t) = \sum_j w_{ij}(\xi) D_j(t) F_j(t) \sigma_j(t) \quad (11)$$

where $D_j(t) \equiv x_j(t)$ and $F_j(t) \equiv U_j(t)/U_{\text{SE}}$ represent the depressing and facilitating mechanisms, respectively, affecting the maximum conductances $w_{ij}(\xi)$, with

$$\begin{aligned} x_j(t+1) &= x_j(t) + \frac{1 - x_j(t)}{\tau_{\text{rec}}} - U_j(t)x_j(t)\sigma_j(t) \\ U_j(t+1) &= U_j(t) + \frac{U_{\text{SE}} - U_j(t)}{\tau_{\text{fac}}} + U_{\text{SE}}[1 - U_j(t)]\sigma_j(t) \end{aligned} \quad (12)$$

Here, the binary character of the neuron variables σ_i is used to substitute the delta functions $\delta(t - t_j^{\text{sp}})$ in Eqs. 6 and 7 by $\sigma_j(t)$. The model described by Eqs. 10–12 can be considered as the more general model of a biologically inspired ANN, which includes dynamic synapses with depressing and facilitating mechanisms. It reduces to the Amari–Hopfield model for static synapses when $x_i = F_i = 1$ (that is $\tau_{\text{rec}}, \tau_{\text{fac}} \ll 1$); $\theta_i = \frac{1}{2} \sum_{j \neq i} w_{ij}(\xi)$ and $w_{ij}(\xi)$ are given by the standard *covariance learning rule*, namely,

$$w_{ij}(\xi) = \frac{1}{Nf(1-f)} \sum_{\mu=1}^M (\xi_i^\mu - f)(\xi_j^\mu - f) \quad (13)$$

where $\zeta^\mu = \{\zeta_i^\mu = 0, 1, i = 1 \dots, N\}$ are M random patterns of neuron activity with mean $\langle \zeta_i^\mu \rangle = f$. To measure the degree of correlation between the current network state and one of these stored patterns it is useful to introduce the overlap function

$$m^\mu(\sigma) \equiv \frac{1}{Nf(1-f)} \sum_i (\zeta_i^\mu - f)\sigma_i \quad (14)$$

Assuming a synchronous and parallel updating of neurons states (e.g., a little dynamics) using [Eq. 10](#) and random unbiased patterns ($f = 1/2$), the previous model can be studied within the standard mean-field approach $\sigma_i \rightarrow \langle \sigma_i \rangle$, where $\langle \cdot \rangle$ means an average over the probability distribution $P_t(\sigma)$. In the simplest case of only a single pattern, or for a finite number of patterns, that is, when the loading parameter $\alpha \equiv M/N \rightarrow 0$ in the thermodynamic limit ($N \rightarrow \infty$), the dynamics of the network is driven by a $6M$ -dimensional discrete map (see Torres et al. [\(2007b\)](#) for details)

$$\mathbf{v}_{t+1} = \mathcal{F}(\mathbf{v}_t) \quad (15)$$

where $\mathcal{F}(X)$ is a complex multidimensional nonlinear function of

$$\mathbf{v}_t = \{m_+^\mu(t), m_-^\mu(t), x_+^\mu(t), x_-^\mu(t), u_+^\mu(t), u_-^\mu(t)\} \quad \mu = 1, \dots, M \quad (16)$$

This vector represents a set of order parameters defined as averages of the microscopic dynamical variables over the sites that are active and inert, respectively, in a given pattern μ , that is

$$c_+^\mu(t) \equiv \frac{1}{Nf} \sum_{i \in \text{Act}(\mu)} c_i(t), \quad c_-^\mu(t) \equiv \frac{1}{N(1-f)} \sum_{i \notin \text{Act}(\mu)} c_i(t) \quad (17)$$

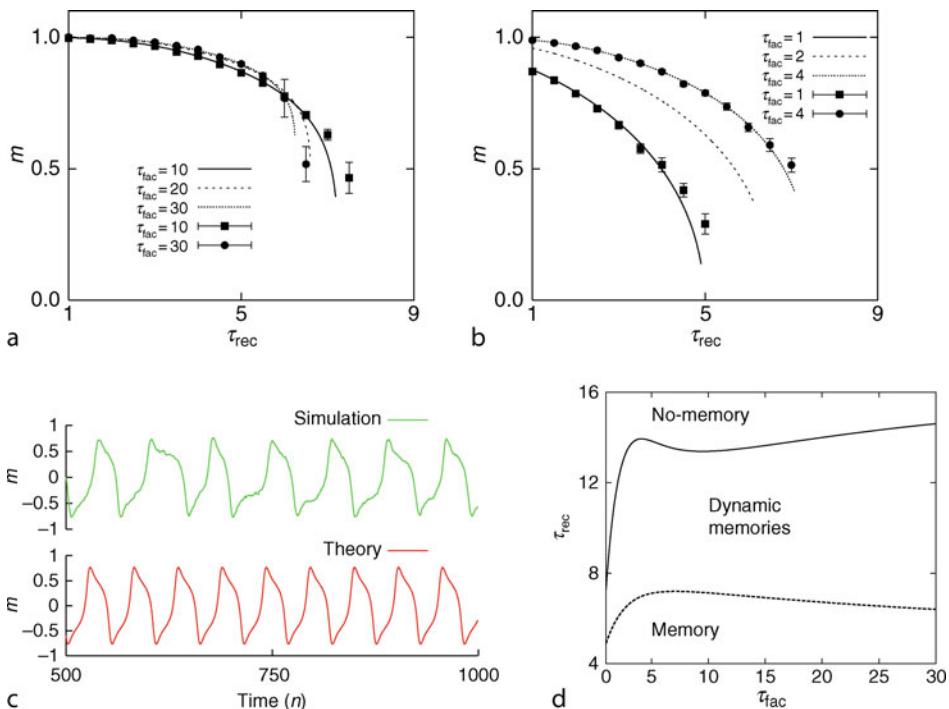
with c_i being $\sigma_i(t)$, $x_i(t)$, and $u_i(t)$, respectively.

One can easily compute the fixed point solutions of the dynamics ([Eq. 15](#)) and study their local stability. Similarly to the Amari–Hopfield standard model one finds that the stored memories ζ^μ are stable attractors in some regions of the space of relevant parameters, such as T , U_{SE} , τ_{rec} and τ_{fac} . Their stability and the error in the memory recall process, however, is highly influenced by the particular value of these parameters (see for instance [Fig. 14a, b](#)). In particular, there are some critical values for which the memories destabilize, and an oscillatory regime, in which the activity of the network is visiting different memories, can emerge. [Figure 14c](#) shows an example of such oscillatory behavior, and a typical phase diagram with different dynamical phases in the (τ_{rec}, τ_{fac}) space is shown in [Fig. 14d](#). The critical values for the destabilization of memories depend on the relative competition between depression and facilitation, which are two processes with, a priori, different effects over the attractors (synaptic depression tries to put the activity of the network out of attractors whereas facilitation induces a faster access of the activity to the attractors). The emergence of oscillations between different patterns of activity induced by activity-dependent synaptic processes has been recently related to the appearance of oscillations between “up and down” activity states observed in cortical areas of living animals (Holzman and Tsodyks [2006](#)).

The instability in the memories induced by an activity-dependent synaptic process, such as depression and facilitation, enhances the sensitivity of the network to external stimuli. An example of the latter is shown in [Fig. 15](#). In this simulation, an external varying stimulus is presented to the network during a temporal window at which τ_{fac} (for the dynamic synapse case) is continuously increased. The stimulus consists of short pulses of amplitude δ ,

Fig. 14

Panels **a** and **b** show the steady state or fixed point memory attractors (as in the standard Amari–Hopfield model), described in terms of the overlap $m \equiv m^1$, which can be reached in the dynamics of the general model of ANN with dynamic synapses. Data points correspond to simulations of Eqs. 6 and 7 for different values τ_{rec} and τ_{fac} , and lines are the corresponding mean field steady-state curves. There are critical values of the synaptic parameters for which these fixed point memory solutions disappear and oscillatory attractors such as that shown in panel **c** can emerge. In Panel **d** a typical phase diagram in the space $(\tau_{\text{rec}}, \tau_{\text{fac}})$ for $T = U_{\text{SE}} = 0.1$ is presented, which shows the critical lines at which the different emergent phases appear.



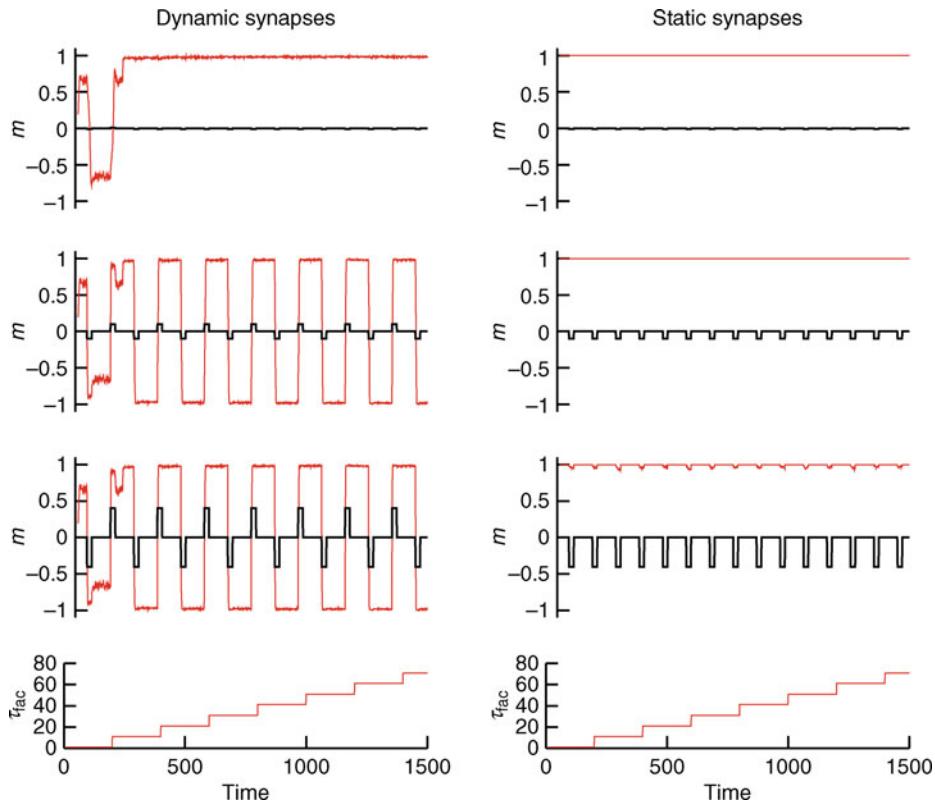
presented at a given frequency such that the activity of the network is already in a memory attractor. The sign of the stimulus changes in such a way that every time it is presented it tries to drive the activity of the network out of the attractor. The analysis of this simulation shows that dynamic synapses allow for a better response to the stimulus even for $\delta \ll 1$ (weak signals). This behavior can be useful for living animals to respond efficiently to a continuously varying environment.

The phenomenology described above is general and it has been also observed in more realistic situations, such as for instance, in networks of integrate and fire neurons, with other type of stored patterns (e.g., more overlapping patterns), for a large number of stored patterns, and for a continuous dynamics (see Pantic et al. (2002)).

Another important issue in the study of ANN is the maximum number of memories M_{max} , per neuron, that the network is able to store and recall without having significant errors, which defines the critical (or maximum) storage capacity of the system. This capacity is

Fig. 15

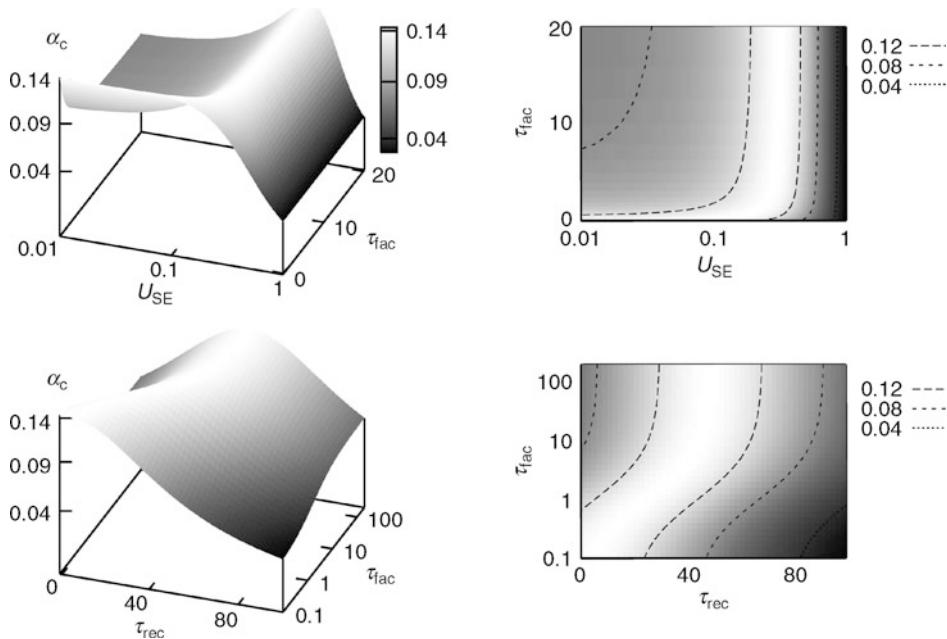
Comparison of the responses to a varying external stimulus in an ANN with dynamic (left) and static (right) synapses, for increasing values of the τ_{fac} . Only in the case of dynamic synapses, can the system optimally respond even for a very weak stimulus.



commonly denoted by the ratio $\alpha_c = M_{\max}/N$. It is well known that the critical storage capacity is affected by certain considerations about actual neural systems, such as constraints in the range of values of the synaptic strength (Fusi and Abbott 2007), the mean activity level of the stored patterns (Tsodyks and Feigelman 1988; Amit and Tsodyks 1991), or the topology of the network (McGraw and Menzinger 2003; Torres et al. 2004). Recently the effect of activity-dependent synaptic processes such as depression and facilitation on storage capacity has been reported (Mejias and Torres 2009). The main results can be summarized in [Fig. 16](#). This shows that synaptic facilitation improves the storage capacity with respect to the case of depressing synapses, for a certain range of the synaptic parameters. Moreover, if the level of depression is not very large, facilitation can increase the critical storage capacity, reaching in some cases the value obtained with static synapses (which is the maximum that one can obtain considering a Hebbian learning rule with unbiased random patterns in a fully connected network). These results suggest that a certain level of facilitation in the synapses might be positive for an efficient memory retrieval, while the function of strongly depressed synapses could be more oriented to other tasks concerning, for instance, the dynamical processing of data, as explained above.

Fig. 16

Maximum storage capacity α_c in a “realistic” model of ANN with dynamic synapses computed within a general mean-field approach and for different sets of the synapse parameters (with $\tau_{rec} = 2$ in the top panels and $U_{SE} = 0.02$ in the bottom panels). Contour lines and different levels of gray define the regions in the space of relevant parameters with similar storage capacity.



4.2.3 Network Topology

Large networks, such as those described above, are usually modeled using fully connected networks. This is convenient because it allows for simple analytical treatments such as mean-field approaches, as seen previously. Extended neural systems, however, present a map of connectivity that can largely differ from such a simple picture and which, in general, are very difficult to determine. An important question to address then is how the network structure can influence the functionality of the system. There have been many theoretical studies focused on this particularly interesting issue. For instance, it has been recently reported that ANN with activity-dependent synapses work optimally for particular network topologies where the level of “heterogeneity” (measured as the number of different connectivity degrees) in the network is maximal (Johnson et al. 2008). Moreover, it is reasonable to think that during development a given neural system can self-organize to reach an optimal topological structure that allows for a better performance of the system during the realization of a particular task (Beggs and Plenz 2003). Finally, recent studies have also shown the complex interplay between network topology and neuron activity to explain the proper or improper functioning of a given neural system, which can be related to brain diseases such as, for instance, epilepsy (Dyhrfjeld-Johnsen et al. 2007; Roxin et al. 2004).

5 Summary

In this chapter we reviewed some theoretical approaches currently used in the modeling of biological neural networks from single cells to large ensembles of neural networks. Models are useful mathematical tools to understand complex emergent behavior observed in living neural media, which arises from different phenomena occurring at subcellular, cellular, network, and system levels, and involving different spatial and temporal scales.

In many cases, models contribute to explain specific aspects of a particular neural system, but there are also efforts to unveil general principles underlying information processing in the brain (Ashby 1960; Rosenblatt 1962; Freeman 1972; von der Malsburg 1981; Hopfield 1982; Rabinovich et al. 2006). In computational neuroscience, both bottom-up and top-down approaches are needed. Often models are accused of being able to fit any particular behavior through the *ad hoc* tuning of their frequently large parameter space. In fact, biology also provides different ways to achieve similar neural dynamics, and self-regulation mechanisms rely on this. It is important to emphasize that the progress in computational neuroscience research will go hand in hand with the development of new experimental protocols to observe and stimulate neural activity, which will constrain the parameter space and validate model hypotheses.

Computational neuroscience is a relatively young discipline that has a lot of work ahead in the study of universal paradigms that neural systems use for coding, decoding, and to achieve robust and flexible information processing. Artificial intelligence will benefit from the work of computational neuroscience, whose results also contribute to bio-inspired robotics, and the design of novel brain-machine interfaces for medical and industrial applications.

References

- Abarbanel HDI, Huerta R, Rabinovich MI, Rulkov NF, Rowat P, Selverston AI (1996) Synchronized action of synaptically coupled chaotic model neurons. *Neural Comp* 8(8):1567–1602
- Abbott LF, Valera JA, Sen K, Nelson SB (1997) Synaptic depression and cortical gain control. *Science* 275(5297):220–224
- Amari S (1972) Characteristics of random nets of analog neuron-like elements. *IEEE Trans Syst Man Cybern* 2:643–657
- Amit D, Tsodyks M (1991) Quantitative study of attractor neural networks retrieving at low spike rates: II. low-rate retrieval in symmetric networks. *Network: Comput Neural Syst* 2:275–294
- Arshavsky YI, Beloozerova IN, Orlovsky GN, Panchin YV, Pavlova GA (1985) Control of locomotion in marine mollusc *Clione limacina*. i. efferent activity during actual and fictitious swimming. *Exp Brain Res* 58(2):255–293
- Arshavsky YI, Grillner S, Orlovsky GN, Panchin YV (1991) Central generators and the spatiotemporal pattern of movements. In: Fagard J, Wolff PH (eds) *The development of timing control*. Elsevier, Amsterdam, pp 93–115
- Ashby WR (1960) *Design for a brain*, 2nd edn. Wiley, New York
- Azouz R, Gray CM (2000) Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons *in vivo*. *Proc Natl Acad Sci USA* 97(14):8110–8115
- Barrie JM, Freeman WJ, Lenhart MD (1996) Spatiotemporal analysis of prepyriform, visual, auditory, and somesthetic surface EEGs in trained rabbits. *J Neurophysiol* 76(1):520–539
- Beenhakker MP, Nusbaum MP (2004) Mechanosensory activation of a motor circuit by coactivation of two projection neurons. *J Neurosci* 24(30):6741–6750
- Beggs JM, Plenz D (2003) Neuronal avalanches in neocortical circuits. *J Neurosci* 23(35):11167–11177
- Bertram R, Sherman A, Stanley EF (1996) Single-domain/bound calcium hypothesis of transmitter release and facilitation. *J Neurophysiol* 75(5):1919–1931
- Brette R et al (2007) Simulation of networks of spiking neurons: a review of tools and strategies. *J Comp Neurosci* 23(3):349–398
- Buia CI, Tiesinga PHE (2005) Rapid temporal modulation of synchrony in cortical interneuron networks with synaptic plasticity. *Computational*

- Neuroscience: Trends in Research 2005. *Neurocomputing* 6566:809–815
- Cazelles B, Courbage M, Rabinovich MI (2001) Anti-phase regularization of coupled chaotic maps modelling bursting neurons. *Europhys Lett* 56(4): 504–509
- Cortes JM, Torres JJ, Marro J, Garrido PL, Kappen HJ (2006) Effects of fast presynaptic noise in attractor neural networks. *Neural Comp* 18(3):614–633
- Cymbalyuk G, Gaudry O, Masino M, Calabrese R (2002) Bursting in leech heart interneurons: cell-autonomous and network-based mechanisms. *J Neurosci* 22(24):10580–10592
- Destexhe A, Marder E (2004) Plasticity in single neuron and circuit computations. *Nature* 431 (7010):789–795
- Destexhe A, Mainen ZF, Sejnowski TJ (1994) An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Comp* 6(1):14–18
- Dobrunz LE, Stevens CF (1997) Heterogeneity of release probability, facilitation, and depletion at central synapses. *Neuron* 18(6):995–1008
- Dyrhfjeld-Johnsen J, Santhakumar V, Morgan RJ, Huerta R, Tsimring L, Soltesz I (2007) Topological determinants of epileptogenesis in large-scale structural and functional models of the dentate gyrus derived from experimental data. *J Neurophysiol* 97(2):1566–1587
- Elson R, Selverston AI, Huerta R, Rabinovich MI, Abarbanel HDI (1998) Synchronous behavior of two coupled biological neurons 81(25):5692–5695
- Elson R, Huerta R, Abarbanel HDI, Rabinovich MI, Selverston AI (1999) Dynamic control of irregular bursting in an identified neuron of an oscillatory circuit. *Phys Rev Lett. J Neurophysiol* 82(1): 115–122
- FitzHugh R (1961) Impulses and physiological states in theoretical models of nerve membrane. *Biophys J* 1(6):445–466
- Franks KM, Stevens CF, Sejnowski TJ (2003) Independent sources of quantal variability at single glutamatergic synapses. *J Neurosci* 23(8):3186–3195
- Freeman W (1972) Progress in theoretical biology, vol 2. Academic, New York
- Fusi S, Abbott L (2007) Limits on the memory storage capacity of bounded synapses. *Nat Neurosci* 10(4): 485–493
- Garcia-Sanchez M, Huerta R (2003) Design parameters of the fan-out phase of sensory systems. *J Comp Neurosci* 15(1):5–17
- Gerstner W, Kistler W (2002) Spiking neuron models. Cambridge University Press, Cambridge
- Getting PA (1989) Emerging principles governing the operation of neural networks. *Ann Rev Neurosci* 12:185–204
- Gutkin BS, Ermentrout GB (1998) Dynamics of membrane excitability determine interspike interval variability: a link between spike generation mechanisms and cortical spike train statistics. *Neural Comp* 10(5):1047–1065
- Hebb DO (1949) The organization of behavior. Wiley, New York
- Hindmarsh JL, Rose RM (1984) A model of neuronal bursting using three coupled first order differential equations. *Proc R Soc Lond B* 221(1222):87–102
- Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol* 117(4):500–544
- Holcman D, Tsodyks M (2006) The emergence of up and down states in cortical networks. *PLoS Comput Biol* 2(3):174–181
- Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci USA* 79(8):2554–2558
- Huerta R, Varona P, Rabinovich MI, Abarbanel HDI (2001) Topology selection by chaotic neurons of a pyloric central pattern generator. *Biol Cybern* 84(1): L1–L8
- Izhikevich E (2004) Which model to use for cortical spiking neurons? *IEEE Trans Neural Netw* 15(5):1063–1070
- Johnson S, Marro J, Torres JJ (2008) Functional optimization in complex excitable networks. *EPL* 83:46006 (1–6)
- Kamiya H, Zucker RS (1994) Residual Ca^{2+} and short-term synaptic plasticity. *Nature* 371(6498): 603–606
- Kandel ER, Schwartz JH, Jessell TM (2000) Principles of neural science, 4th edn. McGraw-Hill, New York, pp 10–11
- Kepcs A, Lisman J (2003) Information encoding and computation with spikes and bursts. *Network: Comput Neural Syst* 14(1):103–118
- Koch C (1999) Biophysics of computation. Oxford University Press, New York
- Koch C, Segev I (1998) Methods in neuronal modeling: from ions to networks, 2nd edn. MIT Press, London, pp 6–17
- Korn H, Faure P (2003) Is there chaos in the brain? II. experimental evidence and related models. *C R Biol* 326(9):787–840
- Lapicque L (1907) Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J Physiol Pathol Gen* 9:620–635
- Latorre R, Rodriguez FB, Varona P (2006) Neural signatures: multiple coding in spiking bursting cells. *Biol Cybern* 95(2):169–183
- Laurent G, Stopfer M, Friedrich RW, Rabinovich MI, Volkovskii A, Abarbanel HDI (2001) Odor encoding as an active, dynamical process: experiments,

- computation, and theory. *Annu Rev Neurosci* 24:263–297
- Levi R, Camhi JM (2000) Population vector coding by the giant interneurons of the cockroach. *J Neurosci* 20(10):3822–3829
- Levi R, Varona P, Arshavsky YI, Rabinovich MI, Selverston AI (2004) Dual sensorymotor function for a molluscan statocyst network. *J Neurophysiol* 91(1):336–345
- Levi R, Varona P, Arshavsky YI, Rabinovich MI, Selverston AI (2005) The role of sensory network dynamics in generating a motor program. *J Neurosci* 25(42):9807–9815
- Lewis JE, Kristan WB (1998) Quantitative analysis of a directed behavior in the medicinal leech: implications for organizing motor output. *J Neurosci* 18(4):1571–1582
- Marder E, Bucher D (2007) Understanding circuit dynamics using the stomatogastric nervous system of lobsters and crabs. *Annu Rev Physiol* 69:291–316
- Marder E, Calabrese RL (1996) Principles of rhythmic motor pattern generation. *Physiol Rev* 76(3): 687–717
- Markram H, Wang Y, Tsodyks M (1998) Differential signaling via the same axon of neocortical pyramidal neurons. *Proc Natl Acad Sci USA* 95(9):5323–5328
- Matveev V, Wang XJ (2000) Differential short-term synaptic plasticity and transmission of complex spike trains: to depress or to facilitate? *Cerebral Cortex* 10 (11):1143–1153
- McAdams C, Maunsell J (1999) Effects of attention on orientation-tuning functions of single neurons in macaque cortical area V4. *J Neurosci* 19(1):431–441
- McCulloch WS, Pitts WH (1943) A logical calculus of ideas immanent in nervous activity. *Bull Math Biophys* 5:115–133
- McGraw PM, Menzinger M (2003) Topology and computational performance of attractor neural networks. *Phys Rev E* 68(4 Pt 2):047102
- Mejias JE, Torres JJ (2008) The role of synaptic facilitation in spike coincidence detection. *J Comp Neurosci* 24(2):222–234
- Mejias JE, Torres JJ (2009) Maximum memory capacity on neural networks with short-term synaptic depression and facilitation. *Neural Comput* 21(3): 851–871
- Morris C, Lecar H (1981) Voltage oscillations in the barnacle giant muscle fiber. *Biophys J* 35(1):193–213
- Nagumo J, Arimoto S, Yoshizawa S (1962) An active pulse transmission line simulating nerve axon. *Proc IRE* 50:2061–2070
- Nowotny T, Huerta R (2003) Explaining synchrony in feed-forward networks: are McCulloch-Pitts neurons good enough? *Biol Cybern* 89(4):237–241
- Panchin YV, Arshavsky YI, Deliagina TG, Popova LB, Orlovsky GN (1995) Control of locomotion in marine mollusk *Clione limacina*. ix. neuronal mechanisms of spatial orientation. *J Neurophysiol* 75(5):1924–1936
- Pantic L, Torres JJ, Kappen HJ, Gielen SCA (2002) Associative memory with dynamic synapses. *Neural Comp* 14(12):2903–2923
- Peretto P (1992) An introduction to the modeling of neural networks. Cambridge University Press, Cambridge
- Pinault D, Smith Y, Deschnes M (1997) Dendrodendritic and axoaxonic synapses in the thalamic reticular nucleus of the adult rat. *J Neurosci* 17(9):3215–3233
- Pinto RD, Varona P, Volkovskii AR, Szucs A, Abarbanel HD, Rabinovich MI (2000) Synchronous behavior of two coupled electronic neurons. *Phys Rev E* 62(2 Pt B):2644–2656
- Prinz A (2006) Insights from models of rhythmic motor systems. *Curr Opin Neurobiol* 16(6):615–620
- Rabinovich MI, Varona P, Selverston AI, Abarbanel HDI (2006) Dynamical principles in neuroscience. *Rev Mod Phys* 78:1213–1265
- Rabinovich MI, Huerta R, Varona P, Afraimovich VS (2008) Transient cognitive dynamics, metastability, and decision making. *PLoS Comput Biol* 4(5): e1000072
- Ramirez J, Tryba A, Pena F (2004) Pacemaker neurons and neuronal networks: an integrative view. *Curr Opin Neurobiol* 6(6):665–674
- Reyes MB, Huerta R, Rabinovich MI, Selverston AI (2003) Artificial synaptic modification reveals a dynamical invariant in the pyloric CPG. *Eur J Appl Physiol* 102(6):667–675
- Rosenblatt F (1962) Principles of neurodynamics: perceptions and the theory of brain mechanisms. Spartan Books, New York
- Roxin A, Riecke H, Solla SA (2004) Self-sustained activity in a small-world network of excitable neurons. *Phys Rev Lett* 92(19):198101
- Rulkov NF (2001) Regularization of synchronized chaotic bursts. *Phys Rev Lett* 86(1):183–186
- Rulkov NF (2002) Modeling of spiking-bursting neural behavior using two-dimensional map. *Phys Rev E* 65(4 Pt 1):041922
- Selverston AI (2005) A neural infrastructure for rhythmic motor patterns. *Cell Mol Neurobiol* 25(2): 223–244
- Selverston AI, Moulins M (eds) (1987) The crustacean stomatogastric system. Springer, Berlin
- Selverston AI, Rabinovich MI, Abarbanel HDI, Elson R, Szcs A, Pinto RD, Huerta R, Varona P (2000) Reliable circuits from irregular neurons: a dynamical approach to understanding central pattern generators. *J Physiol (Paris)* 94(5–6):357–374
- Simmers AJ, Moulins M (1988) A disynaptic sensorimotor pathway in the lobster stomatogastric system. *J Neurophysiol* 59(3):740–756

- Stein SG, Grillner S, Selverston AI, Douglas GS (eds) (1997) *Neurons, networks, and motor behavior*. MIT Press, Cambridge
- Stiesberg GR, Reyes MB, Varona P, Pinto RD, Huerta R (2007) Connection topology selection in central pattern generators by maximizing the gain of information. *Neural Comp* 19(4):974–993
- Swiercz W, Cios KJ, Staley K, Kurgan L, Accurso F, Sagel S (2006) A new synaptic plasticity rule for networks of spiking neurons. *IEEE Trans Neural Netw* 17(1): 94–105
- Szucs A, Varona P, Volkovskii AR, Abarbanel HD, Rabinovich MI, Selverston AI (2000) Interacting biological and electronic neurons generate realistic oscillatory rhythms. *Neuroreport* 11(11):563–569
- Szucs A, Pinto RD, Rabinovich MI, Abarbanel HD, Selverston AI (2003) Synaptic modulation of the interspike interval signatures of bursting pyloric neurons. *J Neurophysiol* 89(3):1363–1377
- Torres JJ, Garrido PL, Marro J (1997) Neural networks with fast time-variation of synapses. *J Phys A: Math Gen* 30:7801–7816
- Torres JJ, Munoz MA, Marro J, Garrido PL (2004) Influence of topology on the performance of a neural network. *Computational Neuroscience: Trends in Research.* 58–60:229–234
- Torres JJ, Cortes JM, Marro J (2007a) Information processing with unstable memories. *Neurocomputing AIP Conf Proc* 887:115–128
- Torres JJ, Cortes JM, Marro J, Kappen HJ (2007b) Competition between synaptic depression and facilitation in attractor neural networks. *Neural Comp* 19(10):2739–2755
- Tsodyks M, Feigelman M (1988) The enhanced storage capacity in neural networks with low activity level. *Europhys Lett* 6:101–105
- Tsodyks MV, Markram H (1997) The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc Natl Acad Sci USA* 94(2):719–723
- Tsodyks MV, Pawelzik K, Markram H (1998) Neural networks with dynamic synapses. *Neural Comp* 10(4):821–835
- Varona P, Ibarz JM, López L, Herreras O (2000) Macroscopic and subcellular factors shaping population spikes. *J Neurophysiol* 83(4):2192–2208
- Varona P, Torres JJ, Huerta R, Abarbanel HDI, Rabinovich MI (2001a) Regularization mechanisms of spiking-bursting neurons. *Neural Netw* 14(6–7): 865–875
- Varona P, Torres JJ, Abarbanel HDI, Rabinovich MI, Elson RC (2001b) Dynamics of two electrically coupled chaotic neurons: experimental observations and model analysis. *Biol Cybern* 84(2): 91–101
- Varona P, Rabinovich MI, Selverston AI, Arshavsky YI (2002) Winnerless competition between sensory neurons generates chaos: a possible mechanism for molluscan hunting behavior. *Chaos* 12(3): 672–677
- von der Malsburg C (1981) The correlation theory of brain function. MPI Biophysical Chemistry, Internal Report
- Zeck GM, Masland RH (2007) Spike train signatures of retinal ganglion cell types. *Eur J Neurosci* 26(2): 367–380

18 Neural Networks in Bioinformatics

Ke Chen¹ · Lukasz A. Kurgan²

¹Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada

kchen1@ece.ualberta.ca

²Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada

lkurgan@ece.ualberta.ca

1	<i>Introduction</i>	566
2	<i>Biological Background</i>	567
3	<i>Neural Network Architectures in Protein Bioinformatics</i>	569
4	<i>Applications of Neural Networks in Protein Bioinformatics</i>	573
5	<i>Summary</i>	579

Abstract

Over the last two decades, neural networks (NNs) gradually became one of the indispensable tools in bioinformatics. This was fueled by the development and rapid growth of numerous biological databases that store data concerning DNA and RNA sequences, protein sequences and structures, and other macromolecular structures. The size and complexity of these data require the use of advanced computational tools. Computational analysis of these databases aims at exposing hidden information that provides insights which help with understanding the underlying biological principles. The most commonly explored capability of neural networks that is exploited in the context of bioinformatics is prediction. This is due to the existence of a large body of raw data and the availability of a limited amount of data that are annotated and can be used to derive the prediction model. In this chapter we discuss and summarize applications of neural networks in bioinformatics, with a particular focus on applications in protein bioinformatics. We summarize the most often used neural network architectures, and discuss several specific applications including prediction of protein secondary structure, solvent accessibility, and binding residues.

1 Introduction

The term “bioinformatics” was coined relatively recently, that is, it did not appear in the literature until 1991 (Boguski 1998). However, the first studies that concerned the field of bioinformatics appeared already in the 1960s when the first protein and nucleic acid sequence database was established. The National Institutes of Health (NIH) defines bioinformatics as “research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral, or health data, including those to acquire, store, organize, archive, analyze, or visualize such data” (NIH Working Definition of Bioinformatics and Computational Biology 2000). We note that bioinformatics is usually constrained to molecular genetics and genomics. In a review by Luscombe et al. (2001), this term is defined as “conceptualizing biology in terms of macromolecules (in the sense of physical chemistry) and then applying ‘informatics’ techniques (derived from disciplines such as applied math, computer science, and statistics) to understand and organize the information associated with these molecules, on a large-scale.” The key observations concerning the above definition are that bioinformatics research is interdisciplinary, that is, it requires knowledge of physics, biochemistry, and informatics, and that it concerns large-scale analysis, that is, only scalable computational methods can be used. Since bioinformatics spans a wide variety of research areas, that is, sequence analysis, genome annotation, evolutionary biology, etc., we are not able to discuss all these research topics. Instead, we concentrate on the approaches concerning protein bioinformatics, that is, the scope of this chapter is limited to the application of bioinformatics in protein-related topics.

The last two decades observed an increased interest in the application of machine learning techniques, and particularly artificial neural networks (NNs), in protein bioinformatics. The most common application of the NNs is prediction; we assume that prediction concerns targets that are both discrete and real valued. The popularity of NNs stems from two key advantages that distinguish them from many other machine-learning methods. First, after the NN model is trained, the use of the model to perform prediction is very efficient, that is, computations are fast. This allows for a high throughput prediction of massive amounts of

data, which is an inherent feature of a significant majority of bioinformatics projects. Second, NN-based models provide high-quality results for many prediction tasks, for example, the leading methods in protein secondary structure prediction and protein solvent accessibility prediction are based on NNs. These successful applications raised the profile of NNs, which are currently being applied in dozens of other prediction tasks.

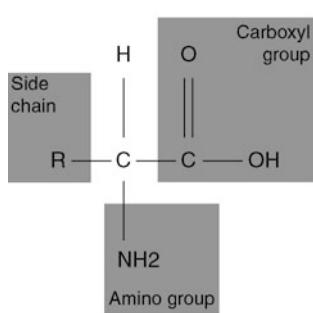
First, we introduce the relevant biological background. Next, we summarize the most popular NN architectures that are applied in protein bioinformatics and the key prediction methods that utilize NNs. Finally, we provide a more detailed analysis of NN-based solutions for the prediction of protein secondary structure, solvent accessibility, and binding residues.

2 Biological Background

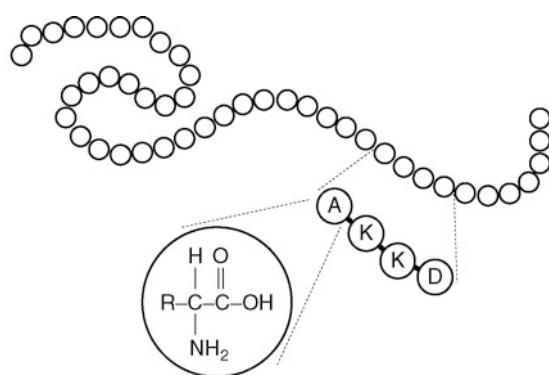
Proteins are essential elements of virtually all living organisms. They participate in every process within cells. For instance, some proteins serve as enzymes that catalyze biochemical reactions which are vital to metabolism. Proteins are also important in cell signaling, immune responses, cell adhesion, and the cell cycle, to name just a few of their functions. They are large polymeric organic molecules which are composed of amino acids (also called residues). Amino acid (AA) is a small molecule that includes an amino ($-\text{NH}_2$) (except the proline amino acid) and a carboxyl ($-\text{COOH}$) group that are linked to a carbon atom. The AA formula, $\text{NH}_2\text{CHRCOOH}$, where N, H, C, and O are the nitrogen, hydrogen, carbon, and oxygen atoms, respectively, also incorporates R which denotes an organic substituent (so-called side chain), see Fig. 1 (Panel A). There are a total of 20 AAs that make up all proteins. They all share the same NH_2CHCOOH group and have different R-group. The side chains determine physiochemical properties, such as charge, weight, and hydrophobicity, of

Fig. 1

Panel A shows the chemical structure of AAs; the side chain (R-group) differentiates the structure of different AAs. Panel B shows a protein chain (linear sequence) composed of AAs where each circle represents one AA.



Panel A



Panel B

individual AAs. AAs are abbreviated using either three-letter or one-letter encoding. In the one-letter case, the 20 AAs are encoded as A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y. The AAs are joined together in a linear form through chemical bonds between the carboxyl group and the amino groups of two adjacent AAs, see [Fig. 1](#) (Panel B).

The protein structure is defined at four levels, which include the primary, secondary, tertiary, and quaternary structures, see [Fig. 2](#):

- *Primary structure* is the linear order of AAs, also called amino acid sequence. The AA sequence is translated from genes (DNA).
 - *Secondary structure* is defined as regular and repetitive spatially local structural patterns in the protein structure. The secondary structure is stabilized by hydrogen bonds. The most common secondary structures are helix, strand, and coil. Secondary structures are present in different regions of the same and different protein molecules.

Fig. 2

Examples of different levels of protein structure. Panels A, B, and C show the structure of a globular domain of the human prion protein. Panel A shows the AA sequence (using one-letter encoding) and the corresponding secondary structure for each AA (using encoding in which H, E, and C stand for helix, strand, and coil, respectively) and also a graphical format where the horizontal line denotes the coil, waves denotes helices, and arrows denotes strands. Panel B shows a cartoon representation of the spatial arrangement of the secondary structures. Helices are shown in red, strands in yellow, and coils as black lines. Panel C shows the tertiary structure of the protein in which individual atoms are represented using spheres. Panel D shows the quaternary structure of a microtubule, which is assembled from α -tubulin (dark gray spheres) and β -tubulin (light gray spheres) proteins. The tubulin is assembled into a hollow cylindrical shape.

LGGYMLGSAMSRPIIHFGSDYEDRYYRENMRHYPNQVYRCPDEYSQNQNFVHDVCVNITI (AA sequence)
CCCCCECCCCCCCCCCCCCHHHHHHHHHHHCCCCCECCCCCCCCCCCCCHHHHHHHHHHH (Sec. structure)



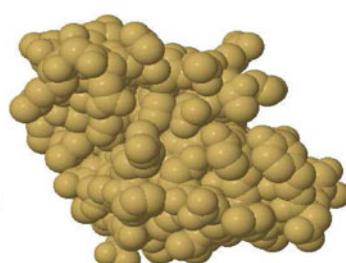
KQHTVTTTKGENTFTEDVKMMERVVEQMCITQYERCSQAYYQR
HHHHHHHHHHHHCCCCCHHHHHHHHHHHHHHHHHHHHHHHHHHHCCCC

(Continued AA sequence)
(Continued sec. structure)

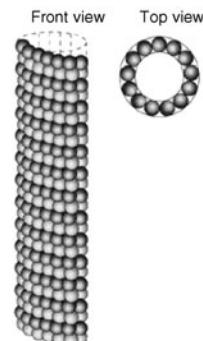
Panel A



Panel B



Panel C



Panel D

- *Tertiary structure* defines the overall three-dimensional shape of a single protein molecule. It concerns the spatial arrangement of the secondary structures and is represented by the coordinates of all atoms in the protein. It is generally believed that the tertiary structure of a given protein is defined by its primary sequence and that each protein has a unique tertiary structure.
- *Quaternary structure* is the arrangement of multiple protein structures in a multi-subunit complex. The individual proteins are assembled into a larger molecule usually with a given geometrical shape, for example, protofilament or a spherical shape. For instance, a microtubule is the assembly of α -tubulin and β -tubulin proteins which takes the form of a hollow cylindrical filament.

While as of January 2009 the primary structure is known for over 6.4 million nonredundant proteins, the corresponding structure is known for only about 55,000 proteins. We emphasize that knowledge of the structure is of pivotal importance for learning and manipulating a protein's function, which for instance is exploited in modern drug design. The significant and widening gap between the set of known protein sequences and protein structures motivates the development of machine learning models that use the known structures to predict structures for the unsolved sequences.

3 Neural Network Architectures in Protein Bioinformatics

Although more than a dozen NN architectures have been developed and adopted, one of the first and simplest architectures, the feedforward neural network (FNN), is the most frequently applied in protein bioinformatics. Besides FNN, the recurrent neural network (RNN) and the radial basis function neural network (RBF) architectures also found several applications in the prediction of bioinformatics data.

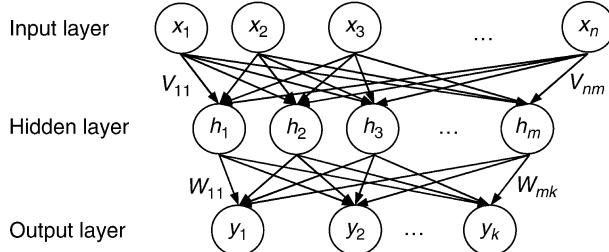
A common feature of all prediction applications in protein bioinformatics is the necessity to convert the input (biological) data into the data that can be processed by the NN. This usually involves encoding of the biological data into a fixed-size feature vector. For instance, the primary protein structure is represented as a variable length string of characters with an alphabet of 20 letters (AAs), see [Fig. 2](#) (Panel A). This sequence is converted into a vector of numerical features that constitutes the input to the NN. For instance, the vector could include 20 counts of the occurrence of the 20 amino acids in the sequence. The following discussion assumes that the input data are already encoded into the feature vector.

3.1 Feedforward Neural Networks

The FNN architecture usually consists of three layers, an input layer, a hidden layer, and an output layer. The input layer accepts the input feature vector and the output layer generates the predictions. The hidden layer is responsible for capturing the prediction model. Each layer consists of a number of nodes and each node in a given layer connects with every other node in the following layer, see [Fig. 3](#). The connections are associated with weights v_{ij} and w_{ij} between the i th node in one layer and the j th node in the next layer. The nodes process the input values, which are computed as the weighted sum of values passed from the previous layer, using activation functions. The two most frequently used activation functions are:

Fig. 3

Architecture of FNN. The input layer contains n nodes (which equals the number of features in the input feature vector), the hidden layer contains m nodes and the output layer contains k nodes. The weight between the i th node of the input layer and the j th node of the hidden layer is denoted by v_{ij} . The weight between the i th node of the hidden layer and the j th node of the output layer is denoted by w_{ij} .



$$\begin{aligned}\phi(v_i) &= \tanh(v_i) \\ \phi(v_i) &= (1 + e^{-v_i})^{-1}\end{aligned}$$

where v_i is the weighted sum of the inputs. The values of the former hyperbolic tangent function range between -1 and 1 , while the values of the latter logistic function range between 0 and 1 . Some applications also utilize radial basis activation functions.

Learning using the FNN-based prediction model is performed by adjusting the connection weight values to minimize the prediction error on training data. For a given input feature vector $\{x_i\}$, the observations (the prediction outcomes) are denoted as $\{y_i\}$. The goal of the FNN is to find a function $f: X \rightarrow Y$, which describes the relation between inputs X and observations Y . The merit of function f is measured with a cost function $C = E[(f(x_i) - y_i)^2]$. For a training dataset with n samples, the cost function is

$$C = \sum_{i=1}^n \frac{[f(x_i) - y_i]^2}{n}$$

Based on the amount of error associated with the outputs of the network in comparison with the expected result (cost function), the adjustment of the connection weights is carried out using a backpropagation algorithm. The n input feature vectors are fed multiple times (each presentation of the entire training dataset is called an epoch) until the weight values do not change or a desired value of the cost function is obtained.

FNN is the most widely applied among the NN architectures in protein bioinformatics. The applications include:

- Prediction of the secondary structure of protein (Jones 1999; Rost et al. 1994; Dor and Zhou 2007a; Hung and Samudrala 2003; Petersen et al. 2000; Qian and Sejnowski 1988). The aim of these methods is to predict the secondary structure state (helix, strand, or coil) for every AA in the input protein sequence.
- Prediction of relatively solvent accessibility of protein residues (Rost and Sander 1994; Garg et al. 2005; Adamczak et al. 2005; Ahmad et al. 2003; Dor and Zhou 2007b; Pollastri

et al. 2002a). The solvent accessibility is defined as a fraction of a surface area of a given AA that is accessible to the solvent. The AAs with high solvent accessibility are usually on the protein surface.

- Prediction of binding residues (Jeong et al. 2004; Ahmad and Sarai 2005; Zhou and Shan 2001; Ofran and Rost 2007). The binding residues are those AAs in a given protein that are involved in interactions with another molecule. The interactions could concern other proteins, DNA, RNA, ions, etc., and they usually implement protein functions.
- Prediction of transmembrane regions (Gromiha et al. 2005; Natt et al. 2004; Jacoboni et al. 2001). Some proteins are embedded into cell membranes and they serve as pumps, channels, receptors, and energy transducers for the cell. The goal of this prediction method is to find which AAs in the input protein sequence are embedded into the membrane.
- Prediction of subcellular location of proteins (Zou et al. 2007; Cai et al. 2002; Reinhardt and Hubbard 1998; Emanuelsson et al. 2000). These methods predict the location of the proteins inside a cell. The locations include cytoplasm, cytoskeleton, endoplasmic reticulum, Golgi apparatus, mitochondrion, nucleus, etc.

3.2 Recurrent Neural Networks

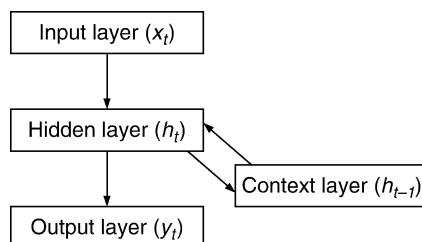
A recurrent neural network (RNN) is a modification to the FNN architecture. In this case, a “context” layer is added, and this layer retains information across observations. In each iteration, a new feature vector is fed into the input layer. The previous contents of the hidden layer are copied to the context layer and then fed back into the hidden layer in the next iteration, see Fig. 4.

When an input feature vector is fed into the input layer, the RNN processes are as follows:

1. Copy the input vector values into the input nodes.
2. Compute hidden node activations using net input from input nodes and from the nodes in the context layer.
3. Compute output node activations.
4. Compute the new weight values using the backpropagation algorithm.
5. Copy new hidden node weights to the context layer.

Fig. 4

Architecture of RNN. Like FNN, RNN also contains an input layer, a hidden layer, and an output layer. An additional context layer is connected to the hidden layer.



Since the trainable weights, that is, weights between the input and hidden layers and between the hidden and output layers, are feedforward only, the standard backpropagation algorithm is applied to learn the weight values. The weights between the context and the hidden layers play a special role in the computation of the cost function. The error values they receive come from the hidden nodes and so they depend on the error at the hidden nodes at the t th iteration. During the training of the RNN model we consider a gradient of a cost (error) function which is determined by the activations at both the present and the previous iterations.

The RNN architecture was successfully applied in the prediction of:

- Beta-turns (Kirschner and Frishman 2008). Beta turns are the most frequent subtypes of coils, which are one of the secondary protein structures.
- Secondary structure of proteins (Chen and Chaudhari 2007).
- Continuous B-cell epitopes (Saha and Raghava 2006). B-cell epitopes are the antigenic regions of proteins recognized by the binding sites of immunoglobulin molecules. They play an important role in the development of synthetic vaccines and in disease diagnosis. The goal of this prediction method is to find AAs that correspond to the epitopes.
- Number of residue contacts (Pollastri et al. 2002b), which is defined as the number of contacts a given AA makes in the three-dimensional protein molecule. The knowledge of the contacts helps in learning the tertiary protein structure.

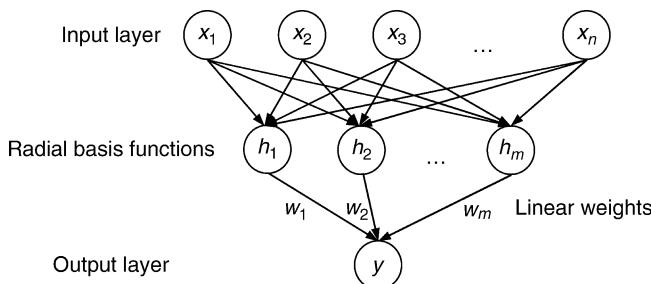
3.3 Radial Basis Function Neural Networks

Radial basis function (RBF) neural networks also incorporate three layers: an input layer, a hidden layer with a nonlinear RBF activation function, and a linear output layer, see  Fig. 5. During the process of training the RBF model:

1. The input vectors are mapped onto each RBF in the hidden layer. The RBF is usually implemented as a Gaussian function. The Gaussian functions are parameterized, that is, values of the center and spread are established, using the training dataset. The commonly

 **Fig. 5**

Architecture of RBF neural network. The network is fully connected between the input and the hidden layers (each node in the input layer is connected with all nodes in the hidden layer), and all the weights are usually assumed to be equal to 1. The nodes in the hidden layer are fully connected with a single node in the output layer, and the weight values are optimized to minimize the cost function.



used methods include K-means clustering or, alternatively, a random subset of the training vectors can be used as the centers.

2. In regression problems (the prediction outcomes are real values), the output layer is a linear combination of values produced by the hidden layer, which corresponds to the mean predicted output. In prediction problems (the prediction outcomes are discrete), the output layer is usually implemented using a sigmoid function of a linear combination of hidden layer values, representing a posterior probability.

RBF networks are faster to train when compared with FNN and RNN. They also have an advantage of not suffering from local minima in the same way as FNN, that is, the FNN may not be able to find globally best solution, but it may get stuck in a local minimum of the cost function. This is because the only parameters that are adjusted in the learning process of the RBF network are associated with the linear mapping from the hidden layer to the output layer. The linearity ensures that the error surface is quadratic and therefore it has a single, usually relatively easy to find, minimum. At the same time, the quality of the prediction is usually higher when using a properly designed and trained FNN.

RBF networks were utilized in several applications that include:

- Prediction of inter-residue contact maps (Zhang and Huang 2004). The contact maps include binary entries that define whether a given AA is or is not in contact with any other AA in the tertiary structure. The knowledge of contacts helps in the reconstruction of the tertiary protein structure.
- Prediction of protease cleavage sites (Yang and Thomson 2005). Protease cleavage is performed by enzymes, which are proteins that catalyze biological reactions. Knowledge of how a given protease cleaves the proteins is important for designing effective inhibitors to treat some diseases. This prediction method aims at finding AAs in the protein sequence that are involved in this interaction.
- Prediction of targets for protein-targeting compounds (Niwa 2004). This method aims at the prediction of biological targets (proteins) that interact with given chemical compounds. This has applications in drug design where large libraries of chemical compounds are screened to find compounds that interact with a given protein and which, as a result, modify (say, inhibit) the protein's function.

One of the important parameters in the design of any of the three above mentioned architectures, that is, FNN, RNN, and RBFNN, is the number of nodes. The number of input nodes usually equals the number of input features. Most commonly, there is only one output node that corresponds to the predicted outcome, although in some cases NNs are used to generate multiple outcomes simultaneously, that is, prediction of the protein secondary structure requires three outcomes. The number of nodes in the hidden layer is chosen by the designer of the network. This number depends on the application and the desired quality of the prediction.

4 Applications of Neural Networks in Protein Bioinformatics

NNs are used in a variety of protein bioinformatics applications. They can be categorized into:

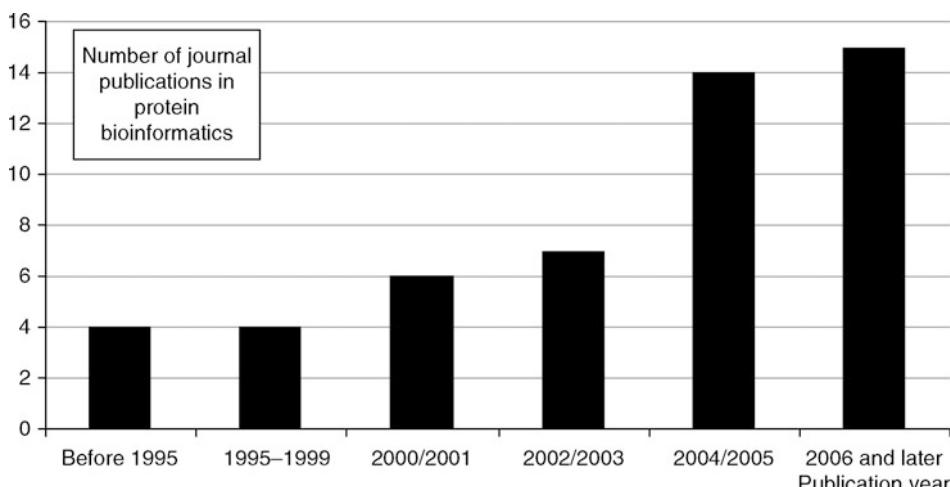
- Prediction of protein structure including secondary structure and secondary structure content, contact maps, structural contacts, boundaries of structural domains, specific types of local structures like beta-turns, etc.

- Prediction of binding sites and ligands, which includes prediction of binding residues and prediction of various properties of the binding ligands.
- Prediction of protein properties such as physicochemical proteins, localization in the host organism, etc.

Specific applications include prediction of a number of residue contacts (Pollastri et al. 2002), protein contact maps (Zhang and Huang 2004), helix-helix (Fuchs et al. 2009) and disulfide contacts (Martelli et al. 2004), beta and gamma turns (Kirschner and Frishman 2008; Kaur and Raghava 2003, 2004), secondary structure (Jones 1999; Rost et al. 1994; Dor and Zhou 2007a; Hung and Samudrala 2003; Petersen et al. 2000; Qian and Sejnowski 1988; Chen and Chaudhari 2007), domain boundaries (Ye et al. 2008), transmembrane regions (Gromiha et al. 2005; Natt et al. 2004; Jacoboni et al. 2001), binding sites and functional sites (Jeong et al. 2004; Ahmad and Sarai 2005; Zhou and Shan 2001; Ofran and Rost 2007; Yang and Thomson 2005; Lin et al. 2005; Lundsgaard et al. 2008; Blom et al. 1996; Ingrell et al. 2007), residue solvent accessibility (Rost and Sander 1994; Garg et al. 2005; Adamczak et al. 2005; Ahmad et al. 2003; Dor and Zhou 2007b; Pollastri et al. 2002), subcellular location (Zou et al. 2007; Cai et al. 2002; Reinhardt and Hubbard 1998; Emanuelsen et al. 2000), secondary structure content (Muskal and Kim 1992; Cai et al. 2003; Ruan et al. 2005), backbone torsion angles (Xue et al. 2008; Kuang et al. 2004), protein structural class (Chandonia and Karplus 1995; Cai and Zhou 2000), signal peptides (Plewczynski et al. 2008; Sidhu and Yang 2006), continuous B-cell epitopes (Saha and Raghava 2006), binding affinities, toxicity, and pharmacokinetic parameters of organic compounds (Vedani and Dobler 2000), biological targets of chemical compounds (Niwa 2004), and prediction of spectral properties of green fluorescent proteins (Nantaseenamat et al. 2007). We observe a growing interest in applying NNs in this domain, see ➤ Fig. 6. The NN-based applications in protein bioinformatics were published in a number of

Fig. 6

Number of journal publications (y-axis) concerning the applications of NNs in protein bioinformatics in the last two decades. The included publications do not constitute an exhaustive list of corresponding studies, but rather they provide a set of the most significant and representative developments.



high-impact scientific journals such as (in alphabetical order) Bioinformatics; BMC Bioinformatics; Gene; IEEE/ACM Transactions on Computational Biology and Bioinformatics; Journal of Computational Chemistry; Journal of Computer-Aided Molecular Design; Journal of Medicinal Chemistry; Journal of Molecular Biology; Nucleic Acids Research; PLoS Computational Biology; Protein Science; Proteins; and Proteomics. The number, scope, and quality of the above venues strongly indicate the important role of this research.

The prediction of the secondary structure, residue solvent accessibility, and binding sites attracted the most attention in the context of the NN-based solutions. Therefore, the following sections concentrate on these three topics.

4.1 Prediction of Protein Secondary Structure with Neural Networks

Protein secondary structure is defined as a regular and repetitive spatially local structural pattern in protein structures. Several methods are used to define the protein secondary structure from a protein's three-dimensional structure. The most commonly used method is the Dictionary of Protein Secondary Structure (DSSP) (Kabsch and Sander 1983), which assigns eight types of secondary structures based on hydrogen-bonding patterns. These types include 3/10 helix, alpha helix, pi helix, extended strand in parallel and/or antiparallel β -sheet conformation, isolated β -bridge, hydrogen bonded turn, bend, and coil. The eight-state secondary structure is often aggregated into a three-state secondary structure. The first three types are combined into the helix state, the following two types into the strand state, and the last three types into the coil state. Most of the existing computational methods predict the three-state secondary structure instead of the eight-state structure. The main goal of these methods is to obtain the secondary structure using only AA sequences of the protein as the input.  [Figure 2](#) shows the AA sequence, the corresponding secondary structure for each AA, the spatial arrangement of the secondary structure, and the overall three-dimensional structure of the human prion protein. This protein is associated with several prion diseases such as fatal familial insomnia and Creutzfeldt–Jakob disease.

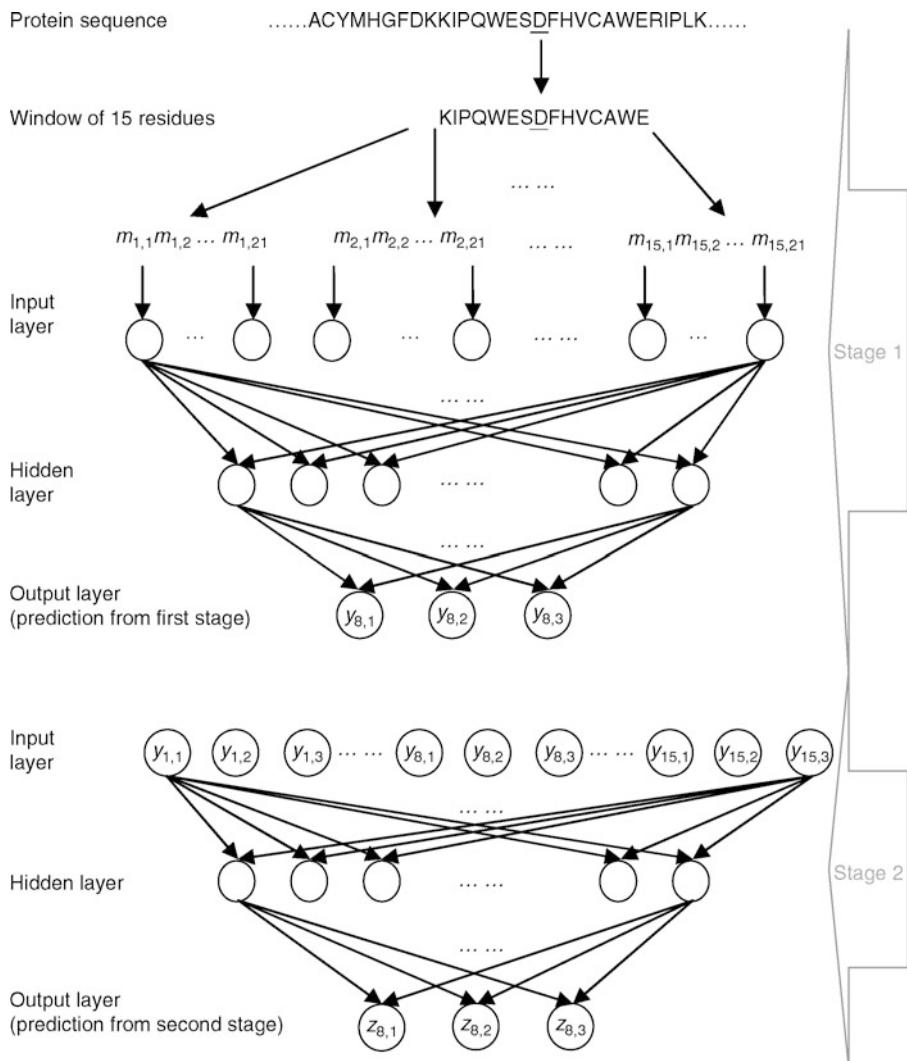
The first study concerning the prediction of protein secondary structure using an NN appeared in 1988 (Qian and Sejnowski 1988). This model is a typical three-layer FNN in which the input layer contains $13 \times 21 = 273$ nodes representing a stretch of 13 continuous AAs in the sequence, and the output layer contains three nodes representing the three secondary structure states. Each AA in the sequence is encoded using 21 binary features indicating the type of the AA at a given position in the sequence. This early method was trained using a very small dataset of 106 protein sequences, which limited its quality.

One of the most successful and commonly used models for the prediction of protein secondary structure, named PSIPRED, was proposed by Jones in 1999 (Jones 1999). It is a two-stage NN that takes a position-specific scoring matrix (PSSM), which is generated from the protein sequence using the PSI-BLAST (Position Specific Iterated Basic Local Alignment Search Tool) algorithm (Altschul et al. 1997) as the input. The architecture of PSIPRED is summarized in  [Fig. 7](#).

In the first stage, the input protein sequence is represented by the PSSM using a window of size 15 which is centered over the predicted AA. PSSM includes 20 dimensions for each AA, which correspond to substitution scores for each of the 20 AAs. The scores quantify which AAs are likely to be present/absent at a given position in the sequence in a set of known sequences that are similar to the sequence being predicted. This is based on the assumption that

Fig. 7

Architecture of PSIPRED algorithm. The algorithm is two-stage and includes two 3-layer FNNs, where the output of the first stage network feeds into the input to the second stage network. In the first stage, a window of 15 positions over the PSSM profile generated by the PSI-BLAST program from the input protein sequence is used. Each position in the input is represented by a vector of 21 values (the i th AA in the window is represented as $m_{i,1}m_{i,2} \dots m_{i,21}$). The 21×15 values are fed into the input layer. The output layer in the first stage NN contains three nodes that represent the probabilities of forming helix, strand, and coil structures (the predicted probabilities for the central AA in the window are represented as $y_{8,1}$, $y_{8,2}$, and $y_{8,3}$). These probabilities, using a window of 15 positions, are fed into the second-stage NN. The output from the second-stage NN is the final prediction that represents the probabilities of three types of the secondary structure: $z_{8,1}$, $z_{8,2}$, and $z_{8,3}$.



similarity in the sequence often implies similarity in the structure. The positive scores indicate that a given AA substitution occurs more frequently than expected by chance, while negative scores indicate that the substitution occurs less frequently than expected. The 20 scores from the PSSM together with a single feature that indicates the terminus of the sequence are fed into the input layer of the first-stage NN. As a result, the input layer contains $15 \times 21 = 315$ nodes. The hidden layer contains 75 nodes and the output layer contains three nodes which indicate the probabilities of the three secondary structure states.

In the second stage, the predicted probabilities of the secondary structures from the first stage for a window of 15 AAs centered over the position being predicted are fed into the input layer. The second layer exploits the fact that secondary structures form segments in the protein sequence, see  Fig. 2a, and thus information about the structure of the AAs in the adjacent positions in the sequence is helpful to determine the structure of a given AA. The input layer contains $4 \times 15 = 60$ nodes (the value indicating the terminus of the sequence is also included), the hidden layer contains 60 nodes, and the output layer contains three nodes. The PSIPRED method can be accessed, as a web server, at <http://bioinf.cs.ucl.ac.uk/psipred/>. Interested users can also download a stand-alone version of this popular prediction method.

One of the recently proposed NN-based methods performs the prediction of the secondary structure using a cascaded bidirectional recurrent neural network (BRNN) (Chen and Chaudhari 2007). Similar to the PSIPRED design, the first BRNN (sequence-to-structure BRNN) predicts the secondary structure based on the input AA sequences. The second BRNN (structure-to-structure BRNN) refines the raw predictions from the first BRNN. The learning algorithm used to develop this method is based on the backpropagation.

The last two decades observed the development of several methods based on NN for the prediction of protein secondary structure. The performance of the methods mainly depends on the representation of the protein sequence and the size of the training dataset. Since the beta-sheets (strands adjacent in the tertiary structure) are established between AAs that are far away in the sequence, the window-based methods (including all present methods for the prediction of protein secondary structure) are inherently incapable of grasping the long-range interactions, which results in a relatively poor result for strands.

4.2 Prediction of Binding Sites with Neural Networks

A protein performs its function through interactions with other molecules, called ligands, which include another protein, DNA, RNA, small organic compounds, or metal ions. Knowledge of the binding sites, which are defined as the AAs that directly interact with the other molecules, is crucial to understand the protein's function. More specifically, an AA is a part of the binding site if the distance from at least one atom of this AA to any atom of the ligand is less than a cutoff threshold D . The values of D vary in different studies and they usually range between 3.5 and 6 Å (Zhou and Shan 2001; Ofran and Rost 2007; Ahmad et al. 2004; Kuznetsov et al. 2006).

In one of the recent works by Jeong and colleagues, the FNN architecture is used for the prediction of RNA-binding sites (Jeong et al. 2004). Each AA in the input protein sequence is encoded by a vector of 24 values, of which 20 values indicate the AA type (using binary encoding), one value represents the terminus of the sequence, and the remaining three values correspond to the probabilities of three types of the secondary structure predicted using the PHD program. Using a window of size of 41 residues, the corresponding design includes

$24 \times 41 = 984$ input nodes. The hidden layer includes 30 nodes and the output layer consists of a single node that provides the prediction.

In another recent study by Ahmad and Sarai, a three-layer FNN is utilized for the prediction of DNA-binding sites (Ahmad and Sarai 2005). The architecture of this model is relatively simple, that is, 100 nodes in the input layer, 2 nodes in the hidden layer, and 1 node in the output layer. The input layer receives values from the PSSM computed over the input protein sequence with the window size of 5, which results in 100 features per AA.

Prediction of protein–protein interaction sites uses designs that are similar to the designs utilized in the prediction of DNA/RNA-binding sites. Zhou and Shan proposed a three-layer FNN to predict the protein–protein interaction sites (Zhou and Shan 2001). In their design, the PSSM and solvent-accessible area generated by the DSSP program (Kabsch and Sander 1983) for the predicted AAs and the 19 spatially nearest neighboring surface AAs make up the input. As a result, the input layer contains $21 \times 20 = 420$ nodes. The hidden layer includes 75 nodes. This method predicts the protein–protein interaction sites from the protein’s three-dimensional structure, since this information is necessary to compute the relative solvent accessibility values and to find the 19 spatially nearest AAs. In a recent study by Ofran and Rost, a classical FNN model is used for the prediction of protein–protein interaction sites from the protein sequence (Ofran and Rost 2007). In this case, AAs in the input protein sequence are represented using PSMM, predicted values of solvent accessibility, predicted secondary structure state, and a conservation score. The window size is set to include eight AAs surrounding the position that is being predicted, and the above-mentioned information concerning these nine amino acids is fed into the input layer.

NNs were also applied for prediction of metal-binding sites (Lin et al. 2005), binding sites for a specific protein family, that is, the binding sites of MHC I (Lundegaard et al. 2008), and prediction of functional sites, that is, the cleavage sites (Blom et al. 1996) and phosphorylation sites (Ingrell et al. 2007).

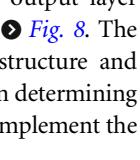
4.3 Prediction of Relative Solvent Accessibility with Neural Networks

Relative solvent accessibility (RSA) reflects the percentage of the surface area of a given AA in the protein sequence that is accessible to the solvent. RSA value, which is normalized to the [0, 1] interval, is defined as the ratio between the solvent accessible surface area (ASA) of an AA within a three-dimensional structure and the ASA of its extended tripeptide (Ala-X-Ala) conformation:

$$\text{RSA} = \frac{\text{RSA in 3D structure}}{\text{RSA in extended tripeptide conformation}}$$

The first study that concerned prediction of RSA from the protein sequence was published in 1994 by Rost and Sander (Rost and Sander 1994). In this work, the AA is encoded by the percentage of the occurrence of each AA type at this position in the sequence in multiple sequence alignment, which is similar to the values provided in the PSSM matrix. The input to the two-layer FNN is based on a window of size 9 which is centered on the AA that is being predicted and is used, which results in 9×20 features, together with the AA composition of the entire protein sequence, length of the sequence (using four values), and distance of the window from two termini of the sequence (using four values for each terminus). As a result, the network has a total of $180 + 20 + 4 + 8 = 212$ nodes in the input

layer. The output layer contains one node representing the predicted RSA value and no hidden layer is used in this model.

The past decade observed the development of several NN-based methods for the prediction of RSA values (Garg et al. 2005; Adamczak et al. 2005; Ahmad et al. 2003; Dor and Zhou 2007b; Pollastri et al. 2002). These methods share similar architectures and therefore we discuss one representative model proposed by Garg et al. (2005). This method is a two-stage design in which both stages are implemented using FNNs. Two sources of information are used to generate inputs for the NNs from the protein sequence, the PSSM profile, and the secondary structure predicted with the PSIPRED algorithm. The input features are extracted using a window of size 11 centered on the AA that is being predicted. The values from PSSM in the window are fed into the first NN. This results in the input layer with $11 \times 21 = 231$ nodes. The hidden layer contains ten nodes and the output layer has one node. In the second stage NN, the predicted RSA values of the AAs in the window and the predicted probabilities of the three secondary structure types predicted by PSIPRED in the same window are fed into the input layer. This results in $11 \times 4 = 44$ nodes in the input layer. The hidden layer includes ten nodes and the single node in the output layer corresponds to the final prediction. The architecture of this method is shown in  Fig. 8. The second layer exploits the observation that information about the secondary structure and solvent accessibility of the AAs in the adjacent positions in the sequence is useful in determining the solvent accessibility of a given AA. We observe that a similar design is used to implement the PSIPRED method.

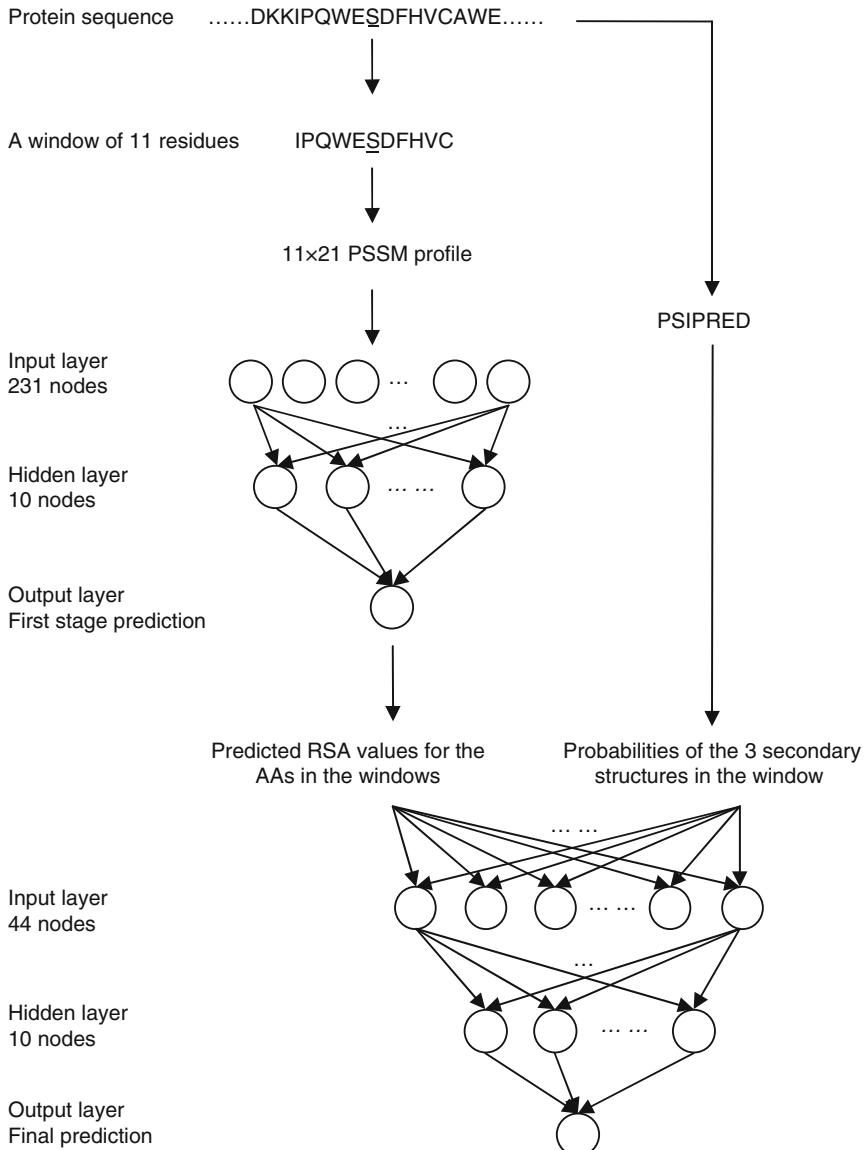
5 Summary

We summarized the applications of neural networks (NN) in bioinformatics, with a particular focus on protein bioinformatics. We show that numerous applications that aim at predictions of a variety of protein-related information, such as structure, binding sites, and localization, are designed and implemented using NNs. The most popular architecture used in these methods is a simple three-layer feedforward NN, although other architectures such as RBF and recurrent NNs are also applied. Some of the protein bioinformatics applications use multilayered designs in which two (or more) NNs are used in tandem. We show that the popularity of the NN-based designs has been growing over the last decade. Three applications that enjoy the most widespread use are discussed in greater detail. They include protein secondary structure prediction, prediction of binding sites, and prediction of relative solvent accessibility. We contrast and analyze the architectures of the corresponding NN models. We conclude that the extent and quality of the applications that are based on NNs indicate that this methodology provides sound and valuable results for the bioinformatics community.

We acknowledge several other useful resources that discuss the applications of a broader range of machine learning techniques in bioinformatics. Although none of these contributions is solely devoted to NNs, some of them discuss NNs together with other similar techniques. A survey by Narayanan and colleagues discusses applications of classification methods (nearest neighbor and decision trees), NNs, and genetic algorithms in bioinformatics (Narayanan et al. 2002). Another survey contribution by Kapetanovic and coworkers concerns clustering and classification algorithms, including NNs and support vector machines (Kapetanovic et al. 2004). The most recent review by Fogel discusses a host of computational intelligence techniques, including NNs, fuzzy systems, and evolutionary computation, in the context of

Fig. 8

Architecture of the model proposed in Garg et al. (2005) for the prediction of relative solvent accessibility. The method includes two stages implemented using FNNs. In the first stage, a window of 11 AAs is used, and each AA is represented by a vector of 21 values. The vector is taken from the PSSM profile generated by the PSI-BLAST algorithm. The output layer of the first stage generates one value that represents the predicted RSA value, which is further refined using the second stage. The predicted RSA values and the secondary structure probabilities predicted using PSIPRED of the AAs in the window are fed into the second-stage FNN. The output from the second-stage NN constitutes the final predicted RSA value.



bioinformatics (Fogel 2008). Several other surveys that do not treat NNs but which focus on the use of other related techniques in bioinformatics were published in recent years. They include a paper by Byvatov and Schneider (2003) that concerns applications of support vector machines; a contribution by Saeys et al. (2007) that discusses feature selection methods; a survey concerning Bayesian networks by Wilkinson (2007); a review of supervised classification, clustering, and probabilistic graphical models by Larranaga et al. (2006); and a recent contribution by Miller et al. (2008) that focuses on clustering.

References

- Adamczak R, Porollo A, Meller J (2005) Combining prediction of secondary structure and solvent accessibility in proteins. *Proteins* 59:467–475
- Ahmad S, Gromiha MM, Sarai A (2003) Real value prediction of solvent accessibility from amino acid sequence. *Proteins* 50:629–635
- Ahmad S, Gromiha MM, Sarai A (2004) Analysis and prediction of DNA-binding proteins and their binding residues based on composition, sequence and structural information. *Bioinformatics* 20:477–486
- Ahmad S, Sarai A (2005) PSSM-based prediction of DNA binding sites in proteins. *BMC Bioinformatics* 6:33
- Altschul SF, Madden TL, Schaffer AA, Zhang JH, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 17:3389–3402
- Blom N, Hansen J, Blaas D, Brunak S (1996) Cleavage site analysis in picornaviral polyproteins: discovering cellular targets by neural networks. *Protein Sci* 5:2203–2216
- Boguski MS (1998) Bioinformatics – a new era. *Trends Guide Bioinformatics* (Suppl S):1–3
- Byvatov E, Schneider G (2003) Support vector machine applications in bioinformatics. *Appl Bioinformatics* 2(2):67–77
- Cai YD, Zhou GP (2000) Prediction of protein structural classes by neural network. *Biochimie* 82:783–785
- Cai YD, Liu XJ, Chou KC (2002) Artificial neural network model for predicting protein subcellular location. *Comput Chem* 26:179–182
- Cai YD, Liu XJ, Chou KC (2003) Prediction of protein secondary structure content by artificial neural network. *J Comput Chem* 24:727–731
- Chandonia JM, Karplus M (1995) Neural networks for secondary structure and structural class predictions. *Protein Sci* 4:275–285
- Chen J, Chaudhari N (2007) Cascaded bidirectional recurrent neural networks for protein secondary structure prediction. *IEEE/ACM Trans Comput Biol Bioinform* 4:572–582
- Dor O, Zhou Y (2007a) Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins* 66:838–845
- Dor O, Zhou Y (2007b) Real-SPINE: an integrated system of neural networks for real-value prediction of protein structural properties. *Proteins* 68:76–81
- Emanuelsson O, Nielsen H, Brunak S, von Heijne G (2000) Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *J Mol Biol* 300:1005–1016
- Fogel GB (2008) Computational intelligence approaches for pattern discovery in biological systems. *Brief Bioinform* 9(4):307–316
- Fuchs A, Kirschner A, Frishman D (2009) Prediction of helix-helix contacts and interacting helices in polytopic membrane proteins using neural networks. *Proteins* 74:857–871
- Garg A, Kaur H, Raghava GP (2005) Real value prediction of solvent accessibility in proteins using multiple sequence alignment and secondary structure. *Proteins* 61:318–324
- Gromiha MM, Ahmad S, Suwa M (2005) TMBETA-NET: discrimination and prediction of membrane spanning beta-strands in outer membrane proteins. *Nucleic Acids Res* 33:W164–167
- Hung LH, Samudrala R (2003) PROTINFO: secondary and tertiary protein structure prediction. *Nucleic Acids Res* 31:3296–3299
- Ingrall CR, Miller ML, Jensen ON, Blom N (2007) Net-PhosYeast: prediction of protein phosphorylation sites in yeast. *Bioinformatics* 23:895–897
- Jacoboni I, Martelli PL, Fariselli P, De Pinto V, Casadio R (2001) Prediction of the transmembrane regions of beta-barrel membrane proteins with a neural network-based predictor. *Protein Sci* 10:779–787
- Jeong E, Chung IF, Miyano S (2004) A neural network method for identification of RNA-interacting residues in protein. *Genome Inform* 15:105–116
- Jones DT (1999) Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* 292:195–202
- Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22:2577–2637

- Kapetanovic IM, Rosenfeld S, Izmirlian G (2004) Overview of commonly used bioinformatics methods and their applications. *Ann NY Acad Sci* 1020:10–21
- Kaur H, Raghava GP (2003) A neural-network based method for prediction of gamma-turns in proteins from multiple sequence alignment. *Protein Sci* 12:923–929
- Kaur H, Raghava GP (2004) A neural network method for prediction of beta-turn types in proteins using evolutionary information. *Bioinformatics* 20:2751–2758
- Kirschner A, Frishman D (2008) Prediction of beta-turns and beta-turn types by a novel bidirectional Elman-type recurrent neural network with multiple output layers (MOLEBRNN). *Gene* 422:22–29
- Kuang R, Leslie CS, Yang AS (2004) Protein backbone angle prediction with machine learning approaches. *Bioinformatics* 20:1612–1621
- Kuznetsov IB, Gou Z, Li R, Hwang S (2006) Using evolutionary and structural information to predict DNA-binding sites on DNA-binding proteins. *Proteins* 64:19–27
- Larranaga P, Calvo B, Santana R, Bielza C, Galdiano J, Inza I, Lozano JA, Armananzas R, Santafe G, Perez A, Robles V (2006) Machine learning in bioinformatics. *Brief Bioinformatics* 7(1):86–112
- Lin CT, Lin KL, Yang CH, Chung IF, Huang CD, Yang YS (2005) Protein metal binding residue prediction based on neural networks. *Int J Neural Syst* 15:71–84
- Lundegaard C, Lamberth K, Harndahl M, Buus S, Lund O, Nielsen M (2008) NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8–11. *Nucleic Acids Res* 36:W509–512
- Luscombe NM, Greenbaum D, Gerstein M (2001) What is bioinformatics? A proposed definition and overview of the field. *Methods Inf Med* 40:346–358
- Martelli PL, Fariselli P, Casadio R (2004) Prediction of disulfide-bonded cysteines in proteomes with a hidden neural network. *Proteomics* 4:1665–1671
- Miller DJ, Wang Y, Kesidis G (2008) Emergent unsupervised clustering paradigms with potential application to bioinformatics. *Front Biosci* 13:677–690
- Muskal SM, Kim SH (1992) Predicting protein secondary structure content. A tandem neural network approach. *J Mol Biol* 225:713–727
- Nantasesnamat C, Isarankura-Na-Ayudhya C, Tansila N, Naenna T, Prachayositkul V (2007) Prediction of GFP spectral properties using artificial neural network. *J Comput Chem* 28:1275–1289
- Narayanan A, Keedwell EC, Olsson B (2002) Artificial intelligence techniques for bioinformatics. *Appl Bioinformatics* 1(4):191–222
- Natt NK, Kaur H, Raghava GP (2004) Prediction of transmembrane regions of beta-barrel proteins using ANN- and SVM-based methods. *Proteins* 56:11–18
- NIH Working Definition of Bioinformatics and Computational Biology (2000) BISTIC Definition Committee, <http://www.bistic.nih.gov/>
- Niwa T (2004) Prediction of biological targets using probabilistic neural networks and atom-type descriptors. *J Med Chem* 47:2645–2650
- Ofran Y, Rost B (2007) Protein-protein interaction hotspots carved into sequences. *PLoS Comput Biol* 3:e119
- Petersen TN, Lundegaard C, Nielsen M, Bohr H, Bohr J, Brunak S, Gippert GP, Lund O (2000) Prediction of protein secondary structure at 80% accuracy. *Proteins* 41:17–20
- Plewcynski D, Slabinski L, Ginalski K, Rychlewski L (2008) Prediction of signal peptides in protein sequences by neural networks. *Acta Biochim Pol* 55:261–267
- Pollastri G, Baldi P, Fariselli P, Casadio R (2002a) Prediction of coordination number and relative solvent accessibility in proteins. *Proteins* 47:142–153
- Pollastri G, Baldi P, Fariselli P, Casadio R (2002b) Prediction of coordination number and relative solvent accessibility in proteins. *Proteins* 47:142–153
- Qian N, Sejnowski TJ (1988) Predicting the secondary structure of globular proteins using neural network models. *J Mol Biol* 202:865–884
- Reinhardt A, Hubbard T (1998) Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Res* 26:2230–2236
- Rost B, Sander C (1994) Conservation and prediction of solvent accessibility in protein families. *Proteins* 20:216–226
- Rost B, Sander C, Schneider R (1994) PHD – an automatic mail server for protein secondary structure prediction. *Comput Appl Biosci* 10:53–60
- Ruan J, Wang K, Yang J, Kurgan LA, Cios KJ (2005) Highly accurate and consistent method for prediction of helix and strand content from primary protein sequences. *Artif Intell Med* 35:19–35
- Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. *Bioinformatics* 23(19):2507–2517
- Saha S, Raghava GP (2006) Prediction of continuous B-cell epitopes in an antigen using recurrent neural network. *Proteins* 65:40–48
- Sidhu A, Yang ZR (2006) Prediction of signal peptides using bio-basis function neural networks and decision trees. *Appl Bioinformatics* 5:13–19
- Vedani A, Dobler M (2000) Multi-dimensional QSAR in drug research. Predicting binding affinities, toxicity and pharmacokinetic parameters. *Prog Drug Res* 55:105–135
- Wilkinson DJ (2007) Bayesian methods in bioinformatics and computational systems biology. *Brief Bioinformatics* 8(2):109–116

- Xue B, Dor O, Faraggi E, Zhou Y (2008) Real-value prediction of backbone torsion angles. *Proteins* 72:427–433
- Yang ZR, Thomson R (2005) Bio-basis function neural network for prediction of protease cleavage sites in proteins. *IEEE Trans Neural Netw* 16:263–274
- Ye L, Liu T, Wu Z, Zhou R (2008) Sequence-based protein domain boundary prediction using BP neural network with various property profiles. *Proteins* 71:300–307
- Zhang GZ, Huang DS (2004) Prediction of inter-residue contacts map based on genetic algorithm optimized radial basis function neural network and binary input encoding scheme. *J Comput Aided Mol Des* 18:797–810
- Zhou HX, Shan Y (2001) Prediction of protein interaction sites from sequence profile and residue neighbor list. *Proteins* 44:336–343
- Zou L, Wang Z, Huang J (2007) Prediction of subcellular localization of eukaryotic proteins using position-specific profiles and neural network with weighted inputs. *J Genet Genomics* 34:1080–1087

19 Self-organizing Maps

Marc M. Van Hulle

Laboratorium voor Neurofysiologie, K.U. Leuven, Belgium

marc@neuro.kuleuven.be

1	<i>Introduction</i>	586
2	<i>SOM Algorithm</i>	589
3	<i>Applications of SOM</i>	596
4	<i>Extensions of SOM</i>	599
5	<i>Growing Topographic Maps</i>	600
6	<i>Recurrent Topographic Maps</i>	605
7	<i>Kernel Topographic Maps</i>	609
8	<i>Conclusion</i>	618

Abstract

A topographic map is a two-dimensional, nonlinear approximation of a potentially high-dimensional data manifold, which makes it an appealing instrument for visualizing and exploring high-dimensional data. The self-organizing map (SOM) is the most widely used algorithm, and it has led to thousands of applications in very diverse areas. In this chapter we introduce the SOM algorithm, discuss its properties and applications, and also discuss some of its extensions and new types of topographic map formation, such as those that can be used for processing categorical data, time series, and tree-structured data.

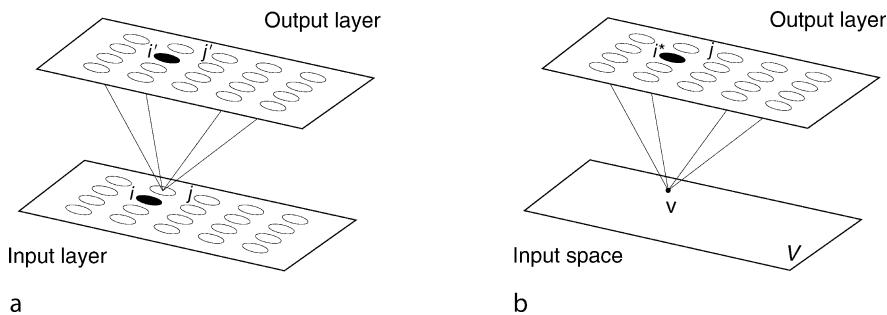
1 Introduction

One of the most prominent features of the mammalian brain is the *topographical* organization of its sensory cortex: neighboring nerve cells (neurons) can be driven by stimuli originating from neighboring positions in the sensory input space, and neighboring neurons in a given brain area project to neighboring neurons in the next area. In other words, the connections establish a so-called *neighborhood-preserving* or *topology-preserving* map, or *topographic map* for short. In the visual cortex, this is called a *retinotopic* map; in the somatosensory cortex, a *somatotopic* map (a map of the body surface); and in the auditory cortex, a *tonotopic* map (of the spectrum of possible sounds).

The study of topographic map formation, from a theoretical perspective, started with basically two types of self-organizing processes, gradient-based learning and competitive learning, and two types of network architectures (● Fig. 1) (for a review, see Van Hulle [2000]). In the first architecture, which is commonly referred to as the Willshaw–von der

■ Fig. 1

(a) Willshaw–von der Malsburg model. Two isomorphic, rectangular lattices of neurons are shown: one represents the input layer and the other the output layer. Neurons are represented by circles: filled circles denote active neurons (“winning” neurons); open circles denote inactive neurons. As a result of the weighted connections from the input to the output layer, the output neurons receive different inputs from the input layer. Two input neurons are labeled (i, j) as well as their corresponding output layer neurons (i', j') . Neurons i and i' are the only active neurons in their respective layers. (b) Kohonen model. The common input all neurons receive is directly represented in the input space, $v \in V \subseteq \mathbb{R}^d$. The “winning” neuron is labeled as i^* : its weight (vector) is the one that best matches the current input (vector).



Malsburg model (Willshaw and von der Malsburg 1976), there are two sets of neurons, arranged in two (one- or) two-dimensional layers or *lattices* (► Fig. 1a). (A lattice is an undirected graph in which every non-border vertex has the same, fixed number of incident edges, and which usually appears in the form of an array with a rectangular or simplex topology.) Topographic map formation is concerned with learning a mapping for which neighboring neurons in the input lattice are connected to neighboring neurons in the output lattice.

The second architecture is far more studied, and is also the topic of this chapter. Now we have continuously valued inputs taken from the input space \mathbb{R}^d , or the data manifold $V \subseteq \mathbb{R}^d$, which need not be rectangular or have the same dimensionality as the lattice to which it projects (► Fig. 1b). To every neuron i of the lattice A corresponds a reference position in the input space, called the weight vector $w_i = [w_{ij}] \in \mathbb{R}^d$. All neurons receive the same input vector $v = [v_1, \dots, v_d] \in V$. Topographic map formation is concerned with learning a map Ψ_A of the data manifold V (gray shaded area in ► Fig. 2), in such a way that neighboring lattice neurons, i, j , with lattice positions r_i, r_j , code for neighboring positions, w_i, w_j , in the input space (cf., the inverse mapping, Ψ). The forward mapping, Φ , from the input space to the lattice is not necessarily topology-preserving – neighboring weights do not necessarily correspond to neighboring lattice neurons – even after learning the map, due to the possible mismatch in dimensionalities of the input space and the lattice (see, e.g., ► Fig. 3). In practice, the map is represented in the input space in terms of neuron weights that are connected by straight lines, if the corresponding neurons are the nearest neighbors in the lattice (e.g., see the left panel of ► Fig. 2 or ► Fig. 3). When the map is topology preserving, it can be used for visualizing the data distribution by projecting the original data points onto the map. The advantage of having a flexible map, compared to, for example, a plane specified by principal components analysis (PCA), is demonstrated in ► Fig. 4. We observe that the three classes are better separated with a topographic map than with PCA. The most popular learning algorithm

► Fig. 2

Topographic mapping in the Kohonen architecture. In the *left panel*, the topology-preserving map Ψ_A of the data manifold $V \subseteq \mathbb{R}^d$ (gray-shaded area) is shown. The neuron weights, w_i, w_j , are connected by a straight line since the corresponding neurons i, j in the lattice A (*right panel*), with lattice coordinates r_i, r_j , are nearest neighbors. The forward mapping Φ is from the input space to the lattice; the backward mapping Ψ is from the lattice to the input space. The learning algorithm tries to make neighboring lattice neurons, i, j , code for neighboring positions, w_i, w_j , in the input space.

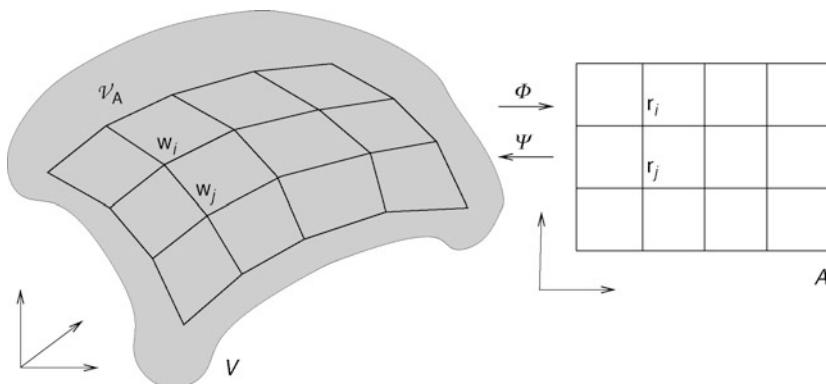
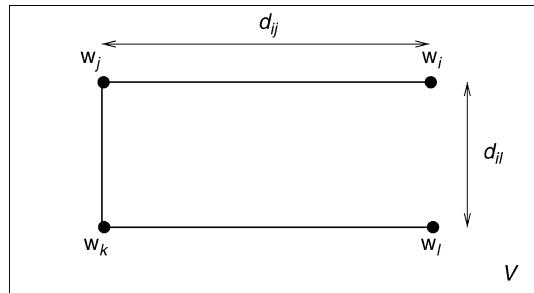
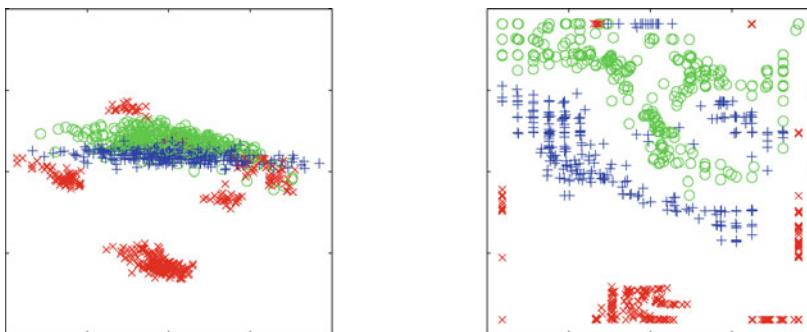


Fig. 3

Example of a one-dimensional lattice consisting of four neurons i, j, k, l in a two-dimensional rectangular space. The distance between the weight vectors of neurons i, j , d_{ij} is larger than between that of neurons i, l , d_{il} . This means that, at least in this example, neighboring neuron weights do not necessarily correspond to neighboring neurons in the lattice.

**Fig. 4**

Oil flow data set visualized using PCA (*left panel*) and a topographic map (*right panel*). The latter was obtained with the generative topographic map (GTM) algorithm (Bishop et al. 1996, 1998). Since the GTM performs a nonlinear mapping, it is better able to separate the three types of flow configurations: laminar (red crosses), homogeneous (blue pluses), and annular (green circles). (Bishop 2006, reprinted with permission.)



for this architecture is the self-organizing map (SOM) algorithm developed by Teuvo Kohonen (1982, 1984), whence this architecture is often referred to as Kohonen's model.

1.1 Chapter Overview

We start with the basic version of the SOM algorithm where we discuss the two stages of which it consists: the competitive and the cooperative ones. The discussion then moves on to the topographic ordering properties of the algorithm: how it unfolds and develops topographically ordered maps, whether there exists a mathematical proof of ordering, and whether topological defects in the map could still occur after the learning process has ended. We also discuss the convergence properties of the algorithm, and in what sense the converged weights are modeling the input density (is the weight density a linear function of the input density?).

We then discuss applications of the SOM algorithm, thousands of which have been reported in the open literature. Rather than attempting an extensive overview, the applications are grouped into three areas: vector quantization, regression, and clustering. The latter is the most important one since it is a direct consequence of the data visualization and exploration capabilities of the topographic map. A number of important applications are highlighted, such as WEBSOM (self-organizing maps for internet exploration) (Kaski et al. 1998) for organizing large document collections; PicSOM (Laaksonen et al. 2002) for content-based image retrieval; and the emergent self-organizing maps (ESOM) (Ultsch and Mörchen 2005), for which the MusicMiner (Risi et al. 2007) is considered, for organizing large collections of music, and an application for classifying police reports of criminal incidents.

Later, an overview of a number of extensions of the SOM algorithm is given. The motivation behind these is to improve the original algorithm, or to extend its range of applications, or to develop new ways to perform topographic map formation.

Three important extensions of the SOM algorithm are then given in detail. First, we discuss the growing topographic map algorithms. These algorithms consider maps with a dynamically defined topology so as to better capture the fine structure of the input distribution. Second, since many input sources have a temporal characteristic, which is not captured by the original SOM algorithm, several algorithms have been developed based on a recurrent processing of time signals (recurrent topographic maps). It is a heavily researched area since some of these algorithms are capable of processing tree-structured data. Third, another topic of current research is the kernel topographic map, which is in line with the “kernelization” trend of mapping data into a feature space. Rather than Voronoi regions, the neurons are equipped with overlapping activation regions, in the form of kernel functions, such as Gaussians. Important future developments are expected for these topographic maps, such as the visualization and clustering of structure-based molecule descriptions, and other biochemical applications.

Finally, a conclusion to the chapter is formulated.

2 SOM Algorithm

The SOM algorithm distinguishes two stages: the *competitive* stage and the *cooperative* stage. In the first stage, the best matching neuron is selected, that is, the “winner,” and in the second stage, the weights of the winner are adapted as well as those of its immediate lattice neighbors. Only the minimum Euclidean distance version of the SOM algorithm is considered (also the dot product version exists, see Kohonen 1995).

2.1 Competitive Stage

For each input $\mathbf{v} \in V$, the neuron with the smallest Euclidean distance (“winner-takes-all,” WTA) is selected, which we call the “winner”:

$$i^* = \arg \min_i \|\mathbf{w}_i - \mathbf{v}\| \quad (1)$$

By virtue of the minimum Euclidean distance rule, we obtain a Voronoi tessellation of the input space: to each neuron corresponds a region in the input space, the boundaries of which are perpendicular bisector planes of lines joining pairs of weight vectors (the gray-shaded area

in [Fig. 5](#) is the Voronoi region of neuron j). Remember that the neuron weights are connected by straight lines (links or edges): they indicate which neurons are nearest neighbors in the lattice. These links are important for verifying whether the map is topology preserving.

2.2 Cooperative Stage

It is now crucial to the formation of topographically ordered maps that the neuron weights are not modified independently of each other, but as topologically related subsets on which similar kinds of weight updates are performed. During learning, not only the weight vector of the winning neuron is updated, but also those of its lattice neighbors, which end up responding to similar inputs. This is achieved with the neighborhood function, which is centered at the winning neuron, and decreases with the lattice distance to the winning neuron. (Besides the neighborhood function, also the neighborhood set exists, consisting of all neurons to be updated in a given radius from the winning neuron [see Kohonen, (1995)].

The weight update rule in incremental mode is given by (with incremental mode we mean that the weights are updated each time an input vector is presented, contrasted with batch mode where the weights are only updated after the presentation of the full training set [“batch”]):

$$\Delta \mathbf{w}_i = \eta \Lambda(i, i^*, \sigma_A(t)) (\mathbf{v} - \mathbf{w}_i), \quad \forall i \in A \quad (2)$$

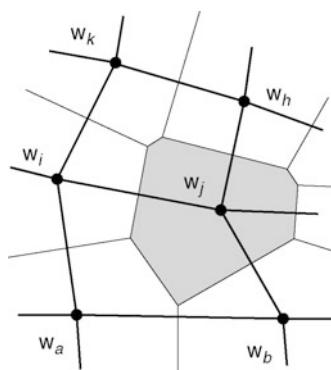
with Λ the neighborhood function, that is, a scalar-valued function of the lattice coordinates of neurons i and i^* , \mathbf{r}_i and \mathbf{r}_{i^*} , mostly a Gaussian:

$$\Lambda(i, i^*) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma_A^2}\right) \quad (3)$$

with range σ_A (i.e., the standard deviation). (we further drop the parameter $\sigma_A(t)$ from the neighborhood function to simplify the notation.) The positions \mathbf{r}_i are usually taken to be

Fig. 5

Definition of quantization region in the self-organizing map (SOM). Portion of a lattice (thick lines) plotted in terms of the weight vectors of neurons a, \dots, k , in the two-dimensional input space, that is, w_a, \dots, w_k .



the nodes of a discrete lattice with a regular topology, usually a two-dimensional square or rectangular lattice. An example of the effect of the neighborhood function in the weight updates is shown in [Fig. 6](#) for a 4×4 lattice. The parameter σ_A , and usually also the learning rate η , are gradually decreased over time. When the neighborhood range vanishes, the previous learning rule reverts to standard unsupervised competitive learning (UCL) (note that the latter is unable to form topology-preserving maps, pointing to the importance of the neighborhood function).

As an example, a 10×10 square lattice is trained with the SOM algorithm on a uniform square distribution $[-1, 1]^2$, using a Gaussian neighborhood function of which the range $\sigma_A(t)$ is decreased as follows:

$$\sigma_A(t) = \sigma_{A0} \exp\left(-2\sigma_{A0} \frac{t}{t_{\max}}\right) \quad (4)$$

with t the present time step, t_{\max} the maximum number of time steps, and σ_{A0} the range We take spanned by the neighborhood function at $t = 0$. We take $t_{\max} = 100,000$ and $\sigma_{A0} = 5$, and the learning rate $\eta = 0.01$. The initial weights (i.e., for $t = 0$) are chosen randomly from the same square distribution. Snapshots of the evolution of the lattice are shown in [Fig. 7](#). We observe that the lattice initially tangles, then contracts, unfolds, and expands so as to span the input distribution. This two-phased convergence process is an important property of the SOM algorithm, and it has been thoroughly studied from a mathematical viewpoint in the following terms: (1) the topographic ordering of the weights and, thus, the formation of

Fig. 6

The effect of the neighborhood function in the SOM algorithm. Starting from a perfect arrangement of the weights of a square lattice (full lines), the weights nearest to the current input (indicated with the cross) receive the largest updates, those further away smaller updates, resulting in the updated lattice (dashed lines).

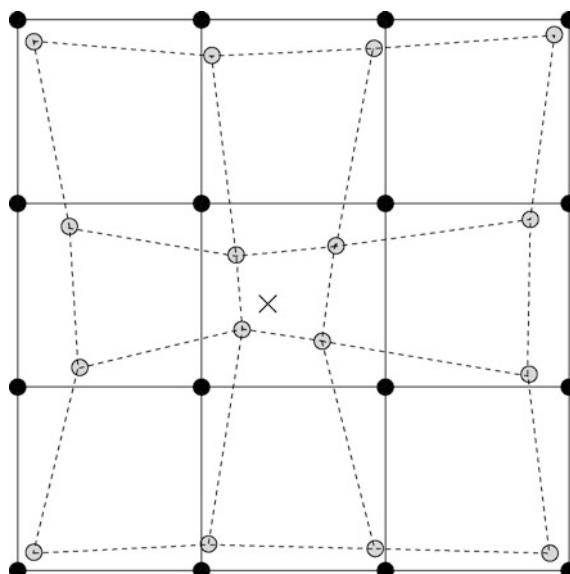
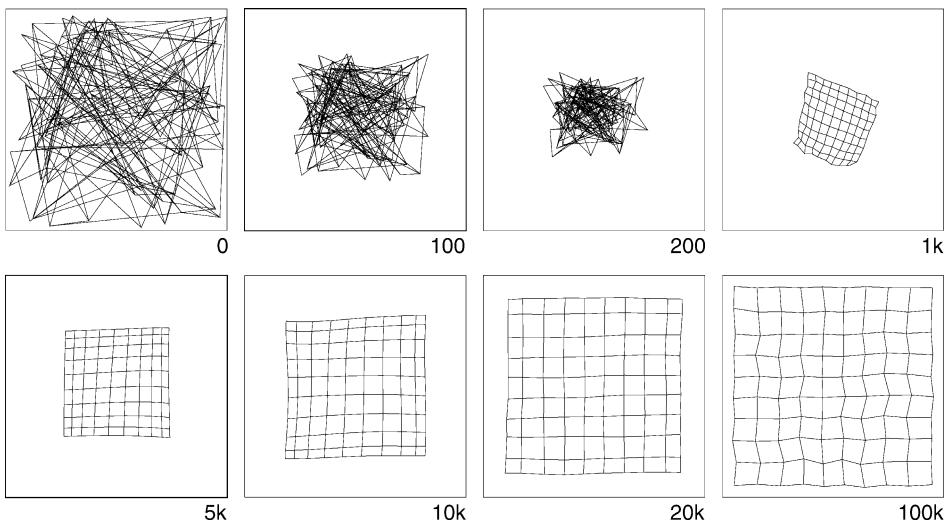


Fig. 7

Evolution of a 10×10 lattice with a rectangular topology as a function of time. The outer squares outline the uniform input distribution. The values given below the squares represent time.



topology-preserving mappings and (2) the convergence of these weights (energy function minimization). Both topics will be discussed next. Finally, the astute reader would have noticed that at the end of the learning phase, the lattice is smooth, but then suddenly becomes more erratic. This is an example of a phase transition, and it has been widely studied for the SOM algorithm (see Der and Herrmann (1993)).

Finally, since the speed of the convergence depends on the learning rate, a version without one has also been developed, called batch map (Kohonen 1995):

$$\mathbf{w}_i = \frac{\sum_{\mu} \Lambda(i^*, i) \mathbf{v}^{\mu}}{\sum_{\mu} \Lambda(i^*, i)}, \quad \forall i \quad (5)$$

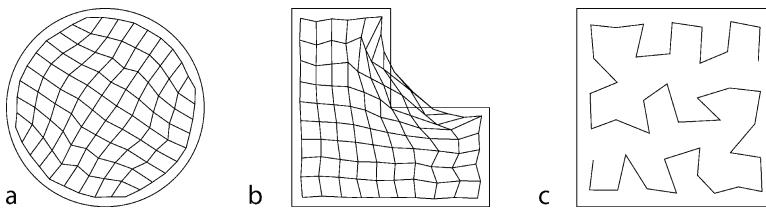
and it leads to a faster convergence of the map.

2.3 Topographic Ordering

In the example of [Fig. 7](#), a two-dimensional square lattice has been used for mapping a two-dimensional uniform, square distribution. One can also use the same lattice for mapping a non-square distribution, so that there is a topological mismatch, for example, a circular and an L-shaped distribution ([Fig. 8a, b](#)). The same lattice and simulation setup is used as before, but now only the final results are shown. Consider first the circular distribution: the weight distribution is now somewhat nonuniform. For the L-shaped distribution, one sees that there are several neurons outside the support of the distribution, and some of them even have a zero (or very low) probability of being active: hence, they are often called “dead” units. It is hard to find a better solution for these neurons without clustering them near the inside corner of the L-shape.

Fig. 8

Mapping of a 10×10 neuron lattice onto a circular (a) and an L-shaped (b) uniform distribution, and a 40-neuron one-dimensional lattice onto a square uniform distribution (c).



We can also explore the effect of a mismatch in lattice dimensionality. For example, we can develop a one-dimensional lattice (“chain”) in the same two-dimensional square distribution as before. (Note that it is now impossible to preserve all of the topology.) We see that the chain tries to fill the available space as much as possible (❷ Fig. 8c): the resulting map approximates the so-called space-filling *Peano curve*. (A Peano curve is an infinitely and recursively convoluted fractal curve which represents the continuous mapping of, for example, a one-dimensional interval onto a two-dimensional surface; Kohonen (1995) pp. 81, 87.)

2.3.1 Proofs or Ordering

It is clear that the neighborhood function plays a crucial role in the formation of topographically ordered weights. Although this may seem evident, the ordering itself is very difficult to describe (and prove!) in mathematical terms. The mathematical treatments that have been considered are, strictly speaking, only valid for one-dimensional lattices developed in one-dimensional spaces. Cottrell and Fort (1987) presented a mathematical stringent (but quite long) proof of the ordering process for the one-dimensional case. For a shorter constructive proof, one can refer to Kohonen (1995, pp. 100–105); for an earlier version, see Kohonen (1984, pp. 151–154). The results of Kohonen (1984) and Cottrell and Fort (1987) were extended by Erwin and coworkers (1992) to the more general case of a monotonically decreasing neighborhood function. However, the same authors also state that a strict proof of convergence is unlikely to be found for the higher-than-one-dimensional case.

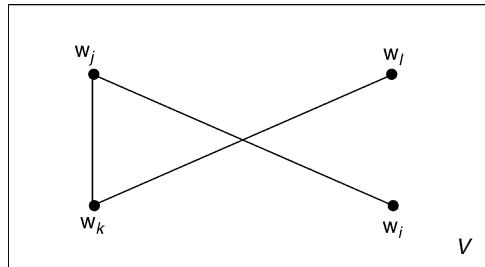
2.3.2 Topological Defects

As said before, the neighborhood function plays an important role in producing topographically ordered lattices; however, this does not imply that one is guaranteed to obtain it. Indeed, if one decreases the neighborhood range too fast, then there could be topological defects (Geszti 1990; Heskes and Kappen 1993). These defects are difficult to iron out, if at all, when the neighborhood range vanishes. In the case of a chain, one can obtain a so-called *kink* (❸ Fig. 9).

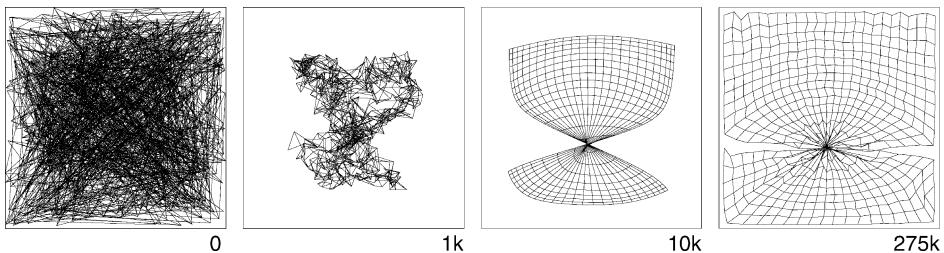
Consider, as a simulation example, a rectangular lattice sized $N = 24 \times 24$ neurons with the input samples taken randomly from a two-dimensional uniform distribution, $p(\mathbf{v})$, within

Fig. 9

Example of a topological defect ("kink") in a chain consisting of 4 neurons i, j, k, l in a two-dimensional rectangular space.

**Fig. 10**

Example of the formation of a topological defect called "twist" in a 24×24 lattice. The evolution is shown for different time instances (values below the squares).



the square $[0,1]^2$. The initial weight vectors are randomly drawn from this distribution. Now incremental learning is performed and the range is decreased as follows:

$$\sigma_1(t) = \sigma_{10} \exp\left(-2 \sigma_{10} \frac{t}{t_{\max}}\right) \quad (6)$$

but now with t the present time step and $t_{\max} = 275,000$. For the learning rate, $\eta = 0.015$ is taken. The evolution is shown in [Fig. 10](#). The neighborhood range was too rapidly decreased since the lattice is twisted and, even if one continues the simulation, with zero neighborhood range, the *twist* will not be removed.

2.4 Weight Convergence, Energy Function

Usually, in neural network learning algorithms, the weight update rule performs gradient descent on an energy function E (also termed error, cost, or distortion objective function):

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}} \quad (7)$$

so that convergence to a local minimum in E can be easily shown, for example, in an average sense. However, contrary to intuition, the SOM algorithm *does not* perform gradient descent

on an energy function as long as the neighborhood range has not vanished (when $\sigma_A = 0$, an energy function exists: one is minimizing the mean squared error (MSE) due to the quantization of the input space into Voronoi regions). Hence, strictly speaking, one cannot judge the degree of optimality achieved by the algorithm during learning.

In defense of the lack of an energy function, Kohonen emphasizes that there is no theoretical reason why the SOM algorithm *should* ensue from such a function (Kohonen 1995, p. 122) since (1) the SOM algorithm is aimed at developing (specific) topological relations between clusters in the input space and (2) it yields an approximative solution of an energy function (by virtue of its connection with the Robbins–Munro stochastic approximation technique), so that convergence is not a problem in practice.

Besides this, a lot of effort has been devoted to developing an energy function that is minimized during topographic map formation (Tolat 1990; Kohonen 1991; Luttrell 1991; Heskes and Kappen 1993; Erwin et al. 1992). Both Luttrell (1991) and Heskes and Kappen (1993) were able to show the existence of an energy function by modifying the definition of the winner. Heskes and Kappen first ascribed a *local error*, e_i , to each neuron, i , at time t :

$$e_i(\mathbf{W}, \mathbf{v}, t) = \frac{1}{2} \sum_{j \in A} \Lambda(i, j) \|\mathbf{v} - \mathbf{w}_j\|^2 \quad (8)$$

with $\mathbf{W} = [\mathbf{w}_i]$ the vector of all neuron weights, and then defined the winner as the neuron for which the local error is minimal:

$$i^* = \arg \min_i \sum_j \Lambda(i, j) \|\mathbf{v} - \mathbf{w}_j\|^2 \quad (9)$$

This was also the equation introduced by Luttrell (1991), but with $\sum_j \Lambda(i, j) = 1$. The actual weight update rule remains the same as in the original SOM algorithm and, thus, still considers several neurons around the winner. The solution proposed by Kohonen (1991) takes a different starting point, but it leads to a more complicated learning rule for the winning neuron (for the minimum Euclidean distance SOM algorithm, see Kohonen (1995, pp. 122–124)).

2.4.1 Weight Density

The next question is, “In what sense are the converged weights modeling the input probability density?” What is the distribution of the weights versus that of the inputs? Contrary to what was originally assumed (Kohonen 1984), the weight density at convergence, also termed the (inverse of the) magnification factor, is not a linear function of the input density $p(\mathbf{v})$.

Ritter and Schulten (1986) have shown that, for a one-dimensional lattice, developed in a one-dimensional input space:

$$p(w_i) \propto p(\mathbf{v})^{\frac{2}{3}} \quad (10)$$

in the limit of an infinite density of neighbor neurons (continuum approximation). Furthermore, when a discrete lattice is used, the continuum approach undergoes a correction, for example, for a neighborhood set with σ_A neurons around each winner neuron ($\Lambda(i, i^*) = 1$ iff $|r_i - r_{i^*}| \leq \sigma_A$):

$$p(w_i) \propto p^z \quad (11)$$

with:

$$\alpha = \frac{2}{3} - \frac{1}{3\sigma_A^2 + 3(\sigma_A + 1)^2}$$

For a discrete lattice of N neurons, it is expected that for $N \rightarrow \infty$ and for minimum mean-squared error (MSE) quantization, in d -dimensional space, the weight density will be proportional to (Kohonen 1995):

$$p(\mathbf{w}_i) \propto p^{\frac{1}{1+d}}(\mathbf{v}) \quad (12)$$

or that in the one-dimensional case:

$$p(w_i) \propto p(v)^{\frac{1}{3}} \quad (13)$$

The connection between the continuum and the discrete approach was established for the one-dimensional case by Ritter (1991), for a discrete lattice of N neurons, with $N \rightarrow \infty$, and for a neighborhood set with σ_A neurons around each “winner” neuron.

Finally, regardless of the effect of the neighborhood function or set, it is clear that the SOM algorithm tends to undersample high probability regions and oversample low probability ones. This affects the separability of clusters: for example; when the clusters overlap, the cluster boundary will be more difficult to delineate in the overlap region than for a mapping which has a linear weight distribution (Van Hulle 2000).

3 Applications of SOM

The graphical map displays generated by the SOM algorithm are easily understood, even by nonexperts in data analysis and statistics. The SOM algorithm has led to literally thousands of applications in areas ranging from automatic speech recognition, condition monitoring of plants and processes, cloud classification, and microarray data analysis, to document and image organization and retrieval (for an overview, see Centre (2003 <http://www.cis.hut.fi/research/som-bibl/>)). The converged neuron weights yield a model of the training set in three ways: vector quantization, regression, and clustering.

3.1 Vector Quantization

The training samples are modeled in such a manner that the average discrepancy between the data points and the neuron weights is minimized. In other words, the neuron weight vectors should “optimally” quantize the input space from which the training samples are drawn, just like one would desire for an adaptive vector quantizer (Gersho and Gray 1991). Indeed, in standard unsupervised competitive learning (UCL), and also the SOM algorithm, when the neighborhood has vanished (“zero-order” topology), the weight updates amount to centroid estimation (usually the mean of the samples which activate the corresponding neuron) and minimum Euclidean distance classification (Voronoi tessellation), and we attempt to minimize the mean squared error due to quantization, or some other quantization metric that one wishes to use. In fact, there exists an intimate connection between the batch version of the UCL rule and the zero-order topology SOM algorithm, on the one hand, and the generalized Lloyd algorithm for building vector quantizers, on the other hand (Luttrell 1989, 1990) (for a

review, see Van Hulle (2000)). Luttrell showed that the neighborhood function can be considered as a probability density function of which the range is chosen to capture the noise process responsible for the distortion of the quantizer's output code (i.e., the index of the winning neuron), for example, due to noise in the communication channel. Luttrell adopted for the noise process a zero-mean Gaussian, so that there is theoretical justification for choosing a Gaussian neighborhood function in the SOM algorithm.

3.2 Regression

One can also interpret the map as a case of non-parametric regression: no prior knowledge is assumed about the nature or shape of the function to be regressed. Non-parametric regression is perhaps the first successful statistical application of the SOM algorithm (Ritter et al. 1992; Mulier and Cherkassky 1995; Kohonen 1995): the converged topographic map is intended to capture the principal dimensions (principal curves and principal manifolds) of the input space. The individual neurons represent the “knots” that join piecewise smooth functions, such as splines, which act as interpolating functions for generating values at intermediate positions. Furthermore, the lattice coordinate system can be regarded as an (approximate) global coordinate system of the data manifold (● Fig. 2).

3.3 Clustering

The most widely used application of the topographic map is clustering, that is, the partitioning of the data set into subsets of “similar” data, without using prior knowledge about these subsets. One of the first demonstrations of clustering was by Ritter and Kohonen (1989). They had a list of 16 animals (birds, predators, and preys) and 13 binary attributes for each one of them (e.g., large size or not, hair or not, two legged or not, can fly or not, etc.). After training a 10×10 lattice of neurons with these vectors (supplemented with the 1-out-of-16 animal code vector, thus, in total, a 29-dimensional binary vector), and labeling the winning neuron for each animal code vector, a natural clustering of birds, predators, and prey appeared in the map. The authors called this the “semantic map.”

In the previous application, the clusters and their boundaries were defined by the user. In order to visualize clusters more directly, one needs an additional technique. One can compute the mean Euclidean distance between a neuron's weight vector and the weight vectors of its nearest neighbors in the lattice. The maximum and minimum of the distances found for all neurons in the lattice is then used for scaling these distances between 0 and 1; the lattice then becomes a gray scale image with white pixels corresponding to, for example, 0 and black pixels to 1. This is called the U-matrix (Ultsch and Siemon 1990), for which several extensions have been developed to remedy the oversampling of low probability regions (possibly transition regions between clusters) (Ultsch and Mörchen 2005).

An important example is WEBSOM (Kaski et al. 1998). Here, the SOM is used for organizing document collections (“document map”). Each document is represented as a vector of keyword occurrences. Similar documents then become grouped into the same cluster. After training, the user can zoom into the map to inspect the clusters. The map is manually or automatically labeled with keywords (e.g., from a man-made list) in such a way that, at each zoom level, the same density of keywords is shown (so as not to clutter the map with text).

WEBSOM has also been applied to visualizing clusters in patents based on keyword occurrences in patent abstracts (Kohonen et al. 1999).

An example of a content-based image retrieval system is the PicSOM (Laaksonen et al. 2002). Here, low-level features (color, shape, texture, etc.) of each image are considered. A separate two-dimensional SOM is trained for each low-level feature (in fact, a hierarchical SOM). In order to be able to retrieve one particular image from the database, one is faced with a semantic gap: how well do the low-level features correlate with the image contents? To bridge this gap, *relevance feedback* is used: the user is shown a number of images and he/she has to decide which ones are relevant and which ones are not (close or not to the image the user is interested in). Based on the trained PicSOM, the next series of images shown are then supposed to be more relevant, and so on.

For high-dimensional data visualization, a special class of topographic maps called emergent self-organizing maps (ESOM) (Ultsch and Mörchen 2005) can be considered. According to Ultsch, emergence is the ability of a system to produce a phenomenon on a new, higher level. In order to achieve emergence, the existence and cooperation of a large number of elementary processes is necessary. An emergent SOM differs from the traditional SOM in that a very large number of neurons (at least a few thousand) are used (even larger than the number of data points). The ESOM software is publicly available from <http://databionic-esom.sourceforge.net/>.

As an example, Ultsch and coworkers developed the *MusicMiner* for organizing large collections of music (Risi et al. 2007). Hereto, low-level audio features were extracted from raw audio data, and static and temporal statistics were used for aggregating these low-level features into higher-level ones. A supervised feature selection was performed to come up with a nonredundant set of features. Based on the latter, an ESOM was trained for clustering and visualizing collections of music. In this way, consistent clusters were discovered that correspond to music genres (Fig. 11). The user can then navigate the sound space and interact with the maps to discover new songs that correspond to his/her taste.

Fig. 11

Organizing large collections of music by means of an ESOM trained on high-level audio features (*MusicMiner*). Shown is the map with several music genres labeled. (Risi et al. 2007, reprinted with permission.)



As an example an ESOM application in the realm of text mining, the case of police reports of criminal incidents is considered. When a victim of a violent incident makes a statement to the police, the police officer has to judge whether, for example, domestic violence is involved. However, not all cases are correctly recognized and are, thus, wrongly assigned the label “nondomestic violence.” Because it is very time consuming to classify cases and to verify whether or not the performed classifications are correct, text mining techniques and a reliable classifier are needed. Such an automated triage system would result in major cost and time savings. A collection of terms (thesaurus), obtained by a frequency analysis of key words, was constructed, consisting of 123 terms. A 50×82 toroidal lattice (in a toroidal lattice, the vertical and horizontal coordinates are circular) was used, and trained with the ESOM algorithm on 4,814 reports of the year 2007; the validation set consisted of 4,738 cases (of the year 2006). The outcome is shown in [Fig. 12](#) (Poelmans 2008, unpublished results).

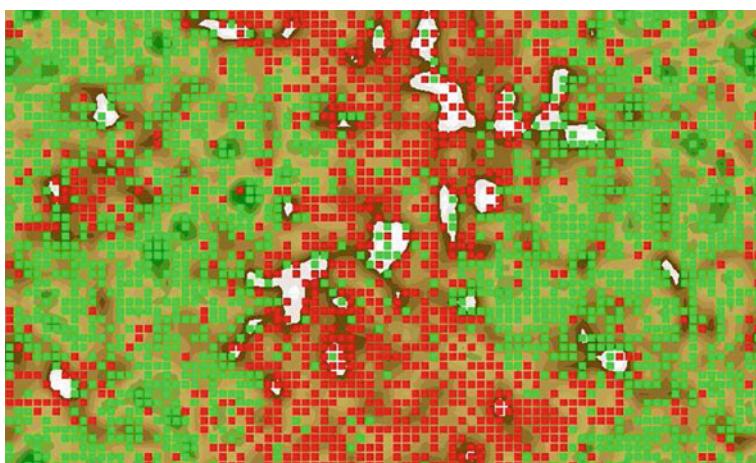
4 Extensions of SOM

Although the original SOM algorithm has all the necessary ingredients for developing topographic maps, many adapted versions have emerged over the years (for references, see Kohonen (1995) and <http://www.cis.hut.fi/research/som-bibl/> which contains over 7,000 articles). For some of these, the underlying motivation was to improve the original algorithm, or to extend its range of applications, while for others the SOM algorithm has served as a source of inspiration for developing new ways to perform topographic map formation.

One motivation was spurred by the need to develop a learning rule that performs gradient descent on an energy function (as discussed above). Another was to remedy the occurrence of

Fig. 12

Toroidal lattice trained with the ESOM algorithm showing the distribution of domestic violence cases (red squares) and nondomestic violence cases (green squares). The background represents the sum of the distances between the weight vector of each neuron and those of its nearest neighbors in the lattice, normalized by the largest occurring sum (white) (i.e., the U-matrix). It is observed that the domestic violence cases appear in one large cluster and a few smaller clusters, mostly corresponding to violence in homosexual relationships. (Courtesy of Jonas Poelmans.)



dead units, since they do not contribute to the representation of the input space (or the data manifold). Several researchers were inspired by Grossberg's idea (1976) of adding a "conscience" to frequently winning neurons to feel "guilty" and to reduce their winning rates. The same heuristic idea has also been adopted in combination with topographic map formation (DeSieno 1988; Van den Bout and Miller 1989; Ahalt et al. 1990). Others exploit measures based on the local distortion error to equilibrate the neurons' "conscience" (Kim and Ra 1995; Chinrungrueng and Séquin 1995; Ueda and Nakano 1993). A combination of the two conscience approaches is the learning scheme introduced by Bauer and coworkers (1996).

A different strategy is to apply a competitive learning rule that minimizes the mean absolute error (MAE) between the input samples v and the N weight vectors (also called the *Minkowski metric* of power one) (Kohonen 1995, pp. 120, 121) (see also Lin et al. (1997)). Instead of minimizing a (modified) distortion criterion, a more natural approach is to optimize an information-theoretic criterion directly. Linsker was among the first to explore this idea in the context of topographic map formation. He proposed a principle of *maximum information preservation* (Linsker 1988) – *infomax* for short – according to which a processing stage has the property that the output signals will optimally discriminate, in an information-theoretic sense, among possible sets of input signals applied to that stage. In his 1989 article, he devised a learning rule for topographic map formation in a probabilistic WTA network by maximizing the average mutual information between the output and the signal part of the input, which was corrupted by noise (Linsker 1989). Another algorithm is the vectorial boundary adaptation rule (VBAR) which considers the region spanned by a quadrilateral (four neurons forming a square region in the lattice) as the quantization region (Van Hulle 1997a, b), and which is able to achieve an equiprobabilistic map, that is, a map for which every neuron has the same chance to be active (and, therefore, maximizes the information-theoretic entropy).

Another evolution is the growing topographic map algorithms. In contrast to the original SOM algorithm, its growing map variants have a dynamically defined topology, and they are believed to better capture the fine structure of the input distribution. They will be discussed in the next section.

Many input sources have a temporal characteristic, which is not captured by the original SOM algorithm. Several algorithms have been developed based on a recurrent processing of time signals and a recurrent winning neuron computation. Also tree structured data can be represented with such topographic maps. Recurrent topographic maps will be discussed in this chapter.

Another important evolution is the kernel-based topographic maps: rather than Voronoi regions, the neurons are equipped with overlapping activation regions, usually in the form of kernel functions, such as Gaussians (☞ Fig. 19). Also for this case, several algorithms have been developed, and a number of them will be discussed in this chapter.

5 Growing Topographic Maps

In order to overcome the topology mismatches that occur with the original SOM algorithm, as well as to achieve an optimal use of the neurons (cf., dead units), the geometry of the lattice has to match that of the data manifold it is intended to represent. For that purpose, several so-called growing (incremental or structure-adaptive) self-organizing map algorithms have been developed. What they share is that the lattices are gradually built up and, hence, do not have a predefined structure (i.e., number of neurons and possibly also lattice dimensionality)

( Fig. 14). The lattice is generated by a successive insertion (and possibly an occasional deletion) of neurons and connections between them. Some of these algorithms can even guarantee that the lattice is free of topological defects (e.g., since the lattice is a subgraph of a Delaunay triangulation, see further). The major algorithms for growing self-organizing maps will be briefly reviewed. The algorithms are structurally not very different; the main difference is with the constraints imposed on the lattice topology (fixed or variable lattice dimensionality). The properties common to these algorithms are first listed, using the format suggested by Fritzke (1996).

- The network is an undirected graph (lattice) consisting of a number of nodes (neurons) and links or edges connecting them.
- Each neuron, i , has a weight vector \mathbf{w}_i in the input space V .
- The weight vectors are updated by moving the winning neuron i^* , and its topological neighbors, toward the input $\mathbf{v} \in V$:

$$\Delta \mathbf{w}_{i^*} = \eta_{i^*} (\mathbf{v} - \mathbf{w}_{i^*}) \quad (14)$$

$$\Delta \mathbf{w}_i = \eta_i (\mathbf{v} - \mathbf{w}_i), \quad \forall i \in \mathcal{N}_{i^*} \quad (15)$$

with \mathcal{N}_{i^*} the set of direct topological neighbors of neuron i^* (neighborhood set), and with η_{i^*} and η_i the learning rates, η_{i^*}, η_i .

- At each time step, the local error at the winning neuron, i^* , is accumulated:

$$\Delta E_{i^*} = (\text{error measure}) \quad (16)$$

The error term is coming from a particular area around \mathbf{w}_{i^*} , and is likely to be reduced by inserting new neurons in that area. A central property of these algorithms is the possibility to choose an arbitrary error measure as the basis for insertion. This extends their applications from unsupervised learning ones, such as data visualization, combinatorial optimization, and clustering analysis, to supervised learning ones, such as classification and regression. For example, for vector quantization, $\Delta E_{i^*} = \|\mathbf{v} - \mathbf{w}_{i^*}\|^2$. For classification, the obvious choice is the classification error. All models reviewed here can, in principle, be used for supervised learning applications by associating output values to the neurons, for example, through kernels such as radial basis functions. This makes most sense for the algorithms that adapt their dimensionality to the data.

- The accumulated error of each neuron is used to determine (after a fixed number of time steps) where to insert new neurons in the lattice. After an insertion, the error information is locally redistributed, which increases the probability that the next insertion will be somewhere else. The local error acts as a kind of memory where much error has occurred; the exponential decay of the error stresses more the recently accumulated error.
- All parameters of the algorithm stay constant over time.

5.1 Competitive Hebbian Learning and Neural Gas

Historically, the first algorithm to develop topologies was introduced by Martinet and Schulten, and it is a combination of two methods: competitive Hebbian learning (CHL) (Martinetz 1993) and the neural gas (NG) (Martinetz and Schulten 1991).

The principle behind CHL is simple: for each input, create a link between the winning neuron and the second winning neuron (i.e., with the second smallest Euclidean distance to the input), if that link does not already exist. Only weight vectors lying in the data manifold develop links between them (thus, nonzero input density regions). The resulting graph is a subgraph of the (induced) Delaunay triangulation (☞ Fig. 13), and it has been shown to optimally preserve topology in a very general sense.

In order to position the weight vectors in the input space, Martinetz and Schulten (1991) have proposed a particular kind of vector quantization method, called neural gas (NG). The main principle of NG is for each input v update the k nearest-neighbor neuron weight vectors, with k decreasing over time until only the winning neuron's weight vector is updated. Hence, one has a neighborhood function but now in input space. The learning rate also follows a decay schedule. Note that the NG by itself does not delete or insert any neurons. The NG requires finetuning of the rate at which the neighborhood shrinks to achieve a smooth convergence and proper modeling of the data manifold.

The combination of CHL and NG is an effective method for topology learning. The evolution of the lattice is shown in ☞ Fig. 14 for a data manifold that consists of three-, two-, and one-dimensional subspaces (Martinetz and Schulten 1991). We see that the lattice has successfully filled and adapted its dimensionality to the different subspaces. For this reason, visualization is only possible for low-dimensional input spaces (hence, it is not suited for data visualization purposes where a mapping from a potentially high-dimensional input space to a low-dimensional lattice is desired). A problem with the algorithm is that one needs to decide *a priori* the number of neurons, as required by the NG algorithm (Fritzke 1996): depending on the complexity of the data manifold, very different numbers may be appropriate. This problem is overcome in the growing neural gas (GNG) (Fritzke 1995a) (see ☞ Sect. 5.2).

☞ Fig. 13

Left panel: Delaunay triangulation. The neuron weight positions are indicated with open circles; the thick lines connect the nearest neighbor weights. The borders of the Voronoi polygons, corresponding to the weights, are indicated with thin lines. **Right panel:** Induced Delaunay triangulation. The induced triangulation is obtained by masking the original triangulation with the input data distribution (two disconnected gray shaded regions).

(Fritzke 1995a, reprinted with permission.)

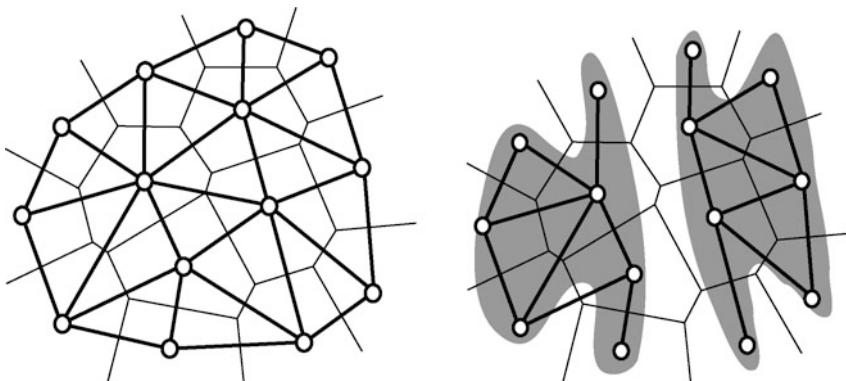
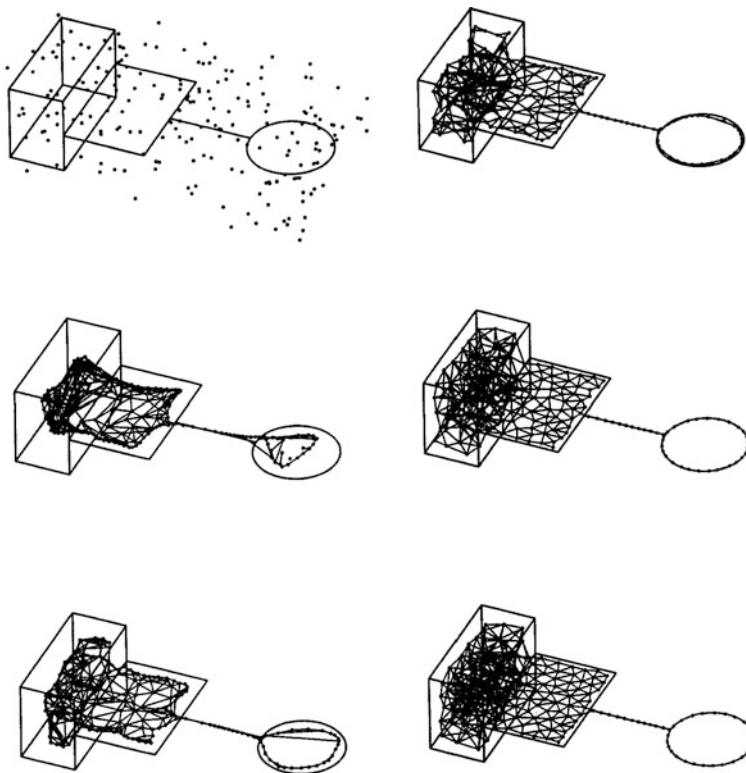


Fig. 14

Neural gas algorithm, combined with competitive Hebbian learning, applied to a data manifold consisting of a right parallelepiped, a rectangle, and a circle connecting a line. The dots indicate the positions of the neuron weights. Lines connecting neuron weights indicate lattice edges. Shown are the initial result (top left), and further the lattice after 5,000, 10,000, 15,000, 25,000, and 40,000 time steps (top-down the first column, then top-down the second column). (Martinetz and Schulten 1991, reprinted with permission.)

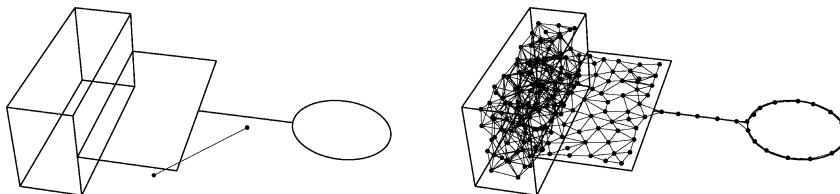


5.2 Growing Neural Gas

Contrary to CHL/NG, the growing neural gas (GNG) poses no explicit constraints on the lattice. The lattice is generated, and constantly updated, by the competitive Hebbian learning technique (CHL, see above; Martinetz 1993). The algorithm starts with two randomly placed and connected neurons (Fig. 15, left panel). Unlike the CHL/NG algorithm, after a fixed number, λ , of time steps, the neuron, i , with the largest accumulated error is determined and a new neuron inserted between i and one of its neighbors. Hence, the GNG algorithm exploits the topology to position new neurons between existing ones, whereas in the CHL/NG the topology is not influenced by the NG algorithm. Error variables are locally redistributed and another λ time step is performed. The lattice generated is a subgraph of a Delaunay triangularization, and can have different dimensionalities in different regions of the data manifold. The end-result is very similar to CHL/NG (Fig. 15, right panel).

Fig. 15

Growing neural gas algorithm applied to the same data configuration as in [Fig. 14](#). Initial lattice (*left panel*) and lattice after 20,000 time steps (*right panel*). Note that the last one is not necessarily the final result because the algorithm could run indefinitely. (Fritzke 1995a, reprinted with permission.)



5.3 Growing Cell Structures

In the growing cell structures (GCS) algorithm (Fritzke 1994), the model consists of hypertetrahedrons (or simplices) of a dimensionality chosen in advance (hence, the lattice dimensionality is fixed). Note that a d_A -dimensional hypertetrahedron has d_A+1 vertices, with d_A the lattice dimensionality, and $d_A \leq d$, with d the input space dimensionality. Examples for $d = 1, 2$, and 3 are a line, a triangle, and a tetrahedron, respectively.

The model is initialized with exactly one hypertetrahedron. Always after a prespecified number of time steps, the neuron, i , with the maximum accumulated error is determined and a new neuron is inserted by splitting the longest of the edges emanating from i . Additional edges are inserted to rebuild the structure in such a way that it consists only of d_A -dimensional hypertetrahedrons: Let the edge which is split connect neurons i and j , then the newly inserted neuron should be connected to i and j and with all *common* topological neighbors of i and j .

Since the GCS algorithm assumes a fixed dimensionality for the lattice, it can be used for generating a dimensionality reducing mapping from the input space to the lattice space, which is useful for data visualization purposes.

5.4 Growing Grid

In the growing grid algorithm (GG) (Fritzke 1995b), the lattice is a rectangular grid of a certain dimensionality d_A . The starting configuration is a d_A -dimensional hypercube, for example, a 2×2 lattice for $d_A = 2$, a $2 \times 2 \times 2$ lattice for $d_A = 3$, and so on. To keep this structure consistent, it is necessary to always insert complete (hyper-)rows and (hyper-)columns. Since the lattice dimensionality is fixed, and possibly much smaller than the input space dimensionality, the GG is useful for data visualization.

Apart from these differences, the algorithm is very similar to the ones described above. After λ time steps, the neuron with the largest accumulated error is determined, and the longest edge emanating from it is identified, and a new complete hyper-row or -column is inserted such that the edge is split.

5.5 Other Algorithms

There exists a wealth of other algorithms, such as the dynamic cell structures (DCS) (Bruske and Sommer 1995), which is similar to the GNG; the growing self-organizing map (GSOM, also called hypercubical SOM) (Bauer and Villmann 1997), which has some similarities to GG but it adapts the lattice dimensionality; incremental grid growing (IGG), which introduces new neurons at the lattice border and adds/removes connections based on the similarities of the connected neurons' weight vectors (Blackmore and Miikkulainen 1993); and one that is also called the growing self-organizing map (GSOM) (Alahakoon et al. 2000), which also adds new neurons at the lattice border, similar to IGG, but does not delete neurons, and which contains a spread factor to let the user control the spread of the lattice.

In order to study and exploit hierarchical relations in the data, hierarchical versions of some of these algorithms have been developed. For example, the growing hierarchical self-organizing map (GHSOM) (Rauber et al. 2002) develops lattices at each level of the hierarchy using the GG algorithm (insertion of columns or rows). The orientation in space of each lattice is similar to that of the parent lattice, which facilitates the interpretation of the hierarchy, and which is achieved through a careful initialization of each lattice. Another example is adaptive hierarchical incremental grid growing (AHIGG) (Merkl et al. 2003) of which the hierarchy consists of lattices trained with the IGG algorithm, and for which new units at a higher level are introduced when the local (quantization) error of a neuron is too large.

6 Recurrent Topographic Maps

6.1 Time Series

Many data sources such as speech have a temporal characteristic (e.g., a correlation structure) that cannot be sufficiently captured when ignoring the order in which the data points arrive, as in the original SOM algorithm. Several self-organizing map algorithms have been developed for dealing with sequential data, such as those using:

- fixed-length windows, for example, the time-delayed SOM (Kangas 1990), among others (Martinetz et al. 1993; Simon et al. 2003; Vesanto 1997);
- specific sequence metrics (Kohonen 1997; Somervuo 2004);
- statistical modeling incorporating appropriate generative models for sequences (Bishop et al. 1997; Tiño et al. 2004);
- mapping of temporal dependencies to spatial correlation, for example, as in traveling wave signals or potentially trained, temporally activated lateral interactions (Euliano and Principe 1999; Schulz and Reggia 2004; Wiemer 2003);
- recurrent processing of time signals and recurrent winning neuron computation based on the current input and the previous map activation, such as with the temporal Kohonen map (TKM) (Chappell and Taylor 1993), the recurrent SOM (RSOM) (Koskela et al. 1998), the recursive SOM (RecSOM) (Voegtlin 2002), the SOM for structured data (SOMSD) (Hagenbuchner et al. 2003), and the merge SOM (MSOM) (Strickert and Hammer 2005).

Several of these algorithms proposed recently, which shows the increased interest in representing time series with topographic maps. For some of these algorithms, also tree

structured data can be represented (see later). We focus on the recurrent processing of time signals, and we briefly describe the models listed. A more detailed overview can be found elsewhere (Barreto and Araújo 2001; Hammer et al. 2005). The recurrent algorithms essentially differ in the context, that is, the way by which sequences are internally represented.

6.1.1 Overview of Algorithms

The TKM extends the SOM algorithm with recurrent self-connections of the neurons, such that they act as leaky integrators (☞ Fig. 16a). Given a sequence $\{v_1, \dots, v_t\}$, $v_j \in \mathbb{R}^d$, $\forall j$, the integrated distance ID_i of neuron i with weight vector $w_i \in \mathbb{R}^d$ is:

$$ID_i(t) = \alpha \|v_t - w_i\|^2 + (1 - \alpha)ID_i(t - 1) \quad (17)$$

with $\alpha \in (0, 1)$ a constant determining the strength of the context information, and with $ID_i(0) \stackrel{d}{=} 0$. The winning neuron is selected as $i^*(t) = \operatorname{argmin}_i ID_i(t)$, after which the network is updated as in the SOM algorithm. ☞ Equation 17 has the form of a leaky integrator, integrating previous distances of neuron i , given the sequence.

The RSOM uses, in essence, the same dynamics; however, it integrates over the directions of the individual weight components:

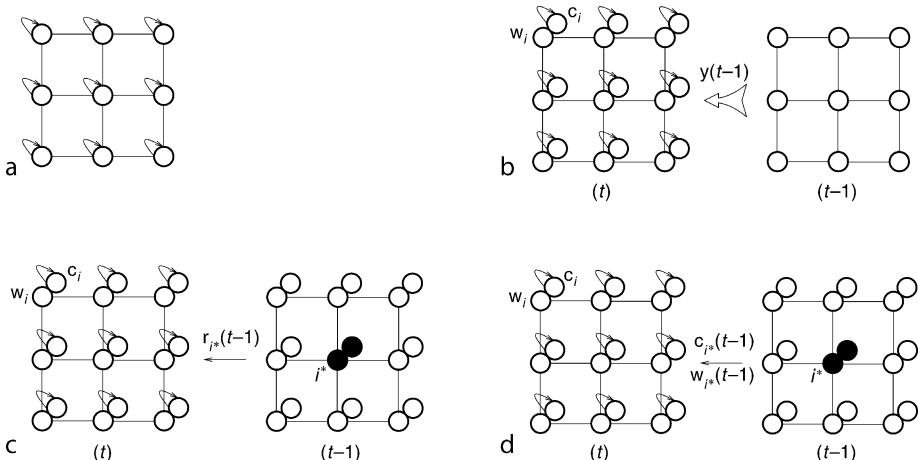
$$ID_{ij}(t) = \alpha(v_{jt} - w_{ij}) + (1 - \alpha)ID_{ij}(t - 1) \quad (18)$$

so that the winner is then the neuron for which $\|ID_{ij}(t)\|^2$ is the smallest. It is clear that this algorithm stores more information than the TKM. However, both the TKM and the RSOM compute only a leaky average of the time series and they do not use any explicit context.

The RecSOM is an algorithm for sequence prediction. A given sequence is recursively processed based on the already computed context. Hereto, each neuron i is equipped with a

Fig. 16

Schematic representation of four recurrent SOM algorithms: TKM (a), RecSOM (b), SOMSD (c), and MSOM (d). Recurrent connections indicate leaky integration; double circles indicate the neuron's weight- and context vectors; i^* and the filled circles indicate the winning neuron; (t) and (t - 1) represent the current and the previous time steps, respectively.



weight and, additionally, a context vector $\mathbf{c}_i \in \mathbb{R}^N$ that stores an activation profile of the whole map, indicating in which context the weight vector should arise (Fig. 16b). The integrated distance is defined as:

$$ID_i(t) = \alpha \|\mathbf{v}_t - \mathbf{w}_i\|^2 + \beta \|\mathbf{y}(t-1) - \mathbf{c}_i\|^2 \quad (19)$$

with $\mathbf{y}(t-1) = [\exp(-ID_1(t-1)), \dots, \exp(-ID_N(t-1))]$, $\alpha, \beta > 0$ constants to control the respective contributions from pattern and context matching, and with $ID_i(0) \triangleq 0$. The winner is defined as the neuron for which the integrated distance is minimal. The equation contains the exponential function in order to avoid numerical explosion: otherwise, the activation, ID_i , could become too large because the distances with respect to the contexts of all N neurons could accumulate. Learning is performed on the weights as well as the contexts, in the usual way (thus, involving a neighborhood function centered around the winner): the weights are adapted toward the current input sequences, the contexts toward the recursively computed contexts \mathbf{y} .

The SOMSD was developed for processing labeled trees with fixed fan-out k . The limiting case of $k = 1$ covers sequences. We further restrict ourselves to sequences. Each neuron has, besides a weight, a context vector $\mathbf{c}_i \in \mathbb{R}^{d_A}$, with d_A the dimensionality of the lattice. The winning neuron i^* for a training input at time t is defined as (Fig. 16c):

$$i^* = \arg \min \alpha \|\mathbf{v}_t - \mathbf{w}_i\|^2 + (1 - \alpha) \|r_{i^*}(t-1) - \mathbf{c}_i\|^2 \quad (20)$$

with r_{i^*} the lattice coordinate of the winning neuron. The weights \mathbf{w}_i are moved in the direction of the current input, as usual (i.e., with a neighborhood), and the contexts \mathbf{c}_i in the direction of the lattice coordinates with the winning neuron of the previous time step (also with a neighborhood).

The MSOM algorithm accounts for the temporal context by an explicit vector attached to each neuron that stores the preferred context of that neuron (Fig. 16d). The MSOM characterizes the context by a “merging” of the weight and the context of the winner in the previous time step (whence the algorithm’s name: merge SOM). The integrated distance is defined as:

$$ID_i(t) = \alpha \|\mathbf{w}_i - \mathbf{v}_t\|^2 + (1 - \alpha) \|\mathbf{c}_i - \mathbf{C}_t\|^2 \quad (21)$$

with $\mathbf{c}_i \in \mathbb{R}^d$, and with \mathbf{C}_t the expected (merged) weight/context vector, that is, the context of the previous winner:

$$\mathbf{C}_t = \gamma \mathbf{c}_{i^*}(t-1) + (1 - \gamma) \mathbf{w}_{i^*}(t-1) \quad (22)$$

with $\mathbf{C}_0 \triangleq 0$. Updating of \mathbf{w}_i and \mathbf{c}_i are then done in the usual SOM way, thus, with a neighborhood function centered around the winner. The parameter, α , is controlled so as to maximize the entropy of the neural activity.

6.1.2 Comparison of Algorithms

Hammer and coworkers (2004) pointed out that several of the mentioned recurrent self-organizing map algorithms share their principled dynamics, but differ in their internal representations of context. In all cases, the context is extracted as the relevant part of the activation of the map in the previous time step. The notion of “relevance” thus differs between the algorithms (see also Hammer et al. 2005). The recurrent self-organizing algorithms can be

divided into two categories: the representation of the context in the data space, such as for the TKM and MSOM, and in a space that is related to the neurons, as for SOMSD and RecSOM. In the first case, the storage capacity is restricted by the input dimensionality. In the latter case, it can be enlarged simply by adding more neurons to the lattice. Furthermore, there are essential differences in the dynamics of the algorithms. The TKM does not converge to the optimal weights; RSOM does it but the parameter α occurs both in the encoding formula and in the dynamics. In the MSOM algorithm, they can be controlled separately. Finally, the algorithms differ in memory and computational complexity (RecSOM is quite demanding, SOMSD is fast, and MSOM is somewhere in the middle), the possibility to apply different lattice types (such as hyperbolic lattices; Ritter 1998), and their capacities (MSOM and SOMSD achieve the capacity of finite state automata, but TKM and RSOM have smaller capacities; RecSOM is more complex to judge).

As an example, Voegtlín (2002) used the Mackey–Glass time series, a well-known one-dimensional time-delay differential equation, for comparing different algorithms:

$$\frac{dv}{dt} = bv(t) + \frac{av(t-\tau)}{1 + v(t-\tau)^{10}} \quad (23)$$

which for $\tau > 16.8$ generates a chaotic time series. Voegtlín used $a = 0.2$, $b = -0.1$, and $\tau = 17$. A sequence of values is plotted in Fig. 17, starting from uniform input conditions. For training, the series is sampled every three time units. This example was also taken up by Hammer et al. (2004) for comparing their MSOM. Several 10×10 maps were trained using 150,000 iterations; note that the input dimensionality $d = 1$ in all cases. Fig. 18 shows the quantization error plotted as a function of the index of the past input (index = 0 means the present). The error is expressed in terms of the average standard deviation of the given sequence and the winning neuron's receptive field over a window of 30 time steps (i.e., delay vector). We observe large fluctuations for the SOM, which is due to the temporal regularity of the series and the absence of any temporal coding by the SOM algorithm. We also observe that the RSOM algorithm is not really better than the SOM algorithm. On the contrary, the RecSOM, SOMSD, and MSOM algorithms (the MSOM was trained with a neural gas neighborhood function, for details see Strickert and Hammer (2003a)) display a slow increase in error as a function of the past, but with a better performance for the MSOM algorithm.

Fig. 17

Excerpt from the Mackey–Glass chaotic time series. (Strickert and Hammer 2003b, reprinted with permission.)

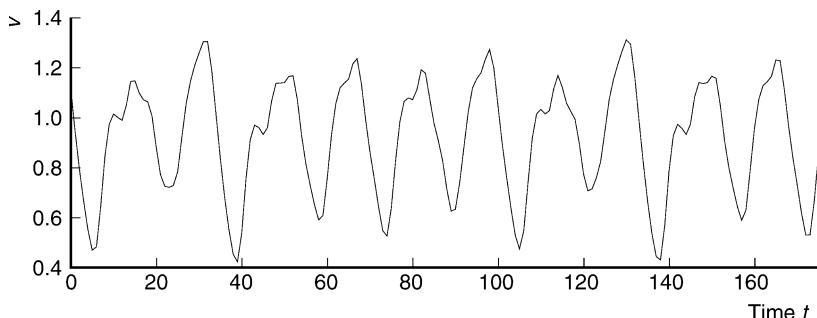
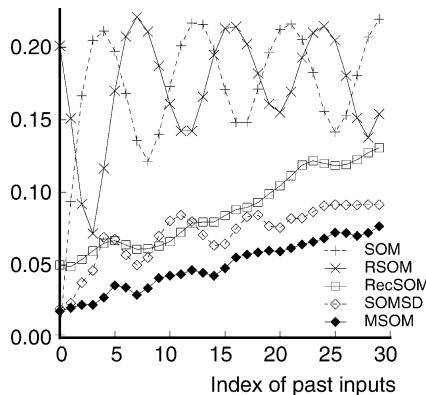


Fig. 18

Temporal quantization error of different algorithms for the Mackey–Glass time series plotted as a function of the past (index = 0 is present). (Strickert and Hammer 2003b, reprinted with permission.)



6.2 Tree Structures

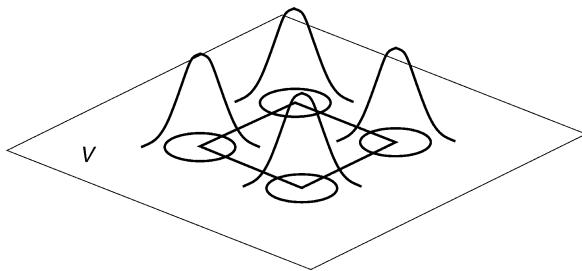
Binary trees, and also trees with limited fan-out k , have been successfully processed with the SOMSD and the MSOM by extending the neuron's single context vector to several context vectors (one for each subtree). Starting from the leafs of a tree, the integrated distance ID of a tree with a given label and the k subtrees can be determined, and the context defined. The usual learning can then be applied to the weights and contexts. As a result of learning, a topographic mapping of trees according to their structure and labels arises. Up to now, only preliminary results of the capacities of these algorithms for tree structures have been obtained.

7 Kernel Topographic Maps

Rather than developing topographic maps with disjoint and uniform activation regions (Voronoi tessellation), such as in the case of the SOM algorithm (☞ Fig. 5), and its adapted versions, algorithms have been introduced that can accommodate neurons with overlapping activation regions, usually in the form of kernel functions, such as Gaussians (☞ Fig. 19). For these *kernel-based topographic maps*, or *kernel topographic maps*, as they are called (they are also sometimes called *probabilistic topographic maps* since they model the input density with a kernel mixture), several learning principles have been proposed (for a review, see Van Hulle (2009)). One motivation to use kernels is to improve, besides the biological relevance, the density estimation properties of topographic maps. In this way, we can combine the unique visualization properties of topographic maps with an improved modeling of clusters in the data. Usually, homoscedastic (equal variance) Gaussian kernels are used, but heteroscedastic (differing variances) Gaussian kernels and other kernel types have also been adopted. The following sections will review the kernel-based topographic map formation algorithms and mention a number of applications. The diversity in algorithms reflects the differences in strategies behind them. As a result, these algorithms have their specific strengths (and weaknesses) and, thus, their own application types.

Fig. 19

Kernel-based topographic map. Example of a 2×2 map (cf. rectangle with thick lines in V -space) for which each neuron has a Gaussian kernel as output function. Normally, a more condensed representation is used where, for each neuron, a circle is drawn with center the neuron weight vector, and radius the kernel range.



7.1 SOM Algorithm Revisited

The starting point is again Kohonen's SOM algorithm. To every neuron a homoscedastic Gaussian kernel is associated with the center corresponding to the neuron's weight vector. Kostiainen and Lampinen (2002) showed that the SOM algorithm can be seen as the equivalent of a maximum likelihood procedure applied to a homoscedastic Gaussian mixture density model, but with the exception that a winner neuron (and, thus, kernel) is selected (the definition of the “winner” i^* [Eq. 1] is equivalent to looking for the Gaussian kernel with the largest output). The position of the winner's kernel is then updated, and possibly also those of other kernels, given the neighborhood function. In a traditional maximum likelihood procedure, there are no winners, and all kernels are updated (Redner and Walker 1984). This means that, for example, for a vanishing neighborhood range, a Gaussian kernel's center is only updated when that neuron is the winner, hence, contrary to the classical case of Gaussian mixture density modeling, the tails of the Gaussian kernels do not lead to center updates (they disappear “under” other kernels), which implies that the kernel radii will be underestimated.

7.2 Elastic Net

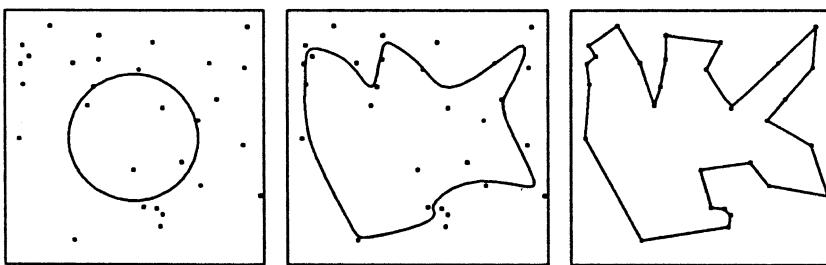
Durbin and Willshaw's (1987) elastic net can be considered as one of the first accounts on kernel-based topographic maps. The elastic net was used for solving the traveling salesman problem (TSP). In the TSP, the objective is to find the shortest, closed tour that visits each city once and that returns to its starting point (e.g., the right panel in Fig. 20). When we represent the location of each city by a point v^μ in the two-dimensional input space $V \subseteq \mathbb{R}^2$, and a tour by a sequence of N neurons – which comprise a ring or closed chain A – then a solution to the TSP can be envisaged as a mapping from V -space onto the neurons of the chain. Evidently, we expect the neuron weights to coincide with the input points (“cities”) at convergence.

The algorithm of the elastic net can be written as follows (in our format):

$$\Delta \mathbf{w}_i = 2\eta \left(\sum_{\mu} \Lambda^{\mu}(i)(v^{\mu} - \mathbf{w}_i) + \kappa(\mathbf{w}_{i+1} - 2\mathbf{w}_i + \mathbf{w}_{i-1}) \right), \quad \forall i \quad (24)$$

Fig. 20

One-dimensional topographic map used for solving the traveling salesman problem. The lattice has a ring topology (closed chain); the points represent cities and are chosen randomly from the input distribution demarcated by the square box. The evolution of the lattice is shown for three time instants, at $t = 0$ (initialization); 7,000; and 10,000 (from left to right). The weights of the lattice at $t = 0$ form a circle positioned at the center of mass of the input distribution. (Reprinted from Ritter and Schulten (1988), ©1988 IEEE.)



where each weight, \mathbf{w}_i , represents a point on the elastic net. The first term on the right-hand side is a force that drags each point \mathbf{w}_i on the chain A toward the cities \mathbf{v}^μ , and the second term is an elastic force that tends to keep neighboring points on the chain close to each other (and thus tends to minimize the overall tour length). The function $\Lambda^\mu(i)$ is a *normalized* Gaussian:

$$\Lambda^\mu(i) = \frac{\exp(-\|\mathbf{v}^\mu - \mathbf{w}_i\|^2/2\sigma_A^2)}{\sum_j \exp(-\|\mathbf{v}^\mu - \mathbf{w}_j\|^2/2\sigma_A^2)} \quad (25)$$

with \mathbf{w}_i the center of the Gaussian and σ_A its range, which is gradually decreased over time (as well as η , and also κ). By virtue of this kernel, the elastic net can be viewed as a homoscedastic Gaussian mixture density model, fitted to the data points by a penalized maximum likelihood term (for a formal account, see Durbin et al. (1989)). The elastic net algorithm looks similar to Kohonen's SOM algorithm except that $\Lambda(i,j)$ has been replaced by $\Lambda^\mu(i)$, and that a second term is added. Interestingly, the SOM algorithm can be used for solving the TSP even without the second term (Ritter et al. 1992), provided we take more neurons in our chain than cities, and that we initialize the weights on a circle (a so-called N -gon) positioned at the center of mass of the input distribution. An example of the convergence process for a 30-city case using a $N = 100$ neuron chain is shown in **Fig. 20**.

The elastic net has been used for finding trajectories of charged particles with multiple scattering in high-energy physics experiments (Gorbunov and Kisel 2006), and for predicting the protein folding structure (Ball et al. 2002). Furthermore, it has been used for clustering applications (Rose et al. 1993). Finally, since it also has a close relationship with “snakes” in computer vision (Kass et al. 1987) (for the connection, see Abrantes and Marques (1995)), the elastic net has also been used for extracting the shape of a closed object from a digital image, such as finding the lung boundaries from magnetic resonance images (Gilson et al. 1997).

7.3 Generative Topographic Map

The generative topographic map (GTM) algorithm (Bishop et al. 1996, 1998) develops a topographic map that attempts to find a representation for the input distribution $p(\mathbf{v})$,

$\mathbf{v} = [v_1, \dots, v_d]$, $\mathbf{v} \in V$, in terms of a number L of latent variables $\mathbf{x} = [x_1, \dots, x_L]$. This is achieved by considering a nonlinear transformation $\mathbf{y}(\mathbf{x}, \mathbf{W})$, governed by a set of parameters \mathbf{W} , which maps points in the latent variable space to the input space, much the same way as the lattice nodes in the SOM relate to positions in V -space (inverse mapping Ψ in [Fig. 2](#)). If one defines a probability distribution, $p(\mathbf{x})$, on the latent variable space, then this will induce a corresponding distribution, $p(\mathbf{y}|\mathbf{W})$, in the input space.

As a specific form of $p(\mathbf{x})$, Bishop and coworkers take a discrete distribution consisting of a sum of delta functions located at the N nodes of a regular lattice:

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i) \quad (26)$$

The dimensionality, L , of the latent variable space is typically less than the dimensionality, d , of the input space so that the transformation, \mathbf{y} , specifies an L -dimensional manifold in V -space. Since $L < d$, the distribution in V -space is confined to this manifold and, hence, is singular. In order to avoid this, Bishop and coworkers introduced a noise model in V -space, namely, a set of radially symmetric Gaussian kernels centered at the positions of the lattice nodes in V -space. The probability distribution in V -space can then be written as follows:

$$p(\mathbf{v}|\mathbf{W}, \sigma) = \frac{1}{N} \sum_{i=1}^N p(\mathbf{v}|\mathbf{x}_i, \mathbf{W}, \sigma) \quad (27)$$

which is a homoscedastic Gaussian mixture model. In fact, this distribution is a *constrained* Gaussian mixture model since the centers of the Gaussians cannot move independently from each other but are related through the transformation, \mathbf{y} . Moreover, when the transformation is smooth and continuous, the centers of the Gaussians will be topographically ordered by construction. Hence, the topographic nature of the map is an intrinsic feature of the latent variable model and is not dependent on the details of the learning process. Finally, the parameters \mathbf{W} and σ are determined by maximizing the log-likelihood:

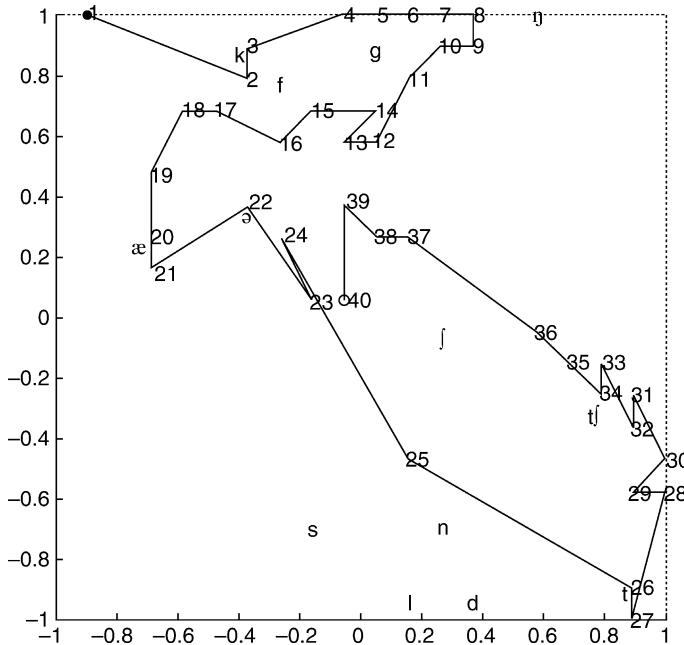
$$\ln \mathcal{L}(\mathbf{W}, \sigma) = \ln \prod_{\mu=1}^M p(\mathbf{v}^\mu|\mathbf{W}, \sigma) \quad (28)$$

and which can be achieved through the use of an expectation-maximization (EM) procedure ([Dempster et al. 1977](#)). Because a single two-dimensional visualization plot may not be sufficient to capture all of the interesting aspects of complex data sets, a hierarchical version of the GTM has also been developed ([Tiňo and Nabney 2002](#)).

The GTM has been applied to visualize oil flows along multiphase pipelines, where the phases are oil, water, and gas, and the flows can be one of the three types: stratified, homogeneous, and annular ([Bishop et al. 1996](#)) ([Fig. 4](#), right panel). It has been applied to visualize electropalatographic (EPG) data for investigating the activity of the tongue in normal and pathological speech ([Carreira-Perpiñán and Renals 1998](#)) ([Fig. 21](#)). It has also been applied to the classification of *in vivo* magnetic resonance spectra of controls and Parkinson's patients ([Axelson et al. 2002](#)), to word grouping in document data sets (using the newsgroup data set benchmark) and the exploratory analysis of web navigation sequences ([Kabán 2005](#)), and to spatiotemporal clustering of transition states of a typhoon from image sequences of cloud patterns ([Kitamoto 2002](#)). In another application, the GTM is used for microarray data analysis (gene expression data) with the purpose of finding low-confidence value genes ([D'Alimonte et al. 2005](#)).

Fig. 21

Visualization of the trajectory in a 20×20 GTM lattice of the activity of the tongue (electropalatographic (EPG) data) of speaker RK for the utterance fragment “I prefer Kant to Hobbes for a good bedtime book.” (Carreira-Perpiñán and Renals 1998, reprinted with permission.)



7.4 Regularized Gaussian Mixture Modeling

Heskes (2001) was able to show the direct correspondence between minimum distortion topographic map formation and maximum likelihood Gaussian mixture density modeling for the homoscedastic case. The starting point was the traditional distortion (vector quantization) formulation of the self-organizing map:

$$F_{\text{quantization}} = \sum_{\mu} \sum_i P(i|\mathbf{v}^{\mu}) \sum_j \Lambda(i,j) \frac{1}{2} \|\mathbf{v}^{\mu} - \mathbf{w}_j\|^2 \quad (29)$$

with $P(i|\mathbf{v}^{\mu})$ the probability that input \mathbf{v}^{μ} is assigned to neuron i with weight \mathbf{w}_i (i.e., the posterior probability, and with $\sum_i P(i|\mathbf{v}^{\mu}) = 1$ and $P(i|\mathbf{v}^{\mu}) \geq 0$). Even if one assigns \mathbf{v}^{μ} to neuron i , there exists a confusion probability $\Lambda(i,j)$ that \mathbf{v}^{μ} is assigned to neuron j . An annealed version of the self-organizing map is obtained if one adds an entropy term:

$$F_{\text{entropy}} = \sum_{\mu} \sum_i P(i|\mathbf{v}^{\mu}) \log \left(\frac{P(i|\mathbf{v}^{\mu})}{Q_i} \right) \quad (30)$$

with Q_i the prior probability (the usual choice is $Q_i = \frac{1}{N}$, with N the number of neurons in the lattice. The final (free) energy is now:

$$F = \beta F_{\text{quantization}} + F_{\text{entropy}} \quad (31)$$

with β playing the role of an inverse temperature. This formulation is very convenient for an EM procedure. The expectation step leads to:

$$P(i|\mathbf{v}^\mu) = \frac{Q_i \exp\left(-\frac{\beta}{2} \sum_j A(i,j) \|\mathbf{v}^\mu - \mathbf{w}_j\|\right)}{\sum_s Q_s \exp\left(-\frac{\beta}{2} \sum_j A(s,j) \|\mathbf{v}^\mu - \mathbf{w}_j\|\right)} \quad (32)$$

and the maximization step to:

$$\mathbf{w}_i = \frac{\sum_\mu \sum_j P(j|\mathbf{v}^\mu) A(j,i) \mathbf{v}^\mu}{\sum_\mu \sum_j P(j|\mathbf{v}^\mu) A(j,i)} \quad (33)$$

which is also the result reached by Graepel and coworkers (1998) for the soft topographic vector quantization (STVQ) algorithm (see the next section). Plugging (Eq. 32) into (Eq. 31) leads to an error function, which allows for the connection with a maximum likelihood procedure, for a mixture of homoscedastic Gaussians, when the neighborhood range vanishes ($A(i,j) = \delta_{ij}$). When the neighborhood is present, Heskes showed that this leads to a term added to the original likelihood.

As an application, Heskes considers market basket analysis. Given are a list of transactions corresponding to the joint set of products purchased by a customer at a given time. The goal of the analysis is to map the products onto a two-dimensional map (lattice) such that neighboring products are “similar.” Similar products should have similar conditional probabilities of buying other products. In another application, he considers the case of transactions in a supermarket. The products are summarized in product groups, and the co-occurrence frequencies are given. The result is a two-dimensional density map showing clusters of products that belong together, for example, a large cluster of household products (Fig. 22).

7.5 Soft Topographic Vector Quantization

Another approach that considers topographic map formation as an optimization problem is the one introduced by Klaus Obermayer and coworkers (Graepel et al. 1997, 1998). They start from the following cost function:

$$E(\mathbf{W}) = \frac{1}{2} \sum_\mu \sum_i c_{\mu,i} \sum_j A(i,j) \|\mathbf{v}^\mu - \mathbf{w}_j\|^2 \quad (34)$$

with $c_{\mu,i} \in \{0, 1\}$ and for which $c_{\mu,i} = 1$ if \mathbf{v}^μ is assigned to neuron i , else $c_{\mu,i} = 0$ ($\sum_i c_{\mu,i} = 1$); the neighborhood function obeys $\sum_j A(i,j) = 1$. The \mathbf{w}_b , $\forall i$, for which this function is minimal, are the optimal ones. However, the optimization is a difficult task, because it depends both on binary and continuous variables and has many local minima. To avoid this, a technique known as deterministic annealing is applied: the optimization is done on a smooth function parametrized by a parameter β , the so-called free energy. When β is small, the function is smooth and only one global minimum remains; when large, more of the structure of the original cost

Fig. 22

Visualization of market basket data in which 199 product groups are clustered based on their co-occurrence frequencies with other products (Heskes (2001), ©2001 IEEE).



function is reflected in the free energy. One starts with a low value of β and attempts to keep track of the minimum through higher values of β .

The application of the principle of maximum entropy yields the free energy (Graepel et al. 1997):

$$F = -\frac{1}{\beta} \log \sum_{c_{\mu,i}} \exp(-\beta E) \quad (35)$$

which leads to probabilistic assignments of inputs \mathbf{v}^μ to neurons, $P(i|\mathbf{v}^\mu)$, $\forall i$, that is, the posterior probabilities, and which are given by:

$$P(i|\mathbf{v}^\mu) = \frac{\exp(-\frac{\beta}{2}\sum_j A(i,j)\|\mathbf{v}^\mu - \mathbf{w}_j\|^2)}{\sum_s \exp(-\frac{\beta}{2}\sum_j A(s,j)\|\mathbf{v}^\mu - \mathbf{w}_j\|^2)} \quad (36)$$

The fixed point rule for the kernel centers is then:

$$\mathbf{w}_i = \frac{\sum_{\mu} \sum_j P(j|\mathbf{v}^{\mu}) A(j,i) \mathbf{v}^{\mu}}{\sum_{\mu} \sum_j P(j|\mathbf{v}^{\mu}) A(j,i)}, \quad \forall i \quad (37)$$

The updates are done through an EM scheme. We observe that the latter equation is identical to Heskes' rule for regularized Gaussian mixture modeling (☞ Eq. 33).

The STVQ has been generalized to the soft topographic mapping for proximity data (STMP), which can be used for clustering categorical data, given a matrix of pairwise proximities or dissimilarities, which is one of the first accounts of this nature in the topographic map literature. A candidate application is the DNA microarray data sets where

the data can be described by matrices with the columns representing tissue samples and the rows genes, and the entries in the matrix correspond to the strength of the gene expression. In Seo and Obermayer (2004), a modified version of the STMP is used for clustering documents (“document map”).

7.6 Heteroscedastic Gaussian Kernel Topographic Map Formation

In the literature, only few approaches exist that consider heteroscedastic kernels, perhaps because the kernel radius in the homoscedastic case is often used in an annealing schedule, as shown above in the STVQ and the elastic net algorithms. When using heteroscedastic kernels, a better density estimate is expected. Several algorithms for heteroscedastic kernels have been developed (for a review, see Van Hulle (2009)). We briefly mention a few here.

Bearing in mind what was said earlier about the SOM in connection to Gaussian mixture modeling, one can extend the original batch map, [Eq. 5](#), to the heteroscedastic case (Van Hulle 2009):

$$\begin{aligned}\mathbf{w}_i &= \frac{\sum_{\mu} \Lambda(i^*, i) \mathbf{v}^{\mu}}{\sum_{\mu} \Lambda(i^*, i)} \\ \sigma_i^2 &= \frac{\sum_{\mu} \Lambda(i^*, i) \|\mathbf{v} - \mathbf{w}_i\|^2 / d}{\sum_{\mu} \Lambda(i^*, i)}, \quad \forall i\end{aligned}\tag{38}$$

with $i^* = \text{argmax}_i K_i$ (which is no longer equivalent to $i^* = \text{argmin}_i \|\mathbf{v} - \mathbf{w}_i\|$, but which is required since we now have heteroscedastic kernels), that is, an *activity-based* definition of “winner-takes-all,” rather than a minimum *Euclidean distance*-based one. Notice again that, by the definition of the winner, the tails of the kernels are cut off, since the kernels overlap.

Recently we introduced (Van Hulle 2005a) a learning algorithm for kernel-based topographic map formation of heteroscedastic Gaussian mixtures that allows for a unified account of distortion error (vector quantization), log-likelihood and Kullback–Leibler divergence, and that generalizes Heskes’ (2001) algorithm to the heteroscedastic case.

There is also the heuristic approach suggested by Yin and Allinson (2001), which is minimizing the Kullback–Leibler divergence, based on an idea introduced by Benaim and Tomasini (1991), for the homoscedastic case. Albeit these authors only suggested an incremental, gradient-based learning procedure (thus, with a learning rate), we can cast their format into a fixed point learning scheme:

$$\begin{aligned}\mathbf{w}_i &= \frac{\sum_{\mu} \Lambda(i^*, i) P(i|\mathbf{v}^{\mu}) \mathbf{v}^{\mu}}{\sum_{\mu} \Lambda(i^*, i) P(i|\mathbf{v}^{\mu})} \\ \sigma_i^2 &= \frac{\sum_{\mu} \Lambda(i^*, i) P(i|\mathbf{v}^{\mu}) \|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2 / d}{\sum_{\mu} \Lambda(i^*, i) P(i|\mathbf{v}^{\mu})}\end{aligned}\tag{39}$$

with the winner neuron defined as $i^* = \text{argmax}_i P(i|\mathbf{v}^{\mu})$, thus, the neuron with the largest posterior probability.

In a still different approach, an input to lattice transformation Φ is considered that admits a kernel function, a Gaussian (Van Hulle 2002a) $\langle \Phi(\mathbf{v}), \Phi(\mathbf{w}_i) \rangle = K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$:

$$K(\mathbf{v}, \mathbf{w}_i, \sigma_i) = \exp\left(-\frac{\|\mathbf{v} - \mathbf{w}_i\|^2}{2\sigma_i^2}\right)\tag{40}$$

When performing topographic map formation, we require that the weight vectors are updated, so as to minimize the expected value of the squared Euclidean distance $\|\mathbf{v} - \mathbf{w}_i\|^2$ and, hence, following our transformation Φ , we instead wish to minimize $\|\Phi(\mathbf{v}) - \Phi(\mathbf{w}_i)\|^2$, which we will achieve by performing gradient descent with respect to \mathbf{w}_i . This leads to the following fixed point rules to which we have added a neighborhood function:

$$\begin{aligned}\mathbf{w}_i &= \frac{\sum_{\mu} A(i, i^*) K(\mathbf{v}^{\mu}, \mathbf{w}_i, \sigma_i) \mathbf{v}^{\mu}}{\sum_{\mu} A(i, i^*) K(\mathbf{v}^{\mu}, \mathbf{w}_i, \sigma_i)} \\ \sigma_i^2 &= \frac{1}{\rho d} \frac{\sum_{\mu} A(i, i^*) K(\mathbf{v}^{\mu}, \mathbf{w}_i, \sigma_i) \|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2}{\sum_{\mu} A(i, i^*) K(\mathbf{v}^{\mu}, \mathbf{w}_i, \sigma_i)}\end{aligned}\quad (41)$$

with ρ a scale factor (a constant) designed to relax the local Gaussian (and d large) assumption in practice, and with $i^* = \arg \max_{i \in A} K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$.

Rather than having a real-valued neural activation, one could also threshold the kernel into a binary variable: in the kernel-based maximum entropy rule (kMER), a neuron, i , is activated by input \mathbf{v} when $\|\mathbf{w}_i - \mathbf{v}\| < \sigma_i$, where σ_i is the kernel radius of neuron i , and which defines a hyperspherical activation region, S_i (Van Hulle 1998). The membership function, $\mathbb{1}_i(\mathbf{v})$, equals unity when neuron i is activated by \mathbf{v} , else it is zero. When there are no neurons active for a given input, the neuron that is positioned closest to that input is defined active. The incremental learning rules for the weights and radii of neuron i are as follows:

$$\begin{aligned}\Delta \mathbf{w}_i &= \eta \sum_j A(i, j) \Xi_i(\mathbf{v}) \text{sign}(\mathbf{v} - \mathbf{w}_i) \\ \Delta \sigma_i &= \eta \left(\frac{\rho_r}{N} (1 - \mathbb{1}_i(\mathbf{v})) - \mathbb{1}_i(\mathbf{v}) \right)\end{aligned}\quad (42)$$

with $\text{sign}(\cdot)$, the sign function taken component-wise, η the learning rate, $\Xi_i(\mathbf{v}) = \frac{\mathbb{1}_i}{\sum_j \mathbb{1}_j}$ a fuzzy membership function, and $\rho_r = \frac{\rho N}{N - \rho}$. It can be shown that the kernel ranges converge to the case where the average probabilities become equal, $\langle \mathbb{1}_i \rangle = \frac{\rho}{N}, \forall i$. By virtue of the latter, kMER is said to generate an equiprobabilistic topographic map (which avoids dead units). The algorithm has been considered for a wide range of applications, such as shape clustering (Van Hulle and Gautama 2004), music signal clustering (Van Hulle 2000), and the linking of patent and scientific publication databases (Deleus and Van Hulle 2001). More recently, a fixed point version, called batch map kMER, was introduced (Gautama and Van Hulle 2006), and applied to handwritten numerals clustering.

7.7 Kernels Other Than Gaussians

In principle, kernels other than Gaussians could be used in topographic map formation. For example, Heskes pointed out that his regularized mixture modeling approach could, in principle, accommodate any kernel of the exponential family, such as the gamma, multinomial, and the Poisson distributions (Heskes 2001).

In another case, the kernel is considered for which the differential entropy of the kernel output will be maximal given a Gaussian input, that is, the incomplete gamma distribution kernel (Van Hulle 2002b).

Another type of kernel is the Edgeworth-expanded Gaussian kernel, which consists of a Gaussian kernel multiplied by a series of Hermite polynomials of increasing order, and of which the coefficients are specified by (the second- and higher-order) cumulants (Van Hulle 2005b).

In still another case, a mixture of Bernoulli distributions is taken (Verbeek et al. 2005) for the specific purpose to better encode binary data (e.g., word occurrence in a document). This also leads to an EM algorithm for updating the posteriors, as well as the expected joint log-likelihood with respect to the parameters of the Bernoulli distributions. However, as the posteriors become quite peaked for higher dimensions, for visualization purposes, a power function of them was chosen. Several applications have been demonstrated, including word grouping in document data sets (newsgroup data set) and credit data analysis (from the University of California, Irvine (UCI) machine learning repository <http://archive.ics.uci.edu/ml/>).

7.8 Future Developments

An expected development is to go beyond the limitation of the current kernel-based topographic maps that the inputs need to be vectors (we already saw the extension toward categorical data). But in the area of structural pattern recognition, more powerful data structures can be processed, such as strings, trees, and graphs. The SOM algorithm has already been extended toward strings (Kohonen and Somervuo 1998) and graphs, which include strings and trees (Günter and Bunke 2002; Seo and Obermayer 2004; Steil and Sperduti 2007) (see also the SOMSD and the MSOM algorithms above). However, new types of *kernels* for strings, trees, and graphs have been suggested in the support vector machine literature (thus, outside the topographic map literature) (for reviews, see Shawe-Taylor and Cristianini (2004) and Jin et al. (2005)). The integration of these new types of kernels into kernel-based topographic maps is yet to be done, but could turn out to be a promising evolution for biochemical applications, such as visualizing and clustering sets of structure-based molecule descriptions.

8 Conclusion

In this chapter we introduced the self-organizing map (SOM) algorithm, discussed its properties, limitations, and types of application, and reviewed a number of extensions and other types of topographic map formation algorithms, such as the growing, the recurrent, and the kernel topographic maps. We also indicated how recent developments in topographic maps enable us to consider categorical data, time series, and tree-structured data, widening further the application field toward microarray data analysis, document analysis and retrieval, exploratory analysis of web navigation sequences, and the visualization of protein structures and long DNA sequences.

Acknowledgments

The author is supported by the Excellence Financing program (EF 2005) and the CREA Financing program (CREA/07/027) of K.U.Leuven, the Belgian Fund for Scientific Research – Flanders (G.0234.04 and G.0588.09), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P6/29), and the European Commission (NEST-2003-012963, STREP-2002-016276, IST-2004-027017, and ICT-2007-217077).

References

- Abrantes AJ, Marques JS (1995) Unified approach to snakes, elastic nets, and Kohonen maps. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'95). IEEE Computer Society, Los Alamitos, CA, vol 5, pp 3427–3430
- Ahalt SC, Krishnamurthy AK, Chen P, Melton DE (1990) Competitive learning algorithms for vector quantization. *Neural Netw* 3:277–290
- Alahakoon D, Halgamuge SK, Srinivasan B (2000) Dynamic self organising maps with controlled growth for knowledge discovery. *IEEE Trans Neural Netw* (Special issue on knowledge discovery and data mining) 11(3):601–614
- Axelson D, Bakken IJ, Gribbestad IS, Ehrnholm B, Nilsen G, Aasly J (2002) Applications of neural network analyses to *in vivo* ¹H magnetic resonance spectroscopy of Parkinson disease patients. *J Magn Reson Imaging* 16(1):13–20
- Ball KD, Erman B, Dill KA (2002) The elastic net algorithm and protein structure prediction. *J Comput Chem* 23(1):77–83
- Barreto G, Araújo A (2001) Time in self-organizing maps: an overview of models. *Int J Comput Res* 10(2):139–179
- Bauer H-U, Villmann T (1997) Growing a hypercubical output space in a self-organizing feature map. *IEEE Trans Neural Netw* 8(2):218–226
- Bauer H-U, Der R, Herrmann M (1996) Controlling the magnification factor of self-organizing feature maps. *Neural Comput* 8:757–771
- Benaim M, Tomasini L (1991) Competitive and self-organizing algorithms based on the minimization of an information criterion. In: Proceedings of 1991 international conference in artificial neural networks (ICANN'91). Espoo, Finland. Elsevier Science Publishers, North-Holland, pp 391–396
- Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
- Bishop CM, Svensén M, Williams CKI (1996) GTM: a principled alternative to the self-organizing map. In: Proceedings 1996 International Conference on Artificial Neural Networks (ICANN'96). Bochum, Germany, 16–19 July 1996. Lecture notes in computer science, vol 1112. Springer, pp 165–170
- Bishop CM, Hinton GE, and Strachan IGD (1997) In: Proceedings IEE fifth international conference on artificial neural networks. Cambridge UK, 7–9 July 1997, pp 111–116
- Bishop CM, Svensén M, Williams CKI (1998) GTM: the generative topographic mapping. *Neural Comput* 10:215–234
- Blackmore J, Miikkulainen R (1993) Incremental grid growing: encoding high-dimensional structure into a two-dimensional feature map. In: Proceedings of IEEE international conference on neural networks. San Francisco, CA. IEEE Press, Piscataway, NJ, vol 1, pp 450–455
- Bruske J, Sommer G (1995) Dynamic cell structure learns perfectly topology preserving map. *Neural Comput* 7(4):845–865
- Carreira-Perpiñán MÁ, Renals S (1998) Dimensionality reduction of electropalatographic data using latent variable models. *Speech Commun* 26(4):259–282
- Centre NNR (2003) Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ), Helsinki University of Technology. <http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html>
- Chappell G, Taylor J (1993) The temporal Kohonen map. *Neural Netw* 6:441–445
- Chinrungrueng C, Séquin CH (1995) Optimal adaptive k -means algorithm with dynamic adjustment of learning rate. *IEEE Trans Neural Netw* 6:157–169
- Cottrell M, Fort JC (1987) Etude d'un processus d'auto-organisation. *Ann Inst Henri Poincaré* 23:1–20
- D'Alimonte D, Lowe D, Nabney IT, Sivaraksa M (2005) Visualising uncertain data. In: Proceedings European conference on emergent aspects in clinical data analysis (EACDA2005). Pisa, Italy, 28–30 September 2005. <http://ciml.di.unipi.it/EACDA2005/papers.html>
- Deleus FF, Van Hulle MM (2001) Science and technology interactions discovered with a new topographic map-based visualization tool. In: Proceedings of 7th ACM SIGKDD international conference on knowledge discovery in data mining. San Francisco, 26–29 August 2001. ACM Press, New York, pp 42–50
- Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood for incomplete data via the EM algorithm. *J R Stat Soc B* 39:1–38
- Der R, Herrmann M (1993) Phase transitions in self-organizing feature maps. In: Proceedings of 1993 international conference on artificial neuron networks (ICANN'93). Amsterdam, The Netherlands, 13–16 September 1993, Springer, New York, pp 597–600
- DeSieno D (1988) Adding a conscience to competitive learning. In: Proceedings of IEEE international conference on neural networks. San Diego, CA, IEEE Press, New York, vol I, pp 117–124
- Durbin R, Willshaw D (1987) An analogue approach to the travelling salesman problem using an elastic net method. *Nature* 326:689–691

- Durbin R, Szeliski R, Yuille AL (1989) An analysis of the elastic net approach to the traveling salesman problem. *Neural Comput* 1:348–358
- Erwin E, Obermayer K, Schulten K (1992) Self-organizing maps: ordering, convergence properties and energy functions. *Biol Cybern* 67:47–55
- Eluliano NR, Principe JC (1999) A spatiotemporal memory based on SOMs with activity diffusion. In: Oja E, Kaski S (eds) Kohonen maps. Elsevier, Amsterdam, The Netherlands, pp 253–266
- Fritzke B (1994) Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Netw* 7(9):1441–1460
- Fritzke B (1995a) A growing neural gas network learns topologies. In: Tesauro G, Touretzky DS, Leen TK (eds) Advances in neural information proceedings systems 7 (NIPS 1994). MIT Press, Cambridge, MA, pp 625–632
- Fritzke B (1995b) Growing grid – a self-organizing network with constant neighborhood range and adaptation strength. *Neural Process Lett* 2(5):9–13
- Fritzke B (1996) Growing self-organizing networks – why? In: European symposium on artificial neural networks (ESANN96). Bruges, Belgium, 1996. D Facto Publications, Brussels, Belgium, pp 61–72
- Gautama T, Van Hulle MM (2006) Batch map extensions of the kernel-based maximum entropy learning rule. *IEEE Trans Neural Netw* 16(2):529–532
- Gersho A, Gray RM (1991) Vector quantization and signal compression. Kluwer, Boston, MA/Dordrecht
- Gesztí T (1990) Physical models of neural networks. World Scientific Press, Singapore
- Gilson SJ, Middleton I, Damper RI (1997) A localised elastic net technique for lung boundary extraction from magnetic resonance images. In: Proceedings of fifth international conference on artificial neural networks. Cambridge, UK, 7–9 July 1997. Maccarenhas Publishing, Stevenage, UK, pp 199–204
- Gorbunov S, Kisel I (2006) Elastic net for stand-alone RICH ring finding. *Nucl Instrum Methods Phys Res A* 559:139–142
- Graepel T, Burger M, Obermayer K (1997) Phase transitions in stochastic self-organizing maps. *Phys Rev E* 56(4):3876–3890
- Graepel T, Burger M, Obermayer K (1998) Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing* 21:173–190
- Grossberg S (1976) Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biol Cybern* 23:121–134
- Günter S, Bunke H (2002) Self-organizing map for clustering in the graph domain. *Pattern Recog Lett* 23:415–417
- Hagenbuchner M, Sperduti A, Tsoi AC (2003) A self-organizing map for adaptive processing of structured data. *IEEE Trans Neural Netw* 14(3):491–505
- Hammer B, Micheli A, Strickert M, Sperduti A (2004) A general framework for unsupervised processing of structured data. *Neurocomputing* 57:3–35
- Hammer B, Micheli A, Neubauer N, Sperduti A, Strickert M (2005) Self organizing maps for time series. In: Proceedings of WSOM 2005. Paris, France, 5–8 September 2005, pp 115–122
- Heskes T (2001) Self-organizing maps, vector quantization, and mixture modeling. *IEEE Trans Neural Netw* 12(6):1299–1305
- Heskes TM, Kappen B (1993) Error potentials for self-organization. In: Proceedings of IEEE international conference on neural networks. San Francisco, CA. IEEE Press, Piscataway, NJ, pp 1219–1223
- Jin B, Zhang Y-Q, Wang B (2005) Evolutionary granular kernel trees and applications in drug activity comparisons, In: Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'05). San Diego, CA, 14–15 November 2005, IEEE Press, Piscataway, NY, pp 1–6
- Kabán A (2005) A scalable generative topographic mapping for sparse data sequences. In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05). Las Vegas, NV, 4–6 April 2005. IEEE Computer Society, vol 1, pp 51–56
- Kass M, Witkin A, Terzopoulos D (1987) Active contour models. *Int J Comput Vis* 1(4):321–331
- Kangas J (1990) Time-delayed self-organizing maps. In: Proceedings IEEE/INNS international Joint Conference on neural networks 1990. San Diego, CA, IEEE, New York, vol 2, pp 331–336
- Kaski S, Honkela T, Lagus K, Kohonen T (1998) WEB-SOM – self-organizing maps of document collections. *Neurocomputing* 21:101–117
- Kim YK, Ra JB (1995) Adaptive learning method in self-organizing map for edge preserving vector quantization. *IEEE Trans Neural Netw* 6:278–280
- Kitamoto A (2002) Evolution map: modeling state transition of typhoon image sequences by spatio-temporal clustering. *Lect Notes Comput Sci* 2534/2002: 283–290
- Kohonen T (1982) Self-organized formation of topologically correct feature maps. *Biol Cybern* 43:59–69
- Kohonen T (1984) Self-organization and associative memory. Springer, Heidelberg
- Kohonen T (1991) Self-organizing maps: optimization approaches. In: Kohonen T, Mäkisara K, Simula O, Kangas J (eds) Artificial neural networks. North-Holland, Amsterdam, pp 981–990
- Kohonen T (1995) Self-organizing maps, 2nd edn. Springer, Heidelberg

- Kohonen T (1997) Self-organizing maps. Springer
- Kohonen T, Somervuo P (1998) Self-organizing maps on symbol strings. *Neurocomputing* 21:19–30
- Kohonen T, Kaski S, Salojärvi J, Honkela J, Paatero V, Saarela A (1999) Self organization of a massive document collection. *IEEE Trans Neural Netw* 11(3): 574–585
- Koskela T, Varsta M, Heikkonen J, Kaski K (1998) Recurrent SOM with local linear models in time series prediction. In: Verleysen M (ed) Proceedings of 6th European symposium on artificial neural networks (ESANN 1998). Bruges, Belgium, April 22–24, 1998. D-Facto, Brussels, Belgium, pp 167–172
- Kostainen T, Lampinen J (2002) Generative probability density model in the self-organizing map. In: Seiffert U, Jain L (eds) Self-organizing neural networks: Recent advances and applications. Physica-Verlag, Heidelberg, pp 75–94
- Laaksonen J, Koskela M, Oja E (2002) PicSOM—self-organizing image retrieval with MPEG-7 content descriptors. *IEEE Trans Neural Netw* 13(4):841–853
- Lin JK, Grier DG, Cowan JD (1997) Faithful representation of separable distributions. *Neural Comput* 9:1305–1320
- Linsker R (1988) Self-organization in a perceptual network. *Computer* 21:105–117
- Linsker R (1989) How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Comput* 1:402–411
- Luttrell SP (1989) Self-organization: a derivation from first principles of a class of learning algorithms. In: Proceedings IEEE international joint conference on neural networks (IJCNN89). Washington, DC, Part I, IEEE Press, Piscataway, NJ, pp 495–498
- Luttrell SP (1990) Derivation of a class of training algorithms. *IEEE Trans Neural Netw* 1:229–232
- Luttrell SP (1991) Code vector density in topographic mappings: scalar case. *IEEE Trans Neural Netw* 2:427–436
- Martinetz TM (1993) Competitive Hebbian learning rule forms perfectly topology preserving maps. In: Proceedings of international conference on artificial neural networks (ICANN93). Amsterdam, The Netherlands, 13–16 September 1993. Springer, London, pp 427–434
- Martinetz T, Schulten K (1991) A “neural-gas” network learns topologies. In: Kohonen T, Mäkisara K, Simula O, Kangas J (eds) Proceedings of International Conference on Artificial Neural Networks (ICANN-91). Espoo, Finland, 24–28 June 1991, vol I, North-Holland, Amsterdam, The Netherlands, pp 397–402
- Martinetz T, Berkovich S, Schulten K (1993) “Neural-gas” network for vector quantization and its application to time-series prediction. *IEEE Trans Neural Netw* 4(4):558–569
- Merkl D, He S, Dittenbach M, Rauber A (2003) Adaptive hierarchical incremental grid growing: an architecture for high-dimensional data visualization. In: Proceedings of 4th workshop on self-organizing maps (WSOM03). Kitakyushu, Japan, 11–14 September 2003
- Mulier F, Cherkassky V (1995) Self-organization as an iterative kernel smoothing process. *Neural Comput* 7:1165–1177
- Rauber A, Merkl D, Dittenbach M (2002) The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Trans Neural Netw* 13(6):1331–1341
- Redner RA, Walker HF (1984) Mixture densities, maximum likelihood and the EM algorithm. *SIAM Rev* 26(2):195–239
- Risi S, Mörschen F, Ultsch A, Lewark P (2007) Visual mining in music collections with emergent SOM. In: Proceedings of workshop on self-organizing maps (WSOM ’07). Bielefeld, Germany, September 3–6, 2007, ISBN: 978-3-00-022473-7, CD ROM, available online at <http://biecoll.ub.uni-bielefeld.de>
- Ritter H (1991) Asymptotic level density for a class of vector quantization processes. *IEEE Trans Neural Netw* 2(1):173–175
- Ritter H (1998) Self-organizing maps in non-Euclidean spaces. In: Oja E, Kaski S (eds) Kohonen maps. Elsevier, Amsterdam, pp 97–108
- Ritter H, Kohonen T (1989) Self-organizing semantic maps. *Biol Cybern* 61:241–254
- Ritter H, Schulten K (1986) On the stationary state of Kohonen’s self-organizing sensory mapping. *Biol Cybern* 54:99–106
- Ritter H, Schulten K (1988) Kohonen’s self-organizing maps: exploring their computational capabilities. In: Proceedings of IEEE international conference on neural networks (ICNN). San Diego, CA. IEEE, New York, vol I, pp 109–116
- Ritter H, Martinetz T, Schulten K (1992) Neural computation and self-organizing maps: an introduction. Addison-Wesley, Reading, MA
- Rose K, Gurewitz E, Fox GC (1993) Constrained clustering as an optimization method. *IEEE Trans Pattern Anal Mach Intell* 15(8):785–794
- Schulz R, Reggia JA (2004) Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps. *Neural Comput* 16(3): 535–561
- Seo S, Obermayer K (2004) Self-organizing maps and clustering methods for matrix data. *Neural Netw* 17(8–9):1211–1229
- Shawe-Taylor J, Cristianini N (2004) Kernel methods in computational biology. MIT Press, Cambridge, MA
- Simon G, Lendasse A, Cottrell M, Fort J-C, Verleysen M (2003) Double SOM for long-term time series

- prediction. In: Proceedings of the workshop on self-organizing maps (WSOM 2003). Hibikino, Japan, September 11–14, 2003, pp 35–40
- Somervuo PJ (2004) Online algorithm for the self-organizing map of symbol strings. *Neural Netw* 17(8–9):1231–1240
- Steil JJ, Sperduti A (2007) Indices to evaluate self-organizing maps for structures. In: Proceedings of the workshop on self-organizing maps (WSOM07) Bielefeld, Germany, 3–6 September 2007. CD ROM, 2007, available online at <http://biecoll.ub.uni-bielefeld.de>
- Strickert M, Hammer B (2003a) Unsupervised recursive sequence processing. In: Verleysen M (ed) European Symposium on Artificial Neural Networks (ESANN 2003). Bruges, Belgium, 23–25 April 2003. D-Side Publications, Evere, Belgium, pp 27–32
- Strickert M, Hammer B (2003b) Neural gas for sequences. In: Proceedings of the workshop on self-organizing maps (WSOM'03). Hibikino, Japan, September 2003, pp 53–57
- Strickert M, Hammer B (2005) Merge SOM for temporal data. *Neurocomputing* 64:39–72
- Tiňo P, Nabney I (2002) Hierarchical GTM: constructing localized non-linear projection manifolds in a principled way. *IEEE Trans Pattern Anal Mach Intell* 24(5):639–656
- Tiňo P, Kabán A, Sun Y (2004) A generative probabilistic approach to visualizing sets of symbolic sequences. In: Kohavi R, Gehrke J, DuMouchel W, Ghosh J (eds) Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2004), Seattle, WA, 22–25 August 2004. ACM Press, New York, pp 701–706
- Tolat V (1990) An analysis of Kohonen's self-organizing maps using a system of energy functions. *Biol Cybern* 64:155–164
- Ultsch A, Siemon HP (1990) Kohonen's self organizing feature maps for exploratory data analysis. In: Proceedings international neural networks. Kluwer, Paris, pp 305–308
- Ultsch A, Mörschen F (2005) ESOM-Maps: Tools for clustering, visualization, and classification with emergent SOM. Technical Report No. 46, Department of Mathematics and Computer Science, University of Marburg, Germany
- Ueda N, Nakano R (1993) A new learning approach based on equidistortion principle for optimal vector quantizer design. In: Proceedings of IEEE NNNSP93, Linthicum Heights, MD. IEEE, Piscataway, NJ, pp 362–371
- Van den Bout DE, Miller TK III (1989) TInMANN: the integer Markovian artificial neural network. In: Proceedings of international joint conference on neural networks (IJCNN89). Washington, DC, 18–22 June 1989, Erlbaum, Englewood Cliffs, NJ, pp II205–II211
- Van Hulle MM (1997a) Topology-preserving map formation achieved with a purely local unsupervised competitive learning rule. *Neural Netw* 10(3): 431–446
- Van Hulle MM (1997b) Nonparametric density estimation and regression achieved with topographic maps maximizing the information-theoretic entropy of their outputs. *Biol Cybern* 77:49–61
- Van Hulle MM (1998) Kernel-based equiprobabilistic topographic map formation. *Neural Comput* 10(7):1847–1871
- Van Hulle MM (2000) Faithful representations and topographic maps: from distortion- to information-based self-organization. Wiley, New York
- Van Hulle MM (2002a) Kernel-based topographic map formation by local density modeling. *Neural Comput* 14(7):1561–1573
- Van Hulle MM (2002b) Joint entropy maximization in kernel-based topographic maps. *Neural Comput* 14(8):1887–1906
- Van Hulle MM (2005a) Maximum likelihood topographic map formation. *Neural Comput* 17(3):503–513
- Van Hulle MM (2005b) Edgeworth-expanded topographic map formation. In: Proceedings of workshop on self-organizing maps (WSOM05). Paris, France, 5–8 September 2005, pp 719–724
- Van Hulle MM (2009) Kernel-based topographic maps: theory and applications. In: Wah BW (ed) Encyclopedia of computer science and engineering. Wiley, Hoboken, vol 3, pp 1633–1650
- Van Hulle MM, Gautama T (2004) Optimal smoothing of kernel-based topographic maps with application to density-based clustering of shapes. *J VLSI Signal Proces Syst Signal, Image, Video Technol* 37:211–222
- Verbeek JJ, Vlassis N, Kröse BJA (2005) Self-organizing mixture models. *Neurocomputing* 63:99–123
- Vesanto J (1997) Using the SOM and local models in time-series prediction. In: Proceedings of workshop on self-organizing maps (WSOM 1997). Helsinki, Finland, 4–6 June 1997. Helsinki University of Technology, Espoo, Finland, pp 209–214
- Voegtlin T (2002) Recursive self-organizing maps. *Neural Netw* 15(8–9):979–992
- Wiemer JC (2003) The time-organized map algorithm: extending the self-organizing map to spatiotemporal signals. *Neural Comput* 15(5):1143–1171
- Willshaw DJ, von der Malsburg C (1976) How patterned neural connections can be set up by self-organization. *Proc Roy Soc Lond B* 194:431–445
- Yin H, Allinson NM (2001) Self-organizing mixture networks for probability density estimation. *IEEE Trans Neural Netw* 12:405–411

Section III

Evolutionary Computation

Thomas Bäck

20 Generalized Evolutionary Algorithms

Kenneth De Jong

Department of Computer Science, George Mason University,
Fairfax, VA, USA
kdejong@gmu.edu

1	<i>Introduction</i>	626
2	<i>Simple Evolutionary Algorithms</i>	626
3	<i>Example Application Areas</i>	631
4	<i>More Complex Evolutionary Algorithms</i>	632
5	<i>Summary and Conclusions</i>	634

Abstract

People have been inventing and tinkering with various forms of evolutionary algorithms (EAs) since the 1950s when digital computers became more readily available to scientists and engineers. Today we see a wide variety of EAs and an impressive array of applications. This diversity is both a blessing and a curse. It serves as strong evidence for the usefulness of these techniques, but makes it difficult to see “the big picture” and make decisions regarding which EAs are best suited for new application areas. The purpose of this chapter is to provide a broader “generalized” perspective.

1 Introduction

Some of the earliest examples of evolutionary algorithms (EAs) appeared in the literature in the 1950s (see, e.g., Box 1957 or Friedberg 1959). Readers interested in more details of this early history are encouraged to see the excellent summary in Fogel (1998). However, three developments that took place in the 1960s have had the most direct and lasting impact on the field today. At the Technical University of Berlin, Ingo Rechenberg and Hans-Paul Schwefel developed an algorithm they called an evolution strategy (ES) in order to solve difficult real-valued parameter optimization problems. At the same time, Larry Fogel and his colleagues at UCLA developed a technique they called evolutionary programming (EP) to automate the process of constructing finite state machines that controlled the behavior of intelligent agents. Independently, John Holland at the University of Michigan developed a class of techniques called genetic algorithms (GAs) for designing robust adaptive systems. Readers interested in more details of these important historical developments are encouraged to consult one or more of the readily available books such as Fogel (1995), Schwefel (1995), and Holland (1975).

Historically, there was little interaction among these groups until the early 1990s. By then each group had developed their own notation and nomenclature. As the interactions became more frequent, it became clear that there was a need for a more general perspective in which GAs, EP, and ESs were seen as important EA instances of a unified “evolutionary computation” (EC) framework (see, e.g., Bäck 1996).

The influence of these historical developments cannot be understated. Many of the EAs used today are variants of one of the historical forms (ES, EP, or GA). At the same time, we are the beneficiaries today of 50 years of exploring, understanding, and extending these ideas. The result is a pretty clear picture of “simple” EAs and a sense of how best to extend and apply them, as indicated in the following sections.

2 Simple Evolutionary Algorithms

The word *evolution* has several meanings. It is used in a general sense to describe a process of incremental change such as “evolving” medical procedures. When we use the term *evolutionary algorithm*, we have a more specific meaning in mind, namely, a Darwinian-like evolutionary process that involves a population of individuals dynamically changing over time with some sort of “survival-of-the-fittest” selection pressure. One might imagine algorithms of this sort whose purpose is to model an existing evolutionary system and explore possible perturbations to such systems. In this case, there is a need for model fidelity to

capture the details of particular evolutionary systems. However, as noted in [Sect. 1](#), the historical motivation for developing EAs was to harness their intrinsic adaptive capabilities to solve difficult computational problems. As such, these EAs are *inspired* by the features of natural systems, but not constrained by them. These algorithms solve problems in the following way:

- They maintain a population of individuals that represent possible solutions to a problem.
- Individuals have a notion of “fitness” associated with them representing the quality of the solution they represent. That fitness is used to bias the population dynamics (i.e., the birth/death cycle).
- Individuals are selected to be parents and produce “offspring,” that is, new solutions that are similar but not identical to the parents.
- Individuals are selected to die (i.e., be deleted from the population).

If one of these EAs is run for several generations (i.e., several birth/death cycles), the fitness-biased selection processes produce a steady increase in the fitness of the population, conveying a strong sense of fitness optimization. This observation leads to the following general template for a simple evolutionary optimization algorithm:

Randomly generate an initial population of size M .

Do until some stopping criterion is met:

 Select some parents to produce some offspring.

 Select some individuals to die.

End Do

Return as the problem solution, the individual with the highest observed fitness.

Of course, this general template is not computationally specific enough to be coded and executed. In order to instantiate this EA template for a particular class of problems, we must answer the following questions:

- How are individuals (problem solutions) represented in the population?
- How big should the population be?
- How are parents selected?
- How are offspring produced from selected parents?
- How many offspring should be produced?
- How are the survivors chosen?

The fact that there is more than one answer to each of these questions was already evident in the 1960s when ES, EP, and GA variants were being developed. As an example, consider parameter optimization problems. A traditional ES approach would be to represent solutions internally as n -dimensional vectors of parameter values and maintain a small parent population (e.g., $M < 10$). Each member of the parent population produces one (or more) offspring by cloning the parent and “mutating” some of the offspring’s parameter values via a Gaussian perturbation operator. The combined parent and offspring populations are sorted by fitness and only the top M individuals survive to the next generation.

By contrast, a traditional GA approach would convert the n -dimensional parameter vectors into an internal binary string representation and maintain a population of moderate size (e.g., $M \in [50, 100]$). Pairs of parents are selected stochastically using a fitness-proportional selection mechanism to produce offspring that inherit combinations of binary substrings

from their parents and a small number of mutations (bit flips). Exactly M offspring are produced. They all survive and the parents all die.

Both of these traditional models were inspired by natural systems, but clearly behave differently as computational procedures for solving search and optimization problems. It is known from various “no-free-lunch” theorems that efficient algorithms incorporate a bias that allows them to exploit the structure inherent in a particular class of problems, and that the same bias is likely to be a hindrance on other problem classes with different structures. This perspective suggests a general strategy for answering the questions listed above: make the choices so as to tune the EA bias to the particular class of problems of interest. That is, view the EA template above as a “meta-heuristic” to be instantiated in problem-specific ways.

This, in turn, requires an understanding of how these EA design decisions affect its problem-solving bias. These issues are explored in the following sections.

2.1 Internal Representation

Choices for the internal representation of problem solutions fall into two biologically inspired categories: genotypic representations and phenotypic representations. Genotypic representations encode problem solutions internally in a manner analogous to the biological genetic code, that is, solutions are represented as strings formed from a universal alphabet (e.g., binary strings in traditional GAs). Reproductive operators are defined to manipulate the universal encodings, significantly increasing the portability of the EA code from one application to the next. Applying such EAs to a new problem class simply requires writing the encoding/decoding procedures.

By contrast, phenotypic representations represent problem solutions directly with no intermediate encoding (e.g., parameter vectors in ESs). Reproductive operators are defined to manipulate the solutions directly. This provides the opportunity to introduce problem-specific biases into the EA, but does imply significant redesign and re-implementation of the reproductive operators.

To illustrate these issues, consider how one might apply an EA to traveling salesperson problems. A GA practitioner would focus on designing a mapping of permutation spaces to/from binary strings, leaving the reproductive operators unchanged. An ES practitioner would focus on replacing the standard Gaussian mutation operator with something more appropriate for permutation spaces.

2.1.1 Reproductive Operators

The most important property of reproductive operators is their ability to use existing solutions as a springboard for creating new and interesting solutions. From a biological perspective, this is the notion of heritability and is achieved in two rather distinct ways: asexual and sexual reproduction. Asexual reproduction involves cloning single parents and then applying a mutation operator to provide some variability. This is the strategy used in traditional ES and EP approaches. Sexual reproduction involves combining elements of more than one parent (generally with a small dose of mutation) to produce offspring that inherit some features from each parent. This is the strategy used in traditional GAs in which offspring are produced from two parents (as is usual in natural systems). However, there is no *a priori* computational reason to limit this to two parents (see, for example, Eiben et al. 1994).

From a search perspective, these two reproductive strategies differ significantly. Asexual reproduction tends to be more of a local search operator producing offspring in a nearby neighborhood of their parents. A key element in the effectiveness of asexual reproduction is the ability to dynamically control the expected distance between parents and offspring, tuning this step size to the properties of the fitness landscape (e.g., the “1:5” step size adaptation rule in traditional ESSs (Schwefel 1995), or more recent “covariance matrix” approaches (Hansen and Ostermeier 2001)).

By contrast, sexual reproduction tends to be more of a global search operator producing offspring that are frequently quite far from their parents. A key element in the effectiveness of sexual reproduction is the ability to recombine modular subcomponents in useful ways. For example, in parameter optimization problems, inheriting coupled (epistatic) values is highly desirable, but generally not known *a priori*. An interesting property of many of the recombination operators in use is that they intrinsically adapt to such couplings, in that, interactions that improve fitness spread throughout the population and thus are more likely to be inherited.

Both sexual and asexual reproductive operators complement each other nicely producing a blend of local and global search. As a consequence, many EAs today use a combination of both.

2.2 Selection Mechanisms

In simple EAs, there are two opportunities for selection: when choosing parents and when determining survivors. Both are opportunities to use fitness information to bias the search. As a first thought, one might consider selecting only the best individuals. But, one must exercise care here since this introduces a certain amount of “greediness” into the search process. In computer science, greedy algorithms are a well-studied class of procedures that have the general property of rapid convergence, but not necessarily to the best solution. For some problem domains and fitness landscapes, rapid convergence to a nearby local optimum is more than adequate. For others, a slower, more diffuse search is required to avoid getting immediately trapped in a local optimum.

So, a key aspect to EA design is determining the right amount of selection pressure (greediness) for a particular application. This is accomplished by making several interrelated observations. The first is that having fitness bias for *both* parent selection and survival selection invariably produces too much greediness and results in rapid convergence to suboptimal solutions. As a consequence, most EAs use fitness-biased selection for only one of the two selection steps, leaving the other one unbiased. To introduce fitness bias, we have a well-studied collection of selection mechanisms to choose from, which is listed here in order of decreasing selection pressure:

- Truncation selection: choose only the N most fit individuals.
- Tournament selection: (repeatedly) choose K individuals at random and keep only the best of the K selected.
- Rank-proportional selection: choose individuals proportional to their fitness rank in the population.
- Fitness-proportional selection: choose individuals proportional to the value of their fitness.

Tournament and fitness-proportional selection have analogs in naturally evolving biological systems. Truncation and rank selection are similar to strategies used in breeding farms. All

have been formally analyzed regarding the degree of selection pressure they induce (see, for example, De Jong 2006).

One can generally choose whether to implement these selection mechanisms as deterministic or stochastic procedures. There is a growing amount of evidence that the stochastic versions converge a bit more slowly but are less likely to get stuck in suboptimal areas (De Jong 2006).

A related observation has to do with whether or not parents and offspring compete with each other for survival. Mathematical models of evolutionary systems frequently adopt a “nonoverlapping” generation assumption: only the offspring survive. This is done primarily to make the mathematical analysis simpler. From a computer science perspective, the choice affects the greediness of an EA. Overlapping generations produce greedier search behavior than nonoverlapping populations. This is why traditional ES procedures come in two forms: “plus” (overlapping) and “comma” (nonoverlapping) strategies (Schwefel 1995), and why there are both nonoverlapping and “steady state” GAs (De Jong 2006).

The final observation that is important to keep in mind when choosing a selection mechanism is the need to achieve an effective balance between exploration and exploitation. In simple EAs, exploration is encouraged by having weaker selection pressure and more reproductive variation, and the converse for exploitation. Effective balance can be achieved in a variety of ways as illustrated by the traditional EAs. Traditional ESs typically have strong survival selection (overlapping generations with truncation selection) and balance that off with a fairly aggressive mutation operator. By contrast, traditional GAs have much weaker selection pressure (fitness-proportional parent selection with nonoverlapping generations) and much milder forms of reproductive variation.

2.3 Population Sizing

In natural systems, the size of a population can grow and shrink based on a variety of environmental factors. To date, EA developers have found little computational advantage for allowing the population size to vary from one generation to the next. Since fixed-size population models are also easier to implement, most EAs maintain a fixed-size population of size M . However, as we have seen, internal to a generation some number K of offspring are produced, and by the end of the generation a survival selection mechanism is used to reduce this intermediate number of $M + K$ individuals back to M .

So, an EA designer has two decisions to make: choosing a value for M and for K . Traditionally, the choices have varied widely for the following reasons. Choices for M are related to the degree of parallel search required. Highly complex, rugged, multi-peaked landscapes need a much higher degree of parallelism (larger M) than, say, a convex quadratic surface. Values for K are associated with feedback delays – the amount of exploration done with the current population before updating it with new information. Larger values of K are needed for more complex, rugged, multi-peaked landscapes. As a consequence, we see effective ESs with $M = 1$ and $K = 1$, effective EPs with $M = 10$ and $K = 10$, and effective GAs with $M = 50$ and $K = 50$. More recently, EAs being used to search complex program spaces have $M = 1,000$ or more.

Since EAs are frequently used on problems with fitness landscapes of unknown (*a priori*) properties, the values of M and K are generally chosen via some preliminary experimentation.

2.4 Fitness Landscapes

For many problems the fitness landscape is defined by the objective function that measures the quality of a solution. Understanding how EAs use fitness feedback to bias the search can often suggest ways to improve an objective function to make it more amenable to evolutionary search. A classic example is that of solving constraint-satisfaction problems. In this case, the formal fitness landscape is defined as 1 (true) if all constraints are satisfied and 0 (false) otherwise, creating a “needle-in-the-haystack” search problem that is lacking any intermediate feedback suggesting areas to look in. The problem becomes significantly more EA friendly if the objective function is modified to give nonuniform feedback on partial solutions (e.g., the number of constraints satisfied).

3 Example Application Areas

At the most general level, these simple EAs can be viewed as parallel adaptive search procedures with a very natural sense of fitness optimization. It is no surprise, then, that the majority of applications involve solving some sort of optimization problem. However, as we will see in this section, the breadth and variety of these applications goes well beyond the scope of traditional mathematical optimization areas.

3.1 Parameter Optimization

Perhaps the most pervasive and best understood EA application area is that of parameter optimization. This is partly because linear vectors of parameters map quite nicely onto our (admittedly simplistic) notion of biological genetic material consisting of linear strands of genes, the values of which affect an organism’s fitness. By analogy, recombining and mutating parameter values provide an interesting and frequently effective strategy for finding optimal combinations of parameter values.

Techniques for solving parameter optimization problems have been developed for many years by the mathematical optimization community. The key difference is that simple EAs make far fewer assumptions about the characteristics of the objective function to be optimized. They do not *a priori* require continuity, differentiability, convexity, etc. Rather, they exhibit a slower, but more robust search behavior even in the presence of noise. At the same time, domain-specific knowledge can be incorporated into an EA via specialized representations and reproductive operators when appropriate and desirable. As a consequence, optimization-oriented EAs continue to provide a nice complement to the existing and more traditional optimization techniques.

3.2 Nonlinear and Variable-Size Structures

Somewhat more challenging are optimization problems in which solutions are most naturally represented as nonlinear structures possibly varying in size. One example of that has already been discussed – the early EP work involving the evolution of finite state machines. To that, we can easily add a variety of useful structures including decision trees, artificial neural networks,

and job-shop schedules. Traditional approaches to such problems focus on developing very problem-specific techniques. The EA approach is more meta-heuristic in the sense that the standard EA framework gets instantiated by choosing an internal representation for the structures and designing effective mutation and recombination operators capable of generating useful offspring. So, for example, one might choose an adjacency list representation for finite state machines and then experiment with various recombination and mutation operators to find ones that produce offspring finite state machines that are interesting variants of their parents.

3.3 Executable Objects

Perhaps the most challenging problems are those in which the structures to be optimized are executable objects. This challenge is actually quite close to our modern view of biological genetic material as a complex program being executed within each cell, involving complex regulatory control mechanisms, and generating phenotypic traits in highly nonlinear ways. In the EA community, this challenge has given rise to entire subgroups focused on how best to evolve executable objects. An interesting open question is the extent to which more traditional programming languages are evolution friendly. As one might expect, languages like C and Java are much better suited for humans than evolution. However, languages like Lisp and rule-oriented languages have produced many striking positive results (see, for example, Koza 1992 or Grefenstette et al. 1990).

3.4 Other Than Optimization Problems

As noted earlier, standard EAs are best described as parallel adaptive search procedures. As such they can be usefully applied whenever one is faced with difficult search problems. For example, many constraint-satisfaction problems are really about finding *any* solution that satisfies all the constraints. Many aspects of data mining involve searching for interesting patterns and/or unusual events. Machine learning problems are generally characterized as searching hypothesis spaces for descriptions of concepts or behaviors. In each case, EAs have been shown to provide effective search strategies that complement more domain-specific approaches.

4 More Complex Evolutionary Algorithms

The EAs discussed in [Sect. 2](#) were characterized as “simple” in the sense that they can easily be described in a few lines of pseudocode. They are also simple from a biological perspective, abstracting away many standard features of natural evolutionary systems. Yet, as we have seen, these simple EAs produce surprisingly powerful and robust search heuristics. However, one might wonder why obvious features of natural evolutionary systems are missing, such as a notion of age, gender, multistranded chromosomes with dominance relationships among gene values, etc. The short answer is that many of these features have been experimented with but appear to provide no useful computational benefit when solving static optimization problems.

On the other hand, the successful application of simple EAs has inspired their application to problem areas that are significantly more complex than standard optimization problems. As a result, there has been a corresponding increase in the complexity of the EAs developed to address these problems. In this section, a few of the more important “complexifications” are described.

4.1 New Parallel EA Models

One of the important features of EAs is their natural parallelism which can be exploited using the increasing array of inexpensive parallel hardware architectures (e.g., multi-core machines, closely coupled multi-cpu clusters, loosely coupled clusters, etc.). In many of the more challenging applications, evaluating the fitness of an individual is the most cpu-intensive EA component. A good example of this is the area of “evolutionary robotics” in which evaluating the fitness of a robot controller is accomplished via a cpu-intensive simulation. In such cases, a simple master-slave configuration provides the ability to evaluate individuals in parallel on multiple loosely coupled machines.

In some EA applications, it is useful to impose a spatial topology on the population, replacing random mating and replacement with local neighborhood interactions (see, for example, Sarma 1998). In this case, computer languages supporting multi-threaded implementations provide a means for implementing efficient parallel local activities on multiprocessor machines.

At the other extreme, “island models” have been shown to be quite useful for solving large and difficult optimization problems (see, e.g., Skolicki and De Jong 2004). These island models represent a coarsely grained parallel framework in which multiple EAs are running on separate “islands” with occasional “migrations” of individuals from one island to another. These models map easily onto loosely coupled networks of machines, allowing a simple means for exploiting the collective computing power of a cluster of machines.

4.2 Multi-population EA Models

In addition to island models, there are a variety of multi-population models that are inspired by the biological notion of coevolving populations in which the fitness of individuals is dependent in part on the individuals in other populations. Classic examples of this in nature are competitive relationships such as predator-prey models or more cooperative relationships such as host-parasite models. As an example, competitive coevolutionary EAs have been used to evolve better game-playing programs (e.g., Rosin and Belew 1997), while cooperative coevolutionary EAs have been successfully applied to difficult design problems that can be broken down into a collection of subproblems (e.g., Potter and De Jong 2000).

4.3 Multi-objective Optimization

Most “real life” design problems involve trying to simultaneously satisfy multiple objectives. Trying to optimize a design for one objective often reduces the desirability of the design with respect to another objective (e.g., power versus fuel efficiency), requiring a designer to identify

acceptable trade-off points. This is often done by finding a set of Pareto-optimal points, that is, points in design space for which it is impossible to improve on one objective without making another worse, and then choosing from among them.

The ability to compute a set of Pareto-optimal points is, in general, a computationally difficult task. Some of the best algorithms to date for doing so are suitably modified EAs (see, for example, Deb 2001 or Coello et al. 2002). The success in this area has led to the organization of conferences focused just on this topic.

4.4 Handling Dynamic Environments

Historically the EC field has focused primarily on solving “static” optimization problems, that is, problems which do not change during the optimization process. However, there are many interesting and difficult optimization problems associated with dynamic environments (e.g., traffic control problems, stock market trading, etc.). As one might expect, standard EAs do not handle dynamic problems very well. However, suitable variations do quite well (see, for example, Branke 2002 or Morrison 2004).

4.5 Exploiting Morphogenesis

One of the most striking differences between simple EAs and natural systems is the fact that in simple EAs there is no concept of infancy, growth, and maturation. Offspring come into existence as mature adults whose fitness is immediately determinable and can be immediately selected to be a parent. Part of the reason for this is that, from a computational point of view, this additional complexity adds little or no value when solving standard static parameter optimization problems. However, as we extend the range of EA applications to problems like evolving complex engineering designs or complex agent behaviors, it becomes increasingly difficult to design effective representations and reproductive operators. If we reflect on how this is accomplished in natural systems, we note that the reproductive operators are manipulating the plans for growing complex objects. That is, there is a clear distinction between the genotype of an individual and its phenotype. A process of growth (morphogenesis) produces the phenotype from the genotype during a maturation process, and fitness is associated with the resulting phenotype.

These observations have led to considerable interest of EAs in “generative representations” capable of producing complex phenotypes from much simpler evolving phenotypes (see, for example, Kicinger et al. 2004). Independently, this notion of generative representation has been extensively explored in other areas of science as well, including biology (e.g., Lindenmayer 1968) and complex systems (e.g., Wolfram 1994).

5 Summary and Conclusions

Although evolutionary computation was one of the early fields to develop around the notion of “inspiration from nature,” it is certainly not the only one. Other familiar examples are “simulated annealing” techniques and “artificial neural networks.” More recently, there has been considerable interest in nature-inspired “ant colony” optimization and “particle swarm”

optimization. This has led to the broader concept of “natural computation” as documented in this handbook.

As emphasized in this chapter, adopting a generalized view of EAs leads naturally to the perception of EAs as a powerful “meta-heuristic,” a template for constructing problem-specific heuristics. This is an important perspective that continues to expand the range of EA applications.

Included in this chapter are a sprinkling of references to more detailed material. Far more information is available from the proceedings of the many EC-related conferences such as GECCO, CEC, PPSN, and FOGA, and from the established journals of the field such as *Evolutionary Computation* (MIT Press), *Transactions on Evolutionary Computation* (IEEE Press), and *Genetic Programming and Evolvable Machines* (Springer). In addition, there is a wealth of information and open-source code available on the Internet.

References

- Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, New York
- Box G (1957) Evolutionary operation: a method for increasing industrial productivity. *Appl Stat* 6(2):81–101
- Branke J (2002) Evolutionary optimization in dynamic environments. Kluwer, Boston, MA
- Coello C, Veldhuizen D, Lamont G (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, Boston, MA
- De Jong K (2006) Evolutionary computation: a unified approach. MIT Press, Cambridge, MA
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
- Eiben A, Raué P, Ruttakay Z (1994) Genetic algorithms with multi-parent recombination. In: Davidor Y, Schwefel H-P, Männer R (eds) Proceedings of the parallel problem solving from nature conference (PPSN III). Springer, Berlin, pp 78–87
- Fogel D (1995) Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, Piscataway, NJ
- Fogel D (1998) Evolutionary computation: the fossil record. IEEE Press, Piscataway, NJ
- Friedberg R (1959) A learning machine: Part 1. *IBM Res J* 3(7):282–287
- Grefenstette J, Ramsey CL, Schultz AC (1990) Learning sequential decision rules using simulation models and competition. *Mach Learn* 5(4):355–381
- Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evolut Comput* 9(2):159–195
- Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI
- Kicinger R, Arciszewski T, De Jong K (2004) Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings. In: Proceedings of the congress on evolutionary computation, Portland, IEEE Press, Piscataway, NJ pp 411–418
- Koza J (1992) Genetic programming. MIT Press, Cambridge, MA
- Lindenmayer A (1968) Mathematical models for cellular interaction in development. *J Theor Biol* 18:280–315
- Morrison R (2004) Designing evolutionary algorithms for dynamic environments. Springer, Berlin
- Potter M, De Jong K (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolut Comput* 8(1):1–29
- Rosin C, Belew R (1997) New methods for competitive coevolution. *Evolut Comput* 5(1):1–29
- Sarma J (1998) An analysis of decentralized and spatially distributed genetic algorithms. Ph.D. thesis, George Mason University
- Schwefel H-P (1995) Evolution and optimum seeking. Wiley, New York
- Skolicki Z, De Jong K (2004) Improving evolutionary algorithms with multi-representation island models. In: Yao X, Burke EK, Lozano JA, Smith J, Merelo Guervos JJ, Bullinaria JA, Rowe JE, Tino P, Kaban A, Schwefel H-P (eds) Proceedings of the parallel problem solving from nature conference (PPSN VIII). Springer, Berlin, pp 420–429
- Wolfram S (1994) Cellular automata and complexity. Addison-Wesley, Reading, MA

21 Genetic Algorithms — A Survey of Models and Methods

Darrell Whitley¹ · Andrew M. Sutton²

¹Department of Computer Science, Colorado State University,
Fort Collins, CO, USA
whitley@cs.colostate.edu

²Department of Computer Science, Colorado State University,
Fort Collins, CO, USA
sutton@cs.colostate.edu

1	<i>The Basics of Genetic Algorithms</i>	638
2	<i>Interpretations and Criticisms of the Schema Theorem</i>	653
3	<i>Infinite Population Models of Simple Genetic Algorithms</i>	655
4	<i>The Markov Model for Finite Populations</i>	658
5	<i>Theory Versus Practice</i>	659
6	<i>Where Genetic Algorithms Fail</i>	662
7	<i>An Example of Genetic Algorithms for Resource Scheduling</i>	664
8	<i>Conclusions</i>	669

Abstract

This chapter first reviews the simple genetic algorithm. Mathematical models of the genetic algorithm are also reviewed, including the schema theorem, exact infinite population models, and exact Markov models for finite populations. The use of bit representations, including Gray encodings and binary encodings, is discussed. Selection, including roulette wheel selection, rank-based selection, and tournament selection, is also described. This chapter then reviews other forms of genetic algorithms, including the steady-state Genitor algorithm and the CHC (cross-generational elitist selection, heterogenous recombination, and cataclysmic mutation) algorithm. Finally, landscape structures that can cause genetic algorithms to fail are looked at, and an application of genetic algorithms in the domain of resource scheduling, where genetic algorithms have been highly successful, is also presented.

1 The Basics of Genetic Algorithms

Genetic algorithms were the first form of evolutionary algorithms to be widely accepted across a diverse set of disciplines ranging from operations research to artificial intelligence. Today, genetic algorithms and other evolutionary algorithms are routinely used as search and optimization tools for engineering and scientific applications.

Genetic algorithms were largely developed by John Holland and his students in the 1960s, 1970s, and 1980s. The term “genetic algorithms” came into common usage with the publication of Ken De Jong’s 1975 PhD dissertation. Holland’s classic 1975 book, *Adaptation in Natural and Artificial Systems* (Holland 1975), used the term “genetic plans” rather than “genetic algorithms.” In the mid-1980s genetic algorithms started to reach other research communities. An explosion of research in genetic algorithms came soon after a similar explosion of research in artificial neural networks. Both areas of research draw inspiration from biological systems as a computational model.

There are several forms of evolutionary algorithms that use simulated evolution as a mechanism to solve problems where the problems can be expressed as a search or optimization problem. Other areas of evolutionary computation, such as evolutionary programming, evolution strategies, and genetic programming, are discussed in other chapters. Compared to other evolutionary algorithms, genetic algorithms put a great deal of emphasis on the combined interactions of selection, recombination, and mutation acting on a genotype. In most early forms of genetic algorithms, recombination was emphasized over mutation. This emphasis still endures in some forms of the genetic algorithm. However, hybridization of genetic algorithms with local search is also very common.

Genetic algorithms emphasize the use of a “genotype” that is decoded and evaluated. These genotypes are often simple data structures. In most early applications, the genotypes (which are sometimes thought of as artificial chromosomes) are bit strings which can be recombined in a simplified form of “sexual reproduction” and can be mutated by bit flips. Because of the bit encoding, it is sometimes common to think of genetic algorithms as function optimizers. However, this does not mean that they yield globally optimal solutions. Instead, Holland (in the introduction to the 1992 edition of his book *Adaptation in Natural and Artificial Systems* Holland (1992)) and De Jong (1993) have both emphasized that these algorithms find competitive solutions rather than optimal solutions, but both also suggest that it is probably best to view genetic algorithms not as an optimization process, but rather as adaptive systems.

At the risk of overemphasizing optimization, an example application from function optimization provides a useful vehicle for explaining certain aspects of these algorithms. For example, consider a control or production process which must be optimized with respect to some evaluation criterion. The input domain to the problem, \mathcal{X} , is the set of all possible configurations to the system. Given an evaluation function f , one seeks to maximize (or minimize) $f(x)$, $x \in \mathcal{X}$. The input domain can be characterized in many ways. For instance, if $\mathcal{X} = \{0, 1, \dots, 2^L - 1\}$ is the search space then our input domain might be the set of binary strings of length L . If we elect to use a “real-valued” encoding, a floating point representation might be used, but in any discrete computer implementation, the input domain is still a finite set.

In the evolutionary algorithm community, it has become common to refer to the evaluation function as a *fitness* function. Technically, one might argue that the objective function $f(x)$ is the evaluation function, and that the fitness function is a second function more closely linked with selection. For example, we might assign fitness to an artificial chromosome based on its rank in the population, assigning a “fitness” of 2.0 to the best member of the population and a fitness of 1.0 to the median member of the population. Thus, the fitness function and evaluation function are not always the same. Nevertheless in common usage, the term fitness function has become a surrogate name for the evaluation function.

Returning to our simple example of an optimization problem, assume that there is a system with three parameters we can control: temperature, pressure, and duration. These are in effect three inputs to a black box optimization problem where inputs from the domain of the function are fed into the black box and a value from the co-domain of the function is produced as an output. The system’s response could be a *quality* measure dependent on the temperature, pressure, and duration parameters.

One could represent the three parameters using a vector of three real-valued parameters, such as

$$\langle 32.56, 18.21, 9.83 \rangle$$

or the three parameters could be represented as bit strings, such as

$$\langle 000111010100, 110100101101, 001001101011 \rangle.$$

Using an explicit bit encoding automatically raises the question as to what precision should be used, and what should be the mapping between bit strings and real values. Picking the right precision can have a large impact on performance. Historically, genetic algorithms have typically been implemented using low precision; using 10 bits per parameter is common in many test functions. Using high precision (e.g., 32 bits per parameters) generally results in poor performance, and real-valued representations should be considered if high precision is required.

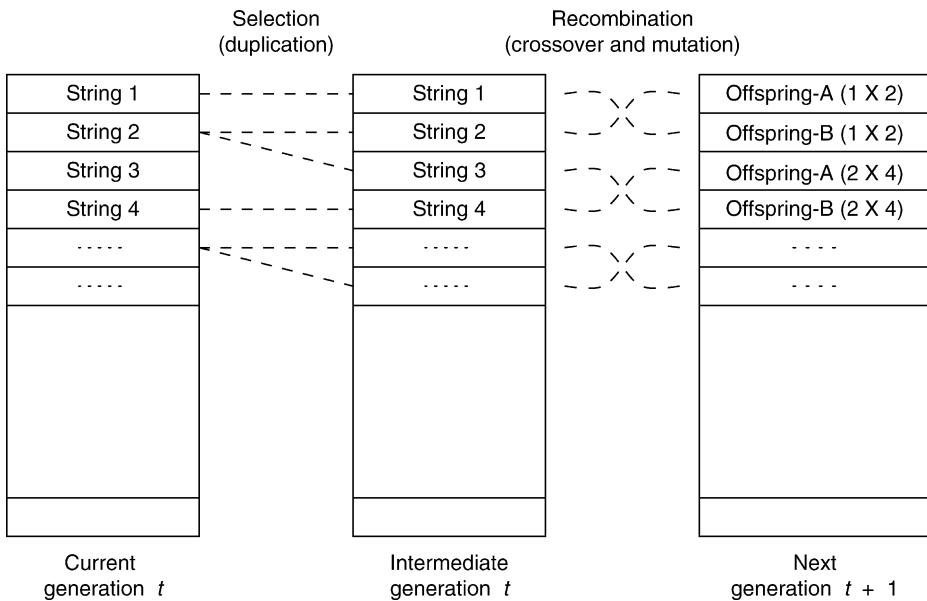
Recombination is central to genetic algorithms. For now, assume that the artificial chromosomes are bit strings and that 1-point crossover will be used. Consider the string 1101001100101101 and another binary string, yxyyxyxxxxyyxy, in which the values 0 and 1 are denoted by x and y. Using a single randomly chosen crossover point, a 1-point recombination might occur as follows.

11010 \ 01100101101

yxyyxy /\ yxxxxyyxy

Fig. 1

One generation is broken down into a selection phase and a recombination phase. This figure shows strings being assigned into adjacent slots during selection. It is assumed that selection randomizes the location of strings in the intermediate population. Mutation can be applied before or after crossover.



Swapping the fragments between the two parents produces the following two offspring.

$$11010yxxyyyxyxxy \quad \text{and} \quad yxxyyxy01100101101$$

Parameter boundaries are ignored. We can also apply a mutation operator. For each bit in the population, mutate with some low probability p_m . For bit strings of length L a mutation rate of $1/L$ is often suggested; for separable problems, this rate of mutation can be shown to be particularly effective.

In addition to mutation and recombination operators, the other key component to a genetic algorithm (or any other evolutionary algorithm) is the selection mechanism. For a genetic algorithm, it is instructive to view the mechanism by which a standard genetic algorithm moves from one generation to the next as a two-stage process. Selection is applied to the current population to create an *intermediate generation*, as shown in Fig. 1. Then, recombination and mutation are applied to the intermediate population to create the *next generation*. The process of going from the current population to the next population constitutes one generation in the execution of a genetic algorithm.

We will assume that our selection mechanism is *tournament selection*. This is not the mechanism used in early genetic algorithms, but it is commonly used today. A simple version of tournament selection randomly samples two strings out of the initial population, then “selects” the best of the two strings to insert into the intermediate generation. If the population size is N , then this must be done N times to construct the intermediate population. We can then apply mutation and crossover to the intermediate population.

1.1 The Canonical Holland-Style Genetic Algorithm

In a Holland-style genetic algorithm, tournament selection is not used; instead *fitness proportional selection* is used. Proportional fitness is defined by f_i/\bar{f} , where f_i is the evaluation associated with string i and \bar{f} is the average evaluation of all the strings in the population. Under fitness proportional selection, fitness is usually being maximized. All early forms of genetic algorithms used fitness proportional selection. Theoretical models also often assume fitness proportional selection because it is easy to model mathematically.

The fitness value f_i may be the direct output of an evaluation function, or it may be scaled in some way. After calculating f_i/\bar{f} for all the strings in the current population, selection is carried out. In the canonical genetic algorithm, the probability that strings in the current population are copied (i.e., duplicated) and placed in the intermediate generation that is proportional to their fitness.

For a maximization problem, if f_i/\bar{f} is used as a measure of fitness for string i , then strings where f_i/\bar{f} is greater than 1.0 have above average fitness and strings where f_i/\bar{f} is less than 1.0 have below average fitness. We would like to allocate more chances to reproduce to those strings that are above average. One way to do this is to directly duplicate those strings that are above average. To do so, f_i is broken into an integer part, x_i , and a remainder, r_i . We subsequently place x_i duplicates of string i directly into the intermediate population and place 1 additional copy with probability r_i .

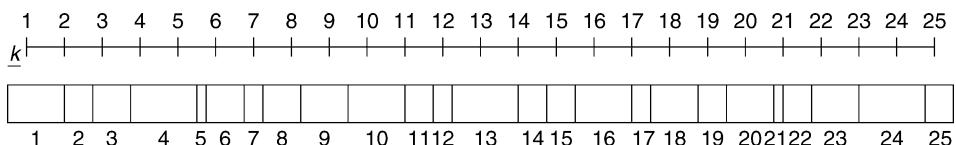
This is efficiently implemented using *stochastic universal sampling*. Assume that the population is laid out in random order as a number line where each individual is assigned space on the number line in proportion to fitness. Now generate a random number between 0 and 1 denoted by k . Next, consider the position of the number $i+k$ for all integers i from 1 to N where N is the population size. Each number $i+k$ will fall on the number line in some space corresponding to a member of the population. The position of the N numbers $i+k$ for $i = 1$ to N in effect selects the members of the intermediate population. This is illustrated in  Fig. 2.

This method is also known as roulette wheel selection: the space assigned to the 25 members of the population in  Fig. 2 could be laid out along the circumference of a circle representing the roulette wheel. The choice of k in effect “spins” the roulette wheel (since only the fractional part of the spins effects the outcome) and determines the position of the evenly spaced pointers, thereby simultaneously picking all N members of the intermediate population. The resulting selection is unbiased (Baker 1987).

After selection has been executed, the construction of the intermediate population is complete. The *next generation* of the population is created from the intermediate population.

 Fig. 2

Stochastic Universal Sampling. The fitnesses of the population can be seen as being laid out on a number line in random order as shown at the bottom of the figure. A single random value, $0 \leq k \leq 1$, shifts the uniformly spaced “pointers” which now select the member of the next intermediate population.



Crossover is applied to randomly paired strings with a probability denoted p_c . The offsprings created by “recombination” go into the next generation (thus replacing the parents). If no recombination occurs, the parents can pass directly into the next generation. However, parents that do not undergo recombination may still be changed due to the application of a mutation operator.

After the process of selection, recombination, and mutation is complete, the next generation of the population can be evaluated. The process of evaluation, selection, recombination, and mutation forms one generation in the execution of a genetic algorithm.

One way to characterize *selective pressure* is to calculate the selection bias toward the best individual in the population compared to the average fitness or the median fitness of the population. This is more useful for “rank-based” than fitness proportional, but it still serves to highlight the fact that fitness proportional selection can result in a number of problems. First, selective pressure can be too strong: too many duplicates are sometimes allocated to the best individual(s) in the population in the early generations of the search. Second, in the later stages of search, as the individuals in the population improve over time, there tends to be less variation in fitness, with more individuals (including the best individual(s)) being close to the population average. As the population average fitness increases, the fitness variance decreases and the corresponding uniformity in fitness values causes selective pressure to go down. When this happens, the search stagnates.

Because of these problems, few implementations of genetic algorithms use simple fitness proportionate reproduction. At the very least, the fitness may be rescaled before fitness proportionate reproduction. Fitness scaling mechanisms are discussed in detail in Goldberg’s textbook *Genetic Algorithm in Search, Optimization and Machine Learning* (Goldberg 1989b). In recent years, the use of rank-based selection and in particular tournament selection has become common.

1.2 Rank-Assigned Tournament Selection

Selection can be based on fitness assigned according to rank, or relative fitness. This can be done explicitly. Assume the population is sorted by the evaluation function. A linear ranking mechanism with selective pressure Z (where $1 < Z \leq 2$) allocates a fitness bias of Z to the top-ranked individual, $2 - Z$ to the bottom-ranked individual, and a fitness bias of 1.0 to the median individual. Note that the difference in selective pressure between the best and the worst member of the population is constant and independent of how many generations have passed. This has the effect of making selective pressure more constant and controlled. Code for linear ranking is given by Whitley (1989).

The use of rank-based selection also transforms the search into what is known as an *ordinal optimization method*. The exact evaluation of sample points from the search space is no longer important. All that is important is the relative value (or relative fitness) of the strings representing sample points from the search space. Note that the fitness function can be an ordinal measure, but the evaluation function itself is not ordinal.

Ordinal optimization may also sometimes relax computation requirements (Ho et al. 1992; Ho 1994). It may be easier to determine $f(x_1) < f(x_2)$ than to exactly compute $f(x_1)$ and $f(x_2)$. An approximate evaluation or even noisy evaluation may still provide enough information to determine the relative ranking, or to determine the relative ranking with high probability.

This can be particularly important when the evaluation is a simulation rather than an exact mathematical objective function.

A fast and easy way to implement ranking is *tournament selection* (Goldberg 1990; Goldberg and Deb 1991). We have already seen a simple form of tournament selection which selects two strings at random and places the best in the intermediate population. In expectation, every string is sampled twice. The best string wins both tournaments and gets two copies in the intermediate population. The median string wins one and loses one and gets one copy in the intermediate population. The worst string loses both tournaments and does not reproduce. In expectation, this produces a linear ranking with a selective pressure of 2.0 toward the best individual. If the winner of the tournament is placed in the intermediate population with probability $0.5 < p < 1.0$, then the selective pressure is less than 2.0. If a tournament size larger than 2 is used and the winner is chosen deterministically, then the selective pressure is greater than 2.0.

We can first generalize tournament selection to generate a linear selection bias less than 2, or a nonlinear bias greater than 2. For stronger selection, pick $t > 2$ individuals uniformly at random and select the best one of those individuals for reproduction. The process is repeated the number of times necessary to achieve the desired size of the intermediate population. The *tournament size*, t , has a direct correspondence with selective pressure because the expected number of times we sample the fittest individual in the intermediate population is t .

The original motivation behind the modern versions of tournament selection is that the algorithm is embarrassingly parallel because each tournament is independent (Suh and Gucht 1987). If one has the same number of processors as the population size, all N tournaments can be run simultaneously in parallel and constructing an intermediate population would require the same amount of time as executing a single tournament. Each processor samples the population t times and selects the best of those individuals.

Poli (2005) has noted that there are two factors that lead to the loss of diversity in regular tournament selection. Due to the randomness of tournaments, some individuals might not get *sampled* to participate in a tournament at all. Other individuals might be sampled but not be *selected* for the intermediate population because the individual loses the tournament.

One way to think of tournament selection (where $t = 2$) is a comparison of strings based on two vectors of random numbers.

```
vector A:  5  4  2  6  2  2  8  3  
vector B:  5  3  6  5  4  3  4  6
```

The integer stored at $A(i)$ identifies a member of the population. We will assume that the lower integers correspond to better evaluations. During the i th tournament, $A(i)$ is compared to $B(i)$ to determine the winner of the tournament. Note that in this example, if the integers 1–8 represent the population members, then individuals 1 and 7 are not *sampled*. Poli points out that the number of individuals neglected in the first decision if two offsprings are produced by recombination is $P(1 - 1/P)^{TP}$ where P is the population size and T is the tournament size. Expressed another way, what Poli calculates is the expected number of population members that fail to appear in vector A or B if these were used to keep track of tournament selection during one generation (or the amount of time needed to sample P new points in the search space). It is also possible to under-sample. If the tournament size is 2, in expectation, all members of the population would be sampled twice if tournament selection were used in combination with a generational genetic algorithm. However, some members of the population might be sampled only once.

Other forms of selection do not have this problem. Universal stochastic sampling guarantees that *all* individuals with above-average fitness get to reproduce, and all individuals below average get a chance to reproduce with a probability proportional to their fitness. Note that fitness in this case could be based on the evaluation function or it could still be a rank-based assignment of space on the roulette wheel in [Fig. 2](#).

Unbiased tournament selection (Sokolov and Whitley 2005) eliminates loss of diversity related to the failure to sample. Unbiased tournament selection also reduces the variance associated with how often an individual is sampled in one generation (variance reduction tournament selection might have been a more accurate name, but is somewhat more cumbersome). Unbiased tournament selection operates much like regular tournament selection except that permutations are used in place of random sampling during tournament construction. Rather than randomly sampling the population (or using random vectors of numbers to sample the population), t random permutations are generated. The i th element in each permutation indexes to a population member: the t population members pointed to by the i th element of each permutation form a tournament. An effective improvement is to use only $t-1$ permutations; the indices from 1 to P in sorted order can serve as one permutation (e.g., as generated in a for-loop). Other permutations can be constructed by sequentially sampling the population without replacement.

An example of unbiased tournament selection for tournament size $t = 3$ can be seen in [Fig. 3](#). Assume that $f(x) = x$ and that lower numbers correspond to better individuals; then the last row, labeled “Winners,” presents the resulting intermediate population as chosen by unbiased tournament selection. Note that permutation 1 is in sorted order. Recall that the selective pressure is partially controlled through the number of permutations.

Unbiased stochastic tournament selection can also be employed for selective pressure $S < 2$. We align two permutations, perform a pairwise comparison of elements, but select the best with $0.5 < p_s < 1.0$. For selective pressure less than or equal to 2 we also enforce the constraint that the i th element of the second permutation is not equal to i . This provides a guarantee that every individual will participate in a tournament twice. This is easily enforced: when generating the i th element of a permutation, we withhold the i th individual from consideration. There are not enough degrees of freedom to guarantee that this constraint holds for the last element of the permutation, but a violation can be fixed by swapping the last element with its immediate predecessor. The constraint is not enforced for $t > 2$ as too much overhead is involved.

To illustrate the congruence between “formula-based” ranking and tournament selection, a simulation was performed to compare three methods of linear ranking: (1) by random sampling using an explicit mathematical formula, (2) using regular tournament selection, and (3) using unbiased tournament selection. The number of times an individual with each rank was selected was measured as the search progressed. [Figure 4](#) presents these measurements

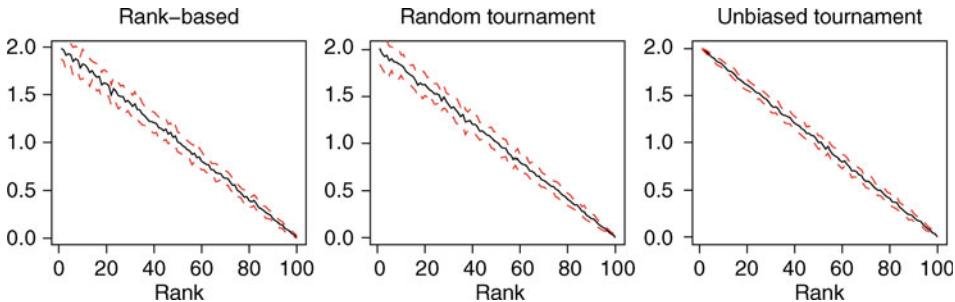
Fig. 3

Unbiased tournament selection with tournament size $t = 3$. In this case, each column represents a slot in the population; the best individual in each column wins the tournament for that slot.

Permutation 1:	1	2	3	4	5	6	7	8
Permutation 2:	6	2	1	5	8	4	3	7
Permutation 3:	2	8	1	4	3	6	7	5
Winners:	1	2	1	4	3	4	3	5

Fig. 4

The number of times an individual with a particular rank was selected for recombination. The results are presented for rank-based, regular, and unbiased tournament selection schemes. *Solid line* is the mean computed across 50 runs with 100 generations/run. *Dashed lines* lie one standard deviation away from the mean.



for a population of size 100 averaged over 50 runs with 100 generations each. A selective pressure of 2.0 was used. The solid line represented the mean and the two dashed lines were one standard deviation away from it. Notice that linear ranking behavior is problem independent because the only measure used in the calculations is the relative goodness of solutions.

As can be seen, formula-based ranking and regular tournament selection are essentially the same. On the other hand, unbiased tournament selection controls for variance by insuring the actual sample in each case is within one integer value of its expected value.

1.3 Different Forms of Representation, Crossover, and Mutation

One of the long-standing debates in the field of evolutionary algorithms involves the use of binary versus real-valued encodings for parameter optimization problems. The genetic algorithms community has largely emphasized bit representations. The main argument for bit encodings is that this representation decomposes the problem into the largest number of smallest possible building blocks and that a genetic algorithm works by processing these building blocks. This viewpoint, which was widely accepted 10 years ago, is now considered to be controversial. On the other hand, the evolution strategies community (Schwefel 1981, 1995; Bäck 1996) has emphasized the use of real-valued encodings for parameter optimization problems. Application-oriented researchers were also among the first in the genetic algorithms community to experiment with real-valued encodings (Davis 1991; Schaffer and Eshelman 1993).

Real-valued encodings can be recombined using simple crossover. The assignment of parameter values can be directly inherited from one parent or the other. But other forms of recombination are also possible, such as blending crossover that averages the real-valued parameter values. This idea might be generalized by considering the two parents as points p_1 and p_2 in a multidimensional space. These points might be combined as a weighted average (using some bias $0 \leq \alpha \leq 1$) such that an offspring might be produced by computing a new point p_0 in the following way:

$$p_0 = \alpha p_1 + (1 - \alpha)p_2$$

One can also compute a centroid associated with multiple parents.

Because real-valued encodings are commonly used in the evolution strategy community, the reader is encouraged to consult the chapter on evolution strategies. This chapter will focus on binary versus Gray coded bit representations.

1.3.1 Binary Versus Gray Representations

A related issue that has long been debated in the evolutionary algorithms community is the relative merit of Gray codes versus standard binary representations for parameter optimization problems. Generally, “Gray code” refers to the *standard binary reflected Gray code* (Bitner et al. 1976); but there are exponentially many possible Gray codes for a particular Hamming space. A Gray code is a bit encoding where adjacent integers are also Hamming distance 1 neighbors in Hamming space.

There are at least two basic ways to compute a Gray code. In general, a Gray matrix can be used that acts as a transform of a standard binary string. The standard binary reflected Gray code encoding matrix has 1s along the diagonal and 1s along the upper minor diagonal and 0s everywhere else. The matrix for decoding the standard binary reflected Gray code has 1s along the diagonal and all of the upper triangle of the matrix is composed of 1 bits. The lower triangle of the matrix is composed of 0 bits. The following is an example of the Graying matrix G (on the left) and the deGraying matrix D (on the right).

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Given a binary string (vector) b , a Graying matrix G , and a deGraying matrix D , bG produces a binary string (vector), which is the binary reflected Gray code of string b . Using the deGraying matrix, $(bG)D = b$.

A faster way to produce the standard binary reflected Gray code is to shift vector b by 1 bit, which will be denoted by the function $s(b)$. Using exclusive-or (denoted by \oplus) over the bits in b and $s(b)$,

$$b \oplus s(b) = bG$$

This produces a string with $L+1$ bits; the last bit is deleted. In implementation, no shift is necessary and one can have exclusive-or pairs of adjacent bits.

Assume that bit strings are assigned to the corner of a hypercube so that the strings assigned to the adjacent points in the hypercube differ by one bit. In general, a Gray code is a permutation of the corners of the hypercube, such that the resulting ordering forms a path that visits all of the points in the space and every point along the path differs from the point before and after it by a single bit flip.

There is no known closed form for computing the number of Gray codes for strings of length L . Given any Gray code (such as the standard binary reflected Gray code), one can (1) permute the order of the bits themselves to create $L!$ different Gray codes, and for each of these Gray codes one can (2) shift the integer assignment to the strings by using modular addition of a constant and still maintain a Gray code. This can be done in 2^L different ways. If all the Gray codes produced by these permutation and shift operations were unique, there would be at least $2^L(L!)$ Gray codes. Different Gray codes have different neighborhood

structures. In practice, it is easy to use the trick of shifting the Gray code by a constant to explore different Gray codes with different neighborhoods. But one can also prove that the neighborhood structure repeats after a shift of $2^L/4$.

The “reflected” part of the standard binary reflected Gray code derives from the following observation. Assume that we have a sequence of 2^L strings and we wish to extend that sequence to 2^{L+1} strings. Let the sequence of 2^L strings be denoted by

$$a \ b \ c \ \dots \ x \ y \ z$$

Without loss of generality, assume the strings can be decoded to correspond to integers such that $a = 0, b = 1, c = 2, \dots, y = (2^L - 2), z = (2^L - 1)$. In standard binary representations, we extend the string by duplicating the sequence and appending a 0 to each string in the first half of the sequence and a 1 to each string in the second half of the sequence. Thus, if we start with a standard binary sequence of strings, then the following is also a standard binary encoding:

$$0a \ 0b \ 0c \ \dots \ 0x \ 0y \ 0z \ 1a \ 1b \ 1c \ \dots \ 1x \ 1y \ 1z$$

However, in a reflected Gray code, we extend the string by duplicating and reversing the sequence; then a 0 is appended to all the strings in the original sequence and a 1 appended to the string in the reflected or reversed sequence.

$$0a \ 0b \ 0c \ \dots \ 0x \ 0y \ 0z \ 1z \ 1y \ 1x \ \dots \ 1c \ 1b \ 1a$$

It is easy to verify that if the original sequence is a Gray code, then the reflected expansion is also a Gray code.

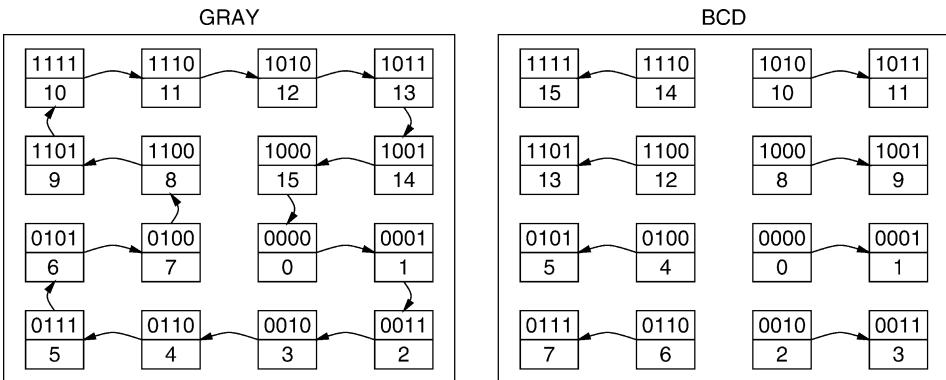
Over all possible discrete functions that can be mapped onto bit strings, the space of any and all Gray code representations and the space of binary representations must be identical. This is another example of what has come to be known as a “no-free-lunch” result (Wolpert and Macready 1995; Radcliffe and Surry 1995; Whitley and Rowe 2008). The empirical evidence suggests, however, that Gray codes are often superior to binary encodings. It has long been known that Gray codes remove Hamming cliffs, where adjacent integers are represented by complementary bit strings: for example, 7 and 8 encoded as 0111 and 1000. Whitley et al. (1996) first made the rather simple observation that every Gray code must preserve the connectivity of the original real-valued functions and that this impacts the number of local optima that exists under Gray and binary representation. This is illustrated in Fig. 5.

A consequence of the connectivity of the Gray code representation is that for every 1-dimensional parameter optimization problem, the number of optima in the Gray coded space must be less than or equal to the number of optima in the original real-valued function. Binary encodings offer no such guarantee. Binary encodings destroy half of the connectivity of the original real-valued function; thus, given a large basin of attraction with a globally competitive local optimum, many of the points near the optimum that are not optima become new local optima under a binary representation. The theoretical and empirical evidence suggests (Whitley 1999) that for parameter optimization problems with a bounded number of optima, Gray codes are better than binary in the sense that Gray codes induce fewer optima.

As for the debate over whether Gray bit encodings are better or worse than real-coded representations, the evidence is very unclear. If high precision is required, real-valued encodings are usually better. In other cases, a lower precision Gray code outperforms a real-valued encoding. This also means that an accurate comparison is difficult. The same parameter optimization problem encoded with high precision real-valued representation versus a low

Fig. 5

Adjacency in 4-bit Hamming space for Gray and standard binary encodings. The binary representation destroys half of the connectivity of the original function.



precision bit encoding results in two different search spaces. There are no clear theoretical or empirical answers to this question.

1.3.2 N-Point and Uniform Recombination

Early mathematical models of genetic algorithms often employed 1-point crossover in conjunction with binary strings. This bias again seems to be connected to the fact that 1-point crossover is easy to model mathematically. It is easy to see that 2-point crossover has advantages over 1-point crossover. In 1-point crossover, bits at opposite ends of the encoding are virtually always separated by recombination. Thus, there is a bias against inheriting bits from the same parents if they are at opposite ends of the encoding. However, with 2-point crossover, there is no longer any bias with regard to the end of the encodings. In 1-point crossover, the “end” of the encoding is a kind of explicit crossover point. 2-Point crossover treats the encoding as if it were circular so that the choice of both of the two crossover points are randomized.

Spears and De Jong (1991) argue for a variable N -point recombination. By controlling the number of recombination points, one can also control the likelihood of the two bits (or real-values) which are close together on one parent string and which are likely to be inherited together. With a small number of crossover points, recombination is less disruptive in terms of assorting which bits are inherited from which parents. With a large number of crossover points, recombination is more disruptive.

Is disruptive crossover good or bad? The early literature on genetic algorithms generally argued disruption was bad. On the other hand, disruption is also a source of new exploration. The most disruptive form of crossover is *uniform crossover*. Syswerda (1989) was one of the first application-oriented researchers to champion the use of uniform crossover. When uniform crossover is applied to a binary encoding, every bit is inherited randomly from one of the two parents. This means that uniform crossover on bit strings can actually be viewed as a blend of crossover and mutation. Any bits that are shared by the two parents must be passed along to

offspring, because no matter which parent the bit comes from, the bit is the same. But if the two parents differ in a particular bit position, then the bit that is inherited is randomly chosen. Another way of looking at uniform crossover is that all bit assignments that are shared by the two parents are automatically inherited. All bits that do not share the same value are set to a random value, because the inheritance is random. In some sense, the bits that are not set to the same value in the parents are determined by mutation: each of their values has an equal probability of being either one or zero.

Does it really matter if bits from a single parent that are close together on the encoding are inherited together or not? The early dogma of genetic algorithms held that inheritance of bits from a single parent that are close to each other on the chromosome was very important; very disruptive recombination, and uniform crossover in particular, should be bad. This in some sense was the central dogma at the time. This bias had a biological source.

This idea was borrowed in part from the biological concept of *coadapted alleles*. A widely held tenant of biological evolution is that distinct fragments of genetic information, or alleles, that work well *together* to enhance survivability of an organism should be inherited together. This view had a very strong influence on Holland's theory regarding the computational power of genetic algorithms: a theory based on the concept of hyperplane sampling.

1.4 Schemata and Hyperplanes

In his 1975 book, *Adaptation in Natural and Artificial Systems* (Holland 1975), Holland develops the concepts of schemata and hyperplane sampling to explain how a genetic algorithm can yield a robust search by implicitly sampling subpartitions of a search space. The idea that genetic algorithms derive their primary search power by hyperplane sampling is now controversial.

The set of strings of length L over a binary alphabet correspond to the vertices of an L -dimensional hypercube. A *hyperplane* is simply a subset of such vertices. These are defined in terms of the bit values they share in common. The concept of schemata is used to describe hyperplanes containing strings that contain particular shared bit patterns. Bits that are shared are represented in the schema; bits that are not necessarily shared are replaced by the * symbol. We will say that a bit string *matches* a particular schema if that bit string can be constructed from the schema by replacing a * symbol with the appropriate bit value. Thus, a 10-bit schema such as 1***** defines a subset that contains half the points in the search space, namely, all the strings that begin with a 1 bit in the search space. All bit strings that match a particular schema are contained in the hyperplane partition represented by that particular schema. The string of all * symbols corresponds to the space itself and is not counted as a partition of the space. There are $3^L - 1$ possible schemata since there are L positions in the bit string and each position can be a 0, 1, or * symbol and we do not count the string with no zeros or ones. A *low order* hyperplane is represented by a schema that has few bits, but many * symbols. The number of points contained in a hyperplane is 2^k where k is the number of * symbols in the corresponding schema.

The notion of a population-based search is critical to the schema-based theory of the search power of genetic algorithms. A population of sample points provides information about numerous hyperplanes; furthermore, low order hyperplanes should be sampled by numerous points in the population. Holland introduced the concept of *intrinsic* or *implicit parallelism* to describe a situation where many hyperplanes are sampled when a population of strings is evaluated; it

has been argued that far more hyperplanes are sampled than the number of strings contained in the population.

Holland's theory suggests that schemata representing competing hyperplanes increase or decrease their representation in the population according to the relative fitness of the strings that lie in those hyperplane partitions. By doing this, more trials are allocated to regions of the search space that have been shown to contain above average solutions.

1.5 An Illustration of Hyperplane Sampling

Holland (1975) suggested the following view of hyperplane sampling. In [Fig. 6](#), a function over a single variable is plotted as a one-dimensional space. The function is to be maximized. Assume the encoding uses 8 bits. The hyperplane 0^{*****} spans the first half of the space and 1^{*****} spans the second half of the space. Since the strings in the 0^{*****} partition are on average better than those in the 1^{*****} partition, we would like the search to be proportionally biased toward this partition. In the middle graph of [Fig. 6](#), the portion of the space corresponding to $**1^{****}$ is shaded, which also highlights the intersection of 0^{*****} and $**1^{****}$, namely, 0^*1^{****} . Finally, in the bottom graph, 0^*10^{****} is highlighted.

One of the assumptions behind the illustration in [Fig. 6](#) is that the sampling of hyperplane partitions is not affected to a significant degree by local minima. At the same time, increasing the sampling rate of regions of the search space (as represented by hyperplanes) that are above average compared to other competing regions does not guarantee convergence to a global optimum. The global optimum could be a relatively isolated peak that might never be sampled, for example. A small randomly placed peak that is not sampled is basically invisible; all search algorithms are equally blind to such peaks if there is no information to guide search.

Nevertheless, good solutions that are globally competitive might be found by such a strategy. This is particularly true if the search space is structured in such a way that pieces of good solutions can be recombined to find better solutions. The notion that hyperplane sampling is a useful way to guide search should be viewed as heuristic. In general, even having perfect knowledge of schema averages up to some fixed order provides little guarantee as to the quality of the resulting search. This is discussed in detail in [Sect. 2](#).

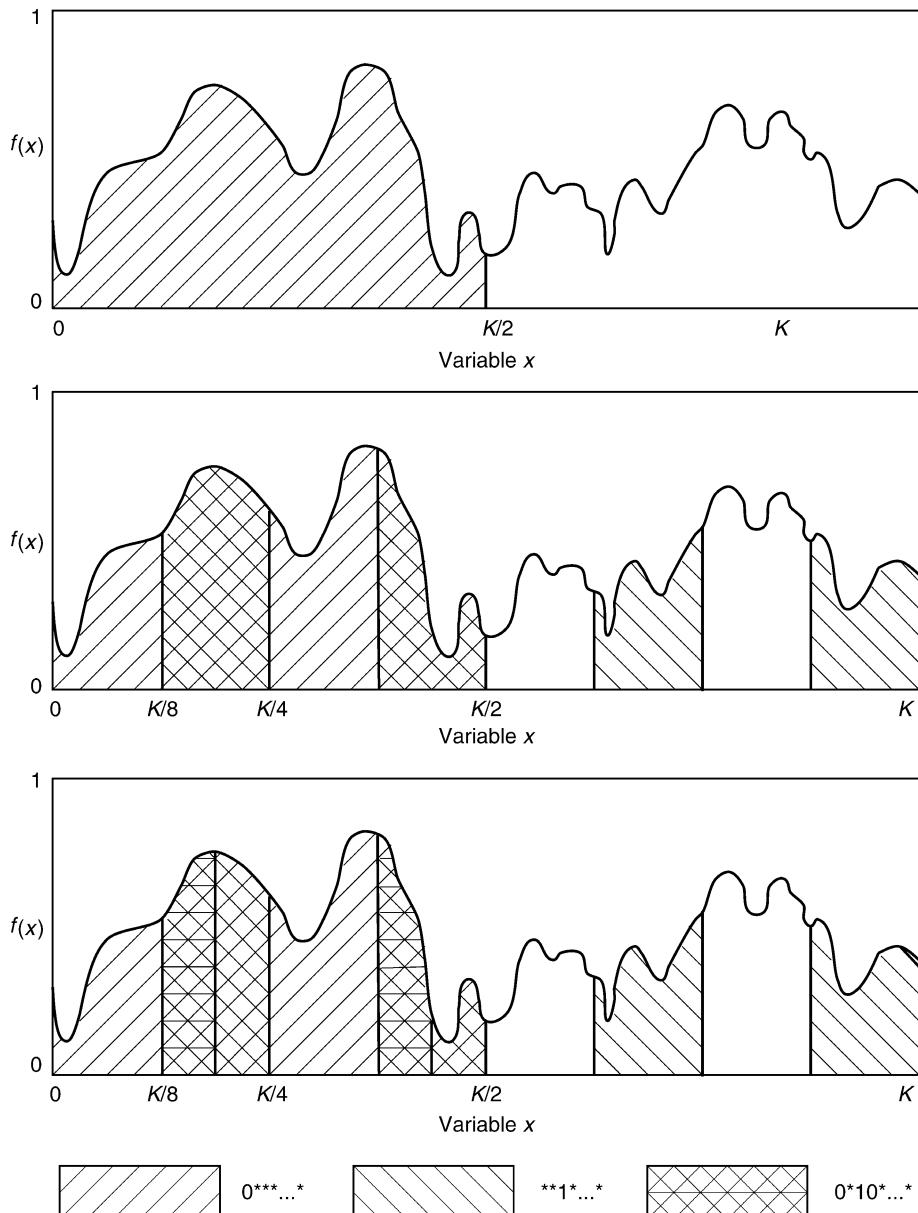
1.6 The Schema Theorem

Holland (1975) developed the *schema theorem* to provide a lower bound on the change in the sampling rate for a single hyperplane from generation t to generation $t+1$. By developing the theorem as a lower bound, Holland was able to make the schema theorem hold independently for every schema/hyperplane. At the same time, as a lower bound, the schema theorem is inexact, and the bounds hold for only one generation into the future. After one generation, the bounds are no longer guaranteed to hold. This weakness is just one of the many reasons that the concept of “hyperplane sampling” is controversial.

Let $P(H, t)$ be the proportion of the population that samples a hyperplane H at time t . Let $P(H, t + \text{intermediate})$ be the proportion of the population that samples hyperplane H after fitness proportionate selection but before crossover or mutation. Let $f(H, t)$ be the average fitness of the strings sampling hyperplane H at time t and denote the population average by \bar{f} . Note that \bar{f} should also have a time index, but this is often not denoted explicitly. This is

Fig. 6

A function and various partitions of hyperspace. Fitness is scaled to a 0 to 1 range in this diagram.



important because the average fitness of the population is *not* constant. Assuming that selection is carried out using fitness proportional selection:

$$P(H, t + \text{intermediate}) = P(H, t) \frac{f(H, t)}{\bar{f}}.$$

Thus, ignoring crossover and mutation, under just selection, the sampling rate of hyperplanes changes according to their average fitness. Put another way, selection “focuses” the search in what appears to be promising regions where the strings sampled so far have above-average fitness compared to the remainder of the search space. Some of the controversy related to “hyperplane sampling” begins immediately with this characterization of selection. The equation accurately describes the focusing effects of selection; the concern, however, is that the focusing effect of selection is not limited to the $3^L - 1$ hyperplanes that Holland considered to be relevant. Selection acts exactly the same way on any arbitrarily chosen subset of the search space. Thus, it acts in exactly the same way on the $2^{(2^L)}$ members of the power set over the set of all strings. While there appears to be nothing special about the sampling rate of hyperplanes under selection, all subsets of strings are not acted on in the same way by crossover and mutation. Some subsets of bit patterns corresponding to schemata are more likely to survive and be inherited in the population under crossover and mutation. For example, the sampling rate of order 1 hyperplanes is not disrupted by crossover and, in general, lower order hyperplanes are less affected by crossover than higher order hyperplanes.

Laying this issue aside for a moment, it is possible to write an exact version of the schema theorem that considers selection, crossover, and mutation. What we want to compute is $P(H, t+1)$, the proportion of the population that samples hyperplane H at the next generation as indexed by $t+1$. First just consider selection and crossover.

$$P(H, t+1) = (1 - p_c)P(H, t) \frac{f(H, t)}{\bar{f}} + p_c \left[P(H, t) \frac{f(H, t)}{\bar{f}} (1 - \text{losses}) + \text{gains} \right]$$

where p_c is the probability of performing a crossover operation. When crossover does not occur (which happens with probability $(1 - p_c)$), only selection changes the sampling rate. However, when crossover does occur (with probability p_c) then we have to consider how crossover and mutation can destroy hyperplane samples (denoted by losses) and how crossover can create new samples of hyperplanes (denoted by gains).

For example, assume we are interested in the schema $11*****$. If a string such as 1110101 were recombined between the first two bits with a string such as 1000000 or 0100000, no disruption would occur in hyperplane $11*****$ since one of the offspring would still reside in this partition. Also, if 1000000 and 0100000 were recombined exactly between the first and second bit, a new independent offspring would sample $11*****$; this is the source of *gains* that is referred to in the above calculation.

We will return to an exact computation, but, for now, instead of computing *losses* and *gains*, what if we compute a bound on them instead? To simplify things, *gains* are ignored and the conservative assumption is made that crossover falling in the significant portion of a schema always leads to disruption. Thus, we now have a bound on the sampling rate of schemata rather than an exact characterization:

$$P(H, t+1) \geq (1 - p_c)P(H, t) \frac{f(H, t)}{\bar{f}} + p_c \left[P(H, t) \frac{f(H, t)}{\bar{f}} (1 - \text{disruptions}) \right].$$

The *defining length* of a schema is based on the distance between the first and last bits in the schema with value either 0 or 1 (i.e., not a * symbol). Given that each position in a schema can be 0, 1, or *, scanning left to right, if I_x is the index of the position of the rightmost occurrence of either a 0 or a 1 and I_y is the index of the leftmost occurrence of either a 0 or a 1, then the defining length is merely $I_x - I_y$. The defining length of a schema representing a hyperplane H

is denoted here by $\Delta(H)$. If 1-point is used, then the defining length can be used to also calculate a bound on disruption:

$$\frac{\Delta(H)}{L-1} (1 - P(H, t))$$

and including this term (and applying simple algebra) yields:

$$P(H, t+1) \geq P(H, t) \frac{f(H, t)}{\bar{f}} \left[1 - p_c \frac{\Delta(H)}{L-1} (1 - P(H, t)) \right]$$

We now have a useful version of the schema theorem (although it does not yet consider mutation). This version assumes that selection for the first parent string is fitness-based and the second parent is chosen randomly. But typically both parents are chosen based on fitness. This can be added to the schema theorem by merely indicating the alternative parent chosen from the intermediate population after selection (Schaffer 1987).

$$P(H, t+1) \geq P(H, t) \frac{f(H, t)}{\bar{f}} \left[1 - p_c \frac{\Delta(H)}{L-1} \left(1 - P(H, t) \frac{f(H, t)}{\bar{f}} \right) \right]$$

Finally, mutation is included. Let $o(H)$ be a function that returns the order of the hyperplane H . The order of H exactly corresponds to a count of the number of bits in the schema representing H that have value 0 or 1. Let the mutation probability be p_m where mutation always flips the bit. Thus, the probability that mutation does not affect the schema representing H is $(1-p_m)^{o(H)}$. This leads to the following expression of the schema theorem.

$$P(H, t+1) \geq P(H, t) \frac{f(H, t)}{\bar{f}} \left[1 - p_c \frac{\Delta(H)}{L-1} \left(1 - P(H, t) \frac{f(H, t)}{\bar{f}} \right) \right] (1 - p_m)^{o(H)}$$

2 Interpretations and Criticisms of the Schema Theorem

For many years, the schema theorem was central to the theory of how genetic algorithms are able to effectively find good solutions in complex search spaces. Groups of bits that are close together on the encoding are less likely to be disrupted by crossover. Therefore, if groups of bits that are close together define a (hyperplane) subregion of the subspace that contains good solutions, selection should increase the representation of these bits in the population, and crossover and mutation should not “interfere” with selection since the probability of disruption should be low. In effect, such groups of bits act as coadapted sets of alleles; these are so important that they have been termed the “building blocks” of genetic search. As different complexes of coadaptive alleles (or bits) emerge in a population, these building blocks are put together by recombination to create even better individuals. The most aggressive interpretation of the schema theorem is that a genetic algorithm would allocate nearly optimal trials to sample different partitions of the search space in order to achieve a near-optimal global search strategy.

There are many different criticisms of the schema theorem. The schema theorem is not incorrect, but arguments have been made that go beyond what is actually proven by the schema theorem. First of all, the schema theorem is an inequality, and it only applies to one generation into the future. So while the bound provided by the schema theorem absolutely holds for one

generation into the future, it provides no guarantees about how strings or hyperplanes will be sampled in future generations.

It is true that the schema theorem does hold true independently for all possible hyperplanes for one generation. However, over multiple generations, the interactions between different subpartitions of the search space (as represented by hyperplanes and schemata) are extremely important. For example, in some search space of size 2^8 suppose that the schemata $11*****$ and $*00*****$ are both “above average” in the current generation of a population in some run of a genetic algorithm. Assume the schema theorem indicates that both will have increasing representation in the next generation. But trials allocated to schemata $11*****$ and $*00*****$ are in conflict because they disagree about the value of the second bit. Over multiple generations, both regions cannot receive increasing trials. These schema are *inconsistent* about what bit value is to be preferred in the second position. The schema theorem does not predict how such inconsistencies will be sorted out.

Whitley et al. (1995b) and Heckendorn et al. (1996) have shown that problems can have varying degrees of consistency in terms of which hyperplanes appear to be promising. For problems that display higher *consistency*, the “most fit” schemata tend to agree about what the values of particular bits should be. In a problem where there is a great deal of consistency, genetic search is usually effective. But other problems can be highly inconsistent, so that the most fit individuals (and sets of individuals as represented by schemata) display a large degree of conflict in terms of what bit values are preferred in different positions. It seems reasonable to assume that a genetic algorithm should do better on problems that display greater consistency, since inconsistency means that the search is being guided by conflicting information (this is very much related to the notion of “deception” but the concept of deception is controversial and much misunderstood).

One criticism of pragmatic significance is that users of the standard or canonical genetic algorithm often use very small populations. The number of positions containing 0 or 1 is referred to as the order of a schema. Thus, $**1*****$ is an order 1 schema, $***0***1$ is an order 2 schema, and $*1**0*1*$ is an order 3 schema. Many users employ a population size of 100 or smaller. In a population of size 100, we would expect 50 samples of any order 1 schema, 25 samples of any order 2 schema, 12.5 samples of any order 3 schema, and exponentially decaying numbers of samples to higher order schema. Thus, if we suppose that the genetic algorithm is implicitly attempting to allocate trials to different regions of the search space based on schema averages, a small population (e.g., 100) is inadequate unless we only care about very low order schemata. Therefore, even if hyperplane sampling is a robust form of heuristic search, the user destroys this potential by using small population sizes. Small populations require that the search rely more on hill-climbing. But in some cases, hill-climbing is very productive.

What if we had perfect schema information? What if we could compute schema information exactly in polynomial time? Rana et al. (1998) have shown that schema information up to any fixed order can be computed in polynomial time for some nondeterministic polynomial time (NP)-Complete problems. This includes Maximum Satisfiability (MAXSAT) problems and NK-Landscapes. This is very surprising. One theoretical consequence of this is captured by the following theorem:

- ▶ If $P \neq NP$ then, in the general case, exact knowledge of the static schema fitness averages up to some fixed order cannot provide information that can be used to guarantee finding a global optimum, or even an above average solution, in polynomial time. (For proofs, see Heckendorn et al. 1999a,b).

This seems like a very negative result. But it is dangerous to overinterpret either positive or negative results. In practice, random MAXSAT problems are characterized by highly inconsistent schema information, so there is really little or no information that can be exploited to guide the search (Heckendorn et al. 1999b). And in practice, genetic algorithms perform very poorly on MAXSAT problems (Rana et al. 1998) unless they are aided by additional hill-climbing techniques. On the other hand, genetic algorithms are known to work well in many other domains. Again, the notion of using schema information to guide search is heuristic.

There are many other criticisms of the schema theorem. In the early literature, too much was claimed about schema and hyperplane processing that was not backed up by solid proofs. It is no longer accepted that genetic algorithms allocate trials in an “optimal way” and it is certainly not the case that the genetic algorithm is guaranteed to yield optimal or even near-optimal solutions. In fact, there are good counterexamples to these claims. On the other hand, some researchers have attacked the entire notion of schema processing as invalid or false. Yet, the schema theorem itself is clearly a valid bound; and, experimentally, in problems where there are clearly defined regions that are above average, the genetic algorithm does quickly allocate more trials to such regions as long as these regions are relatively large.

There is still a great deal of work to be done to understand the role that hyperplane sampling plays in genetic search. Historically, the role of hyperplane sampling has been exaggerated, and the role played by hill-climbing has been underestimated. At the same time, many of the empirical results that call into question how genetic algorithms really work have been carried out on a highly biased sample of test problems. These test problems tend to be *separable*. A separable optimization problem is one in which the parameters are independent in terms of interaction. These problems are inherently easy to solve. For example, using a fixed precision (e.g., 32 bits per parameter), a separable problem can be solved exactly in a time that is a polynomial in the number of parameters by searching each parameter separately. On such problems, hill-climbing is a highly effective search strategy. Perhaps because of these simple test problems, the effectiveness (and speed) of simple hill-climbing has been overestimated.

In the next section, exact models of Holland’s simple genetic algorithm will be introduced.

3 Infinite Population Models of Simple Genetic Algorithms

Goldberg (1987, 1989b) and Bridges and Goldberg (1987) were the first to model critical details of how the genetic algorithm processes infinitely large populations under recombination. These models were independently derived in 1990 by Vose and Whitley in a more precise and generalized form. Vose and Liepins (1991) and Vose (1993) extended and generalized this model while Whitley et al. (1992) and Whitley (1993) introduced another version of the infinite population model that connects the work of Goldberg and Vose. In this section, the Vose model is reviewed and it is shown how the effects of various operators fit into this model.

The models presented here all use fitness proportionate reproduction because it is simpler to model mathematically. Vose (1999) also presents models for rank-based selection.

The vector $p^t \in \mathbb{R}$ is such that the k th component of the vector is equal to the proportional representation of string k at generation t . It is assumed that $n = 2^L$ is the number of points

in the search space defined over strings of length L and that the vector p is indexed 0 to $n-1$. The vector $s^t \in \mathbb{R}$ represents the t th generation of the genetic algorithm after selection and the i th component of s^t is the proportional representation of string i in the population after selection, but before any operators (e.g., recombination, mutation, and local search) are applied. Likewise, p_i^t represents the proportional representation of string i at generation t *before* selection occurs.

The function $r_{i,j}(k)$ yields the probability that string k results from the recombination of strings i and j . (For now, assume that r only yields the results for recombination; the effect of other operators could also be included in r .) Now, using \mathcal{E} to denote expectation,

$$\mathcal{E}\{p_k^{t+1}\} = \sum_{i,j} s_i^t s_j^t r_{i,j}(k) \quad (1)$$

To begin the construction of a general model, we first consider how to calculate the proportional representation of string 0 (i.e., the string composed of all zeros) at generation $t+1$; in other words, we compute p_0^{t+1} . A mixing matrix M is constructed where the (i,j) th entry $m_{i,j} = r_{i,j}(0)$. Here M is built by assuming that each recombination generates a single offspring. The calculation of the change in representation for string $k=0$ is now given by

$$\mathcal{E}\{p_0^{t+1}\} = \sum_{i,j} s_i^t s_j^t r_{i,j}(0) = s^T M s \quad (2)$$

where T denotes transpose. Note that this computation gives the expected representation of a single string, 0, in the next genetic population.

It is simple to see that the exact model for the infinite population genetic algorithm has the same structure as the model on which the schema theorem is based. For example, if we accept that the string of all zeros (denoted here simply by the integer 0) is a special case of a hyperplane, then when $H=0$, the following equivalence holds:

$$\begin{aligned} s^T M s &= P(H, t+1) = (1 - p_c) P(H, t) \frac{f(H, t)}{\bar{f}} \\ &\quad + p_c \left[P(H, t) \frac{f(H, t)}{\bar{f}} (1 - \text{losses}) + \text{gains} \right] \end{aligned} \quad (3)$$

The information about losses and gains is contained in the matrix M . To keep matters simple, assume the probability of crossover is $p_c=1$. Then, in the first row and column of matrix M , the calculation is really the probability of *retaining* a copy of the string, which is $(1 - \text{losses})$ (except at the intersection of the first row and column); the probabilities elsewhere in M are the gains in the above equation. To include the probability of crossover p_c in the model one must include the crossover probability in the construction of the M matrix.

The point of [Eq. 3](#) is that the exact Vose/Liepins model and the model on which the schema theorem is based is really the same. Whitley (1993) presents the calculations of losses and gains so as to make the congruent aspects of the infinite population model and the schema theorem more obvious.

The equations that have been looked at so far tell us how to compute the expected representation of one point in the search space one generation into the future. The key is to generalize this calculation to all points in the search space. Vose and Liepins formalized the

notion that bitwise exclusive-or can be used to access various probabilities from the recombination function r . Specifically,

$$r_{i,j}(k) = r_{i,j}(k \oplus 0) = r_{i \oplus k, j \oplus k}(0). \quad (4)$$

This implies that the mixing matrix M , which was defined such that entry $m_{i,j} = r_{i,j}(0)$, can provide mixing information for any string k just by changing how M is accessed. By reorganizing the components of the vector, s , the mixing matrix M can yield information about the probability $r_{i,j}(k)$. A permutation function, σ , is defined as follows:

$$\sigma_j \langle s_0, \dots, s_{n-1} \rangle^T = \langle s_{j \oplus 0}, \dots, s_{j \oplus (n-1)} \rangle^T \quad (5)$$

where the vectors are treated as columns and n is the size of the search space. The computation

$$(\sigma_q \ s^t)^T M (\sigma_q \ s^t) = p_q^{t+1} \quad (6)$$

thus reorganizes s with respect to string q and produces the expected representation of string q at generation $t+1$. A general operator \mathcal{M} can now be defined over s , which remaps $s^T M s$ to cover all strings in the search space.

$$\mathcal{M}(s) = \langle (\sigma_0 \ s)^T M (\sigma_0 \ s), \dots, (\sigma_{n-1} \ s)^T M (\sigma_{n-1} \ s) \rangle^T \quad (7)$$

This model has not yet addressed how to generate the vector s^t given p^t . A fitness matrix F is defined such that fitness information is stored along the diagonal; the (i,i) th element is given by $f(i)$ where f is the fitness function. Following Vose and Wright (1997),

$$s^t = F p^t / 1^T F p^t \quad (8)$$

since $F p^t = \langle f_0 p_0^t, f_1 p_1^t, \dots, f_{n-1} p_{n-1}^t \rangle$ and the population average is given by $1^T F p^t$.

Vose (1999) refers to this complete model as the \mathcal{G} function. Given any population distribution p , $\mathcal{G}(p)$ can be interpreted in two ways. On one hand, if the population is infinitely large, then $\mathcal{G}(p)$ is the exact distribution of the next population. On the other hand, given any finite population p , if strings in the next population are chosen one at a time, then $\mathcal{G}(p)$ also defines a vector such that element i is chosen for the next generation with probability $\mathcal{G}(p)_i$. This is because $\mathcal{G}(p)$ defines the exact sampling distribution for the next generation. This is very useful when constructing finite Markov models of the simple genetic algorithm.

Next, we look at how mutation can be added to this model in a modular fashion. This could be built directly into the construction of the M matrix. But looking at mutation as an additional process is instructive.

Recall that M is the mixing matrix, which we initially defined to cover only crossover. Define \mathcal{Q} as the mutation matrix. Assuming mutation is independently applied to each bit with the same probability, \mathcal{Q} can be constructed by defining a mutation vector Γ such that component Γ_i is the probability of mutating string i and producing string 0. The vector Γ is the first column of the mutation matrix and in general Γ can be reordered to yield column j of the matrix by reordering such that element $q_{i,j} = \Gamma_{i \oplus j}$.

Having defined a mutation matrix, mutation can now be applied after recombination in the following fashion:

$$p^{t+1,m} = (p^{t+1})^T \mathcal{Q},$$

where $p^{t+1,m}$ is just the p vector at time $t+1$ after mutation has occurred. Mutation also can be done before crossover; the effect of mutation on the vector s immediately after selection produces the following change: $s^T \mathcal{Q}$, or equivalently, $\mathcal{Q}^T s$.

Now we drop the $p^{t+1,m}$ notation and assume that the original p^{t+1} vector is defined to include mutation, such that

$$\begin{aligned} p_0^{t+1} &= (\mathcal{Q}^T s)^T M (\mathcal{Q}^T s) \\ p_0^{t+1} &= s^T (\mathcal{Q} \mathcal{M} \mathcal{Q}^T) s \end{aligned}$$

and we can therefore define a new matrix M_2 such that

$$p_0^{t+1} = s^T M_2 s \quad \text{where } M_2 = (\mathcal{Q} \mathcal{M} \mathcal{Q}^T)$$

As long as the mutation rate is independently applied to each bit in the string, it makes no difference whether mutation is applied before or after recombination. Also, this view of mutation makes it clear how the general mixing matrix can be built by combining matrices for mutation and crossover.

4 The Markov Model for Finite Populations

Nix and Vose (1992) show how to structure the finite population for a simple genetic algorithm. Briefly, the Markov model is an $N \times N$ transition matrix Q , where N is the number of finite populations of K strings and $Q_{i,j}$ is the probability that the k th generation will be population \mathcal{P}_j , given that the $(k-1)$ th population is \mathcal{P}_i .

Let

$$\langle Z_{0,j}, Z_{1,j}, Z_{2,j}, \dots, Z_{r-1,j} \rangle$$

represent a population, where $Z_{x,j}$ represents the number of copies of string x in population j , and $r=2^L$. The population is built incrementally. The number of ways to place the $Z_{0,j}$ copies of string 0 in the population is:

$$\binom{K}{Z_{0,j}}$$

The number of ways $Z_{1,j}$ strings can be placed in the population is:

$$\binom{K - Z_{0,j}}{Z_{1,j}}$$

Continuing for all strings,

$$\binom{K}{Z_{0,j}} \binom{K - Z_{0,j}}{Z_{1,j}} \binom{K - Z_{0,j} - Z_{1,j}}{Z_{2,j}} \dots \binom{K - Z_{0,j} - Z_{1,j} - \dots - Z_{r-2,j}}{Z_{r-1,j}}$$

which yields

$$\frac{K!}{(K - Z_{0,j})! Z_{0,j}!} \frac{(K - Z_{0,j})!}{(K - Z_{0,j} - Z_{1,j})! Z_{1,j}!} \frac{(K - Z_{0,j} - Z_{1,j})!}{(K - Z_{0,j} - Z_{1,j} - Z_{2,j})! Z_{2,j}!} \dots \frac{(K - Z_{0,j} - Z_{1,j} - \dots - Z_{r-2,j})!}{Z_{r-1,j}!}$$

which in turn reduces to

$$\frac{K!}{Z_{0,j}! Z_{1,j}! Z_{2,j}! \dots Z_{r-1,j}!}$$

Let $C_i(y)$ be the probability of generating string y from the finite population P_i . Then

$$Q_{i,j} = \frac{K!}{Z_{0,j}! Z_{1,j}! Z_{2,j}! \dots Z_{r-1,j}!} \prod_{y=0}^{r-1} C_i(y)^{Z_{y,j}}$$

and

$$Q_{i,j} = K! \prod_{y=0}^{r-1} \frac{C_i(y)^{Z_{y,j}}}{Z_{y,j}!}$$

So, how do we compute $C_i(y)$? Note that the finite population P_i can be described by a vector p . Also note that the sampling distribution from which P_j is constructed is given by the infinite population model $\mathcal{G}(p)$ (Vose 1999). Thus, replacing $C_i(y)$ by $\mathcal{G}(p)_y$ yields

$$Q_{i,j} = K! \prod_{y=0}^{r-1} \frac{(\mathcal{G}(p)_y)^{Z_{y,j}}}{Z_{y,j}!}$$

5 Theory Versus Practice

As the use of evolutionary algorithms became more widespread, a number of alternative genetic algorithm implementations have also come into common use. Some of the evolutionary algorithms in common use are based on evolution strategies. Select algorithms closest to traditional genetic algorithms will be reviewed.

The widespread use of alternative forms of genetic algorithms also means there is a fundamental tension between (at least some part of) the theory community and the application community. There are beautiful mathematical models and some quite interesting results for Holland's original genetic algorithm. But there are few results for the alternative forms of genetic algorithms that are used by many practitioners.

5.1 Steady-State and Island Model Genetic Algorithms

Genitor (Whitley and Kauth 1988; Whitley 1989) was the first of what was later termed “steady-state” genetic algorithms by Syswerda (1989). The name “steady-state” is somewhat unfortunate and the term “monotonic” genetic algorithm has been suggested by Alden Wright: these algorithms keep the best solutions found so far, and thus the population average monotonically improves over time. The distinction between steady-state genetic algorithms and regular generational genetic algorithms was also foreshadowed by the evolution strategy community. The Genitor algorithm, for example, can also be seen as a variant of a $(\mu+1)$ -Evolution strategy in terms of its selection mechanism. In contrast, Holland's generational genetic algorithm is an example of a (μ,λ) -Evolution Strategy where $\mu = \lambda$. The genetic algorithms community also used “monotonic” selection for classifier systems in the early 1980s.

Reproduction occurs one individual at a time in the Genitor algorithm. Two parents are selected for reproduction and produce an offspring that is immediately placed back into the population. The worst individual in the population is deleted. Ignoring the worst member of the population, the remainder of the population monotonically improves.

Another major difference between Genitor and other forms of genetic algorithms is that fitness is assigned according to rank rather than by fitness proportionate reproduction. In the original Genitor algorithm, the population is maintained in a sorted data structure. Fitness is assigned according to the position of the individual in the sorted population. This also allows one to prevent duplicates from being introduced into the population. This selection schema also means that the best $N-1$ solutions are always preserved in a population of size N . Goldberg and Deb (1991) have shown that by replacing the worst member of the population, Genitor generates much higher selective pressure than the canonical genetic algorithm.

Steady-state genetic algorithms retain the flavor of a traditional genetic algorithm. But the differences are significant. Besides the additional selective pressure, keeping the best strings seen so far means that the resulting search is more focused. This can be good or bad. The resulting search has a strong hill-climbing flavor, and the algorithm will continue to perturb the (best) strings in the population by crossover and mutation looking for an improved solution. If a population contains adequate building blocks to reach a solution in reasonable time, this focus can pay off. However, if the population does not contain adequate building blocks to generate further improvements, the search will be stuck. A traditional genetic algorithm or a (μ,λ) -evolution strategy allows a certain degree of drift and has the ability to continue moving in the space of possible populations. Thus, the greedy focus of a steady-state genetic algorithm sometimes pays off, and sometimes it does not.

To combat stagnation, it is sometimes necessary to use larger populations with steady-state genetic algorithms, or to use more aggressive mutation. Generally, larger populations result in slower progress, but improved solutions in the long run. Another way to combat stagnation is to use an *island model genetic algorithm*. Steady-state genetic algorithms seem to work better in conjunction with the island model paradigm than generational genetic algorithms. This may be due to the fact that steady-state genetic algorithms are more prone to stagnation.

Instead of using one large population, the population can be broken into several smaller populations. Thus, instead of running one population of size 1,000, one might run five populations of size 200. While the smaller population will tend to converge faster, diversity can be maintained by allowing migration between the subpopulations. For example, a small number of individuals (e.g., 1–5) might migrate from one subpopulation to another every 5–10 generations. It is important that migration be limited in size and frequency, otherwise the set of subpopulation becomes homogeneous too quickly (Starkweather et al. 1990).

In practice, steady-state genetic algorithms such as Genitor are often better optimizers than the canonical generational genetic algorithm. This has especially been true for scheduling problems such as the traveling salesman problem.

Current implementations of steady-state genetic algorithms are likely to use tournament selection instead of explicit rank-based selection. The use of tournament selection makes it unnecessary to keep the population in sorted order for the purposes of selection. But the population must still be sorted to keep track of the worst member of the population over time. Of course, once the population is sorted, inserting a new offspring takes $O(N)$ time, where N is the population size. In practice, poor offspring that are not inserted incur no insertion cost, and by starting from the bottom of the population, the insertion cost is proportional to the fitness of the offspring.

Some researchers have experimented with other ways of deciding which individuals should be replaced after new offspring are generated. Tournament selection can be used in reverse to decide if a new offspring should be allowed into the population and which member of the

current population should be replaced. When tournament selection is used in reverse, the tournament sizes are typically larger. One can also use an aspiration level, so that new offspring are not allowed to compete for entry into the population unless this aspiration level is met.

5.2 CHC

The CHC (Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) (Eshelman 1991; Eshelman and Schaffer 1991) algorithm was created by Larry Eshelman with the explicit idea of borrowing from both the genetic algorithm and the evolution strategy community. CHC explicitly borrows the $(\mu+\lambda)$ strategy of evolution strategies. After recombination, the N best unique individuals are drawn from the parent population and offspring population to create the next generation. This also implies that duplicates are removed from the population. This form of selection is referred to as *truncation selection*. From the genetic algorithm community, CHC builds on the idea that recombination should be the dominant search operator. A bit representation is typically used for parameter optimization problems. In fact, CHC goes so far as to use *only* recombination in the main search algorithm. However, it restarts the search where progress is no longer being made by employing what Eshelman refers to as *cataclysmic mutation*.

Since truncation selection is used, parents can be paired randomly for recombination. However, the CHC algorithm also employs a *heterogeneous recombination* restriction as a method of “incest prevention” (Eshelman 1991). This is accomplished by only mating those string pairs which differ from each other by some number of bits; Eshelman refers to this as a mating threshold. The initial mating threshold is set at $L/4$, where L is the length of the string. If a generation occurs, in which no offspring are inserted into the new population, then the threshold is reduced by 1.

The crossover operator in CHC performs uniform crossover; bits are randomly and independently exchanged, but exactly half of the bits that differ are swapped. This operator, called HUX (Half Uniform Crossover) ensures that offspring are equidistant between the two parents. This serves as a diversity preserving mechanism. An offspring that is *closer* to a parent in terms of Hamming distance will have a tendency to be more similar to that parent in terms of evaluation. Even if this tendency is weak, it can contribute to loss of diversity. If the offspring and parent that are closest in Hamming space tend to be selected together in the next generation, diversity is reduced. By requiring that the offspring be exactly halfway in between the two parents in terms of Hamming distance, the crossover operator attempts to slow down loss of genetic diversity. One could also argue that the HUX operator attempts to maximize the distribution of new samples in new regions of the search space by placing the offspring as far as possible from the two parents while still retaining the bits which the two parents share in common.

No mutation is applied during the regular search phase of the CHC algorithm. When no offspring can be inserted into the population of a succeeding generation and the mating threshold has reached a value of 0, CHC infuses new diversity into the population via a form of restart. Cataclysmic mutation uses the best individual in the population as a template to re-initialize the population. The new population includes one copy of the template string; the remainder of the population is generated by mutating some percentage of bits (e.g., 35%) in the template string.

Bringing this all together, CHC stands for *cross generational elitist selection, heterogeneous recombination* (by incest prevention) and *cataclysmic mutation*, which is used to restart the search when the population starts to converge.

The rationale behind CHC is to have a very aggressive search by using truncation selection which guarantees the survival of the best strings, but to offset the aggressiveness of the search by using highly disruptive uniform crossover. Because of these mechanisms, CHC is able to use a relatively small population size. It generally works well with a population size of 50. Eshelman and Schaffer have reported quite good results using CHC on a wide variety of test problems (Eshelman 1991; Eshelman and Schaffer 1991). Other experiments (c.f. Mathias and Whitley 1994; Whitley et al. 1995) have shown that it is one of the most effective evolutionary algorithms for parameter optimization on many common test problems. Given the small population size and the operators, it seems unreasonable to think of an algorithm such as CHC as a “hyperplane sampling” genetic algorithm. Rather, it can be viewed as an aggressive population-based hill-climber that also uses restarts to quickly explore new regions of the search space. This is also why the small population size is important to the performance of CHC. If the population size is increased, there is some additional potential for increased performance before a restart is triggered. But as with steady-state genetic algorithms, an increase in population size generally results in a small increase in performance after a significant increase in time to convergence/stagnation. The superior performance of CHC using a population of 50 suggests that more restarts have a better payoff than increasing the population size.

6 Where Genetic Algorithms Fail

Do genetic algorithms have a particular mode of failure? There are probably various modes of failure, some of which are common to many search algorithms. If functions are too random, too noisy, or fundamentally unstructured, then search is inherently difficult. “Deception” in the form of misleading hyperplane samples is also a mode of failure that has been studied (Whitley 1991; Goldberg 1989a; Grefenstette 1993).

There is another fundamental mode of failure such that there exists a well-defined set of problems where the performance of genetic algorithms is likely to be poor. Experiments show that various genetic algorithms and local search methods are more or less blind to “ridges” in the search space of parameter optimization problems. In two dimensions, the *ridge problem* is essentially this: a method that searches parallel to the x and y axes cannot detect improving moves that are oriented at a 45° angle to these axes.

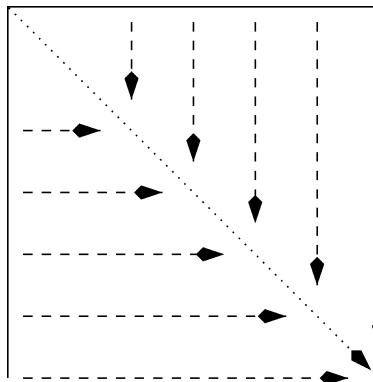
A simplified representation of a ridge problem appears in Fig. 7. Changing one variable at a time will move local search to the diagonal. However, looking in either the x -dimension or the y -dimension, every point along the diagonal appears to be a local optimum. There is actually gradient information if one looks *along* the diagonal; however, this requires either (1) changing both variables at once or (2) transforming the coordinate system of the search space so as to “expose” the gradient information.

The ridge problem is relatively well documented in the mathematical literature on derivative free minimization algorithms (Rosenbrock 1960; Brent 1973). However, until recently, there has been little discussion of this problem in the evolutionary algorithm literature.

Let $\Omega = \{0, 1, \dots, 2^\ell - 1\}$ be the search space which can be mapped onto a hypercube. Elements $x, z \in \Omega$ are *neighbors* when (x, z) is an edge in the hypercube. Bit climbing search algorithms terminate at a *local optimum*, denoted by $x \in \Omega$, such that none of the points in the neighborhood $N(x)$ improve upon x when evaluated by some objective function. Of course, the neighborhood structure of a problem depends upon the coding scheme used. Gray codes are often used for bit representations because, by definition, adjacent integers are adjacent neighbors.

Fig. 7

Local search moves only in the horizontal and vertical directions. It therefore “finds” the diagonal, but gets stuck there. Every point on the diagonal is locally optimal. Local search is blind to the fact that there is gradient information moving along the diagonal.



Suppose the objective function is defined on the unit interval $0 \leq x < 1$. To optimize this function, the interval is discretized by selecting n points. The *natural encoding* is then a map from Ω to the graph that has edges between points x and $x+1$ for all $x = 0, 1, \dots, n-2$.

Under a Gray encoding, adjacent integers have bit representations that are neighbors at Hamming distance 1 (e.g., $3 = 010$, $4 = 110$). Thus a Gray encoding has the following nice property:

- A function $f : \Omega \rightarrow \mathbb{R}$ cannot have more local optima under a Gray encoding than it does under the natural encoding.

A proof first appears in Rana and Whitley (1997); this theorem states that when using a Gray code, local optima of the objective function considered as a function on the unit interval can be destroyed, but no new local optima can be created (☞ Fig. 8).

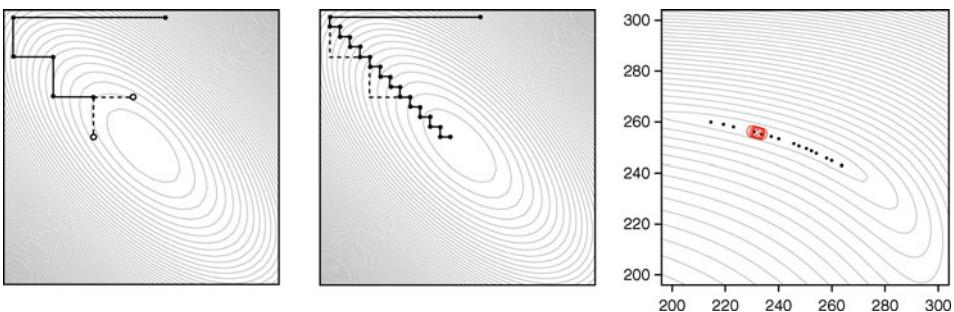
But are there unimodal functions where the natural encoding is multimodal? If the function is one dimensional, the answer is no. However, if the function is not one dimensional, the answer is yes. False local optima are induced on ridges since there are points along the ridge where improving moves become invisible to search.

This limitation is not unique to local search, and it is not absolute for genetic algorithms. Early population sampling can potentially allow the search to avoid being trapped by “ridges.” It is also well known that genetic algorithms quickly lose diversity and then the search must use mutation or otherwise random jumps to move along the ridge. Any method that tends to search one dimension at a time (or to find improvements by changing one dimension at a time via mutation) has the same limitation, including local search and simple “line search” methods.

Salomon (1960) showed that most benchmarks become much more difficult when the problems are *rotated*. Searching a simple two-dimensional elliptical bowl is optimally solved by one iteration of line search when the ellipse is oriented with the x and y axis. However, when the space is rotated 45° , the bowl becomes a ridge and the search problem is more difficult for many search algorithms.

Fig. 8

The two leftmost images show how local search moves on a 2D ridge using different step sizes. Smaller step sizes results in more progress on the ridge, but more steps to get there. The rightmost figure shows points at which local search becomes stuck on a real-world application.



Modern evolution strategies are invariant under rotation. In particular, the covariance matrix adaptation (CMA) evolution strategy uses a form of principal component analysis (PCA) to rotate the search space as additional information is obtained about the local landscape (Hansen 2006). Efforts are currently underway to generalize the concept of a rotationally invariant representation that could be used by different search algorithms (Hansen 2008). Until this work advances, current versions of genetic algorithms still have problems with some ridge structures. Crossover operators such as interval crossover (Schaffer and Eshelman 1993) attempt to deal with the ridge problem but are not as powerful as CMA.

7 An Example of Genetic Algorithms for Resource Scheduling

Resource scheduling problems involve the allocation of time and/or a resource to a finite set of requests. When demand for a resource becomes greater than the supply, conflicting requests require some form of arbitration and the scheduling problem can be posed as an optimization problem over a combinatorial domain. Thus, these problems are combinatorial in nature and are quite distinct from parameter optimization problems. Indeed, genetic algorithms have been very successful on scheduling applications of this nature.

A schedule may attempt to maximize the total number of requests that are filled, or the aggregate value (or priority) of requests filled, or to optimize some other metric of resource utilization. In some cases, a request needs to be assigned a time window on some appropriate resource with suitable capabilities and capacity. In other cases, the request does not correspond to a particular time window, but rather just a quantity of an oversubscribed resource.

One example of resource scheduling is the scheduling of flight simulators (Syswerda 1991). Assume a company has three flight simulators, and 100 people who want to use them. In this case, the simulators are a limited resource. A user might request to use a simulator from 9:00 a.m. to 10:00 a.m. on Tuesday. But if six users want a simulator at this time, then not all requests can be fulfilled. Some users may only be able to utilize the simulators at certain times or on certain days. How does one satisfy as many requests as possible? Or which requests are most important to satisfy? Does it make sense to schedule a user for less time than they requested?

One way to attempt to generate an approximate solution to this problem is to have users indicate a number of prioritized choices. If a user's first choice cannot be filled, then perhaps their second choice of times is available. A hypothetical evaluation function is to have a schedule evaluator that awards ten points for every user that gets their first choice, and five points for those that get their second choice and three points for those that get their third choice, one point for those scheduled (but not in their first, second, or third prioritized requests) and -10 points for user requests that cannot be scheduled.

Is this the best evaluation function for this problem? This raises an interesting issue. In real-world applications, sometimes the evaluation function is not strictly determined and developers must work with users to define a reasonable evaluation function. Sometimes what appears to be an “obvious” evaluation function turns out to be the wrong evaluation function.

Reasonably good solutions to a resource scheduling problem can usually be obtained by using a greedy scheduler which allocates to each request the best available resource at the best available time on a first-come, first-served basis.

The problem with a simple greedy strategy is that requests are not independent – when one request is assigned a slot on a resource, that resource is no longer available (or less available). Thus, placing a single request at its optimal position may preclude the optimal placement of multiple other requests. This is the standard problem with all greedy methods.

A significant improvement on a greedy first-come, first-served strategy is to explore the space of possible permutations of the requests where the permutation defines a priority ordering of the requests to be placed in the schedule. A genetic algorithm can then be applied to search the space of permutations. The fitness function is some measure of cost or quality of the resulting schedule.

The use of a permutation-based representation also requires the construction of a “schedule builder” which maps the permutation of requests to an actual schedule. In a sense, the schedule builder is also a greedy scheduling algorithm. Greedy scheduling on its own can be a relatively good strategy. However, by exploring the space of different permutations, one changes the order in which requests get access to resources; this can also be thought of as changing the order in which requests arrive. Thus, exploring the space of permutations provides the opportunity to improve on the simple greedy scheduler.

The separation of “permutation space” and “schedule space” allows for the use of two levels of optimization. At a lower level, a greedy scheduler converts each permutation into a schedule; at a higher level, a genetic algorithm is used to search the space of permutations. This approach thus creates a strong separation between the problem representation and the actual details of the particular scheduling application. This allows the use of relatively generic “genetic recombination” operators or other local search operators. A more direct representation of the scheduling problem would require search operators that are customized to the application. When using a permutation-based representation, this customization is hidden inside the schedule builder. Changes from one application to another only require that a new schedule builder be constructed for that particular application. This makes the use of permutation-based representations rather flexible.

Whitley et al. (1989) first used a strict permutation-based representation in conjunction with genetic algorithms for real-world applications. However, Davis (1985b) had previously used “an intermediary, encoded representation of schedules that is amenable to crossover operations, while employing a decoder that always yields legal solutions to the problem.” This is also a strategy later adopted by Syswerda (1991) and Syswerda and Palmucci (1991), which he enhanced by refining the set of available recombination operators.

Typically, simple genetic algorithms encode solutions using bit-strings, which enable the use of “standard” crossover operators such as 1-point and 2-point (Goldberg 1989b). However, when solutions for scheduling problems are encoded as permutations, a special crossover operator is required to ensure that the recombination of two parent permutations results in a child that (1) inherits good characteristics of both parents and (2) is still a legal permutation. Numerous crossover operators have been proposed for permutations representing scheduling problems. For instance, Syswerda’s (1991) *order* and *position* crossover are methods for producing legal permutations that inherit various ordering or positional elements from parents.

Syswerda’s order crossover and position crossover differ from other permutation crossover operators such as Goldberg’s Partially Mapped Crossover (PMX) operator (Goldberg and Lingle 1985) or Davis’ order crossover (Davis 1985a) in that no contiguous block is directly passed to the offspring. Instead, several elements are randomly selected by absolute position. These operators are largely used for scheduling applications (e.g., Syswerda 1991; Watson et al. 1999; Syswerda and Palmucci 1991 for Syswerda’s operator) and are distinct from the permutation recombination operators that have been developed for the traveling salesman problem (Nagata and Kobayashi 1997; Whitley et al. 1989). Operators that work well for scheduling applications do not work well for the traveling salesman problem, and operators that work well for the traveling salesman problem do not work well for scheduling.

Syswerda’s order crossover operator can be seen as a generalization of Davis’ order crossover (Davis 1991) that also borrows from the concept of uniform crossover for bit strings. Syswerda’s order crossover operator starts by selecting K uniform-random positions in Parent 2. The corresponding elements from Parent 2 are then located in Parent 1 and reordered, so that they appear in the same relative order as they appear in Parent 2. Elements in Parent 1 that do not correspond to selected elements in Parent 2 are passed directly to the offspring.

Parent 1 :	“A B C D E F G”
Parent 2 :	“C F E B A D G”
Selected Elements :	* * *

The selected elements in Parent 2 are F B and A in that order. A remapping operator reorders the relevant elements in Parent 1 in the same order found in Parent 2.

“A B - - - F - - -” remaps to “FB - - - A - - -”

The other elements in Parent 1 are untouched, thus yielding

“F B C D E A G”

Syswerda also defined a “position crossover.” Whitley and Yoo (1995) prove that Syswerda’s order crossover and position crossover are identical in expectation when order crossover selects K positions and position crossover selects $L - K$ positions over permutations of length L .

7.1 The Coors Warehouse Scheduling Problem

The Coors production facility (circa 1990) consists of 16 production lines, a number of loading docks, and a warehouse for product inventory. At the time this research was originally carried out (Starkweather et al. 1991), each production line could manufacture approximately 500 distinct products. Orders could be filled directly from the production lines or from inventory.

A solution is a priority ordering of customer orders. While the ultimate resource is the product that is being ordered, another limiting resource are the loading docks. Once a customer order is selected to be filled, it is assigned a loading dock and (generally) remains at the dock until it is completely filled, at which point the dock becomes empty and available for another order. All orders compete for product from either the production line or inventory. Note that scheduling also has secondary effects on how much product is loaded from the production line onto a truck (or train) and how much must temporarily be placed in inventory.

The problem representation used here is permutation of customer orders; the permutation queues up the customer orders which then wait for a vacant loading dock. When a dock becomes free, an order is removed from the queue and assigned to the dock.

A simulation is used to compute the evaluation function. Assume we wish to schedule one 24 h period. The simulation determines how long it takes to fill each order and how many orders can be filled in 24 h. The simulation must also track which product is drawn out of inventory, how much product is directly loaded off the production line, and how much product must first go into inventory until it is needed.

Figure 9 illustrates how a permutation is mapped to a schedule. Customer orders are assigned a dock based on the order in which they appear in the permutation; the permutation in effect acts as a customer priority queue. In the right-hand side of the illustration, note that initially customer orders A–I get first access to the docks (in a left to right order). C finishes first, and the next order, J, replaces C at the dock. Order A finishes next and is replaced by K. G finishes next and is replaced by L.

For the Coors warehouse scheduling problem, one is interested in producing schedules that simultaneously achieve two goals. One of these is to minimize the mean time that customer orders remain at dock. Let N be the number of customer orders. Let M_i be the time that the truck or rail car holding customer order i spends at dock. Mean time at dock, \mathcal{M} , is then given by

$$\mathcal{M} = \frac{1}{N} \sum_{i=0}^N M_i.$$

The other goal is to minimize the running average inventory. Let F be the makespan of the schedule. Let \mathcal{J}_t be inventory at time t . The running average inventory, I , is given by

$$I = \frac{1}{F} \sum_{t=0}^F \mathcal{J}_t.$$

Technically this is a multi-objective problem. This problem was transformed into a single-objective problem using a linear combination of the individual objectives:

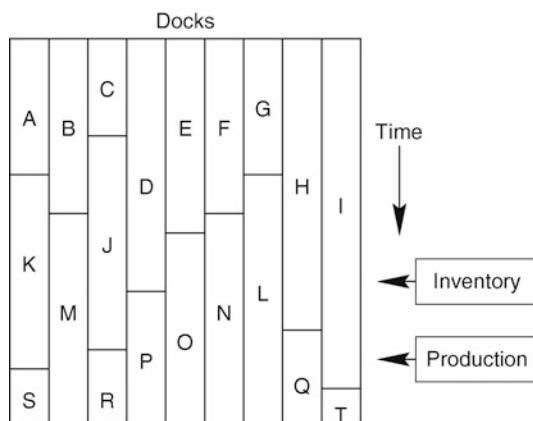
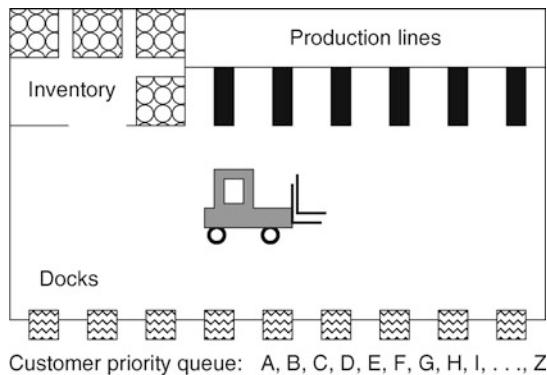
$$\text{obj} = \frac{(\mathcal{M} - \mu_{\mathcal{M}})}{\sigma_{\mathcal{M}}} + \frac{(I - \mu_I)}{\sigma_I} \quad (9)$$

where I represents running average inventory, \mathcal{M} represents order mean time at dock, while μ and σ represent the respective means and standard deviations over a set of solutions.

In Table 1, results are given for the Genitor steady-state genetic algorithm compared to a stochastic hill-climber (Watson et al. 1999). Mean time at dock and average inventory are reported, also with performance and standard deviations over 30 runs. The move operator for the hill-climber was an “exchange operator.” This operator selects two random customers and then swaps their position in the permutation. All of the algorithms reported here

Fig. 9

The warehouse model includes production lines, inventory, and docks. The columns in the schedule represent different docks. Customer orders are assigned to a dock left to right and product is drawn from inventory and the production lines.

**Table 1**

Performance results. The final column indicates a human-generated solution used by Coors

	Genetic algorithm	Hill climber	Coors
Mean time-at-dock			
μ	392.49	400.14	437.55
σ	0.2746	4.7493	n.a.
Average inventory			
μ	364,080	370,458	549,817
σ	1,715	20,674	n.a.

used 100,000 function evaluations. The Genitor algorithm used a population size of 500 and a selective pressure of 1.1.

For our test data, we have an actual customer order sequence developed and used by Coors personnel to fill customer orders. This solution produced an average inventory of 549,817.25 product units and an order mean time of 437.55 min at dock.

Genitor was able to improve the mean time at dock by approximately 9%. The big change, however, is in average inventory. Both Genitor and the Hill Climber show a dramatic reduction in average inventory, meaning more product came out of inventory and that the schedule did a better job of directly loading product off the line. Watson et al. (1999) provide a more detailed discussion of the Coors warehouse scheduling application.

8 Conclusions

This paper has presented a broad survey of both theoretical and practical work related to genetic algorithms. For the reader who may be interested in using genetic algorithms to solve a particular problem, two comments might prove to be useful.

First, the details matter. A small change in representation or a slight modification in how an algorithm is implemented can change algorithm performance. The literature is full of papers that claim one algorithm is better than the other. What is often not obvious is how hard the researchers had to work to get good performance and how sensitive the results are to tuning the search algorithms? Are the benchmarks representative of real-world problems, and do the results generalize?

Second, genetic algorithms are a general purpose approach. An application-specific solution will almost always be better than a general purpose solution. Of course, in virtually every real-world application, application-specific knowledge gets integrated into the evaluation function, the operators, and the representation. Doing this well takes a good deal of experience and intuition about how to solve optimization and search problems which makes the practice of search algorithm design somewhat of a specialized art. This observation leads back to the first point: the details matter, and a successful implementation usually involves keen insight into the problem.

Acknowledgments

This research was partially supported by a grant from the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA9550-08-1-0422. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon. Funding was also provided by the Coors Brewing Company, Golden, Colorado.

References

- Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, Oxford
- Baker J (1987) Reducing bias and inefficiency in the selection algorithm. In: Grefenstette J (ed) GAs and their applications: 2nd international conference, Erlbaum, Hillsdale, NJ, pp 14–21
- Bitner JR, Ehrlich G, Reingold EM (1976) Efficient generation of the binary reflected gray code and its applications. Commun ACM 19 (9):517–521
- Brent R (1973) Algorithms for minimization with derivatives. Dover, Mineola, NY

- Bridges C, Goldberg D (1987) An analysis of reproduction and crossover in a binary coded genetic algorithm. In: Grefenstette J (ed) *GAs and their applications: 2nd international conference*, Erlbaum, Cambridge, MA
- Davis L (1985a) Applying adaptive algorithms to epistatic domains. In: *Proceedings of the IJCAI-85*, Los Angeles, CA
- Davis L (1985b) Job shop scheduling with genetic algorithms. In: Grefenstette J (ed) *International conference on GAs and their applications*. Pittsburgh, PA, pp 136–140
- Davis L (1991) *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York
- DeJong K (1993) Genetic algorithms are NOT function optimizers. In: Whitley LD (ed) *FOGA – 2*, Morgan Kaufmann, Los Altos, CA, pp 5–17
- Eshelman L (1991) The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Rawlins G (ed) *FOGA – 1*, Morgan Kaufmann, Los Altos, CA, pp 265–283
- Eshelman L, Schaffer D (1991) Preventing premature convergence in genetic algorithms by preventing incest. In: Booker L, Belew R (eds) *Proceedings of the 4th international conference on GAs*. Morgan Kaufmann, San Diego, CA
- Goldberg D (1987) Simple genetic algorithms and the minimal, deceptive problem. In: Davis L (ed) *Genetic algorithms and simulated annealing*. Pitman/ Morgan Kaufmann, London, UK, chap 6
- Goldberg D (1989a) Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Syst* 3:153–171
- Goldberg D (1989b) *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, MA
- Goldberg D (1990) A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Tech. Rep. Nb. 90003*. Department of Engineering Mechanics, University of Alabama, Tuscaloosa, AL
- Goldberg D, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins G (ed) *FOGA – 1*, Morgan Kaufmann, San Mateo, CA, pp 69–93
- Goldberg D, Lingle R (1985) Alleles, loci, and the traveling salesman problem. In: Grefenstette J (ed) *International conference on GAs and their applications*. London, UK, pp 154–159
- Grefenstette J (1993) Deception considered harmful. In: Whitley LD (ed) *FOGA – 2*, Morgan Kaufmann, Vail, CO, pp 75–91
- Hansen N (2006) The CMA evolution strategy: a comparing review. In: *Toward a new evolutionary computation: advances on estimation of distribution algorithms*. Springer, Heidelberg, Germany, pp 75–102
- Hansen N (2008) Adaptive encoding: how to render search coordinate system invariant. In: *Proceedings of 10th international conference on parallel problem solving from nature*. Springer, Dortmund, Germany, pp 205–214
- Heckendorn R, Rana S, Whitley D (1999a) Polynomial time summary statistics for a generalization of MAXSAT. In: *GECCO-99*, Morgan Kaufmann, San Francisco, CA, pp 281–288
- Heckendorn R, Rana S, Whitley D (1999b) Test function generators as embedded landscapes. In: *Foundations of genetic algorithms FOGA – 5*, Morgan Kaufmann, Los Altos, CA
- Heckendorn RB, Whitley LD, Rana S (1996) Nonlinearity, Walsh coefficients, hyperplane ranking and the simple genetic algorithm. In: *FOGA – 4*, San Diego, CA
- Ho Y (1994) Heuristics, rules of thumb, and the 80/20 proposition. *IEEE Trans Automat Cont* 39(5): 1025–1027
- Ho Y, Sreenivas RS, Vakili P (1992) Ordinal optimization of discrete event dynamic systems. *Discrete Event Dyn Syst* 2(1):1573–7594
- Holland J (1975) *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI
- Holland JH (1992) *Adaptation in natural and artificial systems*, 2nd edn. MIT Press, Cambridge, MA
- Schaffer JD, Eshelman L (1993) Real-coded genetic algorithms and interval schemata. In: Whitley LD (ed) *FOGA – 2*, Morgan Kaufmann, Los Altos, CA
- Mathias KE, Whitley LD (1994) Changing representations during search: a comparative study of delta coding. *J Evolut Comput* 2(3):249–278
- Nagata Y, Kobayashi S (1997) Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. In: Bäck T (ed) *Proceedings of the 7th international conference on GAs*, Morgan Kaufmann, California, pp 450–457
- Nix A, Vose M (1992) Modelling genetic algorithms with Markov chains. *Ann Math Artif Intell* 5:79–88
- Poli R (2005) Tournament selection, iterated coupon-collection problem, and backward-chaining evolutionary algorithms. In: *Foundations of genetic algorithms*, Springer, Berlin, Germany, pp 132–155
- Radcliffe N, Surry P (1995) Fundamental limitations on search algorithms: evolutionary computing in perspective. In: van Leeuwen J (ed) *Lecture notes in computer science*, vol 1000, Springer, Berlin, Germany
- Rana S, Whitley D (1997) Representations, search and local optima. In: *Proceedings of the 14th national conference on artificial intelligence AAAI-97*. MIT Press, Cambridge, MA, pp 497–502
- Rana S, Heckendorn R, Whitley D (1998) A tractable Walsh analysis of SAT and its implications for genetic algorithms. In: *AAAI98*, MIT Press, Cambridge, MA, pp 392–397

- Rosenbrock H (1960) An automatic method for finding the greatest or least value of a function. *Comput J* 3:175–184
- Salomon R (1960) Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. *Biosystems* 39(3):263–278
- Schaffer JD (1987) Some effects of selection procedures on hyperplane sampling by genetic algorithms. In: Davis L (ed) *Genetic algorithms and simulated annealing*. Morgan Kaufmann, San Francisco, CA, pp 89–130
- Schwefel HP (1981) *Numerical optimization of computer models*. Wiley, New York
- Schwefel HP (1995) *Evolution and optimum seeking*. Wiley, New York
- Sokolov A, Whitley D (2005) Unbiased tournament selection. In: Proceedings of the 7th genetic and evolutionary computation conference. The Netherlands, pp 1131–1138
- Spears W, Jong KD (1991) An analysis of multi-point crossover. In: Rawlins G (ed) *FOGA – 1*, Morgan Kaufmann, Los Altos, CA, pp 301–315
- Starkweather T, Whitley LD, Mathias KE (1990) Optimization using distributed genetic algorithms. In: Schwefel H, Männer R (eds) *Parallel problem solving from nature*. Springer, London, UK, pp 176–185
- Starkweather T, McDaniel S, Mathias K, Whitley D, Whitley C (1991) A comparison of genetic sequencing operators. In: Booker L, Belew R (eds) *Proceedings of the 4th international conference on GAs*. Morgan Kaufmann, San Mateo, CA, pp 69–76
- Suh J, Gucht DV (1987) Distributed genetic algorithms. Tech. rep., Indiana University, Bloomington, IN
- Syswerda G (1989) Uniform crossover in genetic algorithms. In: Schaffer JD (ed) *Proceedings of the 3rd international conference on GAs*, Morgan Kaufmann, San Mateo, CA
- Syswerda G (1991) Schedule optimization using genetic algorithms. In: Davis L (ed) *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, chap 21
- Syswerda G, Palmucci J (1991) The application of genetic algorithms to resource scheduling. In: Booker L, Belew R (eds) *Proceedings of the 4th international conference on GAs*, Morgan Kaufmann, San Mateo, CA
- Vose M (1993) Modeling simple genetic algorithms. In: Whitley LD (ed) *FOGA – 2*, Morgan Kaufmann, San Mateo, CA, pp 63–73
- Vose M (1999) *The simple genetic algorithm*. MIT Press, Cambridge, MA
- Vose M, Liepins G (1991) Punctuated equilibria in genetic search. *Complex Syst* 5:31–44
- Vose M, Wright A (1997) Simple genetic algorithms with linear fitness. *Evolut Comput* 2(4):347–368
- Watson JP, Rana S, Whitley D, Howe A (1999) The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling. *J Scheduling* 2(2):79–98
- Whitley D (1999) A free lunch proof for gray versus binary encodings. In: GECCO-99, Morgan Kaufmann, Orlando, FL, pp 726–733
- Whitley D, Kauth J (1988) GENITOR: A different genetic algorithm. In: *Proceedings of the 1988 Rocky Mountain conference on artificial intelligence*, Denver, CO
- Whitley D, Rowe J (2008) Focused no free lunch theorems. In: GECCO-08, ACM Press, New York
- Whitley D, Yoo NW (1995) Modeling permutation encodings in simple genetic algorithm. In: Whitley D, Vose M (eds) *FOGA – 3*, Morgan Kaufmann, San Mateo, CA
- Whitley D, Starkweather T, Fuquay D (1989) Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: Schaffer JD (ed) *Proceedings of the 3rd international conference on GAs*. Morgan Kaufmann, San Francisco, CA
- Whitley D, Das R, Crabb C (1992) Tracking primary hyperplane competitors during genetic search. *Ann Math Artif Intell* 6:367–388
- Whitley D, Beveridge R, Mathias K, Graves C (1995a) Test driving three 1995 genetic algorithms. *J Heuristics* 1:77–104
- Whitley D, Mathias K, Pyeatt L (1995b) Hyperplane ranking in simple genetic algorithms. In: Eshelman L (ed) *Proceedings of the 6th international conference on GAs*. Morgan Kaufmann, San Francisco, CA
- Whitley D, Mathias K, Rana S, Dzubera J (1996) Evaluating evolutionary algorithms. *Artif Intell J* 85:1–32
- Whitley LD (1989) The GENITOR algorithm and selective pressure: why rank based allocation of reproductive trials is best. In: Schaffer JD (ed) *Proceedings of the 3rd international conference on GAs*. Morgan Kaufmann, San Francisco, CA, pp 116–121
- Whitley LD (1991) Fundamental principles of deception in genetic search. In: Rawlins G (ed) *FOGA – 1*, Morgan Kaufmann, San Francisco, CA, pp 221–241
- Whitley LD (1993) An executable model of the simple genetic algorithm. In: Whitley LD (ed) *FOGA – 2*, Morgan Kaufmann, Vail, CO, pp 45–62
- Wolpert DH, Macready WG (1995) No free lunch theorems for search. Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM

22 Evolutionary Strategies

Günter Rudolph

Department of Computer Science, TU Dortmund, Dortmund, Germany
guenter.rudolph@tu-dortmund.de

1	<i>Historical Roots</i>	674
2	<i>Algorithmic Skeleton: The ES Metaheuristic</i>	675
3	<i>Design Principles: Instantiation of the Metaheuristic</i>	676
4	<i>Advanced Adaptation Techniques in \mathbb{R}^n</i>	681
5	<i>Further Reading</i>	696

Abstract

Evolutionary strategies (ES) are part of the all-embracing class of evolutionary algorithms (EA). This chapter presents a compact tour through the developments regarding ES from the early beginning up to the recent past before advanced techniques and the state-of-the-art techniques are explicated in detail. The characterizing features that distinguish ES from other subclasses of EA are discussed as well.

Prelude

This chapter on evolutionary strategies (ES) has a number of predecessors (Rechenberg 1978; Bäck et al. 1991; Schwefel and Rudolph 1995; Beyer and Schwefel 2002), each of them is still worth reading as they reveal which techniques have been developed in the course of time and which methods were considered as the state of the art at that particular time. Here, the aim is to update the comprehensive overview provided in Beyer and Schwefel (2002) to the present state of the art. To this end, a condensed version of Beyer and Schwefel (2002) is presented in [Sects. 1–3](#) before advanced techniques and the current standard are introduced in [Sect. 4](#). Finally, [Sect. 5](#) reports ongoing developments that may become part of a subsequent state-of-the-art survey on evolutionary strategies in the near future.

1 Historical Roots

In the 1960s, difficult multimodal and noisy optimization problems from engineering sciences awaited their solution at the Technical University of Berlin, Germany. At that time, mathematical models or numerical simulations of these particular problems were not available. As a consequence, the optimization had to be done experimentally with the real object at the hardware level. Three students, Peter Bienert, Ingo Rechenberg, and Hans-Paul Schwefel, envisioned an automated cybernetic system that alters the object parameters mechanically or electrically, runs the experiment, measures the outcome of the experiment, and uses this information in the context of some optimization strategy for the decision on how to alter the object parameters of the real object for the experiment in the subsequent iteration. Obvious candidates for the optimization methods in the framework of this early “hardware-in-the-loop” approach were all kinds of gradient-like descent methods for minimization tasks. But these methods failed. Inspired by lectures on biological evolution, they tried a randomized method that may be seen as the simplest algorithm driven by mutation and selection – a method nowadays known as $(1 + 1)$ -ES. This approach was successful (Rechenberg 1965) and it was extended in various directions including first theoretical results (Schwefel 1964, 1977; Rechenberg 1973) during the next decade. After some years of only few novelties, the research activities gained momentum again in the late 1980s making evolutionary strategies an active field of research since that time.

Since early theoretical publications mainly analyzed simple ES without recombination, somehow the myth arose that ES put more emphasis on mutation than on recombination. This is a fatal misconception! Recombination has been an important ingredient of ES from the very beginning and this is still valid today.

2 Algorithmic Skeleton: The ES Metaheuristic

A metaheuristic may be defined as an algorithmic framework of iterative nature that specifies the sequence of abstract operations. Only if the abstract operations are instantiated by concrete operations the metaheuristic turns into a real algorithm. As a consequence, many variants of instantiated metaheuristics are possible. And this holds true also for the metaheuristic termed an *evolutionary algorithm* that works as follows:

Algorithm 1 Metaheuristic: Evolutionary Algorithm

```

initialize population of individuals
evaluate all individuals by fitness function
repeat
    select individuals (parents) for reproduction
    vary selected individuals in randomized manner to obtain new individuals (offspring)
    evaluate offspring by fitness function
    select individuals for survival from offspring and possibly parents according to fitness
until stopping criterion fulfilled

```

The metaheuristic known as *evolutionary strategies* narrows the degrees of freedom in the instantiation of the EA metaheuristic. These characterizing features are

1. Selection for reproduction is unbiased.
2. Selection for survival is ordinal and deterministic.
3. Variation operators are parametrized and hence controllable.
4. Individuals consist of candidate solution and control parameters.

This leads to the ES metaheuristic:

Algorithm 2 Metaheuristic: Evolutionary Strategy

```

initialize population of individuals
evaluate all individuals by fitness function
repeat
    select parents for reproduction uniformly at random
    vary selected individuals in randomized manner to obtain offspring
    evaluate offspring by fitness function
    rank offspring and possibly parents according to fitness
    select individuals with best ranks for survival
until stopping criterion fulfilled

```

Next, the ES metaheuristic is narrowed further and specified in a more formal manner by following Schwefel and Rudolph (1995) in essence. A population at generation $t \geq 0$ is denoted by $P^{(t)}$. An individual $p \in P^{(t)}$ is a pair $p = (x, \Psi)$ where $x \in X$ is an element of the feasible set X and Ψ is a finite set of strategy parameters of arbitrary kind. The objective function $f: X \rightarrow \mathbb{R}$ is termed fitness function and it maps an element from the feasible set X to a value in \mathbb{R} . Now we are in the position to specify the $(\mu / \rho, \kappa, \lambda)$ -ES, where μ denotes the number of parents, λ the number of offspring, ρ the number of parents participating in

Algorithm 3 $(\mu/\rho, \kappa, \lambda)$ -ES

```

initialize population  $P^{(0)}$  with  $\mu$  individuals
 $\forall p \in P^{(0)}$ : set  $p.\Psi.\text{age} = 1$ 
set  $t = 0$ 
repeat
   $Q^{(t)} = \{\}$ 
  for  $i = 1$  to  $\lambda$  do
    select  $\rho$  parents  $p_1, \dots, p_\rho \in P^{(t)}$  uniformly at random
     $q = \text{variation } (p_1, \dots, p_\rho)$  with  $q.\Psi.\text{age} = 0$ 
     $Q^{(t)} = Q^{(t)} \cup \{q\}$ 
  end for
   $P^{(t+1)} = \text{selection of } \mu \text{ best individuals from } Q^{(t)} \cup \{p \in P^{(t)} : p.\Psi.\text{age} < \kappa\}$ 
   $\forall p \in P^{(t+1)}$ : increment  $p.\Psi.\text{age}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

Table 1

Default parameterization associated with original shorthand notation

Algorithm	Parents	Mates	Offspring	Age
$(1 + 1)$	$\mu = 1$	$\rho = 1$	$\lambda = 1$	$\kappa = \infty$
$(1 + \lambda)$	$\mu = 1$	$\rho = 1$	$\lambda \geq 1$	$\kappa = \infty$
$(1, \lambda)$	$\mu = 1$	$\rho = 1$	$\lambda \geq 2$	$\kappa = 1$
$(\mu + 1)$	$\mu \geq 2$	$\rho = 2$	$\lambda = 1$	$\kappa = \infty$
$(\mu + \lambda)$	$\mu \geq 2$	$\rho = 2$	$\lambda \geq 2$	$\kappa = \infty$
(μ, λ)	$\mu \geq 2$	$\rho = 2$	$\lambda \geq \mu$	$\kappa = 1$

producing an offspring, and κ the maximum age of an individual. Thus, we have $\mu, \rho, \lambda \in \mathbb{N}$ and $\kappa \in \mathbb{N} \cup \{\infty\}$ with $\rho \leq \mu$ and, if setting $\kappa < \infty$, we require $\lambda > \mu$.

Prior to 1995, the age parameter was not part of an individual. Only the special cases $\kappa = 1$ (the “comma”-strategy) and $\kappa = \infty$ (the “plus” strategy) were in deployment. **Table 1** contains the default parameterization associated with the original shorthand notation. For example, $(\mu + \lambda)$ -ES usually stands for a $(\mu / 2, \infty, \lambda)$ -ES.

3 Design Principles: Instantiation of the Metaheuristic

In the course of instantiating the $(\mu / \rho, \kappa, \lambda)$ -ES to a real optimization algorithm, three tasks have to be settled:

1. Choice of an appropriate problem representation
2. Choice (and possibly design) of variation operators acting on the problem representation
3. Choice of strategy parameters (includes initialization)

Subsequently, these tasks will be elucidated in more detail.

3.1 Representation

In the ES community, there is no doctrine concerning the usage of a special representation. Rather, the credo is to use the most natural representation for the problem under consideration. For example, if the decision variables only can attain the value 0 and 1 then a bit string is used as representation; and if all the variables can attain real values then a real vector is used as representation. If some variables are binary, some integer-valued and some real-valued then the representation will be a compound or composite representation, that is, a tuple from $\mathbb{B}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{R}^{n_3}$. This flexibility in the representation requires a larger number of variation operators than the fixed-representation case.

3.2 Variation

Suppose the representation has been chosen, that is, the candidate solution x of an individual (x, Ψ) is an element of some n -fold Cartesian product X . Long-time practical experiences within the ES community in designing variation operators has led to three design guidelines:

1. Reachability

Every solution $x \in X$ should be reachable from an arbitrary solution $x_0 \in X$ after a finite number of repeated applications of the variation operator with positive probability bounded from zero.

2. Unbiasedness

Unless knowledge about the problem has been gathered, the variation operator should not favor a particular subset of solutions. Formally, this can be achieved by choosing the maximum entropy distribution that obeys knowledge about the problem as constraints. As can be seen from some examples given later, this automatically leads to distributions obeying the principle of strong causality, that is, small variations are more likely than large variations.

3. Control

The variation operator should have parameters that affect the shape of the distribution. As known from theory, when approaching the optimal solution, the strength of variation must be weakened steadily.

These design guidelines will now be exemplified for binary, integer, and real search spaces.

3.2.1 Binary Search Space

An evolution strategy with $X = \mathbb{B}^n$ was used by Rechenberg (1973) to minimize the linear pseudoboolean function $f(x) = \|x - x^*\|_1$ for some given pattern $x^* \in \mathbb{B}^n$. The variation operators are recombination and mutation.

Let $x, y \in \mathbb{B}^n$ and $k \in \{1, 2, \dots, n-1\}$ uniformly at random. An offspring $z \in \mathbb{B}^n$ is said to be created/recombined by *1-point crossover* if $z_i = x_i$ for $i \leq k$ and $z_i = y_i$ for $i > k$. Offspring z is generated by *uniform crossover* if for each i holds $z_i = x_i$ or $z_i = y_i$ with equal probability. The standard method for mutation flips every bit independently with some mutation probability $p_m \in (0, 1)$.

Do these operators obey the design guidelines? As for reachability: regardless of the result of the recombination operator, the mutation operator can move from every $x \in \mathbb{B}^n$ to an

arbitrary $y \in \mathbb{B}^n$ in a single step with positive probability $p^{H(x,y)} (1-p)^{n-H(x,y)} > 0$. Verifying the unbiasedness requires the definition of a maximum entropy distribution.

Definition 1 Let X be a discrete random variable with $p_k = \mathbb{P}\{X = x_k\}$ for some index set K . The quantity

$$H(X) = - \sum_{k \in K} p_k \log p_k$$

is called the *entropy* of the distribution of X . If X is a continuous random variable with density $f_X(\cdot)$, the entropy is given by

$$H(X) = - \int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx$$

The distribution of a random variable X for which $H(X)$ attains its maximum value is termed the *maximum entropy distribution*. \square

Example 1 Let $p_k = \mathbb{P}\{X = k\}$ for $k = 1, \dots, n$. We are seeking values p_k such that $H(X)$ becomes maximal under the constraint that the sum of the p_k must be 1. This leads to the Lagrangian function

$$L(p, a) = - \sum_{k=1}^n p_k \log p_k + a \left(\sum_{k=1}^n p_k - 1 \right)$$

with partial derivatives

$$\frac{\partial L(p, a)}{\partial p_k} = -1 - \log p_k + a \stackrel{!}{=} 0 \quad (1)$$

and

$$\frac{\partial L(p, a)}{\partial a} = \sum_{k=1}^n p_k - 1 \stackrel{!}{=} 0 \quad (2)$$

Rearrangement of [Eq. 1](#) yields $p_k \stackrel{!}{=} e^{a-1}$, which can be substituted in [Eq. 2](#) leading to

$$\sum_{k=1}^n p_k = \sum_{k=1}^n e^{a-1} = n e^{a-1} \stackrel{!}{=} 1$$

and finally to $p_k \stackrel{!}{=} e^{a-1} \stackrel{!}{=} \frac{1}{n}$. Thus we have shown that the discrete uniform distribution has maximum entropy. \square

The cutpoint k in case of 1-point crossover was drawn uniformly at random among all possible locations. This is the maximum entropy distribution and no cut point has been given any preference. In case of uniform crossover, each bit is selected from the two parents with equal probability. Again, this is the maximum entropy distribution.

Mutations are realized by flipping each bit independently with some probability p_m . This is the maximum entropy distribution for given p_m . But if the distribution of the number K of bit flips is considered, the situation changes. Notice that $K \sim B(n, p_m)$ has binomial distribution with expectation $E[K] = n p_m$ and variance $V[X] = n p_m (1 - p_m)$. The maximum entropy

distribution for a given expectation would be a Boltzmann distribution. If the variance is also given, then one obtains an apparently nameless distribution.

Finally, control of the mutation operator is ensured as we can change the value of p_m . Since recombination and mutation together form the variation operation, the design guideline of providing a control parameter is fulfilled.

3.2.2 Integer Search Space

The first ES for integer search space \mathbb{Z}^n was developed by Schwefel (1964) where mutations were binomially distributed. Here, we shall develop our a mutation operator according to the design guidelines (Rudolph 1994).

Notice that 1-point and uniform crossover can be defined analogous to the binary case. Reachability can be ensured if the support of our mutation distribution is \mathbb{Z}^n . If the random integer vector is to be generated by drawing each component independently with the same distribution, then it suffices to find the maximum entropy distribution with support \mathbb{Z} .

Thus, the question addressed here is: Which discrete distribution with support \mathbb{Z} is symmetric with respect to 0, has a mean deviation $m > 0$ and possesses maximum entropy? The answer requires the solution of the following nonlinear optimization problem:

$$-\sum_{k=-\infty}^{\infty} p_k \log p_k \rightarrow \max!$$

subject to

$$p_k = p_{-k} \quad \forall k \in \mathbb{Z} \quad (3)$$

$$\sum_{k=-\infty}^{\infty} p_k = 1 \quad (4)$$

$$\sum_{k=-\infty}^{\infty} |k| p_k = m \quad (5)$$

$$p_k \geq 0 \quad \forall k \in \mathbb{Z} \quad (6)$$

We may neglect condition (6), if the solution of the surrogate problem fulfills these inequalities. We may therefore differentiate the Lagrangian

$$L(p, a, b) = -\sum_{k=-\infty}^{\infty} p_k \log p_k + a \cdot \left(\sum_{k=-\infty}^{\infty} p_k - 1 \right) + b \cdot \left(\sum_{k=-\infty}^{\infty} |k| p_k - m \right)$$

to obtain the necessary condition (4) and (5) and $-1 - \log p_k + a + b|k| = 0$ or alternatively

$$p_k = e^{a-1} \cdot (e^b)^{|k|} \quad \forall k \in \mathbb{Z} \quad (7)$$

Exploitation of the symmetry condition (3) and substitution of Eq. 7 in Eq. 4 leads to

$$\sum_{k=-\infty}^{\infty} p_k = p_0 + 2 \cdot \sum_{k=1}^{\infty} p_k = e^{a-1} \cdot \left(1 + 2 \cdot \sum_{k=1}^{\infty} (e^b)^{|k|} \right) = 1 \quad (8)$$

Let $q = e^b < 1$ so that $S(q) := \sum_{k=1}^{\infty} q^{|k|} < \infty$ and $q \cdot S'(q) = \sum_{k=1}^{\infty} |k|q^{|k|}$. Then condition (Eq. 8) reads

$$e^{1-a} = 1 + 2 \cdot S(q) = \frac{1+q}{1-q} \quad (9)$$

Substitution of Eq. 7 in Eq. 5 yields

$$\sum_{k=-\infty}^{\infty} |k|p_k = 2 \cdot \sum_{k=1}^{\infty} |k|p_k = 2e^{a-1} \cdot \sum_{k=1}^{\infty} |k|q^{|k|} = 2e^{a-1} \cdot q \cdot S'(q) = m$$

so that with substitution of (Eq. 9) one obtains

$$p = 1 - \frac{m}{(1+m^2)^{1/2} + 1} \quad (10)$$

with $q = 1 - p$. Substitution of Eq. 9 in Eq. 7 yields

$$p_k = \frac{p}{2-p} (1-p)^{|k|}$$

which is the maximum entropy distribution. It can be shown (Rudolph 1994) that a random number with this distribution can be generated by the difference $G_1 - G_2$ of two independent geometrically distributed random numbers G_1 and G_2 with parameter p . Thus, random numbers according to the maximum entropy distribution are easy to generate and the variability of distribution can be controlled by choosing the mean deviation $m > 0$ in Eq. 10.

3.2.3 Continuous Search Space

The continuous search space \mathbb{R}^n imposed the necessity on ES researchers to develop highly sophisticated variation operators. These will be presented in detail in Sect. 4. Here, focus is on the basics only.

As for recombination, 1-point crossover or uniform crossover (also called discrete or dominant recombination) can be implemented analogous to the binary or integer case. Another common recombination operator is termed *intermediary recombination*: Two parents $x, y \in \mathbb{R}^n$ generate offspring $z = \frac{1}{2}(x+y)$ by averaging. This can be generalized to $\rho \geq 2$ parents easily. A randomized version of intermediary recombination would draw a random number u uniformly distributed in $[0, 1]$ so that the offspring $z = ux + (1-u)y$ will be located somewhere on the line between the two parents. Notice that this is a maximum entropy distribution.

Before the choice of mutation distribution is discussed, some terminology has to be introduced.

Definition 2 Let $X = (X_1, X_2, \dots, X_n)'$ be an n -dimensional random vector. The vector

$$\mathbb{E}[X] = (\mathbb{E}[X_1], \mathbb{E}[X_2], \dots, \mathbb{E}[X_n])'$$

is termed its *expectation vector* whereas the $n \times n$ -matrix

$$\text{Cov}[X] = \mathbb{E}[X - \mathbb{E}[X]]\mathbb{E}[X - \mathbb{E}[X]]'$$

is said to be its *covariance matrix*.

□

Matrices are extensively used not only for compact notation (Abadir and Magnus 2005).

Definition 3 A square matrix $A \in \mathbb{R}^{n,n}$ is *symmetric* if $A = A'$, *orthogonal* if $A'A = AA' = I_n$ (where I_n is the unit matrix), *positive definite* (p.d.) if $x'Ax > 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$, *positive semidefinite* (p.s.d.) if $x'Ax \geq 0$ for all $x \in \mathbb{R}^n$, *regular* if $\det(A) \neq 0$ and *singular* if $\det(A) = 0$. \square

Usually, the mutation distribution of ES is the multivariate normal distribution.

Definition 4 (see Rao 1973, p. 518 or Tong 1990, p. 29) An n -dimensional random vector X with $E[X] = v$ and $Cov[X] = \Sigma$ is *multivariate normal* if and only if $c'X$ is univariate normal with $E[c'X] = c' E[X]$ and $Cov[c'X] = c'\Sigma c$ for all $c \in \mathbb{R}^n \setminus \{0\}$. \square

This definition includes also the case if the covariance matrix is only positive semidefinite. If Σ is positive definite, the distribution has a probability density function

$$f_X(x) = \frac{1}{(2\pi)^{n/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x - v)' \Sigma^{-1} (x - v)\right) \quad (11)$$

for $x \in \mathbb{R}^n$ (as p.d. matrices are regular which in turn can be inverted). Since the support of the distribution is \mathbb{R}^n , every point in \mathbb{R}^n can be reached in principle in a single mutation. Moreover, since the multivariate normal distribution has maximum entropy for given expectation vector and covariance matrix (Rao 1973), the requirement of unbiasedness can be fulfilled. But as long as nothing is known about the problem, the covariance matrix should be the unit matrix I_n . The distribution can be controlled by parameter $\sigma > 0$ if a mutation of the current individual $X^{(t)} \in \mathbb{R}^n$ at generation $t \geq 0$ is done as follows: $X^{(t+1)} = X^{(t)} + \sigma Z$ where Z is drawn from a multivariate normal distribution with zero mean and covariance matrix I_n , denoted by $Z \sim N(0, I_n)$. Mechanisms for controlling the mutation distribution are presented in **Sect. 4**.

3.3 Parameterization

The parameterization requires not only the fixation of μ , λ , ρ , and κ but also other parameters that may be specific to the variation operators deployed. This task demands skill, experience, intuition, and instinct. Fortunately, there are methods that facilitate this task (Bartz-Beielstein 2006; Smit and Eiben 2009). Moreover, we have to decide how to initialize our population: This may happen uniformly at random in a certain search region, by stratified or Latin hypercube sampling (McKay et al. 1979) or in accordance with standard NLP methods at/around some given start position if no range can be specified in advance.

4 Advanced Adaptation Techniques in \mathbb{R}^n

All control or adaptation mechanisms tacitly presume that the operational reach of the ES is within a vicinity of its current position $\tilde{x} \in \mathbb{R}^n$ where the fitness function can be appropriately approximated by a second order Taylor expansion if the problem was differentiable, that is,

$$f(x) \approx f(\tilde{x}) + (x - \tilde{x})' \nabla f(\tilde{x}) + \frac{1}{2} (x - \tilde{x})' \nabla^2 f(\tilde{x}) (x - \tilde{x})$$

where $\nabla f(\tilde{x})$ is the gradient and $\nabla^2 f(\tilde{x})$ the Hessian matrix, which is symmetric and positive definite. If the fitness function is quadratic, then the Taylor expansion is exact and the Hessian matrix contains information about the scaling and orientation of the iso-hyperellipsoids that specify positions in the search space with the same fitness value. If the Taylor expansion is not exact, then the Hessian matrix provides at least a reasonable approximation.

Suppose without loss of generality that $f(x) = \frac{1}{2} x' A x$ with p.d. and symmetric square matrix A . Owing to part (a) of Theorem 1 below, we can decompose matrix A via $A = T D T'$.

Theorem 1 (see Abadir and Magnus 2005) *Let $A \in \mathbb{R}^{n,n}$ be a p.d. matrix.*

- (a) *There exist an orthogonal matrix $T \in \mathbb{R}^{n,n}$ and a diagonal matrix $D \in \mathbb{R}^{n,n}$ with positive diagonal entries such that $A = T D T'$ (spectral decomposition), where the columns of T are the eigenvectors of A and the diagonal entries in D are the associated eigenvalues. Moreover, the decomposition can be written as*

$$A = \sum_{i=1}^n d_i t_i t_i'$$

where d_i and t_i are the eigenvalues in D and eigenvectors in T , respectively.

- (b) *There exists a lower triangular matrix R with positive diagonal entries such that $A = R R'$ (Cholesky decomposition). \square*

With this decomposition, the fitness function can be formulated as follows

$$f(x) = \frac{1}{2} x' A x = \frac{1}{2} x' T D T' x = \frac{1}{2} (T' x)' D (T' x) = \frac{1}{2} y' D y = \frac{1}{2} \sum_{i=1}^n d_i y_i^2 \quad (12)$$

where $y = T' x$ describes the rotation of the original coordinate system by orthogonal matrix T' . Now recall that the surface of the unit hyperellipsoid \mathcal{E}_n centered in the origin of \mathbb{R}^n is defined by the set

$$\mathcal{E}_n = \left\{ y \in \mathbb{R}^n : \sum_{i=1}^n \frac{y_i^2}{a_i^2} = 1 \right\}$$

where the $a_i > 0$ are the lengths of the semi-principal axes from the origin to the intersection with the associated axis of the coordinate system. Inspection of [Eq. 12](#) reveals that the eigenvalues of A , that is, the diagonal entries in D , determine the lengths of the semi-principal axes of the ellipsoid: $a_i = \sqrt{2/d_i}$ for $i = 1, \dots, n$. Thus, the smallest eigenvalue of A is associated with the longest semi-principal axis and hence to the strongest curvature of the ellipsoid.

Not surprisingly, the curvature will have some impact on the performance and hence on the control mechanism of the mutation distribution of the ES. This fact will be discussed in detail in this section. The following general result reveals how we can generate arbitrarily shaped multivariate normal distributions.

Theorem 2 (see Rao 1973, p. 522 or Tong 1990, p. 32)

If $X \sim N_n(v, \Sigma)$ and $Y = AX + b$ where $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$, then

$$Y \sim N_m(Av + b, A\Sigma A')$$

\square

For our purposes it suffices to consider the special case with $m = n$, $v = b = 0$ and $\Sigma = I_n$.

Corollary 1 If $X \sim N_n(0, I_n)$ and $Y = AX$ with $A \in \mathbb{R}^{n,n}$ then $Y \sim N_n(0, AA')$. \square

But as long as nothing is known about the problem, one should choose $A = I_n$, which leads to a spherical distribution, that is, points with the same probability are located on the surface of a hypersphere. If A is a diagonal matrix with nonidentical diagonal entries, then the resulting distribution is elliptical and the principal-axes of the ellipsoid with same probabilities are parallel to the axes of the coordinate system. In the general case, the ellipsoid with same probabilities is also rotated.

Apart from the shape of the mutation distribution, its variance has to be controlled. The next result reveals how the mutations' variance is related to a random step length of the ES:

Theorem 3 (see Fang et al. 1990) If $X \sim N_n(0, \sigma^2 I_n)$ then $X \stackrel{d}{=} r \cdot u_n$ where $r \stackrel{d}{=} \|X\|$ is a scalar random variable with $\chi_n(\sigma)$ -distribution and u_n is an n -dimensional random vector that is uniformly distributed on the surface of the unit hypersphere, i.e., $\|u_n\| = 1$ with probability 1. Moreover, r and u_n are independent. \square

Thus, a mutation according to a multivariate normal distribution can be interpreted as a step in random direction u_n with random step length r . It can be shown (see, e.g., Rudolph 1997, p. 21) that

$$\mathbb{E}[\|X\|] = \sigma \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} \sqrt{2} \approx \sigma \sqrt{n - \frac{1}{2} + \frac{1}{16n}} \quad \text{and} \quad \mathbb{V}[\|X\|] \approx \frac{\sigma^2}{2} \left(1 - \frac{1}{8n}\right)$$

indicating that the random step length actually does not vary strongly. Thus, controlling the step sizes in ES means controlling the variances of the mutation distribution and vice versa.

The subsequent sections present different techniques of controlling step sizes and shape of the mutation distributions. Some are of historical interest but, in principle, any technique can sometimes beat another technique on a set of problems as predicted by the NFL theorem (Wolpert and Macready 1997; Auger and Teytaud 2007). Nevertheless, there is a clear recommendation currently: First try the CMA-ES (Sect. 4.3.4) and switch to other methods if not satisfied with the results.

4.1 Control of Step Sizes

4.1.1 Adaptation of Step Sizes Based on Success Frequency

Early experiments (Schwefel 1964) have led to the observation that the step size must not be fixed during the search to achieve good performance. Intuitively, it is easy to accept the argument that the step sizes should become smaller the closer the ES approaches the optimum — otherwise most of the mutations would not lead to improvements. Actually, there exists an optimal step size depending on the current position and the distance to the optimum. Since the position of the optimum is not known, a clever mechanism is required for getting a hint whether the current step size is too large or too small.

Rechenberg (1973) analyzed two extremely different objective functions (called “sphere model” and “corridor model”) theoretically with the result that the largest progress toward the optimum is achieved for a step size that leads to a relative frequency of successful mutations

Algorithm 4 (1 + 1)-ES

choose $X^{(0)} \in \mathbb{R}^n, \sigma^{(0)} > 0$, set $s = 0, t = 0$

repeat

 draw $Z \sim N(0, I_n)$

$Y^{(t)} = X^{(t)} + \sigma^{(t)} Z$

if $f(Y^{(t)}) \leq f(X^{(t)})$ **then**

$s++$

$X^{(t+1)} = Y^{(t)}$

else

$X^{(t+1)} = X^{(t)}$

end if

if $t \bmod p \neq 0$ **then**

$\sigma^{(t+1)} = \sigma^{(t)}$

else

if $\frac{s}{p} < \frac{1}{5}$ **then**

$\sigma^{(t+1)} = \sigma^{(t)}/\gamma$

else

$\sigma^{(t+1)} = \sigma^{(t)} \cdot \gamma$

end if

$s = 0$

end if

 increment t

until stopping criterion fulfilled

(i.e., mutations that lead to a better fitness value) of about 0.270 for the sphere model and about 0.184 for the corridor model. This finding led to the conjecture that for an arbitrary fitness function, the step size should be adjusted to a length that leads to a success frequency between the two extreme values. Choosing $0.200 = \frac{1}{5}$ as a compromise between the extreme cases, Rechenberg's analysis led to the first step size control mechanism in ES, the so-called $\frac{1}{5}$ -success rule that is implemented as follows: Count the number of successful mutations s within a certain period of time p . If s/p is less than $1/5$, then the step size is too large and it should be decreased by a factor $1/\gamma < 1$. Otherwise, the step size is too small and it should be increased by a factor of $\gamma > 1$. Clearly, p should be a multiple of 5 and $\gamma = (\frac{20}{17})^{\frac{1}{5}}$ is recommended in Schwefel (1981).

4.1.2 Mutative Self-Adaptation of Step Sizes

Since the step sizes are altered only every p iterations when using the $\frac{1}{5}$ -rule, the adaptation happens staircase-like and sometimes it may be necessary to alter the step sizes at a quicker rate to achieve optimum performance. Therefore, Schwefel (1977) proposed a mutative step-size adaptation that leads to smoother adjustment of the step sizes (see [Algorithm 5](#)).

An individual now consists of its position $x \in \mathbb{R}^n$ and a single step size $\sigma > 0$. As can be seen in [Algorithm 5](#), the step size $\sigma^{(t)}$ is multiplicatively mutated with a lognormally distributed random number L , denoted $L \sim LN(0, \tau^2)$ where $\tau = 1/\sqrt{n}$. Random variable L is generated as follows: If $N \sim N(0, \tau^2)$ then $L = \exp(N) \sim LN(0, \tau^2)$. The adaptation process works quite well for $\lambda \geq 5$ but larger values are recommended.

Algorithm 5 $(1, \lambda)$ -ES

```

choose  $X^{(0)} \in \mathbb{R}^n$ , set  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$ ,  $L_i \sim LN(0, \tau^2)$ 
     $Y^{(i)} = X^{(t)} + \sigma^{(t)} L_i Z_i$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = L_b \cdot \sigma^{(t)}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

The mutation distribution used so far was a spherically shaped normal distribution, that is, the covariance matrix was the unit matrix. An extension (Bäck et al. 1991) of the method above is a variant with scaled mutations (Algorithm 6). Now an individual additionally has n scaling parameters $s \in \mathbb{R}_+^n$ that are object to mutations as well. The parameters change to $\tau = 1/\sqrt{2\sqrt{n}}$ and $\eta = 1/\sqrt{2n}$. But notice that experiments have revealed (Schwefel 1987) that a scaling parallel to the coordinate axes only is achieved if $\mu > 1$ and multiparent recombination is deployed. Moreover, it is quite useless to learn this kind of scaling since it is helpful for additively separable problems only. There exist specialized methods optimizing additively separable problems much more efficiently than evolutionary algorithms. In short, Algorithm 6 with $\mu = 1$ cannot be recommended for general problems.

Algorithm 6 $(1, \lambda)$ -ES with Mutative Scaling

```

choose  $X^{(0)} \in \mathbb{R}^n$ ,  $\sigma^{(0)} > 0$ , set  $s^{(0)} = (1, \dots, 1)', t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$ ,  $L_i \sim LN(0, \tau^2)$ ,  $S_i \sim LN(0, \eta^2 I_n)$ 
     $Y^{(i)} = X^{(t)} + \sigma^{(t)} \cdot L_i \cdot \text{diag}(s^{(t)}) \cdot \text{diag}(S_i) \cdot Z_i$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = L_b \cdot \sigma^{(t)}$ 
   $s^{(t+1)} = \text{diag}(S_b) \cdot s^{(t)}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.1.3 Mutative Self-Adaptation of Step Sizes with Momentum Adaptation

Ostermeier conjectured that (Ostermeier 1992) mutative scaling is not efficient especially for large dimensional problems since all mutations are drawn independently. Therefore, he

proposed to replace the mutative scaling by a deterministic mechanism that additionally extrapolates good moves in the past to the future (called *momentum adaptation*).

Let vector $m^{(t)} = (m_1^{(t)}, \dots, m_n^{(t)})'$ with $m^{(0)} = 0 \in \mathbb{R}^n$ induce the diagonal matrix $M^{(t)} = \text{diag}(|m_1^{(t)}|, \dots, |m_n^{(t)}|)$ where $|\cdot|$ denotes the absolute value. The random variables $Z_i \sim N(0, I_n)$ and S_i with $P\{S_i = 1\} = P\{S_i = 2/3\} = P\{S_i = 3/2\} = 1/3$ are mutually independent for all $i = 1, \dots, \lambda$. Notice that the global step size σ is now mutated by a discrete random variable S instead of the continuous lognormal random variable L used previously.

The parameter $\gamma \in [0, 1]$ is a learning rate. No recommendations are given. If $\gamma = 0$ then this method reduces to the original $(1, \lambda)$ -ES with (discrete) mutative step size control.

Algorithm 7 $(1, \lambda)$ -ES with Momentum Adaptation

```

choose  $X^{(0)} \in \mathbb{R}^n$ , set  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i$  and  $S_i$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot S_i \cdot \frac{1}{\sqrt{n}} \cdot [m^{(t)} + (I_n + M^{(t)})Z_i]$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = S_b \cdot \sigma^{(t)}$ 
   $m^{(t+1)} = m^{(t)} + \gamma \cdot (I_n + M^{(t)})Z_b$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.1.4 Mutative Self-Adaptation of Step Sizes with Derandomized Scaling

Ostermeier et al. (1994a) followed the idea of Ostermeier (1992) and proposed another deterministic method for adapting the scaling of the mutation distribution. Hints about

Algorithm 8 $(1, \lambda)$ -ES with Derandomized Scaling

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0$ ,  $S^{(0)} = I_n$ ,  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$  and  $G_i$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot G_i \cdot S^{(t)} \cdot Z_i$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = \sigma^{(t)} G_b$ 
  for  $j = 1$  to  $n$  do
     $s_j^{(t+1)} = s_j^{(t)} \exp\left(|Z_{b,i}| - \sqrt{\frac{2}{\pi}}\right)$ 
  end for
   $S^{(t+1)} = \text{diag}(s_1^{(t+1)}, \dots, s_n^{(t+1)})$ 
  increment  $t$ 
until stopping criterion fulfilled

```

an appropriate scaling is now extracted from the best mutation vector Z_b . This base concept turned out to be fruitful in future.

The global step size σ is multiplicatively mutated by the discrete random variable G with $P\{G = g\} = P\{G = 1/g\} = \frac{1}{2}$ for $g = 1, 4$.

Algorithm 9 $(1, \lambda)$ -CSA-ES

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0$ ,  $S^{(0)} = I_n$ ,  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot S^{(t)} \cdot Z_i$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $p^{(t+1)} = (1 - \alpha)p^{(t)} + \alpha Z_b$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\|p^{(t+1)}\| \sqrt{\frac{2-c}{nc}} - 1 + \frac{1}{5n}\right)^{1/\sqrt{n}}$ 
  for  $j = 1$  to  $n$  do
     $s_j^{(t+1)} = s_j^{(t)} \left[ |p_j^{(t+1)}| \cdot \sqrt{\frac{2-c}{c}} + \frac{j}{20} \right]^{1/n}$ 
  end for
   $S^{(t+1)} = \text{diag}(s_1^{(t+1)}, \dots, s_n^{(t+1)})$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.1.5 Cumulative Step-Size Adaptation (CSA)

The concept of extracting information from the best mutations in the previous method was extended by the same authors (Ostermeier et al. 1994b) by extracting information out of the entire sequence of best mutations. This leads to the method of CSA, which has been adopted for many other algorithms. The best mutation vectors Z_b of each iteration are accumulated in a weighted manner in an evolution path p that represents a good estimator of how to mutate in future. The step size control as well as the scaling of the mutation distribution now depends on the evolution path; they are no longer subject to a mutative adaptation. The parameters should be set to $\alpha = c = \sqrt{1/n}$.

4.2 Control of Descent Direction Without Covariances

The methods considered so far generated mutation distributions where the covariance matrix is a diagonal matrix. If most general distributions are required then one needs not only n scaling parameters but additionally $n(n - 1)/2$ further parameters to fully describe a general covariance matrix. Of course, less general distributions with fewer parameters are imaginable. Some authors refrained from maintaining a full covariance matrix due to increased space and computation requirements. Therefore, they proposed methods that can direct mutations toward good descent directions without using a full covariance matrix.

These arguments concerning space and time restrictions appear obsolete nowadays. The main memory of contemporary computers can store huge matrices and if the evaluation of the

fitness function requires a run of a complex simulation, then the time required for matrix computations is neglectable.

4.2.1 Mutative Self-Adaptation of Step Sizes with Correlated Momentum Adaptation

Ostermeier (1992) extended his method with momentum adaptation by estimating correlations from the best mutation Z_b . Random variable S is distributed as in [Sect. 4.1.3](#). Again, $\gamma \in [0, 1)$ is a learning parameter and the experiments reported in Ostermeier (1992) have been made in the range $0.001 \leq \gamma \leq 0.3$. Finally, the range $\gamma \in [\frac{1}{n}, \frac{3}{n}]$ is recommended for $n \geq 3$.

Algorithm 10 $(1, \lambda)$ -ES with Momentum and Correlations

```

choose  $X^{(0)} \in \mathbb{R}^n$ , set  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$ ,  $K_i \sim N(0, \frac{1}{4})$  and  $S_i$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot S_i \cdot \frac{1}{\sqrt{n}} \cdot \left[ (1 + K_i)m^{(t)} + \left( I_n + \frac{1}{c^{(t)}} M^{(t)} \right) \cdot Z_i \right]$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = S_b \cdot \sigma^{(t)}$ 
   $m^{(t+1)} = m^{(t)} + \gamma \cdot \left[ K_b \cdot m^{(t)} + \left( I_n + \frac{1}{c^{(t)}} M^{(t)} \right) \right] Z_b$ 
   $c^{(t+1)} = \max\{c^{(t)} \cdot (1 + \gamma)^{K_b - 1/20}, 1\}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.2.2 Main Vector Adaptation (MVA-ES)

Poland and Zell (2001) came up with the idea to adapt a vector that represents the main descent direction. Actually, they generate mildly correlated mutations. It is evident from the algorithm that they have made heavy use of the concept of the evolution path from the CSA-ES for adapting the main vector.

The parameters are set as follows: $\alpha = \beta = 4/(n + 4)$ and $\gamma = 2/(n + \sqrt{n})^2$. For conceptual reasons, the weighting function is defined as $w(v) = 1 + 2/\|v\|$ but the experiments reveal that $w(v) \equiv 3$ yields best results!

4.2.3 Evolutionary Gradient Search (EGS)

Salomon (1998) observed that sophisticated ES and stochastic gradient methods are quite similar. As a result, he proposed an evolutionary gradient search that approximates the (negative) gradient in a randomized manner by a central differences schema. Notice that $2\lambda + 2$ fitness function evaluations are necessary per iteration. The parameters should obey $\kappa \geq 1$, $\gamma_1 > 1$ and $\gamma_2 \in (0, 1)$. For example, $\gamma_1 = 1.4$ and $\gamma_2 = 1/\gamma_1$. As for η , no recommendation is given, but $\eta = 2$ seems reasonable, especially in the presence of noise.

Algorithm 11 $(1, \lambda)$ -MVA-ES

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0, t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$  and  $N_i \sim N(0, 1)$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot (Z_i + N_i \cdot w(v^{(t)}) \cdot v^{(t)})$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $s^{(t+1)} = (1 - \alpha)s^{(t)} + \sqrt{\alpha(2 - \alpha)}Z_b$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp\left(\frac{\|s^{(t+1)}\| - \sqrt{n-\frac{1}{2}}}{(1 - \beta^{-1})\sqrt{n-\frac{1}{2}}}\right)$ 
   $p^{(t+1)} = (1 - \beta)p^{(t)} + \sqrt{\beta(2 - \beta)}(Z_b + N_b \cdot w(v^{(t)}) \cdot v^{(t)})$ 
   $v^{(t+1)} = (1 - \gamma) \operatorname{sgn}((v^{(t)})' p^{(t)}) \cdot v^{(t)} + \gamma p^{(t)}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.3 Control of Covariances

The first ES that used and adapted a full covariance matrix was the ES with correlated mutations (named CORR-ES) proposed by Schwefel (1981). He conjectured that it would be optimal to align the hyperellipsoid of equal probabilities of the mutation distribution to the hyperellipsoid of equal fitness values if the problem is adequately described by a second order Taylor expansion. If this is true, then it would be optimal (Rudolph 1992) to set the covariance matrix C proportional to the inverse Hessian matrix of the Taylor expansion. This is easily seen from the probability density function (❷ 11) of the multivariate normal distribution where the isolines of equal probabilities are basically determined by the exponential expression $\exp(-\frac{1}{2}x C^{-1} x)$. Thus, if A is the Hessian matrix of the Taylor approximation, then $C^{-1} = A$ is needed to obtain equally aligned hyperellipsoids, or, equivalently, $C = A^{-1}$.

But it is not obvious that this is always the best choice. If the Hessian matrix is known, then we can perform a coordinate transformation to reduce the original problem

Algorithm 12 $(1, (2\lambda + 2))$ -Evolutionary Gradient Search

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0, t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$ 
  end for
   $\bar{Z} = \sum_{i=1}^{\lambda} [f(x^{(t)} - \sigma^{(t)} \cdot Z_i) - f(x^{(t)} + \sigma^{(t)} \cdot Z_i)] \cdot Z_i$ 
   $Y_1^{(t)} = X^{(t)} + \gamma_1 \cdot \frac{\sqrt{n}}{\eta} \cdot \frac{\bar{Z}}{\|\bar{Z}\|}$ 
   $Y_2^{(t)} = X^{(t)} + \gamma_2 \cdot \frac{\sqrt{n}}{\eta} \cdot \frac{\bar{Z}}{\|\bar{Z}\|}$ 
  let  $b \in \{1, 2\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_1^{(t)}), f(Y_2^{(t)})\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \cdot \gamma_b$ 
  increment  $t$ 
until stopping criterion fulfilled

```

$f(x) = \frac{1}{2}x'Ax + b'x + c$, where the Hessian matrix A is p.d. and symmetric, to a simpler problem for which a single global step size is sufficient, that is, the covariance matrix is the unit matrix

$$\begin{aligned} f(Qx) &= \frac{1}{2}(Qx)'A(Qx) + b'(Qx) + c \\ &= \frac{1}{2}x'Q'AQx + b'Qx + c \end{aligned} \tag{13}$$

$$\begin{aligned} &= \frac{1}{2}x'Q'B'BQx + b'Qx + c \\ &= \frac{1}{2}x'(B^{-1})'B'BB^{-1}x + b'B^{-1}x + c \end{aligned} \tag{14}$$

$$= \frac{1}{2}x'x + b'B^{-1}x + c$$

using $A = B'B$ in [Eq. 13](#) and choosing $Q = B^{-1}$ in [Eq. 14](#). Thus, if the Hessian A is known, it can be decomposed to $A = B'B$ owing to part (b) of [Theorem 1](#). It remains to invert matrix B such that the elliptical problem reduces to a spherical problem.

The bad news, however, is that this is never done. Actually, none of the ES methods perform such a coordinate transformation. Rather the mutation vector is scaled and rotated – the problem itself remains unaltered! Using the stochastic decomposition $\sigma QZ \stackrel{d}{=} rQu$ of [Theorem 3](#) and noting that $\nabla f(x) = Ax + b$ what happens is easily seen:

$$\begin{aligned} f(x + rQu) &= \frac{1}{2}(x + rQu)'A(x + rQu) + b'(x + rQu) + c \\ &= \frac{1}{2}(x'Ax + 2rx'AQu + r^2u'Q'AQu) + b'x + rb'Qu + c \\ &= f(x) + rx'AQu + rb'Qu + \frac{1}{2}r^2u'Q'AQu \\ &= f(x) + r(Ax + b + \frac{r}{2}AQu)'Qu \\ &= f(x) + r(\nabla f(x) + \frac{r}{2}AQu)'Qu \\ &= f(x) + r\nabla f(x)'Qu + \frac{r^2}{2}u'Q'AQu \\ &= f(x) + r\nabla f(x)'Qu + \frac{r^2}{2} \end{aligned} \tag{15}$$

if $A = B'B$ and $Q = B^{-1}$ because $u'Q'AQu = u'Q'B'BQu = u'u = \|u\|^2 = 1$ with probability 1. Evidently, $\nabla f(x)'Qu < 0$ is necessary for an improvement, that is, Qu must be a direction of descent. If we assume that Qu and r in [Eq. 15](#) are deterministic, then we know that $Qu = -\nabla f(x)$ yields the steepest descent and the optimal step length becomes $r^* = -\nabla f(x)'Qu = \|\nabla f(x)\|^2$.

In any case, the optimal alignment of the longest principal semiaxis of the mutation hyperellipsoid should point toward the negative gradient for optimal performance and not always toward the longest principal semiaxis of the Hessian matrix. But luckily, the ES drives itself in a situation in which negative gradient and the longest principal semiaxis of the Hessian matrix coincide: Regardless of the starting point, the ES runs roughly in negative gradient direction by construction of the method (try many directions and choose those with steepest descent). Sooner or later, but inevitably, it will reach a position in the search space that is close

to the longest principal semiaxis of the Hessian matrix. Now, the direction of steepest descent and the longest principal semiaxis of the Hessian matrix coincide: Let $f(x) = \frac{1}{2} x' A x + b' x + c$ with $x \in \mathbb{R}^n$, A p.d. and symmetric. If $\xi > 0$ is an eigenvalue of A and v its associated eigenvector, then the gradient direction in $\tilde{x} \in \mathbb{R}^n$ is parallel to the eigenvector provided the current position $\tilde{x} \in \{x \in \mathbb{R}^n : x = -A^{-1}b + s \cdot v, s \in \mathbb{R}\}$ is on the line spanned by the eigenvector: $\nabla f(\tilde{x}) = A\tilde{x} + b = A(-A^{-1}b + s \cdot v) + b = -b + sAv + b = sAv = s\xi \cdot v$.

From now on, the best points to be sampled are close to the longest principal semiaxis of the Hessian matrix so that the situation remains basically unaltered during the subsequent optimization phase and the adaptation process for the covariance matrix can stabilize. Therefore, it is actually optimal to use the inverse Hessian matrix as covariance matrix as soon as the longest principal semiaxis of the Hessian matrix is reached.

4.3.1 Mutative Self-Adaptation of Covariances (MSC)

Schwefel (1981) devised the first method to adapt arbitrary covariance matrices. The method uses the fact that every orthogonal matrix can be decomposed in the product of $n(n-1)/2$ elementary rotation matrices. An elementary rotation matrix $R_{jk}(\omega)$ is an unit matrix except that for the following entries holds: $r_{jj} = r_{kk} = \cos \omega$ and $r_{jk} = -r_{kj} = -\sin \omega$. Owing to **• Theorem 1** every positive definite matrix can be decomposed via $A = T' S S T$ where T is orthogonal and S is diagonal matrices with positive entries. As a consequence, every p.d. matrix can be generated by adapting $n(n-1)/2$ angles (for building T) and n scale parameters (for building S) (Rudolph 1992). Arbitrary elliptical mutation vectors can then be generated as follows: If $Z \sim N(0, I_n)$ then $TSZ \sim N(0, (TS)(TS)')$. As a consequence, an individual $(x, \sigma, \omega) \in \mathbb{R}^n \times \mathbb{R}_+^n \times (-\pi, \pi]^{n(n-1)/2}$ consists of its current position $x \in \mathbb{R}^n$, n positive step sizes (or scaling parameters) $\sigma \in \mathbb{R}_+^n$ and $n(n-1)/2$ angles for the elementary rotations. For the sake of notational clarity, **• Algorithm 13** is described for the case $\mu = 1$. Please note that the CORR-ES has never been used in this manner! Rather, in practice, choose $\mu > 1$ and deploy recombination on object parameters x and strategy parameters σ and ω .

Algorithm 13 $(1, \lambda)$ -CORR-ES or $(1, \lambda)$ -MSC-ES

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0$ ,  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$ ,  $S_i \sim LN(0, \eta^2 I_n)$ ,  $L_i \sim LN(0, \tau^2)$ ,  $W_i \sim N\left(0, \left(\frac{5\pi}{180}\right)^2\right)$ 
     $\sigma_i^{(t+1)} = L_i \cdot \text{diag}(S_i) \cdot \sigma_i^{(t)}$ 
     $\omega_i^{(t+1)} = (\omega_i^{(t)} + W_i + (\pi, \dots, \pi)') \bmod 2(\pi, \dots, \pi)' - (\pi, \dots, \pi)'$ 
     $Y^{(t)} = X^{(t)} + \prod_{j=1}^{n-1} \prod_{k=j+1}^n R_{jk}(\omega_{i,j,k}^{(t+1)}) \cdot \text{diag}(\sigma_i^{(t+1)}) Z_i$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}) : i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = \sigma_b^{(t+1)}$ 
   $\omega^{(t+1)} = \omega_b^{(t+1)}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

4.3.2 Covariance Matrix Estimation by Least Squares (LS-CME-ES)

Rudolph (1992) has proposed to estimate the covariance matrix by least squares estimation from the already evaluated individuals. This method was rediscovered by Auger et al. (2004). The idea is as follows: Assume that the problem is appropriately determined by a second order Taylor expansion: $f(x) = \frac{1}{2} x' A x + b' x + c$ where $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Whenever an individual was evaluated, we have a pair $(x, f(x))$ with known values. But the constants in A , b , and c are unknown. Thus, the above equation has $n(n+1)/2 + n + 1 = (n^2 + 3n + 2)/2 = N$ unknown values if a known pair $(x, f(x))$ was fed in. In this case, this equation is a linear equation where the unknown values for A , b , and c are the variables. To estimate N by the least squares method, at least $N + 1$ equations are needed, that is, at least $N + 1$ evaluated individuals. The estimation of the N parameters requires $O(N^2) = O(n^6)$ operations. The values for b and c are not needed. Finally, a Cholesky decomposition of $A = BB'$ (see \blacktriangleright Theorem 1) delivers B and subsequent inversion yields the matrix $Q = B^{-1}$ required for generating the desired elliptically distributed mutation vector. Although the computational effort of $O(n^6)$ seems daunting initially, this approach may be useful if the dimension is not too large and a single fitness function evaluation requires much time.

4.3.3 Generating Set Adaptation (GSA)

Hansen et al. (1995) deployed a different method to generate elliptically distributed normal random vectors with arbitrary covariance matrix. To understand the process, some results should be recalled. If $n = 1$, $m \geq 2$, $b = 0$, $\Sigma = 1$ and $C \in \mathbb{R}^{m,1}$ then \blacktriangleright Theorem 2 immediately proves the following result:

Corollary 2 *Let $X \sim N_1(0, 1)$ and $Y = X \cdot c$ for some $c \in \mathbb{R}^m$. Then $Y \sim N_m(0, c c')$ with $\det(c c') = 0$.*

Thus, the product of a standard normal random variable and a deterministic vector leads to a multivariate normal random vector with singular covariance matrix. What happens if several random vectors generated in this manner are added?

Theorem 4 (see Rao 1973, p. 524 or Tong 1990, p. 33) *If $X_i \sim N_n(v_i, \Sigma_i)$ independent for $i = 1, \dots, m$ and $a \in \mathbb{R}^m$, then*

$$X = \sum_{i=1}^m a_i X_i \sim N_n\left(\sum_{i=1}^m a_i v_i, \sum_{i=1}^m a_i^2 \Sigma_i\right) \quad \square$$

Notice that there are no restrictions imposed on the covariance matrices, they may be singular as well as regular. Thus, the sum of multivariate normal random vectors with singular covariance matrices is a multivariate normal random vector whose covariance matrix is the sum of the singular covariance matrices. The resulting covariance matrix is regular if the deterministic vectors used are a basis for the vector space \mathbb{R}^n .

Theorem 5 *Let $X \sim N(0, I_n)$, $B = (b_1, b_2, \dots, b_n) \in \mathbb{R}^{n,n}$ where column vectors b_1, \dots, b_n are a basis of \mathbb{R}^n , and let e_i be the i th unit vector. Then*

$$\sum_{i=1}^n X_i b_i \sim N(BB')$$

where BB' is regular.

Proof Thanks to Theorem 2 we know that $BX \sim N(0, BB')$. Since B contains a basis of \mathbb{R}^n its rank is n so that B is regular. It follows that BB' is regular as it has rank n (Abadir and Magnus 2005, p. 81). Notice that $X = \sum_{i=1}^n X_i e_i$ and hence

$$BX = B \sum_{i=1}^n X_i e_i = \sum_{i=1}^n X_i B e_i = \sum_{i=1}^n X_i b_i \sim N(0, BB')$$

which proves the result. \square

What happens if additional singular random vectors are added although the distribution is regular already?

Lemma 1 Let $A \in \mathbb{R}^{n,n}$ be positive semidefinite. If $B \in \mathbb{R}^{n,n}$ is p.s.d., then $A + B$ is positive semidefinite; if $B \in \mathbb{R}^{n,n}$ is p.d., then $A + B$ is positive definite.

Proof Only the second part is shown: By definition, $x'Ax \geq 0$ and $x'Bx > 0$. Consequently, $x'(A + B)x = x'Ax + x'Bx > 0$ which proves that $A + B$ is positive definite. \square

Thus, adding singular normal vectors to a regular normal vector yields a regular normal vector since the covariance matrix stays positive definite. If more than n vectors are used to span \mathbb{R}^n and the set contains a basis then this set of vectors is termed a *generating set* of \mathbb{R}^n .

After these preparations, the process of adapting the covariance matrix becomes transparent. Adapt more than n vectors v_1, \dots, v_m with $m > n$ and generate the random vector via

$$\sum_{i=1}^m v_i Z_i$$

where $Z \sim N(0, I_m)$. Arbitrary covariance matrices can be constructed in this manner.

Algorithm 14 (1, λ)-GSA-ES

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0$ ,  $t = 0$ 
define a generating set  $\{v_1^{(0)}, \dots, v_m^{(0)}\}$  of  $m \geq n$  vectors spanning the  $\mathbb{R}^n$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_m)$  and  $G_i$ 
     $\tilde{Z}_i^{(t)} = \gamma \sum_{j=1}^m Z_{ij} \cdot b_j^{(t)}$ 
     $Y_i^{(t)} = X^{(t)} + \sigma^{(t)} \cdot G_i \cdot \tilde{Z}_i^{(t)}$ 
  end for
  let  $b \in \{1, \dots, \lambda\}$  such that  $f(Y_b^{(t)}) = \min\{f(Y_i^{(t)}): i = 1, \dots, \lambda\}$ 
   $X^{(t+1)} = Y_b^{(t)}$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \cdot G_b$ 
   $v_1^{(t+1)} = (1 - \alpha)v_1^{(t)} + \sqrt{\alpha(2 - \alpha)}G_b\tilde{Z}_b^{(t)}$ 
   $v_{j+1}^{(t+1)} = v_j^{(t)} \text{ for } j = 1, \dots, m - 1.$ 
  increment  $t$ 
until stopping criterion fulfilled

```

In the beginning, it is recommended to set $v_1^{(0)} = 0 \in \mathbb{R}^n$ and $v_j^{(0)} \sim N(0, \frac{1}{n}I_n)$ for $j = 2 \dots m$. The size of the generating set should be in the range $m \in \{n^2, \dots, 2n^2\}$, guaranteeing that

the initial mutation vector has a regular covariance matrix. Notice that the step-size adaptation is mutative again with the random variable G_i distributed as

$$\mathbb{P}\left\{ G_i^\beta = \left(\frac{3}{2}\right)^\beta \right\} = \mathbb{P}\left\{ G_i^\beta = \left(\frac{2}{3}\right)^\beta \right\} = \frac{1}{2}$$

where $\beta = 1/\sqrt{n}$. Moreover, set $\alpha = 1/\sqrt{n}$ and $\gamma = (1 + 1/m)/\sqrt{m}$.

4.3.4 Covariance Matrix Adaptation (CMA)

Hansen and Ostermeier (1996, 2001) and Hansen (2009) finally introduced a covariance matrix adaptation method that received the status of a standard method. Now, the covariance matrix is continuously updated with information gathered from evaluated points.

Algorithm 15 (μ, λ) -CMA-ES

choose $X^{(0)} \in \mathbb{R}^n$; set $\sigma^{(0)} > 0$, $C^{(0)} = T^{(0)} = S^{(0)} = I_n$, $p_c^{(0)} = p_\sigma^{(0)} = 0 \in \mathbb{R}^n$, $t = 0$

repeat

for $i = 1$ to λ **do**

draw $Z_i \sim N(0, I_n)$
 $Y_i^{(t)} = X^{(t)} + \sigma_i^{(t)} \cdot T^{(t)} \dots S^{(t)} Z_i^{(t)}$

end for

let $b_1, b_2, \dots, b_\lambda \in \{1, \dots, \lambda\}$ such that $f(Y_{b_1}^{(t)}) \leq f(Y_{b_2}^{(t)}) \leq \dots \leq f(Y_{b_\lambda}^{(t)})$

$X^{(t+1)} = \sum_{i=1}^{\mu} w_{b_i} \cdot Y_{b_i}^{(t)}$

$p_\sigma^{(t+1)} = (1 - \alpha)p_\sigma^{(t)} + \sqrt{\alpha(2 - \alpha)\mu_{\text{eff}}} \sum_{i=1}^{\mu} w_{b_i} \cdot Z_{b_i}$

$\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp\left(\frac{\alpha \|p_\sigma^{(t+1)}\| - \chi_n}{\delta}\right)$

$p_c^{(t+1)} = (1 - \beta)p_c^{(t)} + \eta \sqrt{\beta(1 - \beta)\mu_{\text{eff}}} T^{(t)} \cdot S^{(t)} \sum_{i=1}^{\mu} w_{b_i} \cdot Z_{b_i}$

$C^{(t+1)} = (1 - \gamma)C^{(t)} + \frac{\gamma}{v} p_c^{(t+1)} \left(p_c^{(t+1)}\right)' + \gamma(1 - v^{-1}) \sum_{i=1}^{\mu} w_{b_i} T^{(t)} S^{(t)} Z_{b_i} (T^{(t)} S^{(t)} Z_{b_i})'$

get $T^{(t+1)}$ and $S^{(t+1)}$ by spectral decomposition of $C^{(t+1)}$

increment t

until stopping criterion fulfilled

The parameters are recommended as follows:

$$\lambda = 4 + \lfloor 3 \log n \rfloor$$

$$\mu = \left\lceil \frac{\lambda + 1}{2} \right\rceil$$

$$w_i = \frac{\log \left(\frac{\lambda + 1}{2i} \right)}{\mu \log \left(\frac{\lambda + 1}{2} \right) - \log(\mu!)} \quad (1)$$

$$\mu_{\text{eff}} = \frac{1}{w' w} \quad (2)$$

$$v = \mu_{\text{eff}} \quad (3)$$

$$\alpha = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 3} \quad (4)$$

$$\begin{aligned}\beta &= \frac{4}{n+4} \\ \gamma &= \frac{1}{v} \frac{2}{(n+\sqrt{2})^2} + \left(1 - \frac{1}{v}\right) \min \left\{1, \frac{2v-1}{(n+2)^2+v}\right\} \\ \delta &= 1 + 2 \max \left\{0, \sqrt{\frac{\mu_{\text{eff}}-1}{n+1}} - 1\right\} + \alpha \\ \chi_n &= \sqrt{n - \frac{1}{2}}\end{aligned}$$

Jastrebski and Arnold (2006) proposed to exploit the information from the worst mutation vectors also, a lesson learned from the analysis of optimally weighted recombination (Rudolph 1997; Arnold 2006): If w is the worst out of λ mutations then it points in a direction of steep ascent. Therefore, $-w$ should point in the direction of steep descent.

4.3.5 Modified Evolutionary Gradient Search (CMA-EGS)

The method presented in Sect. 4.2.3 was modified with covariance adaptation by Arnold and Salomon (2007). It also requires spectral decomposition of the covariance matrix. As this is a costly operation, they recommend to perform it only every $n/10$ iterations.

Algorithm 16 $(1, \lambda)$ -CMA-EGS

```

choose  $X^{(0)} \in \mathbb{R}^n$ ; set  $C^{(0)} = I_n$ ,  $\sigma^{(0)} > 0$ ,  $t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, C^{(t)})$ 
  end for
   $\bar{Z} = \sum_{i=1}^{\lambda} [f(x^{(t)} - \sigma^{(t)} Z_i) - f(x^{(t)} + \sigma^{(t)} Z_i)] \cdot Z_i$ 
   $\hat{Z} = \frac{\sqrt{n}}{\eta} \cdot \frac{\bar{Z}}{\|\bar{Z}\|}$ 
  let  $T^{(t)}$  and  $S^{(t)}$  such that  $C^{(t)} = T^{(t)} S^{(t)} (T^{(t)} S^{(t)})'$  (see text for details)
   $X^{(t+1)} = X^{(t)} + \sigma^{(t)} \cdot T^{(t)} S^{(t)} \cdot \hat{Z}$ 
   $s_c^{(t+1)} = (1-\alpha)s_c^{(t)} + \eta\sqrt{\alpha(2-\alpha)}T^{(t)}S^{(t)} \cdot \hat{Z}$ 
   $s_\sigma^{(t+1)} = (1-\beta)s_\sigma^{(t)} + \eta\sqrt{\beta(2-\beta)}T^{(t)} \cdot \hat{Z}$ 
   $C^{(t+1)} = (1-\gamma)C^{(t)} + \gamma s_c^{(t+1)} (s_c^{(t+1)})'$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp\left(\frac{\|s_\sigma^{(t)}\|^2 - n}{2n(1+\beta^{-1})}\right)$ 
  increment  $t$ 
until stopping criterion fulfilled

```

There is no clear recommendation for $\eta \geq 1$, but according to their experiments $\eta = 2$ seems to be reasonable (especially in the case of noise). The remaining parameters are recommended to be set as follows:

$$\alpha = \beta = \frac{4}{n+4}, \quad \gamma = \frac{2}{(n+\sqrt{2})^2}$$

4.3.6 Covariance Matrix Self-Adaptation (CMSA)

Beyer and Sendhoff (2008) returned to a mutative self-adaptive step length control but adapted the basic mechanism of the covariance update rule. This opens the door for replacing the costly spectral decomposition by the simpler, quicker, and more stable Cholesky decomposition.

Algorithm 17 (μ, λ) -CMSA-ES

```

choose  $\mu > 1, X^{(0)} \in \mathbb{R}^n$ ; set  $\sigma^{(0)} > 0, C^{(0)} = R^{(0)} = I_n, t = 0$ 
repeat
  for  $i = 1$  to  $\lambda$  do
    draw  $Z_i \sim N(0, I_n)$  and  $L_i \sim LN(0, \tau^2)$ 
     $Y_i^{(t)} = X^{(t)} + \sigma_i^{(t)} \cdot L_i \cdot R^{(t)} Z_i^{(t)}$ 
  end for
  let  $b_1, b_2, \dots, b_\lambda \in \{1, \dots, \lambda\}$  such that  $f(Y_{b_1}^{(t)}) \leq f(Y_{b_2}^{(t)}) \leq \dots \leq f(Y_{b_\lambda}^{(t)})$ 
   $X^{(t+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} Y_{b_i}^{(t)}$ 
   $\sigma^{(t+1)} = \sigma^{(t)} \cdot \frac{1}{\mu} \sum_{i=1}^{\mu} L_{b_i}$ 
   $C^{(t+1)} = (1 - \gamma)C^{(t)} + \gamma \sum_{i=1}^{\mu} R^{(t)} Z_{b_i} (R^{(t)} Z_{b_i})'$ 
  get  $R^{(t+1)}$  by Cholesky decomposition of  $C^{(t+1)}$ 
  increment  $t$ 
until stopping criterion fulfilled

```

Only two parameters are used:

$$\tau \in \left[\frac{1}{2\sqrt{2n}}, \frac{1}{\sqrt{2n}} \right] \quad \text{and} \quad \gamma = \frac{2\mu}{2\mu + n(n+1)}$$

The authors reported results comparable to the CMA-ES for small population sizes and considerable better results for large population sizes that are necessary in the presence of noise. If this behavior can be confirmed in future applications, then this method may become a serious competitor for the CMA-ES.

5 Further Reading

In this chapter, only the basic facts and methods have been presented. Many aspects of ES have been left for further reading. For example, Auger and Hansen (2005) recommend to start the CMA-ES as usual but as soon as a termination criterion is fulfilled, the method should be restarted with increased population size. The results reported are impressive. This technique could be useful for other variants as well.

The CMA-ES has been extended to work in the case of multi-objective optimization (Igel et al. 2006) whereas the constraint handling (Kramer and Schwefel 2006) is fairly unexplored. Model-assisted ES have been developed (Emmerich et al. 2002; Ulmer et al. 2004; Hoffmann and Hölemann 2006) as well as methods that work in the presence of noise (Arnold and Beyer 2006; Beyer and Meyer-Nieberg 2006). Also, the technique of niching (Preuss et al. 2005, 2006) for exploring several promising regions simultaneously has been deployed for CMA-ES (Shir et al. 2007). A survey on parallel ES is given in Rudolph (2005).

As already mentioned, there are many other techniques that deserve further reading. Some ideas toward an integration of currently unexplored principles from biological evolution may be found in Rudolph and Schwefel (2008).

References

- Abadir K, Magnus J (2005) Matrix algebra. Cambridge University Press, New York
- Arnold D (2006) Weighted multirecombination evolution strategies. *Theor Comput Sci* 361(1):18–37
- Arnold D, Beyer HG (2006) A general noise model and its effects on evolution strategy performance. *IEEE Trans Evolut Comput* 10(4):380–391
- Arnold D, Salomon R (2007) Evolutionary gradient search revisited. *IEEE Trans Evolut Comput* 11(4):480–495
- Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE congress on evolutionary computation (CEC 2005), vol 2. IEEE Press, Piscataway, NJ, pp 1769–1776
- Auger A, Teytaud O (2007) Continuous lunches are free! In: Thierens D et al. (eds) Proceedings of the genetic and evolutionary computation conference (GECCO 2007). ACM Press, New York, pp 916–922
- Auger A, Schoenauer M, Vanhaecke N (2004) LS-CMA-ES: a second-order algorithm for covariance matrix adaptation. In: Yao X et al. (eds) Proceedings of the 8th international conference on parallel problem solving from nature (PPSN VIII). Springer, Berlin, Germany, pp 182–191
- Bäck T, Hoffmeister F, Schwefel HP (1991) A survey of evolution strategies. In: Belew RK, Booker LB (eds) Proceedings of the fourth international conference on genetic algorithms (ICGA'91). Morgan Kaufmann, San Mateo CA, pp 2–9
- Bartz-Beielstein T (2006) Experimental research in evolutionary computation. The new experimentalism. Springer, Heidelberg
- Beyer HG, Meyer-Nieberg S (2006) Self-adaptation of evolution strategies under noisy fitness evaluations. *Genet Programming Evolvable Mach* 7(4):295–328
- Beyer HG, Schwefel HP (2002) Evolution strategies – a comprehensive introduction. *Nat Comput* 1(1):3–52
- Beyer HG, Sendhoff B (2008) Covariance matrix adaptation revisited – the CMSA evolution strategy. In: Rudolph G et al. (eds) Proceedings of the 10th international conference on parallel problem solving from nature (PPSN X). Springer, Berlin, Germany, pp 123–132
- Emmerich M, Giotis A, Oezdemir M, Bäck T, Giannakoglou K (2002) Metamodel-assisted evolution strategies. In: Merelo J et al. (eds) Proceedings of the 7th international conference on parallel problem solving from nature (PPSN VII). Springer, Berlin, Germany, pp 361–370
- Fang KT, Kotz S, Ng KW (1990) Symmetric multivariate and related distributions. Chapman and Hall, London, UK
- Hansen N (2009) The CMA evolution strategy: A tutorial. Continuously updated technical report. Available via <http://www.lri.fr/~hansen/cmatutorial.pdf>. Accessed April 2009
- Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of the 1996 IEEE international conference on evolutionary computation (ICEC '96). IEEE Press, Piscataway, NJ, pp 312–317
- Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evolut Comput* 9(2):159–195
- Hansen N, Ostermeier A, Gawelczyk A (1995) On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation. In: Eshelman L (ed) Proceedings of the 6th international conference on genetic algorithms (ICGA 6). Morgan Kaufmann, San Francisco, CA, pp 57–64
- Hoffmann F, Hölemann S (2006) Controlled model assisted evolution strategy with adaptive preselection. In: Proceedings of the 2006 international symposium on evolving fuzzy systems. IEEE Press, Piscataway, NJ, pp 182–187
- Igel C, Hansen C, Roth S (2006) Covariance matrix adaptation for multi-objective optimization. *Evolut Comput* 15(1):1–28
- Jastrebski G, Arnold D (2006) Improving evolution strategies through active covariance matrix adaptation. In: Proceedings of the 2006 IEEE congress on evolutionary computation (CEC 2006). IEEE Press, Piscataway, NJ
- Kramer O, Schwefel HP (2006) On three new approaches to handle constraints within evolution strategies. *Nat Comput* 5:363–385
- McKay M, Beckman R, Conover W (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21(2):239–245
- Ostermeier A (1992) An evolution strategy with momentum adaptation of the random number distribution.

- In: Manderick B, Männer R (eds) Proceedings of the 2nd international conference on parallel problem solving from nature (PPSN II). Elsevier, Amsterdam, The Netherlands, pp 197–206
- Ostermeier A, Gawelczyk A, Hansen N (1994a) A derandomized approach to self adaptation of evolution strategies. *Evolut Comput* 2(4):369–380
- Ostermeier A, Gawelczyk A, Hansen N (1994b) Step-size adaptation based on non-local use of selection information. In: Davidor Y et al. (eds) Proceedings of the 3rd international conference on parallel problem solving from nature (PPSN III). Springer, Berlin, Germany, pp 189–198
- Poland J, Zell A (2001) Main vector adaptation: a CMA variant with linear time and space complexity. In: Proceedings of the genetic and evolutionary computation conference (GECCO 2001). Morgan Kaufmann, San Francisco, CA, pp 1050–1055
- Preuss M (2006) Niching prospects. In: Filipic B, Silc J (eds) Bioinspired optimization methods and their applications (BIOMA 2006). Jozef Stefan Institute, Ljubljana, Slovenia, pp 25–34
- Preuss M, Schönemann L, Emmerich M (2005) Counteracting genetic drift and disruptive recombination in $(\mu + 1, \lambda)$ -EA on multimodal fitness landscapes. In: Beyer HG et al. (eds) Proceedings of the 2005 genetic and evolutionary computation conference (GECCO 2005), vol 1. ACM Press, New York, pp 865–872
- Rao C (1973) Linear statistical inference and its applications, 2nd edn. Wiley, New York
- Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, Farnborough
- Rechenberg I (1973) Evolutionsstrategie: optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart
- Rechenberg I (1978) Evolutionsstrategien. In: Schneider B, Ranft U (eds) Simulationsmethoden in der Medizin und Biologie. Springer, Berlin, Germany, pp 83–114
- Rudolph G (1992) On correlated mutations in evolution strategies. In: Männer R, Manderick B (eds) In: Proceedings of the 2nd international conference on parallel problem solving from nature (PPSN II). Elsevier, Amsterdam, The Netherlands, pp 105–114
- Rudolph G (1994) An evolutionary algorithm for integer programming. In: Davidor Y et al. (eds) Proceedings of the 3rd conference on parallel problem solving from nature (PPSN III). Springer, Berlin, Germany, pp 63–66
- Rudolph G (1997) Convergence properties of evolutionary algorithms. Kovač, Hamburg
- Rudolph G (2005) Parallel evolution strategies. In: Alba E (ed) Parallel metaheuristics: a new class of algorithms. Wiley, Hoboken, NJ, pp 155–170
- Rudolph G, Schwefel HP (2008) Simulated evolution under multiple criteria revisited. In: Zurada J (ed) WCCI 2008 Plenary/Invited Lectures. Springer, Berlin, Germany, pp 248–260
- Salomon R (1998) Evolutionary algorithms and gradient search: similarities and differences. *IEEE Trans Evolut Comput* 2(2):45–55
- Schwefel HP (1964) Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin, Hermann Föttinger-Institut für Strömungstechnik
- Schwefel HP (1977) Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Birkhäuser, Basel
- Schwefel HP (1981) Numerical optimization of computer models. Wiley, Chichester, UK
- Schwefel HP (1987) Collective phenomena in evolutionary systems. In: Checkland P, Kiss I (eds) Problems of constancy and change – the complementarity of systems approaches to complexity, Papers presented at the 31st annual meeting of the International Society for General System Research, International Society for General System Research, vol 2. Budapest, pp 1025–1033
- Schwefel HP, Rudolph G (1995) Contemporary evolution strategies. In: Morán F et al. (eds) Advances in artificial life – proceedings of the third European conference on artificial life (ECAL'95). Springer, Berlin, Germany, pp 893–907
- Shir O, Emmerich M, Bäck T (2007) Self-adaptive niching CMA-ES with Mahalanobis metric. In: Proceedings of the 2007 IEEE congress on evolutionary computation (CEC 2007). IEEE Press, Piscataway, NJ
- Smit S, Eiben A (2009) Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the 2009 IEEE congress on evolutionary computation (CEC 2009). IEEE Press, Piscataway, NJ
- Tong Y (1990) The multivariate normal distribution. Springer, New York
- Ulmer H, Streichert F, Zell A (2004) Model assisted evolution strategies. In: Jin Y (ed) Knowledge incorporation in evolutionary computation. Springer, Berlin, Germany, pp 333–358
- Wolpert D, Macready W (1997) No free lunch theorems for optimization. *IEEE Trans Evolut Comput* 1(1):67–82

23 Evolutionary Programming

Gary B. Fogel

Natural Selection, Inc., San Diego, CA, USA

gfogel@natural-selection.com

1	<i>Background</i>	700
2	<i>History</i>	702
3	<i>Extensions</i>	704
4	<i>Recent Applications</i>	705

Abstract

Evolutionary programming (EP) has a long history of development and application. This chapter provides a basic background in EP in terms of its procedure, history, extensions, and application. As one of the founding approaches in the field of evolutionary computation (EC), EP shares important similarities and differences with other EC approaches.

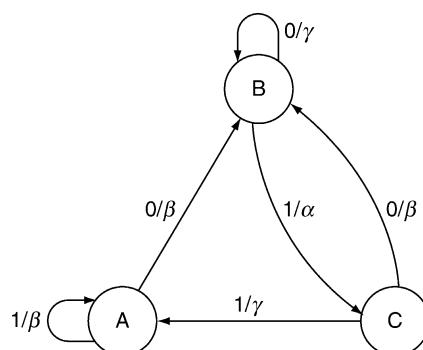
1 Background

Evolutionary programming (EP) is an approach to simulated evolution that iteratively generates increasingly appropriate solutions in the light of a stationary or nonstationary environment and desired fitness function. Standard EP makes use of the same four components that are standard to all evolutionary algorithms (EAs): initialization, variation, evaluation, and selection. This approach to population-based search has its roots in the early work of Lawrence J. Fogel, as described later in [Sect. 2](#). Over generations of simulated evolution, the population of solutions learns about its environment through stochastic trial and error, with a process of selection continually driving the population in more useful directions in the search space.

The basic procedure for canonical EP evolved finite state machines (FSMs) (also called “finite-state automata”) (Mealy 1955; Moore 1957), which were a standard transducer representation in the literature at the time. Finite-state machines operate on a finite set of input symbols, possess a finite number of internal states, and produce symbols from a finite set. The corresponding input-output symbol pairs and set of state transitions specify the behavior of the FSM ([Fig. 1](#)). A population of solutions (i.e., FSM representations) was initialized, generally at random, with memory limitations on the possible number of states in the FSM due to the computers that were being used at the time of its invention in the early 1960s. A known sequence of symbols from the environment was defined as an observed sequence

Fig. 1

A finite-state machine (FSM) representation. Starting in state “A,” input symbols are shown to the left of each slash and response symbols are shown to the right of the slash. Arrows provide connections between states. The path of connections through the FSM produces a series of output symbols from a given input symbol pattern that can be used as a predictor of that symbol environment.



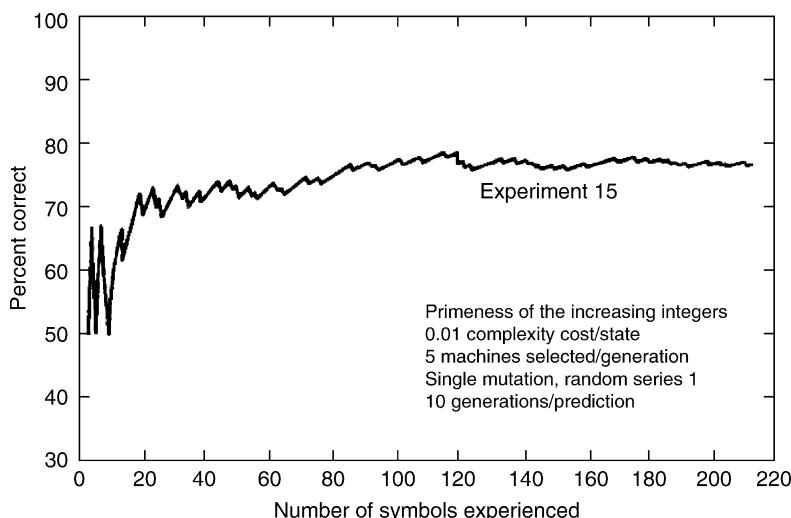
or time series. Existing FSMs in the population were mutated as “parent” solutions to generate “offspring” FSMs. Both parents and offspring were then scored in light of how well they predicted the next symbol in a known sequence of symbols. The best half of the FSMs in the population were retained as parents for the next iteration, with selection removing the remaining half of poor-quality solutions. As required, the best available machine in the population was used to predict the next symbol in the symbol sequence, and the actual symbol was then added to the known available data for model training and optimization. The process was iterated for as long as time permitted (● Fig. 2). This initial approach to EP was first offered in the literature in Fogel (1962a, b) and was studied through a variety of experiments in Fogel et al. (1964, 1965a, b, c) and summarized in Fogel et al. (1966).

1.1 The Canonical EP

For EP, the size of the evolving population, μ , ranges in size depending on the problem and computational processing speed available. A population of solutions was used in all of the early EP experiments, although small populations were typically used in initial experiments. The original process was written in Fortran II and processed on an IBM 7094, which had roughly half the processing speed of an Apple II or Commodore 64 (Fogel 1999). Each solution is evaluated with respect to a fitness function and offspring solutions are generated through a process of variation. For the initial FSM representations, this variation was typically via mutation as there were clear ways in which to vary these representations. The first was to add a state and then randomly assign all of the input-output and input-transition pairs for this

■ Fig. 2

A typical output series for the evolution of finite-state machines using evolutionary programming (EP) to predict primeness of numbers. The evolving population rapidly converged on a useful finite state machine (FSM) after about 100 symbols experienced during the learning process.



state, provided the FSM maximum size does not exceed memory constraints. A second mutation type was to delete a state if there is more than one state in the FSM. A third was to change an output symbol coupled with a specific input symbol in a single state. The fourth was to alter a state transition associated with an input symbol in a single state. Finally, a fifth was to change the starting state itself, if the FSM had more than one state. By adjusting the probability for each of these five mutation strategies and/or the number of mutations per offspring, a diverse set of possible FSMs for offspring solutions was possible. In the canonical version of EP, each parent member i generated λ_i offspring with the above mutation strategies. It should be noted that Fogel et al. (1966) suggested the possibility of other mutation strategies such as recombining representations to generate new solutions; however, these FSM-specific operators were sufficient for early experimentation to demonstrate the utility of the process. In addition, Fogel considered the FSM representations to be equivalent to individual phenotypes being evaluated in an environmental context by selection. As such, the actual methods of modification were less important than simply offering phenotypic change over time in the population of solutions. Thus, the EP process was designed to simulate evolution at the individual level and given that nature only scores phenotypic behavior in a given environment, so long as FSMs were able to generate useful predictions their increasing utility would be guaranteed by the process of evolutionary optimization.

For convenience and with respect to limiting computational requirements, early experiments in EP maintained a constant small population size of three to five FSMs. Each of the m parents was mutated to generate a single offspring, with the resulting $2m$ solutions all competing together for survival to the next generation. The best m solutions were retained as parents. It should be noted that Fogel (1964) and Fogel et al. (1966) noted the possibility of retaining less-fit solutions in the population, but computational hardware requirements were already difficult. Later experiments incorporated tournament selection or proportional-based selection to retain a sampling of m best solutions. The evolutionary process was terminated once a predetermined number of generations had expired, or other criterion had been satisfied.

2 History

Evolutionary programming (EP) was conceived by Lawrence J. Fogel in the early 1960s as an alternative approach to classic artificial intelligence (AI) in computers. While on leave from General Dynamics in 1960 as an assistant to the director of research for the National Science Foundation, Fogel was given the challenge of determining how much America should invest in basic research (Fogel 1999). A portion of this survey focused on methods of artificial intelligence. At the time, two different schools of thought prevailed in AI. The first, a “bionics” approach, attempted to build an artificial brain, neuron by neuron. These approaches were considered by Fogel to show promise for pattern recognition, but not for the generation of intelligence in computers. The second approach, “heuristic programming,” attempted to achieve AI through a series of if-then rules. Modern expert system methods continue to pursue this approach, but there are many issues involved with heuristics such as the quality and qualifications of the experts themselves, how well rules that are written for current situations translate when the environment is nonstationary, and how to make decisions when experts disagree. In light of these shortcomings, Fogel conceived the idea of simulating the evolutionary process on computers – the same process that generated human intelligence over millions

of years of evolution. Such a process does not rely on heuristics, but generates solutions of increasing intellect over time. This approach to AI was presented to his colleagues at the National Science Foundation in June of 1961 but the response was far from positive.

Upon returning to General Dynamics at San Diego, California in June of 1961, the first computer experiments were conducted, together with Al Owens, who was in charge of the computer laboratory. Jack Walsh, a mathematician at General Dynamics became interested in mathematical representations to determine the capabilities of the evolutionary process. The first experiments using FSMs resulted in the publication of *Industrial Research* in 1962 (Fogel 1962a). Several publications resulted from the team of Fogel, Owens, and Walsh between 1962 and 1966 (Fogel 1962b, 1963; Fogel et al. 1964, 1965a, b, c, 1966). Fogel's dissertation in 1964 at the University of California, Los Angeles (the first Ph.D. dissertation in the field of evolutionary computation) (Fogel 1964) described experiments with a population of FSMs where each FSM was exposed to a sequence of observed symbols (representing an environment). A predefined payoff function was used to score the worth of each FSM after input symbols were provided to each FSM and an output prediction was compared to the actual next symbol in the sequence. After all symbols are predicted, a function of the payoff values was used to calculate the overall worth of each FSM. From this set of parent FSMs, offspring FSMs were generated by mutating each parent machine using five possible methods: (1) change an output symbol, (2) change a state transition, (3) add a state, (4) delete a state, and (5) change the initial state. The mutation operators and the number of mutation operations were chosen using a uniform and Poisson probability distribution, respectively, or other specified probability distributions. Like the parent solutions, each offspring solution was then evaluated over the sequence of symbols and scored. FSMs providing the highest payoff were retained as parent solutions for a subsequent generation of FSMs. Standard EP approaches kept the best half of the population to serve as parents for the subsequent generation of solutions and one offspring was generated from each parent solution to maintain a constant population size. The process is continued until all symbols were trained and a prediction of a symbol yet to be encountered is required (equivalent to a testing example after a series of training examples has been learnt). Many of Fogel's early experiments applied evolved FSMs to successively more difficult tasks ranging from simple two-symbol cyclic sequences to eight-symbol cyclic sequences, with noise, to sequences of symbols generated by other FSMs, and even to nonstationary sequences and other number series taken from an article by Flood (1962).

In 1965, Fogel, Owens, and Walsh left General Dynamics to form a company, Decision Science, Inc., in San Diego, California to explore applications of evolutionary programming. This was the first company dedicated to the application of evolutionary computation. The company continued for 17 years applying EP for time series prediction and modeling aircraft combat. Many of these applications were aided by the expertise of George Burgin, another employee of the company. The company was eventually acquired by Titan Systems in 1982.

In 1966, Fogel, Owens, and Walsh published the first book in the field of evolution computation entitled *Artificial Intelligence through Simulated Evolution*, which described the evolution of FSMs in detail and introduced evolutionary computation broadly to the AI community. Surprisingly, the reaction by the AI community was largely negative (see Fogel 1999 for additional details).

Fogel and Burgin (1969) applied EP to discover globally optimal strategies in two-player, zero-sum games. This early application of evolutionary computation to game theory was later extended to gaming situations that outperformed human subjects in nonzero-sum games such as pursuit evasion for aerial combat. Such models evolved the path for an interceptor toward

a moving target in three dimensions and were capable of predicting the position of the target quite successfully, even at this early stage in processing speed. This process, resulting in an intelligently interactive opponent, was called an adaptive maneuvering logic (AML) (Burgin et al. 1975). AML achieved a level of proficiency that even impressed pilots at the level of accomplished U.S. Navy Blue Angels (Burgin 2008). It was at about this same time that Fogel became the second president of the American Cybernetics Society, succeeding Warren S. McCulloch.

Walsh et al. (1970) presented an early application of EP for automatic control of a system with direct comparison to human control. FSMs were used to generate predictions of a future symbol from a sequence of input symbols. In this project, the overall output of the evolved system was itself a combination of optimized submachines. When compared to humans, the evolved models consistently outperformed their human counterparts in terms of prediction accuracy.

Throughout the 1970s, Don Dearholt of New Mexico State University supervised several masters theses on the subject of EP (Cornett 1972; Lyle 1972; Montez 1974; Root 1970; Trellue 1973; Vincent 1976; Williams 1977) in addition to Wirt Atmar's (1976) Ph.D. dissertation, also on EP and its relation to machine intelligence. Dearholt's interest in EP was reviewed in Fogel (2004), and during the 1970s, more evolutionary computation research was published in EP than any other form of simulated evolution. Atmar (1976) provided an early example of simulating evolution in an artificial life setting, and suggested how EP could be designed for what is now termed “evolvable hardware.”

2.1 Philosophical Differences

As mentioned previously, EP was constructed as a simulation of phenotypic evolution in accordance with the neo-Darwinian paradigm that selection operates only on phenotypic behavior rather than directly at the level of the genotype. The underlying code of the phenotype (i.e., the genotype) is only affected indirectly by natural selection. This philosophy represents a “top-down” rather than “bottom-up” approach to optimization. Later approaches, such as Holland's genetic algorithm (GA), focused on a decidedly “bottom-up” view of evolution, relying on the identification, combination, and survival of “good” building blocks (i.e., schemata) to iteratively form better and larger building blocks. Thus, in GA, the importance was directed specifically at the level of the genotype and methods of recombination of representations, rather than the phenotype. The resulting building block hypothesis carries an implicit assumption that fitness is a separable part of the simulated genome. In EP, solutions are only judged on their phenotypic behavior in light of an environment without credit apportionment to the representation. EP and evolution strategies (ES) share great similarity in this regard.

3 Extensions

The canonical EP was extended in many ways over time. For example, in continuous EP, new individuals in the population are added without iterative generations on a continual basis (Fogel and Fogel 1995). This approach is similar to $(\pi+1)$ selection in ES. Self-adaptive strategies have been added to EP to adjust the probabilities of mutation through their own

process of evolutionary optimization (Fogel et al. 1991, 1992). Commonly, in standard evolutionary algorithms, the probabilities associated with variation operators are fixed by the user in advance of the evolutionary process. With self-adaptation, these probabilities are modified concurrent with the evolutionary process, retaining a record of which probabilities and parameters worked best in previous generations and adjusting future probabilities in light of those successes. This approach avoids the necessity for user-tuned probabilities associated with each variation operator and adapts the variances via

$$x_i(k+1) := x_i(k) + v_i(k) \times N_i(0, 1)$$

$$v_i(k+1) := v_i(k) + [\sigma v_i(k)] 1/2 \times N_i(0, 1)$$

where x_i and v_i are the i th components of x and v , respectively, and $N_i(0,1)$ represents a standard normal distribution. To ensure that the variance v_i remains nonnegative, σ was introduced. Further information on this process is offered in Fogel (2006) and Porto (1997). Another extension of EP focused on altering the probability distribution associated with EP applied to continuous parameter optimization where the mutation operator is Gaussian with zero mean and variance. By altering this distribution to Cauchy or other distributions (even combinations of distributions), the efficiency of the search can be improved for some functions (Yao and Liu 1996, 1999; Fogel 1999). Other extensions include the addition of random extinction events to the evolving population (Greenwood et al. 1999). EP has also been used to optimize a wide range of modeling representations from simple FSMs to neural networks (Fogel et al. 1991), to fuzzy if-then rule sets (Fogel and Cheung 2005; Porto et al. 2005). The first real-world application of evolved neural networks with EP was offered by Porto (1989), who evolved neural network classifiers for active sonar returns to classify underwater metal objects from background. This approach was further developed in Porto et al. (1995), and also by Yao (1991), Yao and Liu (1997a, b), and others. The reader is directed to the award-winning review by Yao (1999) for an extensive survey of this approach. Yao and Liu (1997a, b) developed EPNet, an evolutionary system for evolving neural networks that makes use of EP. This approach made use of many concepts of EP described above including continuous EP and was reviewed in Yao and Islam (2008) for evolved neural network ensembles. Further, the approach has been extended by Yao and others to include the evolution of neural network ensembles. Martinez-Estudillo et al. (2006) generated an extension of neural networks optimized by EP via of nodes whose basis function units are products of the inputs raised to a real number power (termed “product units”).

4 Recent Applications

Like many evolutionary algorithms, EP has a wide breadth of applications. For example, within robotic control, Chen et al. (2009) used EP coupled with fuzzy logic for the control of wheeled robots. EP was also used to optimize the control gains for the kinematic control system. In the area of mobile network strategic planning, Portilla-Figueras et al. (2008) applied EP to discover optimal cell size determination. This approach was compared to several standard approaches, and obtained significantly better results in many scenarios. Hoofar (2007) reviewed applications of EP continuous, discrete, and mixed parameter electromagnetic optimization problems including RF circuit design. Guo et al. (2008) introduced a method to optimize wavelet filters using a combination of EP and a chaos (a so-called chaos evolutionary

programming) algorithm. Jamnejad and Hoofar (2004) optimized multifrequency feed horns for reflector antennas of the Jet Propulsion Laboratory/NASA Deep Space Network. The resulting designs were optimized in the light of constraints on return loss, antenna beam width, pattern circularity, and low cross-polarization.

EP has also received broad application in power plant control and optimization in the last decade. Sahoo (2008) evaluated the ability of EP to optimize the performance of a cogeneration system. The method used a canonical EP to search exergoeconomic parameters for the system, and relate these to the monetary cost of running the facility at a particular efficiency of electricity generation. Repeated results on several cogeneration systems demonstrated that EP was able to identify useful parameter settings. Similarly, Contreras-Hernández and Cedeño-Maldonado (2008) used EP to calculate voltage magnitudes for power systems following a power outage. This approach was evaluated on a modified IEEE 30-bus system and found to provide accurate solutions with minimal computational resources. Fong et al. (2007) optimized the installation of solar panels for centralized solar water-heating systems in high-rise buildings to maximize energy savings using EP.

Other applications include optimized methods for impedance-based structural health monitoring approaches (Xu et al. 2004), flow-shop scheduling (Wang and Zheng 2003), and economic load dispatch (Sinha et al. 2003). EP has a long history of application in biological, medical, and chemical applications including molecular docking (Gehlhaar et al. 1995; Kissinger et al. 1999) and optimization of neural networks for quantitative structure–activity prediction (Hecht and Fogel 2007) and prediction of viral tropism from sequence information (Lamers et al. 2008).

Given the no free lunch theorem (Wolpert and Macready 1997) and the realization of the basic similarities of all evolutionary algorithms (population-based random variation and selection), the importance of one particular EA approach versus another has become blurred as the field of evolutionary computation has grown. EP remains a historically important and currently valuable form within this larger community.

References

- Atmar JW (1976) Speculation on the evolution of intelligence and its possible realization in machine form. Doctoral dissertation, New Mexico State University, Las Cruces
- Burgin GH (2008) In memoriam – reflections on Larry Fogel at Decision Science, Inc. (1965–1982). *IEEE Computational Intelligence Magazine* 69–72
- Burgin GH, Fogel LJ, Phelps JP (1975) An adaptive maneuvering logic computer program for the simulation of one-on-one air-to-air combat, NASA Contractor Report NASA CR-2582, Washington, DC
- Chen C-Y, Li T-HS, Yeh Y-C (2009) EP-based kinematic control and adaptive fuzzy sliding-mode dynamic control for wheeled mobile robots. *Info Sci* 179:180–195
- Contreras-Hernández EJ, Cedeño-Maldonado JR (2008) Evolutionary programming applied to branch outage simulation for contingency studies. *Int J Power Energy Syst* 28:48–53
- Cornett FN (1972) An application of evolutionary programming to pattern recognition. Master's thesis, New Mexico State University, Las Cruces
- Flood MM (1962) Stochastic learning theory applied to choice experiments with cats, dogs and men. *Behav Sci* 7:289–314
- Fogel DB (2004) In memoriam Don Dearholt. *IEEE Trans Evol Comput* 8:97–98
- Fogel DB (2006) Evolutionary computation: toward a new philosophy of machine intelligence, 3rd edn. Wiley-IEEE Press, Piscataway, NJ
- Fogel DB, Fogel LJ, Atmar JW (1991) Meta-evolutionary programming. In: Chen RR (ed) Proceedings of 25th Asilomar conference on signals, systems, and computers. IEEE Computer Society, Los Alamitos, CA, pp 540–545
- Fogel DB, Fogel LJ, Atmar JW, Fogel GB (1992) Hierarchic methods of evolutionary programming. In: Fogel DB,

- Atmar W (eds) Proceedings of 1st annual conference on evolutionary programming. Evolutionary Programming Society, La Jolla, CA, pp 175–182
- Fogel GB, Cheung M (2005) Derivation of quantitative structure-toxicity relationships for exotoxicological effects of organic chemicals: evolving neural networks and evolving rules. In: 2005 IEEE congress on evolutionary computation. IEEE, Edinburgh, pp 274–281
- Fogel GB, Fogel DB (1995) Continuous evolutionary programming: analysis and experiments. *Cybernet Syst* 26:79–90
- Fogel LJ (1962a) Autonomous automata. *Indus Res* 4:14–19
- Fogel LJ (August, 1962b) Toward inductive inference automata. In: Proceedings of the congress, International Federation for Information Processing, Munich
- Fogel LJ (1963) Biotechnology: concepts and applications. Prentice-Hall, Englewood, NJ
- Fogel LJ (1964) On the organization of intellect. Doctoral dissertation, University of California, Los Angeles, CA
- Fogel LJ (1990) The future of evolutionary programming. In: Chen RR (ed) Proceedings of 24th Asilomar conference on signals, systems, and computers. Maple Press, Pacific Grove, CA, pp 1036–1038
- Fogel LJ (1999) Intelligence through simulated evolution: forty years of evolutionary programming. Wiley, New York
- Fogel LJ, Burgin GH (1969) Competitive goal-seeking through evolutionary programming. Air Force Cambridge Research Labs, Final Report, Contract AF 19(628)–5927
- Fogel LJ, Owens AJ, Walsh MJ (1964) On the evolution of artificial intelligence. In: Proceedings of 5th National Symp. on Human Factors in Engineering. IEEE, San Diego, CA, pp 63–67
- Fogel LJ, Owens AJ, Walsh MJ (1965a) Intelligent decision-making through a simulation of evolution. *Behav Sci* 11:253–272
- Fogel LJ, Owens AJ, Walsh MJ (1965b) Intelligent decision-making through a simulation of evolution. *IEEE Trans Hum Factors Electron* 6:13–23
- Fogel LJ, Owens AJ, Walsh MJ (1965c) Artificial intelligence through a simulation of evolution. In: Maxfield M, Callahan A, Fogel LJ (eds) Biophysics and cybernetic systems: Proceedings of 2nd cybernetic science symposium. Spartan Books, Washington, DC, pp 131–155
- Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, New York
- Fong KF, Chow TT, Hanby VI (2007) Development of optimal design of solar water heating system by using evolutionary algorithm. *J Solar Energy Eng* 129:499–501
- Gehlhaar DK, Verkhivker GM, Rejto PA, Sherman CJ, Fogel DB, Fogel LJ, Freer ST (1995) Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: Conformationally flexible docking by evolutionary programming. *Chem Biol* 2:317–324
- Greenwood GW, Fogel GB, Ciobanu M (1999) Emphasizing extinction in evolutionary programming. In: Proceedings of 1999 congress on evolutionary computation. IEEE, Washington DC, pp 666–671
- Guo S-M, Chang W-H, Tsai JSH, Zhuang B-L, Chen L-C (2008) JPEG 2000 wavelet filter design framework with chaos evolutionary programming. *Signal Process* 88:2542–2553
- Hecht D, Fogel GB (2007) High-throughput ligand screening via preclustering and evolved neural networks. *IEEE/ACM Trans Comput Biol Bioinform* 4:476–484
- Hoorfar A (2007) Evolutionary programming in electromagnetic optimization: a review. *IEEE Trans Antennas Propagation* 55:523–537
- Jamnejad V, Hoorfar A (2004) Design of corrugated horn antennas by evolutionary optimization techniques. *IEEE, Antennas Wireless Propagation Lett* 3:276–279
- Kissinger CR, Gehlhaar DK, Fogel DB (1999) Rapid automated molecular replacement by evolutionary search. *Acta Crystallogr D Biol Crystallogr* 55:484–491
- Lamers SL, Salemi M, McGrath MS, Fogel GB (2008) Prediction of R5, X4, and R5X4 HIV-1 coreceptor usage with evolved neural networks. *IEEE/ACM Trans Comput Biol Bioinform* 5:291–299
- Lyle MR (1972) An investigation into scoring techniques in evolutionary programming. Master's thesis, New Mexico State University, Las Cruces
- Mealy GH (1955) A method of synthesizing sequential circuits. *Bell Syst Tech J* 34:1054–1079
- Martinez-Estudillo A, Martinez-Estudillo F, Hervas-Martinez C, Garcia-Pedrajas N (2006) Evolutionary product unit based neural networks for regression. *Neural Netw* 19:477–486
- Montez J (1974) Evolving automata for classifying electrocardiograms. Master's thesis, New Mexico State University, Las Cruces
- Moore EF (1957) Gedanken-experiments on sequential machines: automata studies. *Ann Math Stud* 34: 129–153
- Portilla-Figueras JA, Salcedo-Sanz S, Oropesa-García A, Bousño-Calzón C (2008) Cell size determination in WCDMA systems using an evolutionary programming approach. *Comput Oper Res* 35:3758–3768
- Porto VM (1989) Evolutionary methods for training neural networks for underwater pattern classification. 24th Ann. Asilomar conf. on signals, systems, and computers, vol 2, pp 1015–1019
- Porto VW (1997) Evolutionary programming. In: Handbook on evolutionary computation. IOP, Bristol. Oxford University Press, New York
- Porto VW, Fogel DB, Fogel LJ (1995) Alternative neural network training methods. *IEEE Expert* 10(3):16–22

- Porto VW, Fogel DB, Fogel LJ, Fogel GB, Johnson N, Cheung M (2005) Classifying sonar returns for the presence of mines: evolving neural networks and evolving rules. In: Fogel DB Piuri V (eds) 2005 IEEE symposium on computational intelligence for homeland security and personal safety, IEEE Press, Piscataway, NJ, pp 123–130
- Root R (1970) An investigation of evolutionary programming. Master's thesis, New Mexico State University, Las Cruces
- Sahoo PK (2008) Exergoeconomic analysis and optimization of a cogeneration system using evolutionary programming. *Appl Thermal Eng* 28: 1580–1588
- Sinha N, Chakrabarti R, Chattopadhyay PK (2003) Evolutionary programming techniques for economic load dispatch. *IEEE Trans Evol Comput* 7:83–94
- Trellue RE (1973) The recognition of handprinted characters through evolutionary programming. Master's thesis, New Mexico State University, Las Cruces
- Vincent RW (1976) Evolving automata used for recognition of digitized strings. Master's thesis, New Mexico State University, Las Cruces
- Walsh MJ, Burgin GH, Fogel LJ (1970) Prediction and control through the use of automata and their evolution. U.S. Navy Final Report Contract N00014-66-C-0284
- Wang L, Zheng D-Z (2003) A modified evolutionary programming for flow shop scheduling. *Int J Adv Manuf Technol* 22:522–527
- Williams GL (1977) Recognition of hand-printed numerals using evolving automata. Master's thesis, New Mexico State University, Las Cruces
- Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1:67–82
- Xu J, Yang Y, Soh CK (2004) Electromechanical impedance-based structural health monitoring with evolutionary programming. *J Aerospace Eng* 17:182–193
- Yao X (1991) Evolution of connectionist networks. In: Proceedings of the international symposium on AI, reasoning and creativity. Griffith University, Queensland, Australia, pp 49–52
- Yao X (1999) Evolving artificial neural networks. *Proc IEEE* 87:1423–1447
- Yao X, Islam M (February, 2008) Evolving artificial neural network ensembles. *IEEE Comput Mag* 3(1):31–42
- Yao X, Liu Y (1996) Fast evolutionary programming. In: Fogel LJ, Angeline PJ, Bäck T (eds) Evolutionary programming V: Proceedings of 5th annual conference on evolutionary programming. MIT Press, Cambridge, MA, pp 451–460
- Yao X, Liu Y (1997a) EPNet for chaotic time-series prediction. In: Yao X, Kim J-H Furuhashi T (eds) Simulated evolution and learning. Springer, Berlin, pp 146–156
- Yao X, Liu Y (1997b) A new evolutionary system for evolving artificial neural networks. *IEEE Trans Neural Netw* 8:694–713
- Yao X, Liu Y (1999) Evolutionary programming made faster. *IEEE Trans on Evol Prog* 3:82–102

24 Genetic Programming — Introduction, Applications, Theory and Open Issues

*Leonardo Vanneschi*¹ · *Riccardo Poli*²

¹Department of Informatics, Systems and Communication, University of
Milano-Bicocca, Italy

vanneschi@disco.unimib.it

²Department of Computing and Electronic Systems, University of Essex,
Colchester, UK

rpoli@essex.ac.uk

1	<i>Introduction</i>	710
2	<i>The Mechanics of Tree-Based GP</i>	711
3	<i>Examples of Real-World Applications of GP</i>	719
4	<i>GP Theory</i>	720
5	<i>Open Issues</i>	729
6	<i>Conclusions</i>	733

Abstract

Genetic programming (GP) is an evolutionary approach that extends genetic algorithms to allow the exploration of the space of computer programs. Like other evolutionary algorithms, GP works by defining a goal in the form of a quality criterion (or fitness) and then using this criterion to evolve a set (or population) of candidate solutions (individuals) by mimicking the basic principles of Darwinian evolution. GP breeds the solutions to problems using an iterative process involving the probabilistic selection of the fittest solutions and their variation by means of a set of genetic operators, usually crossover and mutation. GP has been successfully applied to a number of challenging real-world problem domains. Its operations and behavior are now reasonably well understood thanks to a variety of powerful theoretical results. In this chapter, we introduce the main definitions and features of GP and describe its typical operations. We then survey some of its applications. We also review some important theoretical results in this field, including some very recent ones, and discuss some of the most challenging open issues and directions for future research.

1 Introduction

Genetic algorithms (GAs) are capable of solving many problems competently. Furthermore, in their simplest realizations, they have undergone a variety of theoretical studies, so that a solid understanding of their properties is now available. Nevertheless, the fixed-length string-type representation of individuals that characterizes GAs is unnatural and overly constraining for a wide set of applications. These include the evolution of computer programs, where, rather obviously, the most natural representation for a solution is a hierarchical, variable size structure rather than a fixed-length character string. In particular, fixed-length strings do not readily support the hierarchical organization of tasks into subtasks typical of computer programs; they do not provide a convenient way of incorporating iteration and recursion; and so on. Above all, traditional GA representations cannot be dynamically varied at run time. The initial choice of string-length limits in advance the number of internal states of the system, thereby setting an upper bound on what the system can learn.

This lack of representation power (already recognized two decades ago, for example, in De Jong (1988)) is overcome by genetic programming (GP), an extension of GAs that largely owes its success to Koza (1992a), who defines GP as:

- ▶ a systematic method for getting computers to automatically solve a problem starting from a high-level statement of what needs to be done (Koza and Poli 2003).

In some senses, GP represents an attempt to accomplish one of the most ambitious goals of computer science: being able to specify what one wants a program to do, but not how, and have the computer figure out an implementation.

At a slightly lower level of abstraction, GP basically works like a GA. As GAs do for fixed length strings of characters, GP genetically breeds a population of computer programs to solve a problem. The major difference between GAs and GP is that in the former the individuals

in the population are fixed-length strings of characters, in the latter they are hierarchical variable-size structures that represent *computer programs*.

Every programming language (e.g., Pascal or C) is capable of expressing and executing general computer programs. Koza, however, chose the *LISP* (*LISt Processing*) language to code his GP implementation. Besides the fact that many sophisticated programming tools were available for LISP at the time, the language presented a variety of advantages. Both programs and data have the same form in LISP, so that it is possible and convenient to treat a computer program as data in the genetic population. This common form of programs and data is the list. Lists can be nested and, therefore, can easily represent hierarchical structures such as *syntax trees*. Syntax trees are particularly suitable as representations of computer programs. Their size, shape, and primitives can be changed dynamically by genetic operators, thereby imposing no *a priori* limit on the complexity of what's evolved by GP. LISP facilitates the programming of such manipulations.

While modern GP implementations are based on C, C++, Java, or Python, rather than LISP, and syntax trees are nowadays often represented using a flattened array-based representation rather than linked lists (typical of LISP), from a logical point of view programs are still treated and manipulated as trees as Koza did. This form of GP will be called *tree-based GP* hereafter.

While the tree-based representation of individuals is the oldest and most common one, it is not the only form of GP that has been employed to evolve computer programs. In particular, in the last few years, a growing attention has been dedicated by researchers to linear genomes (see for instance Brameier and Banzhaf 2001) and graph-based genomes (see for instance Miller 2001). Many other forms of GP, for example, based on grammars or based on the estimation of probability distributions, have also been proposed. However, for space limitations these forms are not covered in this chapter. The interested reader is referred to Poli et al. (2008) for a more comprehensive review.

This chapter is structured as follows. ⚡ Sect. 2 introduces the main definitions and features of GP and its operators. ⚡ Sect. 3 offers an outline of some of the most significant real-world applications of GP. ⚡ Sect. 4 reviews the most significant theoretical results to date on GP, including some recent and exciting new advances. ⚡ Sect. 5 lists some important open issues in GP. ⚡ Sect. 6 provides some conclusions.

2 The Mechanics of Tree-Based GP

In synthesis, the GP paradigm works by executing the following steps:

1. Generate an initial population of computer programs (or individuals).
2. Perform the following steps until a termination criterion is satisfied:
 - (a) Execute each program in the population and assign it a fitness value according to how well it solves the problem.
 - (b) Create a new population by applying the following operations:
 - i. Probabilistically select a set of computer programs for mating, on the basis of their fitness (selection).
 - ii. Copy some of the selected individuals, without modifying them, into the new population (reproduction).

- iii. Create new computer programs by genetically recombining randomly chosen parts of two selected individuals (crossover).
- iv. Create new computer programs by substituting randomly chosen parts of some selected individuals with new randomly generated ones (mutation).
- 3. The best computer program which appeared in any generation is designated as the result of the GP process. This result may be a solution (or an approximate solution) to the problem.

In the following sections, each step of this process is analyzed in detail.

2.1 Representation of GP Individuals

The set of all the possible structures that GP can generate includes all the trees that can be built recursively from a set of function symbols $\mathcal{F} = \{f_1, f_2, \dots, f_n\}$ (used to label internal tree nodes) and a set of terminal symbols $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$ (used to label the leaves of GP trees). Each function in the function set \mathcal{F} takes a fixed number of arguments, known as its *arity*. Functions may include arithmetic operations ($+, -, \times$, etc.), mathematical functions (such as \sin , \cos , \log , \exp), Boolean operations (such as *AND*, *OR*, *NOT*), conditional operations (such as *If-Then-Else*), iterative operations (such as *While-Do*), and other domain-specific functions that may be defined to suit the problem at hand. Each terminal is typically either a variable or a constant.

For example, given the set of functions $\mathcal{F} = \{+, -\}$ and the set of terminals $\mathcal{T} = \{x, 1\}$, a legal GP individual is represented in Fig. 1. This tree can also be represented by the LISP-like S-expression $(+ x (- x 1))$ (for a definition of LISP S-expressions see, for instance, Koza (1992)).

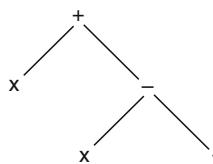
It is good practice to choose function and terminal sets that satisfy two requirements: *closure* and *sufficiency*.

The *closure property* requires that each of the functions in the function set be able to accept, as its arguments, any value and data type that may possibly be returned by any function in the function set and any value and data type that may possibly result from the evaluation of any terminal in the terminal set. In other words, each function should be defined (and behave properly) for any combination of arguments that it may encounter. This property is essential since, clearly, programs must be executed in order to assign them a fitness. A failure in the execution of one of the programs composing the population would lead either to a failure of the whole GP system or to unpredictable results.

The function and terminal sets used in the previous example clearly satisfy the closure property. The sets $\mathcal{F} = \{*, /\}$ and $\mathcal{T} = \{x, 0\}$, however, do not satisfy this property. In fact, each evaluation of an expression containing an operation of division by zero would lead to unpredictable behavior. In practical applications, the division operator is often modified in order to make sure the closure property holds.

Fig. 1

A tree that can be built with the sets $\mathcal{F} = \{+, -\}$ and $\mathcal{T} = \{x, 1\}$.



Respecting the closure property in real-world applications is not always straightforward, particularly if the use of different data types is required by the problem domain. A common example is the mixing of Boolean and numeric functions. For example, if one used a function set composed by Boolean functions (*AND*, *OR*, ...), arithmetic functions (+, −, *, /, ...), comparison functions (>, <, =, ...), and conditionals (*IF THEN ELSE*), expressions such as:

$$\text{IF } ((x > 10 * y) \text{ AND } (y > 0)) \text{ THEN } z + y \text{ ELSE } z * x$$

might easily be evolved. In such cases, introducing typed functions and type-respecting genetic operations in the GP system can help enforce closure. This is achieved, for example, by *strongly typed GP* (Banzhaf et al. 1998), a GP system in which each primitive carries information about its type as well as the types it can call, thus forcing functions calling it to cast arguments into the appropriate types. Using types make even more sense in GP than for human programmers, since human programmers have a mental model of what they are doing, whereas the GP system is completely random in its initialization and variation phases. Furthermore, type consistency reduces the search space, which is likely to improve chances of success for the search.

The *sufficiency property* requires that the set of terminals and the set of functions be capable of expressing a solution to the problem. For instance, a function set including the four arithmetic operations combined with a terminal set including variables and constants respects the sufficiency property if the problem at hand requires the evolution of an analytic function. (Any function can be approximated to the desired degree by a polynomial via Taylor's expansion.) The primitive set $\mathcal{F} = \{+, -\}$, on the other hand, would not be sufficient, since it can only represent linear functions.

For many domains, there is no way of being sure whether sufficiency holds for the chosen set. In these cases, the definition of appropriate sets depends very much on prior knowledge about the problem and on the experience of the GP designer.

2.2 Initialization of a GP Population

The initialization of the population is the first step of the evolution process. It involves the creation of the program structures that will later be evolved. The most common initialization methods in tree-based GP are the *grow* method, the *full* method and the *ramped half-and-half* method (Koza 1992a). These methods will be explained in the following paragraphs. All are controlled by a parameter: the maximum depth allowed for the trees, d .

When the *grow method* is employed, each tree of the initial population is built using the following algorithm:

- A random symbol is selected with uniform probability from the function set \mathcal{F} to be the root of the tree; let n be the arity of the selected function symbol.
- n nodes are selected with uniform probability from the union of the function set and the terminal set, $\mathcal{F} \cup \mathcal{T}$, to be its children;
- For each function symbol within these n nodes, the grow method is recursively invoked, that is, its children are selected from the set $\mathcal{F} \cup \mathcal{T}$, unless the node has a depth equal to $d-1$. In the latter case, its children are selected from \mathcal{T} .

While this method ensures that no parts of the tree are deeper than d , the drawing of primitives from $\mathcal{F} \cup \mathcal{T}$ allows branches to be fully leafed before they reach this maximum

depth. So, initial trees of varied shape and size typically result from the use of the grow method. One should note, however, that the exact distribution of tree shapes depends in a nontrivial way on the ratio between the number of primitives of each arity and the number of terminals. If the primitive set contains a large number of terminals, the trees produced by the grow method tend to be small. On the contrary, if there are a few terminals and many functions with an arity of 2 or more, branches will tend to reach the maximum depth d , resulting in almost full trees.

Instead of selecting nodes from $\mathcal{F} \cup \mathcal{T}$, the *full method* chooses only function symbols until the maximum depth is reached. Then it chooses only terminals. The result is that every branch of the tree goes to the full maximum depth.

As first noted by Koza (1992a), population initialized with the above two methods may lack diversity. To fix this, Koza suggested to use a technique he called *ramped half-and-half*. This works as follows. With the ramped half-and-half method, a fraction $\frac{1}{d}$ of the population is initialized with trees having a maximum depth of 1, a fraction $\frac{1}{d}$ with trees of maximum depth 2, and so on. For each depth group, half of the trees are initialized with the full technique and half with the grow technique. In this way, the initial population is composed by a mix of large, small, full, and unbalanced trees, thus ensuring a certain amount of diversity.

2.3 Fitness Evaluation

Each program in the population is assigned a fitness value, representing the degree to which it solves the problem of interest. This value is calculated by means of some well-defined procedure. Two fitness measures are most commonly used in GP: the *raw fitness* and the *standardized fitness*. They are described below.

Raw fitness is the measurement of fitness that is stated in the natural terminology of the problem itself. In other words, raw fitness is the simplest and most natural way to calculate the degree to which a program solves a problem. For example, if the problem consists of controlling a robot so that it picks up as many of the objects contained in a room as possible, then the raw fitness of a program could simply be the number of objects actually picked up at the end of its execution.

Often, but not always, raw fitness is calculated over a set of *fitness cases*. A fitness case corresponds to a representative situation in which the ability of a program to solve a problem can be evaluated. For example, consider the problem of generating an arithmetic expression that approximates the polynomial $x^4 + x^3 + x^2 + x$ over the set of natural numbers smaller than 10. Then, a fitness case is one of those natural numbers. Suppose $x^2 + 1$ is the functionality of a program one needs to evaluate. Then, $2^2 + 1 = 5$ is the value taken by this expression over the fitness case 2. Raw fitness is then defined as the sum over all fitness cases of the distances between the target values expected from a perfect solution and the values actually returned by the individual being evaluated. Formally, the raw fitness, f_R , of an individual i , calculated over a set of N fitness cases, can be defined as

$$f_R(i) = \sum_{j=1}^N |S(i, j) - C(j)|^k \quad (1)$$

where $S(i, j)$ is the value returned by the evaluation of individual i over fitness case j , $C(j)$ is the correct output value associated to fitness case j and k is some natural number (often either $k=1$ or $k=2$).

Fitness cases are typically a small sample of the entire domain space. The choice of how many fitness cases (and which ones) to use is often a crucial one since whether or not an evolved solution will generalize over the entire domain depends on this choice. In practice, the decision is made on the basis of knowledge about the problem and practical performance considerations (the bigger the training set, the longer the time required to evaluate fitnesses). A first theoretical study on the suitable number of fitness cases to be used has been presented in Giacobini et al. (2002).

Standardized fitness is a reformulation of the raw fitness so that lower numerical values are better. When the smaller the raw fitness the better (as in the case in [Eq. 1](#)), then the standardized fitness, f_S , can simply be equal to the raw fitness, that is $f_S = f_R$. It is convenient and desirable to ensure the best value of standardized fitness is equal to zero. When this is not the case, this can be obtained by subtracting or adding a constant to the raw fitness. If, for a particular problem, the bigger the raw fitness the better, the standardized fitness f_S of an individual can be defined as $f_S = f_R^{\max} - f_R$, where f_R is the raw fitness of the individual and f_R^{\max} is the maximum possible value of raw fitness (which is assumed to be known).

2.4 Selection

Each individual belonging to a GP population can undergo three different operations: genetic operators can be applied to that individual, it can be copied into the next generation as it is, or it can be discarded. *Selection* is the operator that decides whether a chance of reproduction is given to an individual. It is a crucial step for GP, as is for all EAs, since the success and pace of evolution often depends on it.

Many selection algorithms have been developed. The most commonly used are *fitness proportionate selection*, *ranking selection*, and *tournament selection*. All of them are based on fitness: individuals with better fitness have higher probability of being chosen for mating. Selection mechanisms used for GP are typically identical to the ones used for GAs and for other EAs. They will not be discussed in detail in this chapter, since they are introduced elsewhere in this book.

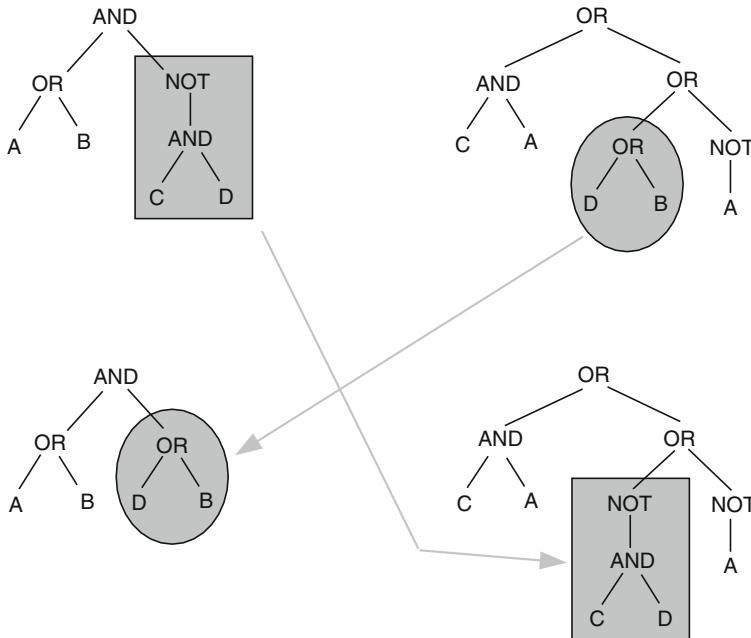
2.5 Crossover

The *crossover* or sexual *recombination* operator creates variation in the population by producing offspring that consist of genetic material taken from the parent. The parents, T_1 and T_2 , are chosen by means of a selection method.

Standard GP crossover (often called *subtree crossover*) (Koza 1992a) begins by independently selecting a random node in each parent – we will call it the *crossover point* for that parent. Usually crossover points are chosen in such a way that internal nodes are picked with a probability of 0.9 and any node (internal node or terminal) with a probability 0.1. The *crossover fragment* for a particular parent is the subtree rooted at the crossover point. An offspring is produced by deleting the crossover fragment of T_1 from T_1 and inserting the crossover fragment of T_2 at the crossover point of T_1 . Some implementations also return a second offspring, which is produced in a symmetric manner. Figure [2](#) shows an example of standard GP crossover. Because entire subtrees are swapped and because of the closure property of the primitive set, crossover always produces syntactically legal programs.

Fig. 2

An example of standard GP crossover. Crossover fragments are indicated by the shaded areas.



It is important to remark that when one of the crossover points is a leaf while the other is the root of a tree the offspring can be much bigger and deeper than the parents. While this may be desirable at early stages of a run, in the presence of the phenomenon of *bloat* – the progressive growth of the code size of individuals in the population generation after generation (more on this in [Sect. 4.4](#)) – some limit may have to be imposed on offspring size. Typically, if the offspring's size or depth is beyond a limit, the offspring is either rejected (and crossover attempted again) or accepted in the population but is given a very low fitness.

Many variants of the standard GP crossover have been proposed in the literature. The most common ones consist in assigning probabilities of being chosen as crossover points to nodes, based on node depth. In particular, it is very common to assign low probabilities to crossover points near the root and the leaves, so as to reduce the occurrence of the phenomenon described above. Another kind of GP crossover is *one-point crossover*, introduced in Poli and Langdon (1997), where the crossover points in the two parents are forced to be at identical positions. This operator was important in the development of a solid GP theory (see [Sect. 4](#)), since it provided a stepping stone that allowed the extension to GP of corresponding GA theory.

2.6 Mutation

Mutation is asexual, that is, it operates on only one parental program.

Standard GP mutation, often called *subtree mutation*, begins by choosing a *mutation point* at random, with uniform probability, within a selected individual. Then, the subtree rooted at the mutation point is removed and a new randomly generated subtree is inserted at that point.

Fig. 3

An example of standard GP mutation.

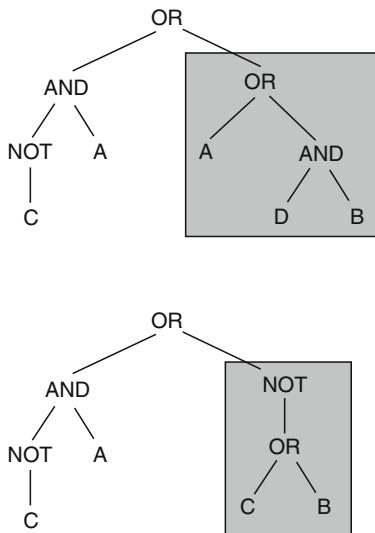


Figure 3 shows an example of standard GP mutation. As it is the case for standard crossover this operation is controlled by a parameter that specifies the maximum depth allowed, thereby limiting the size of the newly created subtree that is to be inserted. Nevertheless, the depth of the generated offspring can be considerably larger than that of the parent.

Many variants of standard GP mutation have been developed. The most commonly used are the ones aimed at limiting the probability of selecting as mutation points the root and/or the leaves of the parent. Another common form of mutation is *point mutation* (Poli and Langdon 1997) that randomly replaces nodes with random primitives of the same arity. Other commonly used variants of GP mutation are *permutation* (or *swap* mutation) that exchanges two arguments of a node and *shrink mutation* that generates a new individual from a parent's subtree. Another form of mutation is *structural mutation* (Vanneschi 2003, 2004). Being consistent with the edit distance measure, this form of mutation is useful in the study of GP problem difficulty (see [Sect. 4.6](#)).

2.7 Other Variants

In GP systems using a *steady state* scheme, the individuals produced by variation operators are immediately inserted into the population without waiting for a full generation to be complete. To keep the population size constant, a corresponding number of individuals need to be removed from the population. The removed individuals are typically the ones with the worst fitness, but versions of GP systems also exist where the removed individuals are randomly selected or where offspring replace their parents. After the new individuals have been inserted into the population, the process is iterated. A GP system using the traditional (non-steady-state) strategy is called *generational*.

Automatically defined functions (ADFs) are perhaps the best-known mechanisms by which GP can evolve and use subroutines inside individuals. In GP with ADFs, each program is represented by a set of trees, one for each ADF, plus one for the main program. Each ADF may have zero, one, two or more variables that play the role of formal parameters. ADFs allow parametrized reuse of code and hierarchical invocation of evolved code. The usefulness of ADFs has been shown by Koza (1994).

In the same vein, Koza defined automatically defined iterations (ADIs), automatically defined loops (ADLs), automatically defined recursions (ADRs), and automatically defined stores (ADSs) (Koza et al. 1999). All these mechanisms may allow a high degree of reusability of code inside GP individuals. However, their use is still not very widespread.

2.8 GP Parameters

The user of a GP has to decide the set of functions \mathcal{F} and the set of terminals \mathcal{T} to be used to represent potential solutions of a given problem and the exact implementation of the genetic operators to be used to evolve programs. This is not all, though. The user must also set some parameters that control evolution.

The most important parameters are population size, stopping criterion, technique used to create the initial population, selection algorithm, crossover type and rate, mutation type and rate, maximum tree depth, steady state vs. generational update, presence or absence of ADFs or other modularity-promoting structures, and presence or absence of elitism (i.e., the guaranteed survival of the best individual(s) found in each generation). The setting of these parameters represents, in general, an important choice for the performance of the GP system, although, based on our experience, we would suggest that population size and anything to do with selection (e.g., elitism, steady state, etc.) are probably the most important ones.

2.9 GP Benchmarks

Koza (1992a) defines a set of problems that can be considered as “typical GP problems,” given that they mimic some important real-life applications and they are simple to define and to apply to GP. For this reason, they have been adopted by the GP research community as a, more or less, agreed-upon set of benchmarks, to be used in experimental studies. They include the following:

- *The Even k Parity Problem*, whose goal is to find a Boolean function of k arguments that returns *true* if an even number of its arguments evaluates to true, and *false* otherwise.
- *The h Multiplexer Problem*, where the goal is to design a Boolean function with h inputs that approximates a multiplexer function.
- *The Symbolic Regression Problem*, that aims at finding a program that matches a given target mathematical function.
- *The Intertwined Spirals problem*, the goal of which is to find a program to classify points in the $x-y$ plane as belonging to one of two spirals.
- *The Artificial Ant on the Santa Fe trail*, whose goal is to find a target navigation strategy for an agent on a limited toroidal grid.

3 Examples of Real-World Applications of GP

Since its early beginnings, GP has produced a cornucopia of results. Based on the experience of numerous researchers over many years, it appears that GP and other evolutionary computation methods have been especially productive in areas having some or all of the following properties:

- The relationships among the relevant variables is unknown or poorly understood.
- Finding the size and shape of the ultimate solution is a major part of the problem.
- Significant amounts of test data are available in computer-readable form.
- There are good simulators to test the performance of tentative solutions to a problem, but poor methods to directly obtain good solutions.
- Conventional mathematical analysis does not, or cannot, provide analytic solutions.
- An approximate solution is acceptable (or is the only result that is ever likely to be obtained).
- Small improvements in performance are routinely measured (or easily measurable) and highly prized.

The literature, which covers more than 5,000 recorded uses of GP, reports an enormous number of applications where GP has been successfully used as an automatic programming tool, a machine learning tool, or an automatic problem-solving engine. It is impossible to list all such applications here. Thus, to give readers an idea of the variety of successful GP applications, below a representative sample is listed (see Poli et al. (2008) for a more detailed analysis):

- *Curve Fitting, Data Modelling, and Symbolic Regression.* A large number of GP applications are in these three fields (see, e.g., Lew et al. 2006 and Keijzer 2004). GP can be used as a stand-alone tool or coupled with one of the numerous existing feature selection methods, or even GP itself can be used to do feature selection (Langdon and Buxton 2004). Furthermore, GP can also be used in cases where more than one output (prediction) is required. In that case, linear GP with multiple output registers, graph-based GP with multiple output nodes, or a GP operating on single trees with primitives on vectors can be used.
- *Image and Signal Processing.* The use of GP to process surveillance data for civilian purposes, such as predicting motorway traffic jams from subsurface traffic speed measurements was suggested in Howard and Roberts (2004); GP found recurrent filters in Esparcia-Alcazar and Sharman (1996); the use of GP to preprocess images, particularly of human faces, to find regions of interest for subsequent analysis was presented in Trujillo and Olague (2006). Classifications of objects and human speech with GP was achieved in Zhang and Smart (2006) and Xie et al. (2006), respectively. A GP system in which the image processing task is split across a swarm of evolving agents and some of its applications have been described in Louchet (2001). Successful GP applications in medical imaging can be found, for instance, in Poli (1996).
- *Financial Trading, Time Series, and Economic Modelling.* Recent papers have looked at the modeling of agents in stock markets (Chen and Liao 2005), evolving trading rules for the S&P 500 (Yu and Chen 2004), and forecasting the Hang Seng index (Chen et al. 1999). Other examples of financial applications of GP include, for example, Jin and Tsang (2006) and Tsang and Jin (2006).

- *Industrial Process Control.* GP is frequently used in industrial process control, although, of course, most industrialists have little time to spend on academic reporting. A notable exception is Dow Chemical, where a group has been very active in publishing results (Castillo et al. 2006; Jordaan et al. 2006). Kordon (2006) describes where industrial GP stands now and how it will progress. Other interesting contributions in related areas are, for instance, Dassau et al. (2006) and Lewin et al. (2006). GP has also been applied in the electrical power industry (Alves da Silva and Abrao 2002).
- *Medicine, Biology, and Bioinformatics.* Among other applications, GP is used in biomedical data mining. Of particular medical interest are very wide data sets, with typically few samples and many inputs per sample. Recent examples include single nuclear polymorphisms (Shah and Kusiak 2004), chest pain (Bojarczuk et al. 2000), and Affymetrix GeneChip microarray data (Langdon and Buxton 2004; Yu et al. 2007).
- *Computational Chemistry.* Computational chemistry is important in the drug industry. However, the interactions between chemicals that might be used as drugs are beyond exact calculation. Therefore, there is great interest in approximate models that attempt to predict either favorable or adverse interactions between proto-drugs and biochemical molecules. These models can be applied very cheaply in advance of the manufacturing of chemicals, to decide which of the myriad of chemicals is worth further study. Examples of GP approaches include Barrett and Langdon (2006), Hasan et al. (2006), and Archetti et al. (2007).
- *Entertainment, Computer Games, and Art.* Work on GP in games includes, among others, (Azaria and Sipper 2005; Langdon and Poli 2005). The use of GP in computer art is reported for instance in Jacob (2000, 2001). Many recent techniques are described in Machado and Romero (2008).
- *Compression.* GP has been used to perform lossy compression of images (Koza 1992b; Nordin and Banzhaf 1996) and to identify iterated functions system, which are used in the domain of fractal compression (Lutton et al. 1995). GP was also used to evolve wavelet compression algorithms (Klappenecker and May 1995) and nonlinear predictors for images (Fukunaga and Stechert 1998). Recently, a GP system called *GP-ZIP* has been proposed that can do for lossless data compression (Kattan and Poli 2008).
- *Human-Competitive Results.* Getting machines to produce competitive results is the very reason for the existence of the fields of AI and machine learning. Koza et al. (1999) proposed that an automatically created result should be considered “human-competitive” if it satisfies at least one of eight precise criteria. These include patentability, beating humans in regulated competitions, producing publishable results, etc. Over the years, dozens of GP results have passed the human-competitiveness test (see Trujillo and Olague 2006 and Hauptman and Sipper 2007).

4 GP Theory

As discussed in [Sect. 3](#), GP has been remarkably successful as a problem-solving and engineering tool. One might wonder how this is possible, given that GP is a nondeterministic algorithm, and, as a result, its behavior varies from run to run. Why can GP solve problems and how? What goes wrong when it cannot? What are the reasons for certain undesirable behaviors, such as bloat? Below a summary of current research will be given.

4.1 Schema Theories

If we could visualize the search performed by GP, we would often find that initially the population looks like a cloud of randomly scattered points, but that, generation after generation, this cloud changes shape and moves in the search space. Because GP is a stochastic search technique, in different runs we would observe different trajectories. If we could see regularities, these might provide us with a deep understanding of how the algorithm is searching the program space for the solutions, and perhaps help us see why GP is successful in finding solutions in certain runs and unsuccessful in others. Unfortunately, it is normally impossible to exactly visualize the program search space due to its high dimensionality and complexity, making this approach nonviable.

An alternative approach to better understanding the dynamics of GP is to study mathematical models of evolutionary search. Exact mathematical models of GP are probabilistic descriptions of the operations of selection, reproduction, crossover, and mutation. They explicitly represent how these operations determine which areas of the program space will be sampled by GP, and with what probability. There are a number of cases where this approach has been very successful in illuminating some of the fundamental processes and biases in GP systems.

Schema theories are among the oldest and the best-known models of evolutionary algorithms (Holland 1975). These theories are based on the idea of partitioning the search space into subsets, called *schemata*. They are concerned with modelling and explaining the dynamics of the distribution of the population over the schemata. Modern GA schema theory (Stephens and Waelbroeck 1999) provides exact information about the distribution of the population at the next generation in terms of quantities measured at the current generation, without having to actually run the algorithm.

The theory of schemata in GP has had a difficult childhood. However, recently, the first exact GP schema theories have become available (Poli 2001), which give exact formulations for the expected number of individuals sampling a schema at the next generation. Initially, these exact theories were only applicable to GP with one-point crossover (see  Sect. 2.5). However, more recently, they have been extended to other types of crossover including most crossovers that swap subtrees (Poli and McPhee 2003a,b).

Other models of evolutionary algorithms include models based on Markov chain theory. These models are discussed in the next section.

4.2 Markov Chains

Markov chains are important in the theoretical analysis of evolutionary algorithms operating on discrete search spaces (e.g., Davis and Principe 1993). Since such algorithms are stochastic but Markovian (e.g., in GAs and GP the population at the next generation typically depends only on the population at the current generation), all one needs to do to study their behavior is to compute with which probability an algorithm in any particular state (e.g., with a particular population composition) will move any other state (e.g., another population composition) at the next generation. These probabilities are stored in a stochastic matrix M , which therefore describes with which probability the algorithm will exhibit all possible behaviors over one generation. By taking powers of the matrix M and multiplying it by the initial probability distribution over states determined by the initialization algorithm, it is then possible to study

the behavior of the algorithm over any number of generations. It is, for example, possible to compute the probability of solving a problem within n generations, the first hitting time, the average and best fitness after n generations, etc.

Markov models have been applied to GP (Poli and McPhee 2003; Mitanskiy and Rowe 2006), but so far they have not been developed as fully as the schema theory model.

4.3 Search Spaces

The models presented in the previous two sections treat the fitness function as a black box. That is, there is no representation of the fact that in GP, unlike in other evolutionary techniques, the fitness function involves the execution of computer programs on a variety of inputs. In other words, schema theories and Markov chains do not tell how fitness is distributed in the search space.

The theoretical characterization of the space of computer programs explored by GP from the point of view of fitness distributions aims at answering this very question (Langdon and Poli 2002). The main result in this area is the discovery that the distribution of functionality of non Turing-complete programs approaches a limit as program length increases. That is, although the number of programs of a particular length grows exponentially with length, beyond a certain threshold, the fraction of programs implementing any particular functionality is effectively constant. This happens in a variety of systems and can be proven mathematically for LISP expressions (without side effects) and machine code programs without loops (e.g., see Langdon and Poli 2002). Recently, Poli and Langdon (2006) started extending these results to Turing complete machine code programs. A mathematical analysis of the halting process based on a Markov chain model of program execution and halting was performed, which derived a scaling law for the *halting probability* for programs as their length is varied.

4.4 Bloat

Very often, the average size (number of nodes) of the programs in a GP population, after a certain number of generations in which it is largely static, starts growing at a rapid pace. Typically, the increase in program size is not accompanied by any corresponding increase in fitness. This phenomenon is known as *bloat*. Its origin has been the focus of intense research for over a decade. Bloat is not only interesting from a scientific point of view: it also has significant practical deleterious effects, slowing down the evaluation of individuals, and often consuming large computational resources. There are several theories of bloat.

For instance, the *replication accuracy theory* (McPhee and Miller 1995) states that the success of a GP individual depends on its ability to have offspring that are functionally similar to the parent: as a consequence, GP evolves toward (bloated) representations that increase replication accuracy.

The nodes in a GP tree can be categorized into two classes: *inactive code* (code that is not executed or has no effect) and *active code* (all code that is not inactive). The *removal bias theory* (Soule and Foster 1998b) observes that inactive code in a GP tree tends to be low in the tree, residing in smaller-than-average-size subtrees. Crossover events excising inactive subtrees produce offspring with the same fitness as their parents. On average the inserted subtree is bigger than the excised one, so such offspring are bigger than average while retaining the fitness of their parent, leading ultimately to growth in the average program size.

The *nature of program search spaces theory* (Langdon et al. 1999) relies on the result mentioned above that above a certain size, the distribution of fitnesses does not vary with size. Since there are more long programs, the number of long programs of a given fitness is greater than the number of short programs of the same fitness. Over time GP samples longer and longer programs simply because there are more of them.

In Poli and McPhee (2003), a *size evolution equation* for GP was developed, which provided an exact formalization of the dynamics of average program size. The original equation was derived from the exact schema theory for GP. This equation can be rewritten in terms of the expected change in average program size as

$$E[\Delta\mu(t)] = \sum_{\ell} \ell \times (p(\ell, t) - \Phi(\ell, t)) \quad (2)$$

where $\Phi(\ell, t)$ is the proportion of programs of size ℓ in the current generation and $p(\ell, t)$ is their selection probability. This equation does not directly explain bloat, but it constrains what can and cannot happen size-wise in GP populations. So, any explanation for bloat (including the theories above) has to agree with it.

The newest, and hopefully conclusive, explanation for bloat has been recently formalized in the so-called *crossover bias theory* (Dignum and Poli 2007; Poli et al. 2007), which is based on [Eq. 2](#). On average, each application of subtree crossover removes as much genetic material as it inserts; consequently crossover on its own does not produce growth or shrinkage. While the *mean* program size is unaffected, however, *higher moments* of the distribution are. In particular, crossover pushes the population toward a particular distribution of program sizes, known as a *Lagrange distribution of the second kind*, where small programs have a much higher frequency than longer ones. For example, crossover generates a very high proportion of single-node individuals. In virtually all problems of practical interest, however, very small programs have no chance of solving the problem. As a result, programs of above average size have a selective advantage over programs of below average size, and the mean program size increases. Because crossover will continue to create small programs, the increase in average size will continue generation by generation.

Numerous empirical techniques have been proposed to control bloat (Soule and Foster 1998; Langdon et al. 1999). These include the use of size and depth limits, the use of genetic operators with an inbuilt anti-bloat bias, the use of multi-objective selection techniques to jointly optimize fitness and minimize program size. Among these are the famous parsimony pressure method (Koza 1992b; Zhang and Mühlenbein 1995) and a recent improvement of if, the covariant parsimony pressure method (Poli and McPhee 2008). The reader is referred to Poli et al. (2008), Chap. 11 for a survey.

4.5 Is GP Guaranteed to Find Solutions to Problems?

While often Markov transition matrices describing evolutionary algorithms are very large and difficult to manage numerically, it is sometimes possible to establish certain properties of such matrices theoretically. An important property in relation to the ability of an algorithm to reach all possible solutions to a problem (given enough time) is the property of *ergodicity*. What it means in practice is that there exist some $k \in \mathbb{N}$ such that all elements of M^k are nonzero. In this case, $\lim_{k \rightarrow \infty} M^k$ is a particular matrix where all elements are nonzero. This means that irrespective of the starting state, the probability of reaching a solution to a problem

(e.g., the global optimum of a function, if one is using EAs for function optimization) is nonzero. As a result, given enough generations, the algorithm is guaranteed to find a solution.

In the context of EAs acting on traditional fixed-length representations, this has led to proving that EAs with nonzero mutation rates are global optimizers with guaranteed convergence (Rudolph 1994). While these results have been extended to more general search spaces (Rudolph 1996) and, as we discussed in [Sect. 4.2](#), Markov chain models of some forms of GP have been recently derived (e.g., see Poli et al. (2004)), the calculations involved in setting up and studying these models are of considerable mathematical complexity. Furthermore, extending the work to infinite search spaces (which is required to model the traditional form of GP where operators can create trees of potentially any size over a number of generations) presents even bigger challenges. So, it is fair to say that at the present time there is no formal mathematical proof that a GP system can always find solutions.

Despite these formal obstacles, it is actually surprisingly easy to find good reasons in support of the conjecture that, with minor modifications, the traditional form of GP is guaranteed to visit a solution to a problem, given enough generations. In fact, if one uses mutation and the mutation operator is such that, albeit with a very low probability, any point in the search space can be generated by mutating any other point, then it is clear that sooner or later the algorithm will visit a solution to the problem. It is then clear that in order to ensure that GP has a guaranteed asymptotic ability to solve all problems, all we need to do is to add to the mix of operators already present in a GP system some form of mutation that has a nonzero probability of generating any tree in the search space. One such mutation operator is the subtree mutation operator described in [Sect. 2.6](#).

4.6 Problem Difficulty in GP

What problems are solvable via GP? Is it possible to measure the difficulty of a problem for GP? In classical algorithms, we have a well-developed complexity theory that allows us to classify problems into complexity classes such that problems in the same class have roughly the same complexity, that is, they consume (asymptotically) the same amount of computational resources, usually time. Although, properly speaking, GP is a randomized heuristic and not an algorithm, it would be nice if we were able to somehow classify problems according to some measure of *difficulty*.

Difficulty studies have been pioneered in the related but simpler field of GAs, by Goldberg and coworkers (e.g., see Goldberg 1989 and Horn and Goldberg 1995). Their approach consisted in constructing functions that should *a priori* be easy or hard for GAs to solve. These approach has been followed by many others (e.g., Mitchell et al. 1992 and Forrest and Mitchell 1993) and have been at least partly successful in the sense that they have been the source of many ideas as to what makes a problem easy or difficult for GAs. One concept that underlies many measures of difficulty is the notion of *fitness landscapes*.

The metaphor of a fitness landscape (Stadler 2002), although not without faults, has been a fruitful one in several fields and is attractive due to its intuitiveness and simplicity. Probably, the simplest definition of fitness landscape in EAs is a plot where the points in the abscissas represent the different individual genotypes in a search space and the ordinates represent the fitness of each of these individuals (Langdon and Poli 2002). If genotypes can be visualized in two dimensions, the plot can be seen as a three-dimensional “map,” which may contain peaks

and valleys. The task of finding the best solution to the problem is equivalent to finding the highest peak (for maximization problems).

One early example of the application of the concept of fitness landscapes to GP is represented by the work of Kinnear (1994), where GP difficulty was related to the shape of the fitness landscape and analyzed through the use of the fitness *auto-correlation function*. Langdon and Poli (2002) took an experimental view of GP fitness landscapes in several works summarized in their book. After selecting important and typical classes of GP problems, they study fitness landscapes either exhaustively, whenever possible, or by randomly sampling the program space when enumeration becomes unfeasible. Their work highlights several important characteristics of GP spaces, such as density and size of solutions and their distribution. This is useful work. Even if the authors' goals were not establishing problem difficulty, it certainly has a bearing on it. More recently, Langdon (2003) has extended his studies for convergence rates in GP for simple machine models (which are amenable to quantitative analysis by Markov chain techniques) to convergence of program fitness landscapes for the same machine models using genetic operators and search strategies to traverse the space. This approach is rigorous because the models are simple enough to be mathematically treatable. The ideas are thus welcome, although their extension to standard GP might prove difficult.

The work of Nikolaev and Slavov (1998) represents the first effort to apply a difficulty measure called *fitness distance correlation* (FDC) (first studied by Jones (1995) for GAs) to GP, with the aim of determining which mutation operator, among a few that they propose, "sees" a smoother landscape on a given problem. Pros and cons of FDC as a general difficulty measure in GP were investigated in Vanneschi (2004), and Tomassini et al. (2005). Those contributions claimed the reliability of FDC and pointed out, as one of its major drawbacks, its lack of predictiveness of FDC (the genotype of the global optima must be known beforehand to calculate it).

In order to overcome this limitation, a new hardness measure, called *negative slope coefficient* (NSC) has been recently presented and its usefulness for GP investigated in Vanneschi et al. (2004, 2006). Results indicated that, although NSC may still need improving and is not without faults, it is a suitable hardness indicator for many well-known GP benchmarks, for some hand-tailored theoretical GP problems and also for some real-life applications (Vanneschi 2007).

Even though interesting, these results still fail to take into account many typical characteristics of GP. For instance, the NSC definition only includes (various kinds of) mutation, but it does not take crossover into account. A similarity/dissimilarity measure related to standard subtree crossover has been presented in Gustafson and Vanneschi (2005, 2008) and Vanneschi et al. (2006). This could be used in the future to build new measures of problem difficulty.

4.7 Does No-Free-Lunch Apply to GP?

Informally speaking, the no-free-lunch (NFL) theory originally proposed by Wolpert and Macready (1997) states that, when evaluated over all possible problems, all algorithms are equally good or bad irrespective of our evaluation criteria.

In the last decade there have been a variety of results which have refined and specialized NFL (see Whitley and Watson 2005). One such result states that if one selects a set of fitness functions that are *closed under permutation*, the expected performance of any search algorithm over that set of problems is constant, that is, it does not depend on the algorithm we choose (Schumacher et al. 2001). What does it mean for a set of functions to be

closed under permutation? A fitness function is an assignment of fitness values to the elements of the search space. A permutation of a fitness function is simply a rearrangement or reshuffling of the fitness values originally allocated to the objects in the search space. A set of problems/fitness functions is closed under permutation, if, for every function in the set, all possible shuffles of that function are also in the set.

The result is valid for any performance measure. Furthermore, Schumacher et al. (2001) showed that it is also the case that two arbitrary algorithms will have identical performance over a set of functions only if that set of functions is closed under permutation.

Among the many extension of NFL to a variety of domains, Woodward and Neil (2003) have made some progress in assessing the applicability of NFL to the search spaces explored by GP. In particular, they argued that there is a free lunch in a search space whenever there is a nonuniform many-to-one genotype–phenotype mapping, and that the mapping from syntax to functionality in GP is one such mapping. The reason why NFL would not normally be applicable to search in program spaces is that there are many more programs than functionalities and that not all functionalities are equally likely. So, interpreting syntax trees as genotypes and functionalities as phenotypes, the GP genotype–phenotype mapping is many-to-one and nonuniform, which invalidates NFL.

Beyond this interesting counterexample, to show that in general not all functionalities are equally likely in program search spaces, Woodward and Neil (2003) referred to the results on the limiting distribution of functionality reviewed above and to the universal distribution (Kirchherr et al. 1997) (informally this states that there are many more programs with a simple functionality than programs with a complex one). The latter result, however, applies to Turing complete languages, that is, to programs with memory and loops. So, Woodward and Neil proof applies only to Turing-complete GP, which is essentially a rarity.

In very recent work (Poli et al. 2009) it has been possible to extend these results to the case of standard GP problems and systems and to ascertain that a many-to-one genotype–phenotype mapping is not the only condition under which NFL breaks down when searching program spaces. The theory is particularly easy to understand because it is based on geometry. It is briefly reviewed in the rest of this section.

As seen above, the fitness of a program p in GP is often the result of evaluating the behavior of p in a number of fitness cases and adding up the results (Poli et al. 2008). That is

$$f(p) = \sum_{i=1}^n g(p(x_i), t(x_i)) \quad (3)$$

where f is the fitness function, $\{x_i\}$ is a set of fitness cases of cardinality n , and g is a function that evaluates the degree of similarity of its arguments. Almost invariably, the function g respects the axioms of a metric, the most common form of g being $g(a, b) = |a - b|^k$ for $k=1$ or $k=2$.

One can consider a finite space of programs $\Omega = \{p_i\}_{i=1}^r$, such as, for example, the space of all possible programs with at most a certain size or depth. A fitness function f over Ω can be represented as a vector $\mathbf{f} = (f_1, \dots, f_r)$ where $f_i = f(p_i)$. Using this vector representation for fitness functions, we can write \bullet Eq. 3 as

$$\mathbf{f} = \left(\sum_{i=1}^n g(p_1(x_i), t(x_i)), \dots, \sum_{i=1}^n g(p_r(x_i), t(x_i)) \right) \quad (4)$$

As we can see from \bullet Eq. 4, if n and the set of fitness cases $\{x_i\}$ are fixed a priori, then a fitness function is fully determined by the value of the vector of target behaviors, $\mathbf{t} = (t_1, t_2, \dots, t_n)$,

where $t_i = t(x_i)$ is fixed. If we focus on the most frequent type of program induction application in GP, symbolic regression, we typically have that the components t_i are scalars representing the desired output for each fitness case.

The function g is a distance (on \mathbb{R}). Because the sum of distances is also a distance, we can define the following distance function:

$$d(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^n g(\mathbf{u}_i, \mathbf{v}_i) \quad (5)$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. With this in hand, we can see that the fitness associated to a program p can be interpreted as the distance between the vector \mathbf{t} and the vector $\mathbf{p} = (p(x_1), p(x_2), \dots, p(x_n))$. That is

$$f(\mathbf{p}) = d(\mathbf{p}, \mathbf{t}) \quad (6)$$

Note that whenever we represent programs using their behavior vector we are essentially focusing on the phenotype-to-fitness mapping, thereby complementing the analysis of Woodward and Neil (2003) summarized above.

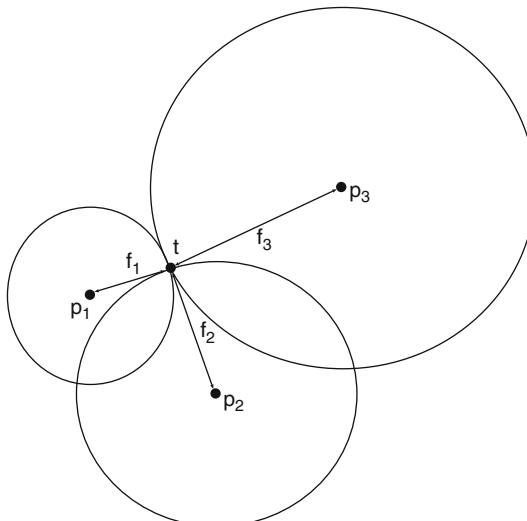
Using distances, [Fig. 4](#) can be rewritten more concisely as

$$\mathbf{f} = (d(\mathbf{p}_1, \mathbf{t}), d(\mathbf{p}_2, \mathbf{t}), \dots, d(\mathbf{p}_r, \mathbf{t})) \quad (7)$$

Note that if we know the fitness f of a program p , we know that the target behavior \mathbf{t} that generated that fitness must be on the surface of a sphere centered on \mathbf{p} (the vector representation of p) and of radius f . So, for every valid symbolic regression fitness function, the target behavior is at the intersection of the spheres centered on the behavior of each program in the search space (see [Fig. 4](#)).

Fig. 4

If g measures the squared difference between two numbers, a valid fitness function requires the spheres centered on each program behavior and with radius given by their corresponding fitness to meet in one point: the target vector \mathbf{t} .



With this geometric representation of symbolic regression problems in hand, it is not difficult to find out under which conditions NFL does or does not hold for GP. In particular, Poli et al. (2009) showed that:

Theorem 1 (NFL for GP). Let $\mathcal{F} = \{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m\}$ be a set of fitness functions of the form in [Eq. 7](#) and let $\mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_m\}$ be the set of target vectors associated to the functions in \mathcal{F} , with \mathbf{t}_i being the vector generating \mathbf{f}_i for all i . The set \mathcal{F} is closed under permutation (and NFL applies to it) if and only if for all target vectors $\mathbf{t} \in \mathcal{T}$ and for all permutations σ of $(1, 2, \dots, r)$ there exists a target vector $\tilde{\mathbf{t}} \in \mathcal{T}$ such that:

$$\sum_{i=1}^n g(p_{\sigma(j)}(x_i), t_i) = \sum_{i=1}^n g(p_j(x_i), \tilde{t}_i) \quad (8)$$

or, equivalently,

$$d(\mathbf{p}_{\sigma(j)}, \mathbf{t}) = d(\mathbf{p}_j, \tilde{\mathbf{t}}) \quad (9)$$

for all $j = 1, 2, \dots, r$.

⦿ [Equation 9](#) is a mathematical statement of the geometric requirements for $\tilde{\mathbf{t}}$ to exist. Namely, the target vector $\mathbf{t} \in \mathcal{T}$ associated to a function $\mathbf{f} \in \mathcal{F}$ must be at the intersection of the spheres centered on programs $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_r$ and having radii f_1, f_2, \dots, f_r , respectively. Permuting the elements of \mathbf{f} via a permutation σ to obtain a new fitness function corresponds to shuffling the radii of the spheres centered on $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_r$. Note these centers remain fixed since they represent the behavior of the programs in the search space. After the shuffling, some spheres may have had their radius decreased, some increased, and some unchanged. If any radius has changed, then \mathbf{t} can no longer be the intersection of the spheres. However, we must be able to find a new intersection $\tilde{\mathbf{t}} \in \mathcal{T}$ or else the new fitness function we have generated is not a symbolic regression fitness function and therefore cannot be a member of \mathcal{F} , which would imply that the set is not closed under permutation.

We should also note that Theorem 1 focuses on the set of program behaviors. So, it tells us something about the nature of the phenotype-to-fitness function. It really states under which conditions there can be an NFL for a searcher exploring program behaviors with no resampling. If a search algorithm instead explores the space of syntactic structures representing programs, in the presence of symmetries (such as those highlighted by Woodward and Neil), then the searcher will produce resampling of program behaviors even if it never resampled the same syntactic structure. So, in the presence of a set of behaviors for which NFL holds, this would unavoidably give the “syntactic” searcher lower average performance than an algorithm that never resampled behaviors.

This theorem is useful since, by designing fitness functions that break its assumptions, we can find conditions where there is a free lunch for GP. It turns out that such conditions are really easy to satisfy. In particular, it is easy to see that if two programs have identical behavior, they must have identical fitness or there cannot be any intersection between the spheres centered around them. Formally:

Theorem 2 Consider a search space which includes at least two programs p_1 and p_2 such that $p_1(x) = p_2(x)$ for all x in the training set. Let a set of symbolic regression problems \mathcal{F} contain a fitness function f induced by a target vector \mathbf{t} such that there exists a third program p_3 in the

search space with fitness $f(p_3) \neq f(p_1) = f(p_2)$. Then \mathcal{F} cannot be closed under permutation and NFL does not hold.

Note that this result is valid for search in the space of syntactic structures representing programs. As one can see, a many-to-one mapping is not required for a free lunch to be available.

Continuing with our geometric investigation of NFL, it is clear that a necessary condition for the existence of a target vector \mathbf{t} , which induces a particular fitness function, is that the triangular inequality be verified. More precisely:

Lemma 1 *If a target vector \mathbf{t} induces a symbolic regression fitness function \mathbf{f} , then for every pair of program behaviors \mathbf{p}_1 and \mathbf{p}_2 the distance $d(\mathbf{p}_1, \mathbf{p}_2)$ between \mathbf{p}_1 and \mathbf{p}_2 (based on the same metric used to measure fitnesses) must not be greater than $f_1 + f_2$.*

From this result, we can see that another common situation where there is incompatibility between an assignment of fitness to programs and the fitness representing a symbolic regression problem is the case in which two programs have different behaviors (and so are represented by two distinct points in the \mathbb{R}^n), but the sum of their fitnesses is smaller than their distance. This leads to the following result:

Theorem 3 *Given a set of symbolic regression functions \mathcal{F} , if there exists any function $\mathbf{f} \in \mathcal{F}$ and any four program behaviors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ in a search space such that $d(\mathbf{p}_1, \mathbf{p}_2) > f_3 + f_4$ then the set is not closed under permutation and NFL does not apply.*

So, it is extremely easy to find realistic situations in which a set of symbolic regression problems is provably not closed under permutation. This implies that, in general, we can expect that there is a free lunch for techniques that sample the space of computer programs for the purpose of fitting data sets. This is particularly important since it implies that, for GP, unlike other areas of natural computing, it is worthwhile to try to come up with new and more powerful algorithms.

5 Open Issues

Despite the successful application of GP to a number of challenging real-world problem domains and the recent theoretical advancements in explaining the behavior and dynamics of GP, there remains a number of significant open issues. These must be addressed for GP to become an even more effective and reliable problem-solving method.

The following are some of the important open issues:

- Despite some preliminary studies trying to remove this limit, at present, GP is fundamentally a “static” process, in the sense that many of its characteristics are fixed once and for all before executing it. For instance, the population size, the language used to code individuals (\mathcal{F} and \mathcal{T} sets), the genetic operators used to explore the search space and the fitness function(s) used are all static.
- Directly related to the previous point, until now GP has been mainly used for static optimization problems. No solid explorations, neither theoretical nor applicative, exist of dynamic optimization problems, where for instance the target function (and thus the fitness value of the candidate solutions) change over time.

- While a large body of literature and well-established results exist concerning the issue of generalization for many non-evolutionary machine-learning strategies, this issue in GP has not received the attention it deserves. Only recently, a few papers dealing with the problem of generalization have appeared. Nonetheless, generalization is one of the most important performance evaluation criteria for artificial learning systems and its foundations for GP still have to be deeply understood and applied.
- Although some constructs for handling modularity in GP have been defined (some of them, like ADFs, have been discussed in [Sect. 2.7](#)) and widely used, still some questions remain unanswered. Can these constructs help in solving substantially more complex problems? Can we achieve modularity, hierarchy, and reuse in a more controlled and principled manner? Can we provide insights and metrics on how to achieve this?

In the rest of this section, some of these grand challenges and open issues will be expanded upon. It is hoped that this chapter will help focus future research in order to deepen our understanding of the method to allow the development of more powerful problem-solving algorithms.

5.1 Dynamic and/or Self-Adapting GP Frameworks

Although the difficulty measures discussed in [Sect. 4.6](#) are useful to statically calculate the hardness of a given problem for GP starting from its high-level specifications, they have not been applied to dynamically improve the search as yet. Indeed, a dynamic modification “on the fly” (i.e., during the evolutionary process) of the fitness function, the population size, the genetic operators, or representation used, driven by one or more hardness measures represents one of the most challenging open issues in GP, as in many other fields of evolutionary computation. One of the major difficulties in the realization of such a dynamic and self-adapting GP environment is probably represented by the fact that both FDC and NSC have always been calculated, until now, using a large sample of the search space and not the evolving population that is typically much smaller. Furthermore, the lack of diversity in populations after some generations may seriously compromise the reliability of these indicators. Nevertheless, this issue remains a challenge and a first attempt in this direction has recently been presented in Wedge and Kell ([2008](#)).

Models using dynamically sized populations, although not based on the values of difficulty indicators, have been presented, among others, in Rochat et al. ([2005](#)), Tomassini et al. ([2004](#)), and Vanneschi ([2004](#)), where the authors show that fixed-size populations may cause some difficulties to GP. They try to overcome these difficulties by removing or adding new individuals to the population in a dynamic way during evolution. In particular, individuals are removed as long as the best or average population’s fitness keeps improving, while new individuals (either randomly generated ones or mutations of the best ones in the population) are added when fitness stagnates. The authors show a set of experimental results that confirm the benefits of the presented methods, at least for the studied problems. These results suggest that fixed-size population unnaturally limits the GP behavior and that, according to what happens in nature, the evolution has a benefit when populations are left free to shrink or increase in size dynamically, according to a set of events. Some of the most relevant and macroscopic of these events are so called *plagues* that usually cause a violent decrease in natural population sizes. Plagues have been successfully applied to GP in Fernandez et al. ([2003a,b](#)) and Fernandez and Martin ([2004](#)). These contributions go in the

same direction as the previous ones: fixed-size populations limit and damage the behavior of GP while the use of dynamically sized populations is often beneficial. Furthermore, the models presented in the above quoted references have been successfully used by Silva (2008) to study and control bloat.

5.2 GP for Dynamic Optimization

Dynamic environments abound and offer particular challenges for all optimization and problem-solving methods. A well-known strategy for survival in dynamic environments is to adopt a population-based approach (Dempsey 2007). This allows a diversity of potential solutions to be maintained, which increases the likelihood that a sufficiently good solution exists at any point in time to ensure the survival of the population in the long term. Dynamic environments can exhibit changes in many different ways including the frequency and degree/size of change. The types of changes might range from relatively small smooth transitions to substantial perturbations in all aspects of the domain (Dempsey 2007).

Given the power of the biological process of evolution to adapt to ever, changing environments, it is surprising that the number of studies applying and explicitly studying their artificial counterpart of GP in dynamic environments have been minimal (Dempsey 2007). Despite the existence of a recent special issue in the GP journal on dynamic environments (Yang et al. 2006), none of the four articles actually dealt with GP directly (e.g., Wang and Wineberg (2006)). While some applications in dynamic environments have been undertaken in the past two years (e.g., Wagner et al. 2007, Hansen et al. 2007, Jakobović and Budin 2006, and Kibria and Li 2006), there has been little analysis of the behavior of GP in these environments. The two main examples have examined bloat (Langdon and Poli 1998) and constant generation (Dempsey 2007).

This seems to hint that we are missing the boat by focusing on static problems where there is a single fixed target. Recent experiments in evolutionary biology simulations suggest that EC/evolution could work efficiently “because” of dynamic environments as opposed to “despite” them (Kashtan et al. 2007).

5.3 Generalization in GP

Generalization is one of the most important performance-evaluation criteria for artificial learning systems. Many results exist concerning the issue of generalization in machine learning, like, for instance, support vector machines (see for instance Smola and Scholkopf 1999). However, this issue in GP has not received the attention it deserves and only recently a few papers dealing with the problem of generalization have appeared (Eiben and Jelasity 2002; Kushchu 2002). A detailed survey of the main contributions on generalization in GP has been done by Kushchu (2002). Another important contribution to the field of generalization in GP is due to the work of Banzhaf and coworkers. In particular, in Francone et al. (1996) they introduce a GP system called *Compiling GP* and they compared its generalization ability with that of other machine-learning paradigms. Furthermore, in Banzhaf et al. (1996) they show the positive effect of an extensive use of the mutation operator on generalization in GP using sparse data sets. In 2006, Da Costa and Landry (2006) have proposed a new GP model called *Relaxed GP*, showing its generalization ability. In 2006, Gagné et al. (2006) have investigated two methods to improve generalization in GP-based learning: (1) the selection

of the best-of-run individuals using a three-data-sets methodology, and (2) the application of parsimony pressure to reduce the complexity of the solutions.

A common design principle among ML researchers is the so called *minimum description length principle* (see for instance Rissanen 1978), which states that the best model is the one that minimizes the amount of information needed to encode it. In this perspective, preference for simpler solutions and over-fitting avoidance seem to be closely related. It is more likely that a complex solution incorporates specific information from the training set, thus over-fitting it, compared to a simpler solution. But, as mentioned in Domingos (1999), this argumentation should be taken with care as too much emphasis on minimizing complexity can prevent the discovery of more complex yet more accurate solutions. Finally, in Vanneschi et al. (2007), authors hint that GP generalization ability (or more precisely, *extrapolation* ability) can be improved by performing a multi-optimization on the training set. A related idea has been used in Gagné et al. (2006) in the domain of binary classification, where a two-objective sort is performed in order to extract a set of nondominated individuals. However, different objectives in the realm of regression problems are used in Vanneschi et al. (2007).

Despite the above mentioned studies, the issue of generalization remains an open one in GP, since no theoretical approach has ever been presented to tackle this problem until now.

5.4 Modularity in GP

The use of modules may be very important to improve GP expressiveness, code reusability, and performance. The best-known of these methods is Koza's ADFs, which is introduced in ➤ Sect. 2.7. The first step toward a theoretically motivated study of ADFs is probably represented by Rosca (1995), where an algorithm for the automatic discovery of building blocks in GP called *Adaptive Representation Through Learning* was presented. In the same year, Spector (1995) introduced techniques to evolve collections of automatically defined macros and showed how they can be used to produce new control structures during the evolution of programs and Seront (1995) extended the concept of code reuse in GP to the concept of generalization, showing how programs (or “concepts,” using Seront’s terminology) synthesized by GP to solve a problem can be reused to solve other ones. Altenberg offers a critical analysis of modularity in evolution in Altenberg (2009), stating that the evolutionary advantages that have been attributed to modularity do not derive from modularity *per se*. Rather, they require that there be an “alignment” between the spaces of phenotypic variation, and the selection gradients that are available to the organism. Modularity in the genotype–phenotype map may make such an alignment more readily attained, but it is not sufficient; the appropriate phenotype–fitness map in conjunction with the genotype–phenotype map is also necessary for evolvability. This contribution is interesting and stimulating, but its applicability to GP remains an open question.

In Jonyer and Himes (2006) the concept of ADFs is extended by using graph-based data mining to identify common aspects of highly fit individuals and modularizing them by creating functions out of the subprograms identified. In Hemberg et al. (2007) the authors state that the ability of GP to scale to problems of increasing difficulty operates on the premise that it is possible to capture regularities that exist in a problem environment by decomposition of the problem into a hierarchy of modules.

Although the use of modularity in GP has helped solve some problems and provide new data/control abstractions, for instance in the form of ADFs, still some issues remain open.

For instance: Has it solved really substantially more complex problems and does it hold the promise to do so in the near future? Are ADFs necessary/sufficient as a formalism to help solve grand-challenge problems, that is to provide scalability? And even more ambitiously: Can we achieve modularity, hierarchy, and reuse in a more controlled and principled manner? Can we provide insights and metrics on how to achieve this? Is it possible to routinely evolve algorithms? How to achieve iteration and/or recursion in a controlled and principled manner? What is the best way to handle the generation and maintenance of constants?

A first attempt to answer this last question is represented by Vanneschi et al. (2006) and Cagnoni et al. (2005) where the authors present a new coevolutionary environment where a GA dynamically evolves constants for GP. Although interesting, this contribution, as many others aimed at improving GP modularity, is mainly empirical, that is, it shows the benefits of the presented method(s) by means of a set of experimental results on some test functions. In order to be able to get a deeper insight of the real usefulness of modularity and code-reuse in GP, more theoretical studies are needed.

6 Conclusions

GP is a systematic method for getting computers to automatically solve a problem starting from its high-level specifications. It extends the model of genetic algorithms to the space of computer programs. In its almost 30 years of existence, it has been successfully applied to a number of challenging real-world problem domains. Furthermore, solid theory exists explaining its behavior and dynamics.

After a general introduction to the mechanics of GP, this chapter has discussed some examples of real-world applications of GP. In general, GP is particularly suitable for applications of curve fitting, data modeling, or symbolic regression, typically characterized by a large amount of data and where little is known about the relationships among the relevant features. Over the years, dozens of GP results have passed the human-competitiveness test, defined by Koza by establishing eight precise criteria, including patentability, beating humans in regulated competitions, producing publishable results, etc. Fields where GP have been successfully applied in the last few years include: image and signal processing; financial trading, time series, and economic modeling; industrial process control; medicine, biology and bioinformatics; computational chemistry; entertainment and computer games; arts; and compression.

Subsequently, we presented some of the most relevant GP theoretical results, including schema theories and Markov chains-based models. We discussed recent studies of the distribution of fitness in GP search spaces and of the distribution of programs' length. We presented various theories explaining the phenomenon of bloat, including the recently introduced crossover bias theory, which is based on a precise equation of the programs' size evolution, and that, hopefully, will give a conclusive explanation of this phenomenon. We gave arguments in support of the conjecture that GP is a global optimizer, that is, it is guaranteed to find a globally optimal solution, given enough time. Furthermore, we presented the most important results obtained in the field of GP problem difficulty, including a discussion of fitness distance correlation and negative slope coefficient, two of the most effective measures of problem difficulty for GP, and the recently introduced subtree crossover similarity/dissimilarity measure, that should help in defining new and hopefully more powerful

difficulty measures. Finally, we summarized results on the existence of a “free lunch” for GP under some weak (and, in practice, often satisfied) conditions.

In the last part of this chapter, hot topics and open issues of GP have been discussed, including the development of a self-adapting framework to dynamically modify the GP configuration (population size, representation, genetic operators, etc.) during the evolution; the study of the usefulness of GP for dynamic optimization problems; the study and better understanding of the concept of generalization and its implications for GP; and the definition of more powerful and self-adaptive methods of using modularity in GP.

Although our chapter could not exhaustively cover the immensity of the field, it is hoped that this chapter will attract newcomers to the field of GP as well as help focus future research, leading to a wider application and deeper understanding of GP as a problem-solving strategy.

References

- Altenberg L (2009) Modularity in evolution: Some low-level questions. In: Rasskin-Gutman D, Callebaut W (eds), *Modularity: Understanding the Development and Evolution of Complex Natural Systems*. MIT Press, Cambridge, MA
- Alves da Silva AP, Abrao PJ (2002) Applications of evolutionary computation in electric power systems. In: Fogel DB et al. (eds), *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pp. 1057–1062. IEEE Press
- Archetti F, Messina E, Lanzeni S, Vanneschi L (2007) Genetic programming for computational pharmacokinetics in drug discovery and development. *Genet Programming Evol Mach* 8(4):17–26
- Azaria Y, Sipper M (2005) GP-gammon: Genetically programming backgammon players. *Genet Programming Evol Mach* 6(3):283–300, Sept. Published online: 12 August 2005
- Banzhaf W, Francone FD, Nordin P (1996) The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In: Ebeling W et al., (ed) *4th International Conference on Parallel Problem Solving from Nature (PPSN96)*, Springer, Berlin, pp. 300–309
- Banzhaf W, Nordin P, Keller RE, Francone FD (1998) *Genetic Programming, An Introduction*. Morgan Kaufmann, San Francisco, CA
- Barrett SJ, Langdon WB (2006) Advances in the application of machine learning techniques in drug discovery, design and development. In: Tiwari A et al. (eds), *Applications of Soft Computing: Recent Trends, Advances in Soft Computing, On the World Wide Web*, 19 Sept.–7 Oct. 2005. Springer, Berlin, 99–110
- Bojarczuk CC, Lopes HS, Freitas AA (July–Aug. 2008) Genetic programming for knowledge discovery in chest-pain diagnosis. *IEEE Eng Med Biol Mag* 19(4):38–44
- Brameier M, Banzhaf W (2001) A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Trans Evol Comput* 5(1):17–26
- Cagnoni S, Rivero D, Vanneschi L (2005) A purely-evolutionary memetic algorithm as a first step towards symbiotic coevolution. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, Edinburgh, Scotland, 2005. IEEE Press, Piscataway, NJ, pp. 1156–1163
- Castillo F, Kordon A, Smits G (2006) Robust pareto front genetic programming parameter selection based on design of experiments and industrial data. In: Riolo RL, et al. (ed) *Genetic Programming Theory and Practice IV*, vol 5 of *Genetic and Evolutionary Computation*, chapter 2. Springer, Ann Arbor, 11–13 May
- Chen S-H, Liao C-C (2005) Agent-based computational modeling of the stock price-volume relation. *Inf Sci* 170(1):75–100, 18 Feb
- Chen S-H, Wang H-S, Zhang B-T (1999) Forecasting high-frequency financial time series with evolutionary neural trees: The case of Hang Seng stock index. In: Arabnia HR, (ed), *Proceedings of the International Conference on Artificial Intelligence, IC-AI '99*, vol 2, Las Vegas, NV, 28 June–1 July. CSREA Press pp. 437–443
- Da Costa LE, Landry JA (2006) Relaxed genetic programming. In: Keijzer M et al., editor, *GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, vol 1, Seattle, Washington, DC, 8–12 July. ACM Press pp. 937–938
- Dassau E, Grosman B, Lewin DR (2006) Modeling and temperature control of rapid thermal processing. *Comput Chem Eng* 30(4):686–697, 15 Feb
- Davis TE, Principe JC (1993) A Markov chain framework for the simple genetic algorithm. *Evol Comput* 1(3): 269–288

- De Jong KA (1988) Learning with genetic algorithms: An overview. *Mach Learn* 3:121–138
- Dempsey I (2007) Grammatical evolution in dynamic environments. Ph.D. thesis, University College Dublin, Ireland
- Dignum S, Poli R (2007) Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In: Thierens, D et al. (eds), *GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, vol 2 London, 7–11 July 2007. ACM Press, pp. 1588–1595
- Domingos P (1999) The role of Occam's razor in knowledge discovery. *Data Mining Knowl Discov* 3(4):409–425
- Eiben AE, Jelasity M (2002) A critical note on experimental research methodology in EC. In: *Congress on Evolutionary Computation (CEC'02)*, Honolulu, HI, 2002. IEEE Press, Piscataway, NJ, pp. 582–587
- Esparcia-Alcazar AI, Sharman KC (Sept. 1996) Genetic programming techniques that evolve recurrent neural networks architectures for signal processing. In: *IEEE Workshop on Neural Networks for Signal Processing*, Seiko, Kyoto, Japan
- Fernandez F, Martin A (2004) Saving effort in parallel GP by means of plagues. In: Keijzer M, et al. (eds), *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, vol 3003 of *LNCS*, Coimbra, Portugal, 5–7 Apr. Springer-Verlag, pp. 269–278
- Fernandez F, Tomassini M, Vanneschi L (2003) Saving computational effort in genetic programming by means of plagues. In: Sarker, R et al. (eds), *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, Camberra, 8–12 Dec. 2003. IEEE Press, pp. 2042–2049
- Fernandez F, Vanneschi L, Tomassini M (2003) The effect of plagues in genetic programming: A study of variable-size populations. In: Ryan, C et al. (ed) *Genetic Programming, Proceedings of EuroGP'2003*, vol 2610 of *LNCS*, Essex, 14–16 Apr. Springer-Verlag, pp. 317–326
- Forrest S, Mitchell M (1993) What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. *Mach Learn* 13:285–319
- Francone FD, Nordin P, Banzhaf W (1996) Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In: Koza JR et al. (ed), *Genetic Programming: Proceedings of the First Annual Conference*, MIT Press, Cambridge, pp. 72–80
- Fukunaga A, Stechert A (1998) Evolving nonlinear predictive models for lossless image compression with genetic programming. In: Koza, JR et al. (eds), *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, WI, 22–25 July, Morgan Kaufmann pp. 95–102
- Gagné C, Schoenauer M, Parizeau M, Tomassini M (2006) Genetic programming, validation sets, and parsimony pressure. In: Collet P et al. (ed), *Genetic Programming, 9th European Conference, EuroGP2006, Lecture Notes in Computer Science, LNCS 3905*, pp. 109–120. Springer, Berlin, Heidelberg, New York
- Giacobini M, Tomassini M, Vanneschi L (2002) Limiting the number of fitness cases in genetic programming using statistics. In: Merelo JJ, et al. (eds), *Parallel Problem Solving from Nature – PPSN VII*, vol 2439 of *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, pp. 371–380
- Goldberg DE (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA
- Gustafson S, Vanneschi L (2005) Operator-based distance for genetic programming: Subtree crossover distance. In: Keijzer, M., et al. (ed), *Genetic Programming, 8th European Conference, EuroGP2005, Lecture Notes in Computer Science, LNCS 3447*, pp. 178–189. Springer, Berlin, Heidelberg, New York
- Gustafson S, Vanneschi L (2008) Operator-based tree distance in genetic programming. *IEEE Trans Evol Comput* 12:4
- Hansen JV, Lowry PB, Meservy RD, McDonald DM (Aug. 2007) Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decis Support Syst* 43(4): 1362–1374, Special Issue Clusters
- Hasan S, Daugelat S, Rao PSS, Schreiber M (June 2006) Prioritizing genomic drug targets in pathogens: Application to mycobacterium tuberculosis. *PLoS Comput Biol* 2(6):e61
- Hauptman A, Sipper M (2007) Evolution of an efficient search algorithm for the mate-in-N problem in chess. In: Ebner, M et al. (eds), *Proceedings of the 10th European Conference on Genetic Programming*, vol 4445 of *Lecture Notes in Computer Science*, Valencia, Spain, 11–13 Apr. Springer pp. 78–89
- Hemberg E, Gilligan C, O'Neill M, Brabazon A (2007) A grammatical genetic programming approach to modularity in genetic algorithms. In: Ebner, M et al. (eds), *Proceedings of the 10th European Conference on Genetic Programming*, vol 4445 of *Lecture Notes in Computer Science*, Valencia, Spain, 11–13 Apr. Springer pp. 1–11
- Holland JH (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI
- Horn J, Goldberg DE (1995) Genetic algorithm difficulty and the modality of the fitness landscapes. In: Whitley D, Vose M (eds), *Foundations of Genetic Algorithms*, vol. 3, Morgan Kaufmann, pp. 243–269
- Howard D, Roberts SC (2004) Incident detection on highways. In: O'Reilly, U-M et al., (eds), *Genetic*

- Programming Theory and Practice II*, chapter 16, Springer, Ann Arbor, 13–15 May pp. 263–282
- Jacob C (May–June 2000) The art of genetic programming. *IEEE Intell Syst* 15(3):83–84, May–June
- Jacob C (2001) *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, San Francisco, CA
- Jakobović D, Budin L (2006) Dynamic scheduling with genetic programming. In: Collet, P et al. (eds), *Proceedings of the 9th European Conference on Genetic Programming*, vol 3905 of Lecture Notes in Computer Science, Budapest, Hungary, 10–12 Apr. Springer pp. 73–84
- Jin N, Tsang E (2006) Co-adaptive strategies for sequential bargaining problems with discount factors and outside options. In: *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, Vancouver, 6–21 July. IEEE Press, pp. 7913–7920
- Jones T (1995) Evolutionary algorithms, fitness landscapes and search. Ph.D. thesis, University of New Mexico, Albuquerque
- Jonyer I, Himes A (2006) Improving modularity in genetic programming using graph-based data mining. In: Sutcliffe GCJ, Goebel RG (eds), *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pp. 556–561, Melbourne Beach, FL, May 11–13 2006. American Association for Artificial Intelligence
- Jordan E, den Doeler J, Smits G (2006) Novel approach to develop structure-property relationships using genetic programming. In: Runarsson TP, et al. (eds), *Parallel Problem Solving from Nature – PPSN IX*, vol 4193 of LNCS, Reykjavik, Iceland, 9–13 Sept. Springer-Verlag pp. 322–331
- Kashtan N, Noor E, Alon U (2007) Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716, August 21
- Kattan A, Poli R (2008) Evolutionary lossless compression with GP-ZIP. In *Proceedings of the IEEE World Congress on Computational Intelligence*, Hong Kong, 1–6 June. IEEE
- Keijzer M (Sept. 2004) Scaled symbolic regression. *Genetic Programming and Evolvable Machines*, 5(3):259–269
- Kibria RH, Li Y (2006) Optimizing the initialization of dynamic decision heuristics in DPLL SAT solvers using genetic programming. In: Collet P, et al. (eds), *Proceedings of the 9th European Conference on Genetic Programming*, vol. 3905 of Lecture Notes in Computer Science, Budapest, Hungary, 10–12 Apr. Springer pp. 331–340
- Kinnear KE Jr (1994) Fitness landscapes and difficulty in genetic programming. In: *Proceedings of the First IEEE Congress on Evolutionary Computation*, IEEE Press, Piscataway, NY, pp. 142–147
- Kirchherr W, Li M, Vitanyi P (1997) The miraculous universal distribution. *Math Intell* 19:7–15
- Klappenecker A, May FU (1995) Evolving better wavelet compression schemes. In: Laine, AF et al. (ed), *Wavelet Applications in Signal and Image Processing III*, vol 2569, San Diego, CA 9–14 July. SPIE
- Kordon A (Sept. 2006) Evolutionary computation at Dow Chemical. *SIGEVOlution*, 1(3):4–9
- Koza J, Poli R (2003) A genetic programming tutorial. In: Burke E (ed) *Introductory Tutorials in Optimization, Search and Decision Support*, Chapter 8. <http://www.genetic-programming.com/jkpdf/burke2003tutorial.pdf>
- Koza JR (1992a) A genetic approach to the truck backer-upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, vol IV, IEEE Press, pp. 310–318
- Koza JR (1992b) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA
- Koza JR (1994) *Genetic Programming II*. The MIT Press, Cambridge, MA
- Koza JR, Bennett FH III, Stiffelman O (1999) Genetic programming as a Darwinian invention machine. In: Poli R, et al. (eds) *Genetic Programming, Proceedings of EuroGP'99*, vol 1598 of LNCS, Goteborg, Sweden, 26–27 May. Springer-Verlag pp. 93–108
- Koza JR, Bennett FH III, Andre D, Keane MA (1999) *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, San Francisco, CA
- Kushchuk I (2002) An evaluation of evolutionary generalization in genetic programming. *Artif Intell Rev* 18(1):3–14
- Langdon WB (2003) Convergence of program fitness landscapes. In: Cantú-Paz, E., et al. (ed) *Genetic and Evolutionary Computation – GECCO-2003*, vol 2724 of LNCS, Springer-Verlag, Berlin, pp. 1702–1714
- Langdon WB, Buxton BF (Sept. 2004) Genetic programming for mining DNA chip data from cancer patients. *Genet Programming Evol Mach*, 5(3): 251–257
- Langdon WB, Poli R (1998) Genetic programming bloat with dynamic fitness. In: Banzhaf W, et al. (eds), *Proceedings of the First European Workshop on Genetic Programming*, vol 1391 of LNCS, Paris, 14–15 Apr. Springer-Verlag, pp. 96–112
- Langdon WB, Poli R (2002) *Foundations of Genetic Programming*. Springer-Verlag
- Langdon WB, Poli R (2005) Evolutionary solo pong players. In: Corne, D et al. (eds), *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol 3, Edinburgh, U.K., 2–5 Sept. IEEE Press pp. 2621–2628

- Langdon WB, Soule T, Poli R, Foster JA (June 1999) The evolution of size and shape. In Spector, L et al. (eds), *Advances in Genetic Programming 3*, chapter 8, pp. 163–190. MIT Press, Cambridge, MA
- Lew TL, Spencer AB, Scarpa F, Worden K, Rutherford A, Hemez F (Nov. 2006) Identification of response surface models using genetic programming. *Mech Syst Signal Process* 20(8):1819–1831
- Lewin DR, Lachman-Shalem S, Grosman B (July 2006) The role of process system engineering (PSE) in integrated circuit (IC) manufacturing. *Control Eng Pract* 15(7):793–802 Special Issue on Award Winning Applications, 2005 IFAC World Congress
- Louchet J (June 2001) Using an individual evolution strategy for stereovision. *Genet Programming Evol Mach* 2(2):101–109
- Lutton E, Levy-Vehel J, Cretin G, Glevarec P, Roll C (1995) Mixed IFS: Resolution of the inverse problem using genetic programming. Research Report No 2631, INRIA
- Machado P, Romero J (eds). (2008) *The Art of Artificial Evolution*. Springer
- McPhee NF, Miller JD (1995) Accurate replication in genetic programming. In: Eshelman L (ed), *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, Pittsburgh, PA 15–19 July Morgan Kaufmann pp. 303–309
- Miller J (2001) What bloat? Cartesian genetic programming on Boolean problems. In: Goodman ED (ed), *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pp. 295–302, San Francisco, CA 9–11 July
- Mitavskiy B, Rowe J (2006) Some results about the Markov chains associated to GPs and to general EAs. *Theor Comput Sci* 361(1):72–110 28 Aug
- Mitchell M, Forrest S, Holland J (1992) The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Varela F, Bourgine P (eds), *Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life*, The MIT Press, pp. 245–254
- Nikolaev NI, Slavov V (1998) Concepts of inductive genetic programming. In: Banzhaf, W., et al. (ed), *Genetic Programming, Proceedings of EuroGP'1998*, vol 1391 of LNCS, Springer-Verlag, pp. 49–59
- Nordin P, Banzhaf W (1996) Programmatic compression of images and sound. In: Koza JR, et al. (eds), *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, CA 28–31 July. MIT Press pp. 345–350
- Poli R (1996) Genetic programming for image analysis. In: Koza JR et al. (eds), *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, CA 28–31 July MIT Press pp. 363–368
- Poli R (2001) Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genet Programming Evol Mach* 2(2):123–163
- Poli R, Langdon WB (1997) Genetic programming with one-point crossover and point mutation. Tech. Rep. CSRP-97-13, University of Birmingham, B15 2TT, U.K., 15
- Poli R, Langdon WB (2006) Efficient Markov chain model of machine code program execution and halting. In: Riolo RL, et al. (eds), *Genetic Programming Theory and Practice IV*, vol 5 of *Genetic and Evolutionary Computation*, chapter 13. Springer, Ann Arbor, 11–13 May
- Poli R, Langdon WB, Dignum S (2007) On the limiting distribution of program sizes in tree-based genetic programming. In: Ebner, M et al. (eds), *Proceedings of the 10th European Conference on Genetic Programming*, vol 4445 of Lecture Notes in Computer Science, Valencia, Spain, 11–13 Apr. Springer pp. 193–204
- Poli R, McPhee NF (Mar. 2003a) General schema theory for genetic programming with subtree-swapping crossover: Part I. *Evol Comput* 11(1):53–66
- Poli R, McPhee NF (June 2003b) General schema theory for genetic programming with subtree-swapping crossover: Part II. *Evol Comput* 11(2):169–206
- Poli R, McPhee NF (2008) Parsimony pressure made easy. In: *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pp. 1267–1274, New York, NY, ACM
- Poli R, McPhee NF, Rowe JE (Mar. 2004) Exact schema theory and Markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. *Genet Programming Evol Mach* 5(1):31–70
- Poli R, McPhee NF, Graff M (2009) Free lunches for symbolic regression. In: *Foundations of Genetic Algorithms (FOGA)*. ACM, forthcoming
- Poli R, Langdon WB, McPhee NF (2008) A Field Guide to Genetic Programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>, (With contributions by J. R. Koza)
- Rissanen J (1978) Modeling by shortest data description. *Automatica* 14:465–471
- Rochat D, Tomassini M, Vanneschi L (2005) Dynamic size populations in distributed genetic programming. In: Keijzer M, et al. (eds), *Proceedings of the 8th European Conference on Genetic Programming*, vol 3447 of Lecture Notes in Computer Science, Lausanne, Switzerland, 30 Mar.–1 Apr. Springer. pp. 50–61
- Rosca JP (1995) Towards automatic discovery of building blocks in genetic programming. In: *Working Notes for the AAAI Symposium on Genetic Programming*, AAAI, pp. 78–85
- Rudolph G (1994) Convergence analysis of canonical genetic algorithm. *IEEE Trans Neural Netw* 5(1): 96–101

- Rudolph G (1996) Convergence of evolutionary algorithms in general search spaces. In: *International Conference on Evolutionary Computation*, pp. 50–54
- Schumacher C, Vose MD, Whitley LD (2001) The no free lunch and problem description length. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Morgan Kaufmann, pp. 565–570
- Seront G (1995) External concepts reuse in genetic programming. In: Siegel EV, Koza JR (eds), *Working Notes for the AAAI Symposium on Genetic Programming*, MIT, Cambridge, MA 10–12 Nov. AAAI pp. 94–98
- Shah SC, Kusiak A (July 2004) Data mining and genetic algorithm based gene/SNP selection. *Artif Intell Med* 31(3):183–196
- Silva S (2008) Controlling bloat: individual and population based approaches in genetic programming. Ph.D. thesis, Universidade de Coimbra, Faculdade de Ciencias e Tecnologia, Departamento de Engenharia Informatica, Portugal
- Smola AJ, Scholkopf B (1999) A tutorial on support vector regression. Technical Report Technical Report Series – NC2-TR-1998-030, NeuroCOLT2
- Soule T, Foster JA (1998a) Effects of code growth and parsimony pressure on populations in genetic programming. *Evol Comput* 6(4):293–309, Winter
- Soule T, Foster JA (1998b) Removal bias: A new cause of code growth in tree based evolutionary programming. In *1998 IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska 5–9 May IEEE Press. pp. 781–186
- Spector L (1995) Evolving control structures with automatically defined macros. In: Siegel EV, Koza JR (eds), *Working Notes for the AAAI Symposium on Genetic Programming*, MIT, Cambridge, MA 10–12 Nov. AAAI pp. 99–105
- Stadler PF (2002) Fitness landscapes. In: Lässig M, Valleriani A (eds), *Biological Evolution and Statistical Physics*, vol 585 of Lecture Notes Physics, pp. 187–207, Heidelberg, Springer-Verlag
- Stephens CR, Waelbroeck H (1999) Schemata evolution and building blocks. *Evol Comput* 7(2):109–124
- Tomassini M, Vanneschi L, Cuendet J, Fernandez F (2004) A new technique for dynamic size populations in genetic programming. In: *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon, 20–23 June. IEEE Press pp. 486–493
- Tomassini M, Vanneschi L, Collard P, Clergue M (2005) A study of fitness distance correlation as a difficulty measure in genetic programming. *Evol Comput* 13(2):213–239, Summer
- Trujillo L, Olague G (2006) Using evolution to learn how to perform interest point detection. In: X Y T et al. (ed.), *ICPR 2006 18th International Conference on Pattern Recognition*, vol 1, IEEE, pp. 211–214. 20–24 Aug
- Tsang E, Jin N (2006) Incentive method to handle constraints in evolutionary. In: Collet P, et al. (eds), *Proceedings of the 9th European Conference on Genetic Programming*, vol 3905 of Lecture Notes in Computer Science, Budapest, Hungary, 10–12 Apr. Springer. pp. 133–144
- Vanneschi L (2004) Theory and practice for efficient genetic programming Ph.D. thesis, Faculty of Sciences, University of Lausanne, Switzerland
- Vanneschi L (2007) Investigating problem hardness of real life applications. In: R. R. et al., (ed), *Genetic Programming Theory and Practice V*, Springer, Computer Science Collection, pp. 107–124, Chapter 7
- Vanneschi L, Clergue M, Collard P, Tomassini M, Vérel S (2004) Fitness clouds and problem hardness in genetic programming. In: Deb K, et al. (eds), *Genetic and Evolutionary Computation – GECCO-2004, Part II*, vol 3103 of Lecture Notes in Computer Science Seattle, WA 26–30 June, Springer-Verlag pp. 690–701
- Vanneschi L, Gustafson S, Mauri G (2006) Using subtree crossover distance to investigate genetic programming dynamics. In: Collet P, et al. (ed), *Genetic Programming, 9th European Conference, EuroGP2006*, Lecture Notes in Computer Science, LNCS 3905, pp. 238–249. Springer, Berlin, Heidelberg, New York
- Vanneschi L, Mauri G, Valsecchi A, Cagnoni S (2006) Heterogeneous cooperative coevolution: strategies of integration between GP and GA. In: Keijzer M, et al. (eds), *GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, vol 1, Seattle, Washington, DC, 8–12 July. ACM Press. pp. 361–368
- Vanneschi L, Rochat D, Tomassini M (2007) Multi-optimization improves genetic programming generalization ability. In: Thierens D, et al. (eds), *GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, vol 2, London, 7–11 July. ACM Press. pp. 1759–1759
- Vanneschi L, Tomassini M, Collard P, Clergue M (2003) Fitness distance correlation in structural mutation genetic programming. In: Ryan, C., et al., (ed), *Genetic Programming, 6th European Conference, EuroGP2003*, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, pp 455–464
- Vanneschi L, Tomassini M, Collard P, Vérel S (2006) Negative slope coefficient. A measure to characterize genetic programming. In: Collet P, et al. (eds), *Proceedings of the 9th European Conference on Genetic Programming*, vol 3905 of Lecture Notes in Computer Science, Budapest, Hungary, 10–12 Apr. Springer. pp. 178–189
- Wagner N, Michalewicz Z, Khouja M, McGregor RR (Aug. 2007) Time series forecasting for dynamic

- environments: The DyFor genetic program model. *IEEE Trans Evol Comput* 11(4):433–452
- Wang Y, Wineberg M (2006) Estimation of evolvability genetic algorithm and dynamic environments. *Genet Programming Evol Mach* 7(4):355–382
- Wedge DC, Kell DB (2008) Rapid prediction of optimum population size in genetic programming using a novel genotype–fitness correlation. In: Keijzer M, et al. (eds), *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, GA, ACM pp. 1315–1322
- Whitley D, Watson JP (2005) Complexity theory and the no free lunch theorem. In: Burke EK, Kendall G (eds), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Chapter 11, pp. 317–339. Springer
- Wolpert D, Macready W (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82
- Woodward JR, Neil JR (2003) No free lunch, program induction and combinatorial problems. In: Ryan C, et al. (eds), *Genetic Programming, Proceedings of EuroGP'2003*, vol 2610 of LNCS, Essex, 14–16 Apr. Springer-Verlag pp. 475–484
- Xie H, Zhang M, Andreae P (2006) Genetic programming for automatic stress detection in spoken English. In: Rothlauf F, et al. (eds), *Applications of Evolutionary Computing, EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC*, vol 3907 of LNCS, pp. 460–471, Budapest, 10–12 Apr. Springer Verlag
- Yang S, Ong Y-S, Jin Y (Dec. 2006) Editorial to special issue on evolutionary computation in dynamic and uncertain environments. *Genet Programming Evol Mach* 7(4):293–294, Editorial
- Yu T, Chen S-H (2004) Using genetic programming with lambda abstraction to find technical trading rules. In: *Computing in Economics and Finance*, University of Amsterdam, 8–10 July
- Yu J, Yu J, Almal AA, Dhanasekaran SM, Ghosh D, Worzel WP, Chinnaian AM (Apr. 2007) Feature selection and molecular classification of cancer using genetic programming. *Neoplasia* 9(4): 292–303
- Zhang B-T, Mühlenbein H (1995) Balancing accuracy and parsimony in genetic programming. *Evol Comput* 3(1):17–38
- Zhang M, Smart W (Aug. 2006) Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recog Lett* 27(11):1266–1274. Evolutionary Computer Vision and Image Understanding

25 The Dynamical Systems Approach — Progress Measures and Convergence Properties

Silja Meyer-Nieberg¹ · Hans-Georg Beyer²

¹Fakultät für Informatik, Universität der Bundeswehr München,
Neubiberg, Germany

silja.meyer-nieberg@unibw.de

²Department of Computer Science, Fachhochschule Vorarlberg,
Dornbirn, Austria
hans-georg.beyer@fhv.at

1	<i>Introduction: Evolutionary Algorithms as Dynamical Systems</i>	742
2	<i>Results from the Local Progress and Dynamical Systems Approach</i>	760
3	<i>The Dynamical Systems Approach in Comparison</i>	808
4	<i>Conclusions</i>	810
5	<i>The Progress Coefficients</i>	811

Abstract

This chapter considers local progress and the dynamical systems approach. The approach can be used for a quantitative analysis of the behavior of evolutionary algorithms with respect to the question of convergence and the working mechanism of these algorithms. Results obtained so far for evolution strategies on various fitness functions are described and discussed before presenting drawbacks and limitations of the approach. Furthermore, a comparison with other analysis methods is given.

1 Introduction: Evolutionary Algorithms as Dynamical Systems

Evolutionary algorithms can be studied in various ways – theoretically and experimentally. This section gives an overview of an analysis approach that models the algorithms as dynamical systems. As in physics and engineering sciences, the time evolution of the systems is due to the forces acting on the systems. Thus, the forces, also referred to as local progress measures, are the basis to understand how evolutionary algorithms function.

The text is divided as follows: first, a short introduction to evolution strategies, the evolutionary algorithm to which the approach was mainly applied, is given. Afterward, a description of how the dynamics of evolution strategies can be transformed into a deterministic dynamical system is given. In the next step, the local progress measures are defined before some results are discussed. The following overview starts with a description of several commonly used fitness functions. Afterward, the dynamical systems and local progress approach is explained in more detail using a simple example. The next section lists the results obtained for undisturbed and uncertain environments. The approach presented has certain drawbacks and limitations. The remaining section is devoted to these aspects, setting the approach used in relation to other theoretical approaches.

1.1 Evolution Strategies

An *evolution strategy* (ES) is a specific evolutionary algorithm (EA) invented in 1963 by Bäck, Rechenberg, and Schwefel. The population-based search heuristic moves through the search space by means of variation, that is, mutation and recombination, and selection. A population consists of several individuals. Each individual represents a candidate solution coded in the object parameters.

The performance of an ES strongly depends on the choice of a so-called strategy parameter, the *mutation strength* also referred to as the *step size*, which controls the spread of the population due to mutation. During an optimization run, the mutation strength must be adapted continuously to allow the ES to travel with sufficient speed. To this end, several methods have been developed – for example, Rechenberg's well-known 1 / 5th-rule (Rechenberg 1973), self-adaptation (Rechenberg 1973; Schwefel 1977), or the cumulative step-size adaptation (CSA) and covariance matrix adaptation (CMA) of Ostermeier, Gawelczyk, and Hansen, for example Ostermeier et al. (1995) and Hansen and Ostermeier (2001).

Fig. 1

The $(\mu/\rho + \lambda)$ - σ SA-ES (cf. Beyer 2001b, p. 8).

```

BEGIN
    g:=0;
    INITIALIZATION ( $\mathcal{P}_\mu^{(0)} := \{(y_m^{(0)}, \sigma_m^{(0)}, F(y_m^{(0)}))\}$ );
    REPEAT
        FOR EACH OF THE  $\lambda$  OFFSPRING DO
             $\mathcal{P}_\rho :=$  REPRODUCTION( $\mathcal{P}_\mu^{(g)}$ );
             $\sigma'_l :=$  RECOMB $\sigma$ ( $\mathcal{P}_\rho$ );
             $\sigma_l :=$  MUTATE $\sigma$ ( $\sigma'_l$ );
             $y'_l :=$  RECOMB $y$ ( $\mathcal{P}_\rho$ );
             $y_l :=$  MUTATE $y$ ( $y'_l, \sigma_l$ );
             $F_l := F(y_l)$ ;
        END
         $\mathcal{P}_\lambda^{(g)} := \{(y_l, \sigma_l, F_l)\}$ ;
        CASE “,”-SELECTION:  $\mathcal{P}_\mu^{(g+1)} :=$  SELECT( $\mathcal{P}_\lambda^{(g)}$ );
        CASE “+”-SELECTION:  $\mathcal{P}_\mu^{(g+1)} :=$  SELECT( $\mathcal{P}_\mu^{(g)}, \mathcal{P}_\lambda^{(g)}$ );
        g:=g+1;
    UNTIL stop;
END

```

Following Beyer (2001b, p. 8), Fig. 1 illustrates the basic algorithm of a multi-parent $(\mu/\rho + \lambda)$ -ES with σ -self-adaptation (σ SA). In a self-adaptive ES, the tuning of the mutation strength is left to the evolution strategy itself. Similar to the object parameters, the strategy parameters are subject to variation. If an offspring is selected, it also has a chance to bequeath its strategy parameters to the offspring generation. That is, self-adaptation assumes a statistic or probabilistic connection between strategy parameters and “good” fitness values.

As Fig. 1 shows, a $(\mu/\rho + \lambda)$ -ES maintains a population $\mathcal{P}_\mu^{(g)}$ of μ candidate solutions at generation g – with the strategy parameters used in their creation. Based on that parent population, λ offspring are created via variation.

Offspring are created as follows: For each offspring, ρ of the μ parents are chosen for recombination leading to the set \mathcal{P}_ρ . The selection of the parents may be deterministic or probabilistic (see, e.g., Beyer and Schwefel 2002; Eiben and Smith 2003).

First, the strategy parameters are changed. The strategy parameters of the chosen ρ parents are recombinated and the result is mutated afterward. The change of the object parameters occurs in the next step. Again, the parameters are first recombinated and then mutated. The newly created strategy parameter σ_l is used in the mutation process. Afterward, the fitness of the offspring is determined.

After the offspring population of λ individuals has been created, the μ best individuals with respect to their fitness values are chosen as the next parental population $\mathcal{P}_\mu^{(g+1)}$. Two selection schemes are generally distinguished: “comma” and “plus”-selection. In the case of “comma”-selection, only the offspring population is considered for selection, that is, only individuals of the offspring population have a chance to be selected into the next parent population. The old parent population is completely discarded.

In the case of “plus”-selection, members of the old parent population and the offspring population may be selected into the succeeding parent population.

There are two types of recombination that are mainly used in practice: intermediate and dominant multi-parent recombination denoted by $(\mu/\mu_B, \lambda)$ and $(\mu/\mu_D, \lambda)$, respectively. Using intermediate recombination for both the object parameters and the mutation strengths, the offspring are generated in the following manner:

1. Compute the mean, $\langle \sigma \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} \sigma_m$, of the mutation strengths σ_m of the parent population.
2. Compute the centroid, $\langle \mathbf{y} \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} \mathbf{y}_m$, of the object vectors \mathbf{y}_m of the μ parents.
3. For all offspring $l \in \{1, \dots, \lambda\}$:
 - a. Derive the new mutation strength by mutating the mean $\langle \sigma \rangle$ according to $\sigma_l = \langle \sigma \rangle \zeta$ where ζ is a random variable, which should fulfill $E[\zeta] \approx 1$ (see Beyer and Schwefel (2002) for a discussion of this and further requirements). Typical choices of ζ 's distribution include the log-normal distribution, the normal distribution, or a two-point distribution (Bäck 1997).
 - b. Generate the object vector \mathbf{y}_l according to $y_i = \langle y_i \rangle + \sigma_l \mathcal{N}(0, 1)$ where y_i is the vector's i th component and $\mathcal{N}(0, 1)$ stands for a standard normally distributed random variable.

Dominant recombination works differently. For each offspring, the recombination vector is created component-wise by randomly choosing one parent for each position and copying the value.

A nonrecombinant multi-parent strategy is denoted as a (μ, λ) -ES or a $(\mu + \lambda)$ -ES. Offspring are generated by choosing one of the parents at random and adding a mutation vector.

After the offspring or the candidate solutions have been created, the μ best solutions (“plus”: offspring and parents, “comma”: offspring) are chosen, according to their fitness, and become the succeeding parent population.

Self-adaptive strategies also retain the strategy parameters for each selected individual. The strategy parameters can be seen as an additional part of an individual's genome – neutral with respect to the fitness but influencing the evolutionary progress. The mutation is usually realized by using the log-normal operator with *learning parameter* τ . The learning parameter τ is a parameter of the log-normal distribution. Similar to the standard deviation of the normal distribution, it controls the spread of the distribution (see [Table 1](#)). Larger values result in a larger spread whereas smaller values lead to a shrink toward an area including one.

Table 1

Common mutation operators for self-adaptive evolution strategies. The parameter τ appearing in the log-normal operator is usually called the *learning parameter*. The random variable U for the two-point distribution is uniformly distributed in the interval $(0, 1]$. The second row for the two-point mutation gives an alternative definition, producing the same random process when using $\alpha = \ln(1 + \beta)$

Operator	Mutation	Random variable
Log-normal	$\sigma' := \zeta \sigma$	$\zeta = e^{\tau \mathcal{N}(0, 1)}$
Meta-EP	$\sigma' := \zeta \sigma$	$\zeta = 1 + \tau \mathcal{N}(0, 1)$
Two-point	$\sigma' := \begin{cases} \sigma(1 + \beta) & \text{if } U(0, 1] \leq 1/2 \\ \sigma/(1 + \beta) & \text{if } U(0, 1] > 1/2 \end{cases}$	
Two-point	$\sigma' := \zeta \sigma$	$\zeta = e^{\alpha \operatorname{sign}(\mathcal{N}(0, 1))}$

Note, however, that the log-normal distribution is restricted to positive values and is not symmetric. Furthermore, the learning parameter controls the spread but *is not* a central moment of the distribution, that is, τ^2 is not the variance. Further alternatives are given in [Table 1](#).

Self-adaptation uses only one kind of information from the evolutionary search: the ranking of the fitness values. In contrast, the cumulative step-size adaptation (CSA) and the standard version of covariance matrix adaptation (CMA) use a so-called *evolution path*, which sums up the actually realized search steps. Therefore, these algorithms do not only use information from the fitness ranking but also related search space information. In the following, the basic CSA-algorithm is described for $(\mu/\mu_b, \lambda)$ -ES:

$$\forall l = 1 \dots \lambda : \mathbf{w}_l^{(g)} := \overrightarrow{\mathcal{N}}_l(0, 1) \quad (1)$$

$$\mathbf{x}^{(g)} := \sigma^{(g)} \mathbf{z}^{(g)} := \frac{\sigma^{(g)}}{\mu} \sum_{m=1}^{\mu} \mathbf{w}_{m;\lambda} \quad (2)$$

$$\mathbf{y}^{(g+1)} := \mathbf{y}^{(g)} + \mathbf{x}^{(g)} \quad (3)$$

$$\mathbf{s}^{(g+1)} := (1 - c)\mathbf{s}^{(g)} + \sqrt{\mu c(2 - c)} \mathbf{z}^{(g)} \quad (4)$$

$$\sigma^{(g+1)} := \sigma^{(g)} \exp\left(\frac{\|\mathbf{s}^{(g+1)}\| - \overline{\chi_N}}{D\overline{\chi_N}}\right) \quad (5)$$

with $\overrightarrow{\mathcal{N}}_l(0, 1)$ as a random vector with standard normally distributed components and D as the dampening constant (Beyer and Arnold [2003b](#)). The mechanism of cumulative step size adaptation is based on the assumption that the step size is adapted optimally when consecutive search steps are uncorrelated or perpendicular: If the mutation strength is too small, the ES takes several steps into the same or similar direction. These steps could be substituted by a single larger step. The control rule should therefore increase the mutation strength and with it the step size. If otherwise the mutation strength is too large, the ES has to retrace its movements – consecutive steps may be antiparallel. In this case, the mutation strength must be decreased.

The *evolutionary path* or *accumulated progress vector* ([Eq. 4](#)) and its length can be used to ascertain which situation is occurring at present. Let one neglect for a moment the influence of selection and assume that selection is purely random. In this case, the consecutive steps are normally distributed. Their weighted sum is therefore also normally distributed, and its length follows a χ -distribution with N degrees of freedom. This explains the appearance of the mean $\overline{\chi_N}$ of the χ_N -distribution in [Eq. 5](#).

Selection changes the situation: If the mutation strength is too large, the ES has to retrace, and antiparallel movements may occur resulting in a path length smaller than the ideal case $\overline{\chi_N}$. The CSA decreases the step size according to [Eq. 5](#). If the mutation strength is too small, the algorithm takes several steps into roughly the same direction, which results in a path length longer than that in the uncorrelated case. The CSA then increases the mutation strength.

Theoretical analyses of ESs performed so far have been devoted mainly to the isotropic mutation case that needs to consider a single mutation parameter – the mutation strength σ – only. The adaptation of arbitrary Gaussian mutation distributions is state-of-the-art in algorithm design, known as the covariance matrix adaptation-ES. This ES uses the concept of the evolutionary path and adapts additionally the full covariance matrix. Detailed descriptions of the CMA and CSA can be found in Hansen and Ostermeier ([1996, 1997, 2001](#)) and Hansen ([2006](#)). Theoretical performance analysis of the CMA-ES is, however, still in its infancy.

Table 2

Types of ESs considered. Note a $(\mu/\mu, \lambda)$ -ES and a (μ, λ) -ES always include the $(1, \lambda)$ -ES as a special case. The groups for the $(1, \lambda)$ -ES only list publications solely devoted to this ES

Evolution strategy	Fitness	References
$(1 + 1)$ -ES	Sphere	Arnold and Beyer (2002a); Beyer (1989, 1993)
$(1 + \lambda)$ -ES	Sphere	Beyer (1993)
$(1, \lambda)$ -ES	Sphere	Beyer and Meyer-Nieberg (2006a); Beyer (1993)
$(\mu/\mu, \lambda)$ -ES	Sphere	Arnold and Beyer (2003b, 2004, 2006a, b); Meyer-Nieberg and Beyer (2005); Beyer et al. (2003, 2004)
		Arnold and Beyer (2000, 2001b, 2002b, c); Beyer (1995a, 1996a); Beyer and Arnold (2003b)
$(\mu/\mu_D, \lambda)$ -ES	Sphere	Beyer (1995a, 1996a)
(μ, λ) -ES	Sphere	Arnold and Beyer (2001a, 2003a); Beyer (1995b)
(λ_{opt}) -ES	Sphere	Arnold (2006b)
(μ, λ) -ES	Linear	Arnold and Beyer (2001a, 2003a)
$(1, \lambda)$ -ES	Ridge	Beyer and Meyer-Nieberg (2006b); Beyer (2001a); Oyman et al. (1998, 2000)
$(1 + \lambda)$ -ES	Ridge	Oyman et al. (1998, 2000)
$(\mu/\mu, \lambda)$ -ES	Ridge	Meyer-Nieberg and Beyer (2007, 2008); Arnold and Beyer (2008); Arnold (2006a); Oyman and Beyer (2000)
$(\mu/\mu_D, \lambda)$ -ES	Ridge	Oyman and Beyer (2000)
$(1, \lambda)$ -ES	General quadratic functions	Beyer and Arnold (1999)
$(\mu/\mu, \lambda)$ -ES	General quadratic functions	Beyer (2004); Beyer and Arnold (2003a)
$(\mu/\mu, \lambda)$ -ES	Positive definite quadratic forms	Arnold (2007); Beyer and Finck (2009)

Table 2 gives an overview of the ES variants analyzed up to now using the progress rate and dynamical system approaches. The analysis of the CSA-ES used a slightly different version of the CSA than previously presented. Instead of Eq. 5, the mutation strength update rule reads

$$\sigma^{(g+1)} = \sigma^{(g)} e^{\frac{\|s^{(g)}\|^2 - N}{2DN}} \quad (6)$$

See, e.g., Arnold (2002a) for a discussion and usual settings of the parameters c and D .

1.2 A Short Introduction to Dynamical Systems

The dynamical systems approach uses concepts that stem from the theory of dynamical systems. This section gives a short introduction to this area. To obtain more information, the reader is referred, for example, to Wiggins (1990) and Braun (1998). The term *dynamical* already points to one of the main characteristics of these systems: they change in time. But how does the system change? Are there any points where the system comes to rest? These are some of the questions that the theory of dynamical systems addresses.

The section starts with some illustrative examples. Afterward, the concept of equilibrium solutions, the rest points of a system, is addressed. The section closes with a short introduction to the stability of equilibrium points.

1.2.1 What Are Dynamical Systems?

First of all, the term *dynamical system* has to be explained. Dynamical systems can be approached in two ways. On the one hand, on an abstract level, a dynamical system is just a difference or differential equation, the properties of which shall be studied. On the other hand, this formal mathematical description generally appears as the result of a modeling process in the analysis of a time-dependent process, for example, the course of a chemical reaction, the spread of a forest fire, the growth of a population, the spreading of an epidemic disease, or the run of an evolutionary algorithm.

A modeling process aims at formalizing and quantifying the important characteristics of the system, for instance, the interactions between entities. If the modeler is interested in how the system evolves in time, the result is a dynamic system. Thus, a dynamical system is a model of a time-dependent system or process expressed in a formal mathematical way.

In general, several classes of dynamical systems can be distinguished. Dynamical systems can be grouped by the nature of the change, that is, whether the changes are deterministic or stochastic. This section considers deterministic systems. For a rigorous introduction to the field of random dynamical systems, the reader is referred to Arnold (2002b).

Dynamical systems can further be classified into discrete time and continuous time dynamical systems. In the first case, the change of the variable, $y \in \mathbb{R}$, occurs at discrete time steps, leading to an *iterated map* or *difference equation*

$$y(t+1) = f(y(t)) \quad (7)$$

with $f: \mathbb{R}^N \rightarrow \mathbb{R}^N$ and starting point $y(0) = y_0$. In this and the following sections, it is assumed that the time always starts at $t_0 = 0$.

A dynamical system in continuous time leads to a *differential equation*

$$\frac{d}{dt}y(t) = g(y(t)) \quad (8)$$

with $g: \mathbb{R}^N \rightarrow \mathbb{R}^N$ and starting point $y(0) = y_0$. The behavior of the system (7) or (8), respectively, depends on (a) the properties of f and g and (b) to some lesser extent on the initial value y_0 .

In the following, some examples of dynamic systems are discussed. Let one consider the situation that a saver opens a bank account with a starting capital y_0 . The bank offers an interest rate of $p\%$ per annum. How does the capital increase?

The problem can be described by

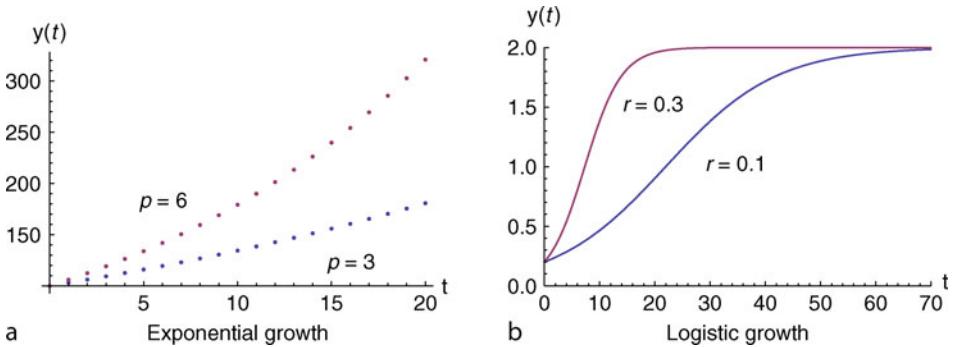
$$y(t+1) = y(t) \left(1 + \frac{p}{100}\right) \quad (9)$$

with the time, t , in discrete years. It is easy to see by performing a few iterations of Eq. 9 starting with $y(0) = y_0$ that the solution of Eq. 9 can be given in closed form as

$$y(t) = y_0 \left(1 + \frac{p}{100}\right)^t \quad (10)$$

Fig. 2

Examples for dynamical systems: (a) shows [Eq. 10](#), a time-discrete exponential growth, with initial value $y_0 = 100$ and two choices for the interest rate. (b) shows [Eq. 13](#) with carrying capacity $C = 2$ and initial value $y_0 = 0.2$.



The capital grows exponentially, see [Fig. 2a](#). The exact value at time t depends on the initial value, whereas the general behavior – the exponential growth – is a result of the form of the change. The form of [Eq. 10](#) follows from the fact that [Eq. 9](#) is a simple linear difference equation.

Let one now consider the growth of a population of bacteria in a petri dish. In experiments, it was found that a bacteria population first experiences an exponential growth which slows down in further progress. The decrease of the growth can be attributed among others to intra-species competition over increasingly scarce resources. Usually, an environment or a habitat (the petri dish) allows a maximal population density, the so-called carrying capacity C . (The population density is determined as (number of individuals)/(size of habitat).) The process can be described by a continuous model

$$\frac{d}{dt}y(t) = ry(t)\left(1 - \frac{y(t)}{C}\right) \quad (11)$$

with r standing for the exponential growth rate. Again, it is possible to obtain a solution in closed form. The calculations require partial fraction decomposition

$$\frac{\frac{d}{dt}y(t)}{ry(t)\left(1 - \frac{y(t)}{C}\right)} = \frac{d}{dt}y(t) \left(\frac{A}{ry(t)} + \frac{B}{\left(1 - \frac{y(t)}{C}\right)} \right) = 1 \quad (12)$$

with A and B to be determined and some knowledge in the area of differential equations. Finally,

$$y(t) = \frac{Cy_0 e^{rt}}{C - y_0 + y_0 e^{rt}} \quad (13)$$

is obtained. [Figure 2b](#) shows two examples of a logistic growth which differ in the size of the parameter r . The larger r is, the faster the carrying capacity is reached. Let one consider a third example. This time, consider the interaction between a predator population and a prey

population. Lotka and Volterra (see, e.g., Hofbauer and Sigmund 2002) used the following difference equation to describe the species' interactions

$$\begin{aligned}\frac{d}{dt}x(t) &= ax(t) - bx(t)y(t) \\ \frac{d}{dt}y(t) &= -cy(t) + dx(t)y(t)\end{aligned}\tag{14}$$

with $x(t)$ denoting the density of the prey population and $y(t)$ the density of the predator. Lotka and Volterra assumed that in the absence of the predator the prey population would grow exponentially, whereas the predator would die out without any prey. In the absence of the prey, the predator decreases exponentially. The parameters a and c determine the growth rate of the prey and the loss rate of the predator. It is assumed that the population growth of the predator depends on its preying and therefore on its meeting the prey. The loss of the prey due to predation is also proportional to the meeting of predator and prey. The constants b and d regulate the respective loss and gain. The Lotka–Volterra predator–prey model is one of the earliest and simplest models in population biology. However, although it is extremely simple, there are no general analytical solutions of [Eq. 14](#) which express x and y as time-dependent functions.

In the next case, the growth of a population, that is, the logistic growth, is reconsidered (Hofbauer and Sigmund 2002). For instance, in the case of insect populations with non-overlapping generations, the growth should be modeled with a difference equation instead of a differential equation leading to

$$y(t+1) = Ry(t) \left(1 - \frac{y(t)}{K}\right)\tag{15}$$

In contrast to the continuous model [\(13\)](#), a general analytical solution to [Eq. 15](#) that depends on the time cannot be obtained.

There are many more examples where it is not possible to obtain a closed analytical solution for a differential or difference equation. The question that remains is what should be done in this case. One solution that comes to mind is to perform simulations or computer experiments, to let the system evolve inside the computer and to draw statistically sound conclusions from the observations. This established method may have some drawbacks, though. What if the behavior of the system depends strongly on the size of some parameters or the initial values, and the simulations did not consider these? Or what if the time horizon of the simulation is not sufficiently long for the system to show the interesting behavior? The study of equilibrium solutions provides another way to obtain more information about a system.

1.2.2 Rest Points: Equilibrium Solutions

Equilibrium solutions are solutions of [Eqs. 8](#) or [7](#) that are the rest points of the system, that is, points where no more changes of the system occur. In other words, an equilibrium solution is a solution \mathbf{y}_{eq} of the discrete system [\(7\)](#) that does not change in time, that is,

$$\mathbf{y}(t+1) = \mathbf{y}(t) = \mathbf{y}_{\text{eq}} \Rightarrow f(\mathbf{y}_q) = \mathbf{y}_q\tag{16}$$

or a solution \mathbf{y}_{eq} of the continuous system [\(8\)](#)

$$\frac{d}{dt}\mathbf{y}(t) = g(\mathbf{y}(t)) = 0 \Rightarrow g(\mathbf{y}(t)) = g(\mathbf{y}_{\text{eq}}) = 0\tag{17}$$

There are many synonyms for equilibrium solutions – including the terms rest point, singularity, fix point, fixed point, stationary point (also solution or state), or steady state (Wiggins 1990, p. 6).

In the examples, the equilibrium solutions can be determined straightforwardly. In the case of the time-discrete exponential growth, [\(Eq. 9\)](#)

$$y(t+1) = y(t) \left(1 + \frac{p}{100}\right) \quad (18)$$

requiring $y(t+1) = y(t)$ leads only to one equilibrium solution, $y_{\text{eq}} = 0$: Only zero capital is not changed by interest yield.

In the case of the time-continuous logistic growth [\(Eq. 11\)](#), one requires $d/(dt)y(t) = 0$ for no change, leading to

$$0 = ry(t) \left(1 - \frac{y(t)}{C}\right) \quad (19)$$

with two solutions $y_{\text{eq}1} = 0$ and $y_{\text{eq}2} = C$, the zero population and the carrying capacity. Again, this immediately makes sense. A population with zero individuals cannot grow and if the maximal allowed capacity is reached, a further increase is not possible.

In the case of the Lotka–Volterra model [\(Eq. 14\)](#), the system only “rests” as a whole, if the predator equation and the prey equation have equilibrium solutions. Therefore, one has to set $d/(dt)x(t) = 0$ and $d/(dt)y(t) = 0$, leading to

$$0 = ax(t) - bx(t)y(t) = x(t)(a - by(t)) \quad (20)$$

$$0 = -cy(t) + dx(t)y(t) = y(t)(-c + dx(t)) \quad (21)$$

One solution, $x_{\text{eq}} = y_{\text{eq}} = 0$, follows immediately. If $x_{\text{eq}} \neq 0$, $a - by(t) = 0$ has to hold. Therefore, $y_{\text{eq}} = a/b$. For $a \neq 0$, $y_{\text{eq}} \neq 0$. In order to have $d/(dt)y(t) = 0$, $-c + dx = 0$ has to be fulfilled. This leads to the pair $x_{\text{eq}} = c/d$ and $y_{\text{eq}} = a/b$. Thus, the Lotka–Volterra model has two pairs of equilibrium solutions. The first one, $x_{\text{eq}} = y_{\text{eq}} = 0$ corresponds to an extinction of both populations, whereas the second corresponds to a coexistence of predator and prey.

Let one reconsider the time-discrete logistic growth model [\(Eq. 15\)](#)

$$y(t+1) = Ry(t) \left(1 - \frac{y(t)}{K}\right) \quad (22)$$

The demand $y(t+1) = y(t)$ leads to the first solution $y_{\text{eq}1} = 0$ and

$$\begin{aligned} 1 &= R \left(1 - \frac{y}{K}\right) \Rightarrow \frac{K}{R} = K - y \\ &\Rightarrow y_{\text{eq}} = K \left(\frac{R-1}{R}\right) \end{aligned} \quad (23)$$

if $R \geq 1$. The second solution starts in zero for $R = 1$ and increases afterward – approaching K for $R \rightarrow \infty$.

Note the difference of the equilibrium solutions between the time-discrete and the time-continuous logistic growth. As the examples showed, there is more than one equilibrium solution in some cases. This immediately raises the question of how the system behaves:

- Are these points ever attained by the system?
- What happens if the system is in one of these fix points and a disturbance occurs. Does it return?

In other words, one has to address the question of the stability of the equilibrium solutions.

1.2.3 On the Stability of Equilibrium Solutions

This section provides a very short introduction to the analysis of the stability of equilibrium solutions. The concept of a local stability, that is, the behavior of solutions that start close to one of the equilibrium solutions, is considered.

First of all, the term “stability” has to be defined. An equilibrium solution is *stable*, if solutions of the system which start close to it remain close for all times. This form of stability is also called *Liapunov stability*. An equilibrium solution is called *asymptotically stable*, if solutions of the dynamical system which start close to the equilibrium solution approach the equilibrium solution (Wiggins 1990, p. 6).

The concept is illustrated by the simple linear system $y(t+1) = ay(t)$, $a \in \mathbb{R}$. The only equilibrium solution is $y_{\text{eq}} = 0$. The size of a decides the behavior of solutions starting close to the equilibrium. Consider a solution $y(t)$ of the linear system which starts close to $y_{\text{eq}} = 0$, for example, $y(t) = 0 + u(t)$ with $y(0) = u(0)$, $|u(0)|$ “small.” Due to the form of the equation, the “new system” leads to $u(t) = a^t u(0)$.

If $|a| < 1$, the system is (asymptotically) stable, since $u(t)$ converges to zero. For $|a| > 1$ it is unstable; the disturbance explodes as the system moves toward infinity. The case of $|a| = 1$ is a special one: The system leaves the equilibrium solution, and does neither return nor does it go toward infinity. It remains at the distance which was given by $|u(0)|$ and may show oscillating behavior depending on the sign of a . This is not a case of asymptotical stability but only of (Liapunov) stability.

Let one consider now the discrete time logistic growth model (❷ 15)

$$y(t+1) = Ry(t) \left(1 - \frac{y(t)}{K}\right) \quad (24)$$

with equilibrium solutions $y_{\text{eq}1} = 0$ and $y_{\text{eq}2} = K(1 - 1/R)$ for $R \geq 1$. In contrast to the previous example, the system is nonlinear. How can stability criteria be derived in this case? In general, one has a system with a nonlinear function $f: \mathbb{R} \rightarrow \mathbb{R}$

$$y(t+1) = f(y(t)) \quad (25)$$

Let $y(t)$ be a solution which starts close to the equilibrium solution y_{eq} , $y(t) = y_{\text{eq}} + u(t)$ with $|u(0) - y_{\text{eq}}| \leq \delta$, $\delta > 0$. How can information be gained on how $y(t)$ behaves? Considering solutions close to the equilibrium solution allows us to apply a Taylor series expansion for the function f around y_{eq}

$$\begin{aligned} y(t+1) &= f(y_{\text{eq}}) + \frac{d}{dy}f(y)|_{y=y_{\text{eq}}} u + \mathcal{O}(|u|^2) \\ y(t+1) - y_{\text{eq}} &= \frac{d}{dy}f(y)|_{y=y_{\text{eq}}} u(t) + \mathcal{O}(|u|^2) \\ u(t+1) &= \frac{d}{dy}f(y)|_{y=y_{\text{eq}}} u(t) + \mathcal{O}(|u|^2) \end{aligned} \quad (26)$$

Neglecting the quadratic and higher terms, it depends on the first derivative of f at the equilibrium solution whether the equilibrium solution is asymptotically stable or not. If

$|d/(dy)f(y_{eq})| < 1$, then y_{eq} is asymptotically stable. If $|d/(dy)f(y_{eq})| > 1$, then y_{eq} is unstable. (This can be proven formally, see, e.g., Wiggins (1990).) If $|d/(dy)f(y_{eq})| = 1$, the equilibrium solution is called *non-hyperbolic* and no information can be gained by using the approach presented. The opposite is a *hyperbolic* equilibrium solution. In this example of a nonlinear one-dimensional system, the equilibrium solution is called hyperbolic if the absolute value of the derivative of the function at the solution is not equal to one. In these cases, the linearization of the system via Taylor series expansion suffices completely for the analysis. If the point is non-hyperbolic, other methods have to be applied. A discussion of alternative ways to proceed can be found for instance in Wiggins (1990).

In the case of the discrete logistic growth [Eq. 15](#), the nonlinear function, f , reads

$$f(y) = Ry \left(1 - \frac{y}{K}\right) \text{ with} \quad (27)$$

$$\frac{d}{dy}f(y) = R \left(1 - 2 \frac{y}{K}\right) \quad (28)$$

The equilibrium solutions are $y_{eq1} = 0$ and $y_{eq2} = K(R - 1)/R$ for $R \geq 1$. It follows immediately that the first solution, $y_{eq1} = 0$, is stable for $R < 1$ and unstable for $R > 1$. In the case of the second solution,

$$\begin{aligned} \frac{d}{dy}f(y_{eq2}) &= R \left(1 - \frac{2y_{eq2}}{K}\right) \\ &= R \left(1 - \frac{2K}{K} \left(\frac{R-1}{R}\right)\right) \\ &= R \left(1 - 2 \left(\frac{R-1}{R}\right)\right) \\ &= R - 2R + 2 = 2 - R \end{aligned} \quad (29)$$

follows. Since stability requires $|d/(dy)f(y_{eq2})| < 1$, one proceeds with determining the zero point of $2 - R$ as $R_0 = 2$. Considering first $R \leq 2$, $2 - R < 1 \Rightarrow 1 < R$ has to hold in order that y_{eq2} is stable. The second equilibrium solution is stable for $R \in (1, 2]$. For $R > 2$, $R - 2 < 1 \Rightarrow R < 3$ must be fulfilled. In short, the second equilibrium point is stable for $R \in (1, 3)$. For all larger growth parameters, both equilibrium solutions are unstable. Note, though, that this system shows a very interesting behavior – exhibiting periodic solutions and moving toward a chaotic regime. Readers interested in a more thorough discussion of the behavior of the general logistic map are referred, for example, to Hofbauer and Sigmund ([2002](#)).

Let one now consider the N -dimensional linear system

$$\mathbf{y}(t+1) = \mathbf{A}\mathbf{y}(t) \quad (30)$$

with $\mathbf{y}(t) \in \mathbb{R}^N$, $\mathbf{A} \in \mathbb{R}^{N \times N}$, and initial value $\mathbf{y}(0)$. The only equilibrium solution is $\mathbf{y}_{eq} = \mathbf{0}$. To analyze the stability, consider a solution, $\mathbf{y}(t)$, starting close to \mathbf{y}_{eq} with $\mathbf{y}(t) = \mathbf{y}_{eq} + \mathbf{u}(t)$, $\|\mathbf{u}(0) - \mathbf{y}_{eq}\| \leq \delta$, $\delta > 0$ and system

$$\mathbf{u}(t+1) = \mathbf{A}\mathbf{u}(t) \quad (31)$$

The solution is given by

$$\mathbf{u}(t) = \mathbf{A}^t \mathbf{y}(0) = \mathbf{A}^t \mathbf{u}(0) \quad (32)$$

Although the system is now in closed form, the behavior of the system is not easy to discern. One way to simplify the analysis is to switch to an alternative coordinate system. Let one

consider the eigenvalues λ_i and eigenvectors \mathbf{v}_i of the quadratic matrix \mathbf{A} (to simplify the discussions, one assumes that \mathbf{A} is diagonalizable) and recall that $\mathbf{Av}_i = \lambda_i \mathbf{v}_i$. Setting $\mathbf{M} := (\mathbf{v}_1, \dots, \mathbf{v}_N)$ and $\Lambda := (\lambda_1 \mathbf{e}_1, \dots, \lambda_N \mathbf{e}_N)$, with \mathbf{e}_i denoting the i th unit vector, it follows that

$$\mathbf{AM} = \mathbf{M}\Lambda \Rightarrow \mathbf{M}^{-1}\mathbf{AM} = \Lambda \quad (33)$$

Setting $\mathbf{z}(t) := \mathbf{M}^{-1}\mathbf{u}(t)$, [Eq. 31](#) changes to

$$\mathbf{z}(t+1) = \Lambda \mathbf{z}(t) \quad (34)$$

with the solution

$$\mathbf{z}(t) = \Lambda^t \mathbf{z}(0) \quad (35)$$

or

$$\mathbf{u}(t) = \mathbf{M}\Lambda^t \mathbf{M}^{-1}\mathbf{u}(0) \quad (36)$$

respectively. The eigenvalues of \mathbf{A} determine the behavior. If $\max |\lambda_i| < 1$, then the system converges to zero: The equilibrium solution is stable. If $\max |\lambda_i| > 1$, the system diverges to infinity and the equilibrium solution is unstable. In both cases, equilibrium solution is hyperbolic. If $\max |\lambda_i| \leq 1$ and $|\lambda_i| = 1$ for at least one eigenvalue, the respective component of \mathbf{z} remains at the initial value. In this case, the system is stable. Note, this only holds for diagonalizable matrices.

As in the one-dimensional case, the findings for the linear system can be transferred to the nonlinear system. Again, a Taylor series expansion around the equilibrium solution can be used. The equivalent to the first derivative in the one-dimensional case is the Jacobian matrix or Jacobian for functions $f: \mathbb{R}^N \rightarrow \mathbb{R}^M$: It gives the partial derivatives of the function components.

Concerning the question of stability, the Jacobian is decisive, or more correctly, its eigenvalues λ_i . If $\max |\lambda_i| < 1$ holds, then the equilibrium solution is asymptotically stable. If $|\lambda_i| > 1$ holds for at least one eigenvalue, then the solution is not stable.

Generally, in the case of time-discrete dynamic systems, an equilibrium solution is called *hyperbolic* if no eigenvalue λ_i of the Jacobian has an absolute value (modulus) equal to one, that is, $|\lambda_i| \neq 1$ for all eigenvalues. If $\max |\lambda_i| < 1$ and $|\lambda_i| = 1$ for at least one eigenvalue, then the equilibrium solution is non-hyperbolic and other methods than the one presented have to be applied.

Note, that the discussion has been limited, in this section, to the stability of the equilibrium solutions of maps or difference equations, respectively. The stability of equilibrium solutions of differential equations has not been considered. However, the discussions can be transferred with minor changes. This becomes clear when considering the simple linear system

$$\frac{d}{dt}y(t) = ay(t) \quad (37)$$

with $a \in \mathbb{R}$. The only equilibrium solution is $y_{\text{eq}} = 0$. Consider a solution $y(t)$ of [Eq. 37](#) that starts close to $y_{\text{eq}} = 0$, that is, $y(t) = y_{\text{eq}} + u(t)$ with $|u(0)| \leq \delta$, $\delta > 0$. Then

$$\begin{aligned} \frac{d}{dt}y(t) &= \frac{d}{dt}u(t) = ay(t) = a(y_{\text{eq}} + u(t)) = au(t) \\ &\Rightarrow \frac{\frac{d}{dt}u(t)}{u(t)} = a \end{aligned} \quad (38)$$

holds, leading to

$$\begin{aligned}
 \frac{\frac{d}{dt}u(t)}{u(t)} &= a \\
 \Rightarrow \frac{d}{dt}\ln(u(t)) &= a \\
 \Rightarrow \ln\left(\frac{u(t)}{u(0)}\right) &= at \\
 \Rightarrow u(t) &= u(0)e^{at}
 \end{aligned} \tag{39}$$

Again, the constant a is decisive for the question of stability. For $a < 0$, the system (39) goes to zero: The equilibrium solution is asymptotically stable. For $a > 0$, the system diverges and the system is not stable. For $a = 0$, the solution remains at the starting point. Again, this is a case of Liapunov stability not of asymptotical stability. As in the case of maps, these findings can be transferred to the nonlinear case using Taylor series expansion and to the N -dimensional case via the eigenvalues of the Jacobian.

1.3 From Stochastic Processes to Dynamical Systems

The run of an evolutionary algorithm is a dynamical movement of the population in the search space. Due to random influences, for example mutation, the movement of an EA is stochastic. In other words, the algorithm induces a stochastic process which can be analyzed in several ways. The process itself takes place in a high-dimensional state space (dimensionality $\geq N$, N – search space dimension). However, it is often preferable to consider derived or aggregated quantities with respect to the task of the analysis: that is, whether the ES converges, how fast the optimizer is approached, how EA parameters influence the behavior, and whether the step-size adaptation mechanism works adequately.

The state of an evolution strategy can be characterized by so-called state variables: variables which characterize the important and interesting features and the system behavior completely. Common variables include the fitness values, the distance to the optimizer, $\hat{\mathbf{y}}$, (depending on the fitness model), and, if the effects of step-size adaptation mechanism shall be considered, the mutation strength.

The task is to model and to analyze the evolution of these state variables over time. In the following, the sphere model (71) is used for further explanations. Since the sphere model consists of functions of the form $f(\mathbf{y}) = g(\|\mathbf{y} - \hat{\mathbf{y}}\|) = g(R)$, the state variables are chosen as the distance to the optimizer $R^{(g)} := \|\mathbf{y}^{(g)} - \hat{\mathbf{y}}\|$ and the mutation strength $\sigma^{(g)}$ at generation g . The dynamics of an ES generate the stochastic process

$$\begin{pmatrix} R^{(g)} \\ \sigma^{(g)} \end{pmatrix} \rightarrow \begin{pmatrix} R^{(g+1)} \\ \sigma^{(g+1)} \end{pmatrix} \tag{40}$$

Till now, no closed solution for the transition kernels could be derived in general. The only exception is a (1, 2)-ES using the two-point rule for the mutation of the mutation strength (see Beyer 1996b or Beyer 2001b, p. 287).

Another way to proceed is to transform the stochastic process into a deterministic dynamical system in a first approximation (mean value dynamics) and into a Gaussian

random process in a second approximation. This is done via a step-by-step approach introduced in Beyer (1996b).

The approach introduces the *evolution equations*: stochastic difference equations or iterated maps, respectively, used to describe the change of the state variables during one generation.

The change of the random variables can be divided into two parts: The first denotes the expected change. The second part covers the random fluctuations and is denoted by ε_R or ε_σ . In their most general form, the evolution equations read

$$R^{(g+1)} = R^{(g)} - E[R^{(g)} - R^{(g+1)} | R^{(g)}, \sigma^{(g)}] + \varepsilon_R(R^{(g)}, \sigma^{(g)}) \quad (41)$$

$$\sigma^{(g+1)} = \sigma^{(g)} \left(1 + E \left[\frac{\sigma^{(g+1)} - \sigma^{(g)}}{\sigma^{(g)}} | R^{(g)}, \sigma^{(g)} \right] \right) + \varepsilon_\sigma(R^{(g)}, \sigma^{(g)}) \quad (42)$$

in the case of self-adaptation. The case of CSA is described below. In \blacktriangleright Eq. 41, a well-known *progress measure* appears: the *progress rate* φ_R . The progress rate measures the expected change of the distance in one generation

$$\varphi_R(\sigma^{(g)}, R^{(g)}) := E[R^{(g)} - R^{(g+1)} | \sigma^{(g)}, R^{(g)}] \quad (43)$$

In the case of the evolution of the mutation strength, a different progress measure is used when self-adaptation is considered. Note, since the mutation of the mutation strength is generally realized by a multiplication with a random variable, \blacktriangleright Eq. 42 gives the relative change. The progress measure is called the (first-order) *self-adaptation response* (SAR) ψ . The SAR gives the expected relative change of the mutation strength in one generation

$$\psi(\sigma^{(g)}, R^{(g)}) := E \left[\frac{\sigma^{(g+1)} - \sigma^{(g)}}{\sigma^{(g)}} | \sigma^{(g)}, R^{(g)} \right]. \quad (44)$$

This progress measure is not used in analyses of CSA-evolution strategies. In CSA, the change of the mutation strength is derandomized. The stochastic influences stem from the mutation of the object parameters and enter the progress vector $\mathbf{z}^{(g)}$ \blacktriangleright Eq. 2. Analyses of the CSA usually consider the evolution of the length of the evolutionary path $\|\mathbf{s}^{(g)}\|$, \blacktriangleright Eq. 4, and the mutation strength \blacktriangleright Eq. 6

$$\|\mathbf{s}^{(g+1)}\|^2 = (1 - c)^2 \|\mathbf{s}^{(g)}\|^2 + 2(1 - c) \sqrt{2\mu(2 - c)} (\mathbf{s}^{(g)})^T \mathbf{z}^{(g)} + \mu c(2 - c) \|\mathbf{z}^{(g)}\|^2 \quad (45)$$

$$\sigma^{(g+1)} = \sigma^{(g)} e^{\frac{\|\mathbf{s}^{(g+1)}\|^2 - N}{2DN}} \quad (46)$$

The fluctuation terms in \blacktriangleright Eqs. 41 and \blacktriangleright 42 present a problem: Their distribution is not known and must be approximated using a reference density. Common approaches comprise an expansion into a Gram–Charlier or Edgeworth series (see, e.g., Kolossa 2006). The reference distribution is usually (but not necessarily) chosen to be the normal distribution. In order to expand an unknown distribution at all, it must be possible to determine some of its moments or cumulants. If this is the case, the approach continues by standardizing the fluctuation terms using the mean and the standard deviations. Clearly, the conditional mean of ε_σ and ε_R is zero. Therefore, only the standard deviation remains to be determined.

The main points of the derivation are explained considering the case of ε_R . The case of the mutation strength is analogous. Let D_φ denote the standard deviation. Therefore, the standardized random part ε'_R is related to ε_R by $\varepsilon_R = D_\varphi \varepsilon'_R$. The standard deviation can

be derived via \blacktriangleright Eq. 41 since its square equals the second conditional moment of ε_R (note $E[\varepsilon_R] = 0$)

$$\begin{aligned} D_\varphi^2(\sigma^{(g)}, R^{(g)}) &= E\left[\varepsilon_R^2 | \sigma^{(g)}, R^{(g)}\right] = E\left[\left(R^{(g+1)} - R^{(g)} + \varphi_R(\sigma^{(g)}, R^{(g)})\right)^2 | \sigma^{(g)}, R^{(g)}\right] \\ &= E\left[\left(R^{(g+1)} - R^{(g)}\right)^2 | \sigma^{(g)}, R^{(g)}\right] - \varphi_R^2(\sigma^{(g)}, R^{(g)}) \end{aligned} \quad (47)$$

The distribution of ε_R is expanded into an Edgeworth series. For the analysis, the expansion is cut off after the first term (cf. Beyer (2001b, p. 265)). That is to say, it is supposed that the deviations from the normal distribution are negligible in the analysis scenario the equations will be applied to. The random variable ε_R reads

$$\varepsilon_R = D_\varphi(\sigma^{(g)}, R^{(g)}) \mathcal{N}(0, 1) + \dots \quad (48)$$

The expectation $E[(R^{(g+1)} - R^{(g)})^2 | \sigma^{(g)}, R^{(g)}]$ appearing in \blacktriangleright Eq. 47 is called the *second-order progress rate*

$$\varphi_R^{(2)}(\sigma^{(g)}, R^{(g)}) := E\left[\left(R^{(g+1)} - R^{(g)}\right)^2 | \sigma^{(g)}, R^{(g)}\right] \quad (49)$$

The random variable ε_σ is obtained similarly. As in the case of the distance, a first-order approach (i.e., the first term of the series expansion) is used

$$\varepsilon_\sigma = D_\psi(\sigma^{(g)}, R^{(g)}) \mathcal{N}(0, 1) + \dots \quad (50)$$

The derivation of the standard deviation is exactly the same. One has

$$\begin{aligned} D_\psi^2(\sigma^{(g)}, R^{(g)}) &= E\left[\varepsilon_\sigma^2 | \sigma^{(g)}, R^{(g)}\right] = E\left[\left(\sigma^{(g+1)} - \sigma^{(g)} - \sigma^{(g)}\psi(\sigma^{(g)}, R^{(g)})\right)^2 | \sigma^{(g)}, R^{(g)}\right] \\ &= (\sigma^{(g)})^2 E\left[\left(\frac{\sigma^{(g+1)} - \sigma^{(g)}}{\sigma^{(g)}}\right)^2 | \sigma^{(g)}, R^{(g)}\right] - (\sigma^{(g)})^2 \psi^2(\sigma^{(g)}, R^{(g)}) \end{aligned} \quad (51)$$

(cf. \blacktriangleright Eq. 42). Again, this introduces a new measure, the *second-order SAR*

$$\psi^{(2)}(\sigma^{(g)}, R^{(g)}) := E\left[\left(\frac{\sigma^{(g+1)} - \sigma^{(g)}}{\sigma^{(g)}}\right)^2 | \sigma^{(g)}, R^{(g)}\right] \quad (52)$$

Using the results obtained so far, the evolution equations can be rewritten as

$$R^{(g+1)} = R^{(g)} - \varphi_R(\sigma^{(g)}, R^{(g)}) + D_\varphi(\sigma^{(g)}, R^{(g)}) \mathcal{N}(0, 1) + \dots \quad (53)$$

$$\begin{aligned} \sigma^{(g+1)} &= \sigma^{(g)} \left(1 + \psi(\sigma^{(g)}, R^{(g)})\right) + D_\psi(\sigma^{(g)}, R^{(g)}) \mathcal{N}(0, 1) + \dots \\ &= \sigma^{(g)} \left(1 + \psi(\sigma^{(g)}, R^{(g)}) + D'_\psi(\sigma^{(g)}, R^{(g)}) \mathcal{N}(0, 1) + \dots\right) \end{aligned} \quad (54)$$

with $D'_\psi = D_\psi / \sigma^{(g)}$.

1.3.1 The Deterministic Evolution Equations

In most analyses, the fluctuation parts are neglected with the exception of Beyer (1996b). The evolution equations without perturbation parts are generally termed *deterministic evolution*

equations (Beyer 2001b). This approach serves well to extract the general characteristics of self-adaptive evolution strategies. The deterministic evolution equations read

$$R^{(g+1)} = R^{(g)} - E[R^{(g)} - R^{(g+1)} | \sigma^{(g)}, R^{(g)}] \quad (55)$$

$$\sigma^{(g+1)} = \sigma^{(g)} \left(1 + E \left[\frac{\sigma^{(g+1)} - \sigma^{(g)}}{\sigma^{(g)}} | \sigma^{(g)}, R^{(g)} \right] \right) \quad (56)$$

for σ SA. In the case of the CSA, \textcircled{O} Eq. 56 is replaced by the system \textcircled{O} 45), \textcircled{O} 46). Analyses can usually be conducted only in the case that the evolution approaches a time-invariant distribution or a stationary (steady) state. In the deterministic approach, this is characterized by $R^{(g+1)} = R^{(g)}$, $\|\mathbf{s}^{(g+1)}\|^2 = \|\mathbf{s}^{(g)}\|^2$ and $\sigma^{(g+1)} = \sigma^{(g)}$. Note, demanding stationarity of the $R^{(g)}$ -evolution equals a complete standstill of the ES. Often more interesting is the evolution equation of the normalized mutation strength $\sigma^{*(g)} := \sigma^{(g)} N / R^{(g)}$

$$\sigma^{*(g+1)} = \sigma^{*(g)} \left(\frac{1 + \psi(\sigma^{*(g)}, R^{(g)})}{1 - \frac{\varphi_R^*(\sigma^{*(g)}, R^{(g)})}{N}} \right) \quad (\sigma\text{SA}) \quad (57)$$

$$\sigma^{*(g+1)} = \sigma^{*(g)} \frac{R^{(g)}}{R^{(g+1)}} e^{\frac{\|\mathbf{s}^{(g+1)}\|^2 - N}{2DN}} \quad (\text{CSA}) \quad (58)$$

with $\varphi_R^* := \varphi_R N / R^{(g)}$ and $\sigma^{*(g+1)} = \sigma^{(g+1)} N / R^{(g+1)}$ since it admits a stationary state without requiring a stationary state of the $R^{(g)}$ -evolution. In the case of the CSA, $\|\mathbf{s}^{(g+1)}\|^2 = \|\mathbf{s}^{(g)}\|^2$ is additionally demanded.

The assumption of the existence of a stationary state is motivated by the observation that it is optimal in many cases for the mutation strength to scale with the distance to the optimizer. Optimal in this case refers to a local progress measure, that is, to a maximal expected gain during one generation.

1.3.2 Including the Fluctuations

Considering the perturbation parts complicates the analysis. \textcircled{O} Equations 53 and \textcircled{O} 54 describe a Markov process or a Gaussian random field, the transition densities p_{tr} of which have to be determined. The variables $R^{(g+1)}$, $R^{(g)}$, $\sigma^{(g+1)}$, and $\sigma^{(g)}$ are all random variables. Assuming decomposability of the joint density of σ and R , the density of the distance R at generation g is denoted with $p(R^{(g)})$ and the density of the mutation strength with $p(\sigma^{(g)})$. As pointed out in Beyer (2001b, p. 313), it generally suffices to concentrate on some of the moments, generally the expected values, and not to determine the complete distribution

$$\begin{aligned} \overline{R^{(g+1)}} &= \int_0^\infty R^{(g+1)} p(R^{(g+1)}) dR^{(g+1)} \\ &= \int_0^\infty \int_0^\infty (R^{(g)} - \varphi_R(R^{(g)}, \sigma^{(g)})) p(\sigma^{(g)}) p(R^{(g)}) dR^{(g)} d\sigma^{(g)} \end{aligned} \quad (59)$$

$$\begin{aligned} \overline{\sigma^{(g+1)}} &= \int_0^\infty \sigma^{(g+1)} p(\sigma^{(g+1)}) d\sigma^{(g+1)} \\ &= \int_0^\infty \int_0^\infty \sigma^{(g)} ((1 + \psi(\sigma^{(g)}, R^{(g)})) p(\sigma^{(g)}) p(R^{(g)}) d\sigma^{(g)} dR^{(g)}) \end{aligned} \quad (60)$$

As can be inferred from [Eqs. 59](#) and [60](#), the transition densities are not needed if only the expected values are to be determined.

An equilibrium or a time-invariant limit distribution of a stochastic process is then characterized by a convergence to an equilibrium distribution, that is, $\lim_{g \rightarrow \infty} p(\sigma^{(g+1)}) = \lim_{g \rightarrow \infty} p(\sigma^{(g)}) = p_\infty(\sigma)$. Note, only the normalized mutation strength converges toward an equilibrium as long as the ES progresses still. If a stationary state is reached, the invariant density solves the eigenvalue equation

$$cp_\infty(\sigma) = \int_0^\infty p_{\text{tr}}(\sigma|\sigma)p_\infty(\sigma) d\sigma \quad (61)$$

with $c = 1$ and p_{tr} the transition density. In general, the equilibrium distribution p_∞ is unknown. As pointed out in Beyer ([2001b](#), p. 318), it is possible to determine p_∞ numerically or even analytically. The results, however, tend to be quite complicated and do not allow a further analytical treatment. Instead of trying to obtain the distribution itself, the expected value is obtained by analyzing the mean value dynamics of the system. Unfortunately, the form of the evolution equations hinders a direct determination of the expectation since in general lower order moments depend on higher order moments leading to a nonending recursion.

1.4 Local Progress Measures

This section briefly describes some local progress measures. Local refers to the concept of time: They describe the (expected) changes for two consecutive generations.

1.4.1 Quality Gain

The *quality gain* is a local performance measure or local progress measure in fitness space. Instead of the expected change in distance to the objective, it gives the expected change of the fitness from one parent population to the next. It depends on the state $\mathcal{P}^{(g)}$ of the evolution strategy at generation g

$$\overline{\Delta Q(\mathbf{y}^{(g)})} := E[F(\mathbf{y}^{(g+1)})_p - F(\mathbf{y}^{(g)})_p | \mathcal{P}^{(g)}] \quad (62)$$

When considering the $(\mu/\mu, \lambda)$ -ES, the fitness of the parent populations is usually defined as the fitness of the centroid, that is, $F(\mathbf{y}^{(g)})_p := F(\langle \mathbf{y}^{(g)} \rangle)$ with $\langle \mathbf{y}^{(g)} \rangle := (1/\mu) \sum_{m=1}^{\mu} \mathbf{y}_m^{(g)}$. Another possible definition for an ES with μ parents is for instance the average fitness $F(\mathbf{y}^{(g)})_p := \langle F(\mathbf{y}^{(g)}) \rangle := (1/\mu) \sum_{m=1}^{\mu} F(\mathbf{y}_m^{(g)})$. The definitions are not equivalent except for $\mu = 1$.

The parental population at $g + 1$ consists of the μ best candidate solutions with respect to their fitness – or with respect to the fitness change to a constant reference point. In the case of “plus”-strategies, the new parental population can contain old parental individuals of generation g . In the case of “comma”-strategies, the populations are disjunct and the determination of [Eq. 62](#) is usually simpler. The expectations can be calculated by obtaining the distribution of the offspring conditional to the state at g and using order statistics (Arnold et al. ([1992](#))). An important quantity is the *local quality change* or *local fitness change* induced by a mutation

$$Q_y(\mathbf{z}) = F(\mathbf{y} + \mathbf{z}) - F(\mathbf{y}) \quad (63)$$

being usually the starting point for the determination of the quality gain and the progress rate.

1.4.2 Progress Rate

The *progress rate* is a local performance measure in parameter space. Progress rates measure the expected change of the distance to the optimizer or other reference points $\hat{\mathbf{y}}$

$$\varphi := E[\|\mathbf{y}_P^{(g)} - \hat{\mathbf{y}}\| - \|\mathbf{y}_P^{(g+1)} - \hat{\mathbf{y}}\| | \mathcal{P}^{(g)}] \quad (64)$$

This is not mandatory, however. On functions with no finite optimizer, for instance linear or ridge functions, the progress rate measures the speed (i.e., the rate of change) in a predefined direction. When considering ridge functions, for example, a progress rate can be defined to measure the expected difference of the axial component for two consecutive generations.

Like the quality gain, the progress rate considers the parent population. Again, there are several possibilities for a definition. On the one hand, it is possible to define the progress rate as the expected change of the distance of two parental centroids

$$\varphi := E[\langle \|\mathbf{y}^{(g)}\| - \hat{\mathbf{y}} \rangle - \langle \|\mathbf{y}^{(g+1)}\| - \hat{\mathbf{y}} \rangle | \mathcal{P}^{(g)}] \quad (65)$$

Alternatively,

$$\varphi := E[\langle \|\mathbf{y}^{(g)} - \hat{\mathbf{y}}\| \rangle - \langle \|\mathbf{y}^{(g+1)} - \hat{\mathbf{y}}\| \rangle | \mathcal{P}^{(g)}] \quad (66)$$

with $\langle \|\mathbf{y}^{(g)} - \hat{\mathbf{y}}\| \rangle := (1/\mu) \sum_{m=1}^{\mu} \|\mathbf{y}_m^{(g)} - \hat{\mathbf{y}}\|$ is also possible. As in the case of the quality gain, both definitions are not interchangeable except for $\mu = 1$. The determination of the progress rate requires either the determination of the distribution of the distances of the m th best individuals, $m = 1, \dots, \mu$, or the determination of the distribution of the distance of the centroid of the μ best individuals. For “plus”-strategies, the best individuals are taken from the offspring and parent population. Again, “comma”-strategies lead to disjunct populations and are easier to analyze. It should be noted that the m th best candidate solution means the solution with m th best fitness, not distance. It is therefore necessary to derive the distribution of the distance connected to the m th best fitness. Unless there is a straightforward relationship of distance and fitness, this task may not be solvable.

The progress rate is often used to derive the efficiency of an ES. The *serial efficiency* is defined as

$$\eta := \frac{\max_{\sigma} \varphi(\sigma)}{\lambda} \quad (67)$$

that is, the maximal progress per offspring or fitness evaluation. The measure can be used to compare different strategies assuming an optimally working step-size adaptation mechanism.

1.4.3 Self-Adaptation Response (SAR)

Self-Adaptation refers to a specific way to adapt control parameters of evolutionary algorithms. In short, if parameters are changed dynamically and according to feedback from the evolutionary process, the control rule is *adaptive*, and it is *self-adaptive* if the control is left to the EA itself. The *self-adaptation response (SAR)* is the expectation of the one-generational change of the mutation strength $\sigma_P^{(g)}$ of the parent population

$$\psi(\sigma_P^{(g)}, \mathcal{P}^{(g)}) := E\left[\frac{\sigma_P^{(g+1)} - \sigma_P^{(g)}}{\sigma_P^{(g)}} | \mathcal{P}^{(g)}\right] \quad (68)$$

given the state of the algorithm at generation g . The relative change is considered because the mutation of the mutation strength is usually realized by a multiplication with a random number.

The SAR is not used in analyses of CSA-evolution strategies. However, the *logarithmic adaptation response*

$$\Delta_\sigma := \ln\left(\frac{\sigma^{(g+1)}}{\sigma^{(g)}}\right) \quad (69)$$

is often considered. While it is not a progress measure per se, the measure is similar to the integrand in [Eq. 68](#) for $\sigma^{(g+1)} \approx \sigma^{(g)}$, since the logarithm has the series expansion $\ln(x) = \sum_{k=1}^{\infty} (1/k)(x-1)^k(-1)^{k-1}$ for $x \neq 0$. For $x \approx 1$, $\ln(x) \approx x - 1$ and $\ln(\sigma^{(g+1)} / \sigma^{(g)}) \approx (\sigma^{(g+1)} - \sigma^{(g)}) / \sigma^{(g)}$.

2 Results from the Local Progress and Dynamical Systems Approach

This section presents results obtained using the local progress and dynamical systems approach. Firstly, some fitness functions that were used in the analyses are introduced. Secondly, the $(\mu/\mu_b, \lambda)$ -ES is used on the sphere to explain how to derive a local progress measure, the quality gain, in detail. Afterward, the case of undisturbed fitness environments is considered. In practical optimization tasks, exact function values or exact information on the position in the search space often cannot be obtained. Therefore, noise is an important factor that has to be taken into account. The results for noisy and robust optimization are presented in the remaining part of this section.

2.1 Fitness Environments Considered

This section gives a short overview of test functions considered in the analysis.

2.1.1 Linear Functions

Linear functions are given by

$$F_{lf}(\mathbf{y}) = \mathbf{c}^T \mathbf{y} \quad (70)$$

with $\mathbf{c} \in \mathbb{R}^N$. The task is usually maximization. Since the “optimum” lies in infinity, the ES has to increase the fitness perpetually. Linear functions can be used to analyze, for instance, the effectiveness of the step-size adaptation mechanism.

2.1.2 Sphere Model

The sphere model is given by

$$F_{sphere}(\mathbf{y}) = g(\|\hat{\mathbf{y}} - \mathbf{y}\|) =: g(R) \quad (71)$$

with $\hat{\mathbf{y}}$ the optimal state and g either a strictly monotonically increasing or strictly monotonically decreasing function of $\|\hat{\mathbf{y}} - \mathbf{y}\|$. The simplest form of the sphere is

$$F_{\text{sphere}}(\mathbf{y}) = \beta \|\hat{\mathbf{y}} - \mathbf{y}\|^{\alpha} =: \beta R^{\alpha} \quad (72)$$

The sphere model is a simple test function, generally modeling more general functions in the vicinity of the optimizer. It is used to gain deeper insights into the working mechanisms of the evolutionary algorithm, its convergence speed and its scaling behavior with respect to the search space dimensionality N .

2.1.3 General Quadratic Functions

An extension of the sphere model is the function class of general quadratic functions

$$F_{\text{GQF}}(\mathbf{y}) = \mathbf{b}^T \mathbf{y} - \mathbf{y}^T \mathbf{Q} \mathbf{y} \quad (73)$$

with \mathbf{Q} a symmetric, positive definite $\mathbb{R}^{N \times N}$ matrix and \mathbf{b} an N -dimensional vector with $b_i \in \mathbb{R}$. The sphere (❷ Eq. 72) with $\alpha = 2$ and $\hat{\mathbf{y}} = \mathbf{0}$ is therefore a special case of general quadratic functions with $\mathbf{b} = \mathbf{0}$ and $\mathbf{Q} = \beta \mathbf{I}$.

2.1.4 Biquadratic Functions

Biquadratic functions are defined by

$$F_{\text{BQF}}(\mathbf{y}) = \sum_{i=1}^n a_i y_i - c_i y_i^4 \quad (74)$$

with $a_i \in \mathbb{R}$ and $c_i \in \mathbb{R}$.

2.1.5 Ridge Functions

The general ridge function, with axis direction \mathbf{v} and parameters α and d determining the shape of the ridge, is given by

$$F_{\text{gR}}(\mathbf{y}) := \mathbf{v}^T \mathbf{y} - d \left(\sqrt{(\mathbf{v}^T \mathbf{y} - \mathbf{y})^T (\mathbf{v}^T \mathbf{y} - \mathbf{y})} \right)^{\alpha} \quad (75)$$

with $d > 0$ and $\alpha > 0$. The vector $\mathbf{v} \in \mathbb{R}^N$ with $\|\mathbf{v}\| = 1$ is called the ridge direction. Usually, a rotated version of ❷ Eq. 75 is considered

$$F_{\text{ridge}}(\mathbf{y}) = y_1 - d \left(\sum_{i=2}^N y_i^2 \right)^{\frac{\alpha}{2}} \quad (76)$$

In this model, the ridge axis aligns with the coordinate axis $y_1 \mathbf{e}_1$ (Beyer 2001a). Ridge functions consist of two parts: a linear gain part and a nonlinear loss part. The loss part in ❷ Eq. 76 resembles an $(N - 1)$ -dimensional sphere. The $(N - 1)$ -terms which make up the sphere component of the ridge can be interpreted as an $(N - 1)$ -dimensional distance to the axis $y_1 \mathbf{e}_1$. Two parameters appear in ❷ Eq. 76: the weighting constant $d > 0$ and the parameter $\alpha > 0$. The latter determines the topology of the fitness landscape. Ridge functions with $\alpha = 1$ are

called *sharp ridges*, with $\alpha = 2$ *parabolic ridges*, and with $\alpha = 3$ *cubic ridges*. The constant d weights the influence of the embedded sphere. In general, if $d \rightarrow 0$, the problem degenerates to the hyperplane $F(\mathbf{y}) = y_1$, whereas for increasing d , the isofitness lines appear more and more parallel to the axis and the problem approaches a sphere model with $F(\mathbf{y}) = -d(\sum_{i=2}^N y_i^2)^{\alpha/2}$.

Ridge functions do not have a finite optimum and therefore may be considered an “ill-posed” problem for ES (Arnold and Beyer 2008): Since the “optimum” lies in infinity, the fitness of the ES must be steadily increased. Improvement is possible in many ways. Generally, there are two viewpoints that may be taken (Beyer 2001a). The first viewpoint states (Oyman 1999, p. 32) the “object variable for the optimum [...]” reads

$$\hat{y}_1 \rightarrow \infty, \forall i \neq 1: \hat{y}_i = 0.$$

This viewpoint derives its justification from seeing ridge functions as the limit of

$$F_c(\mathbf{y}) = y_1 - cy_1^2 - d\left(\sum_{i=2}^N y_i^2\right)^{\alpha/2} \quad (77)$$

for $c \rightarrow 0$ (cf. Oyman 1999 and Beyer 2001a). For every finite c , F_c has an optimal point at $(1/(2c), 0, \dots, 0)^T$. If c decreases, the position on the axis moves toward infinity.

Evolution strategies use local information. They sample the search space randomly and select the μ best offspring, that is, the candidate solutions with the μ highest fitness values. This is the foundation of the second viewpoint which takes a more process-oriented view: The ridge does not have a finite optimum. The algorithm is required to increase the fitness perpetually. This does not necessarily mean that it has to find the ridge. Although the highest fitness value is on the axis for every finite interval, the situation changes if an unbounded search space is considered. Actually, it is not even necessary to require a finite distance to the ridge. Since the search space is infinite, there are infinitely many points in arbitrary distance to the ridge with exactly the same fitness as a position on the axis. As a result, the ES may diverge from the axis – as long as it increases the linear component faster than the loss components, it still increases the overall fitness.

2.2 Example: The $(\mu/\mu_l, \lambda)$ -Evolution Strategy on the Sphere

The determination of the quality gain of the $(\mu/\mu_l, \lambda)$ -ES is used in the following as an illustrative example. The fitness environment considered is the sphere model with $g(\mathbf{y}) = -\|\mathbf{y} - \hat{\mathbf{y}}\|^2$. The quality gain for the $(\mu/\mu_l, \lambda)$ -ES is defined as the expected change

$$\begin{aligned} \overline{\Delta Q} &:= E\left[-\|\langle \mathbf{y}^{(g+1)} \rangle - \hat{\mathbf{y}}\|^2 - \left(-\|\langle \mathbf{y}^{(g)} \rangle - \hat{\mathbf{y}}\|^2\right)\right] \\ &= E[\|\langle \mathbf{y}^{(g)} \rangle - \hat{\mathbf{y}}\|^2 - \|\langle \mathbf{y}^{(g+1)} \rangle - \hat{\mathbf{y}}\|^2] =: E[(R^{(g)})^2 - (R^{(g+1)})^2] =: E[R^2 - r^2] \end{aligned} \quad (78)$$

of the squared distances of two consecutive centroids $\langle \mathbf{y} \rangle = (1/\mu) \sum_{m=1}^{\mu} \mathbf{y}_m$ to the optimizer $\hat{\mathbf{y}}$. The determination of the quality gain requires the determination of the expected value of $E[r^2]$, the squared distance of the centroid of the μ best offspring. An offspring l is created by adding a *mutation vector* \mathbf{x}_l to the centroid of the parent population $\langle \mathbf{y} \rangle$: $\mathbf{y}_l = \langle \mathbf{y} \rangle + \mathbf{x}_l$. Due to its form, the sphere model allows for a very useful simplification of the representation of the original N -dimensional system by a change of the coordinate system: At generation g , the

population's centroid is in a position $\langle \mathbf{y} \rangle$. The distance vector to the optimizer $\hat{\mathbf{y}}$ is well defined and given by \mathbf{R} . First of all, every vector can be expressed by another vector with origin in \mathbf{R} . The vector \mathbf{R} also defines a perpendicular plane. Together, the vector and the base of the plane build an alternative base for the N -dimensional space. Changing the coordinate system, every other vector can be expressed by a vector with origin in \mathbf{R} and with a component in \mathbf{R} -direction and a perpendicular component.

This holds also for \mathbf{r}_l , the distance vector of an offspring l , which can be written as $\mathbf{r}^l = \mathbf{R} - x_R^l \mathbf{e}_R + \mathbf{h}_R^l$ with $x_R^l \mathbf{e}_R$ the part in the plane of \mathbf{R} and \mathbf{h}_R^l in the perpendicular plane (see Fig. 3). The same decomposition can be applied to the distance vector \mathbf{r} of the centroid which is given by $\mathbf{r} = \langle \mathbf{y} \rangle - \hat{\mathbf{y}} + \langle \mathbf{x} \rangle = \mathbf{R} + \langle \mathbf{x} \rangle$. The vector $\langle \mathbf{x} \rangle$ can be decomposed into a part $\langle x_R \rangle \mathbf{e}_R$ in the direction of \mathbf{R} and into a perpendicular part $\langle \mathbf{h}_R \rangle$. The squared length is therefore

$$\begin{aligned} r^2 &= \mathbf{r}^T \mathbf{r} = (\mathbf{R} - \langle x_R \rangle \mathbf{e}_R + \langle \mathbf{h}_R \rangle)^T (\mathbf{R} - \langle x_R \rangle \mathbf{e}_R + \langle \mathbf{h}_R \rangle) \\ &= R^2 - 2R\langle x_R \rangle + \langle x_R \rangle^2 + \langle \mathbf{h}_R \rangle^2 \end{aligned} \quad (79)$$

The vector $\langle \mathbf{x} \rangle$ consists of the μ best mutations $\mathbf{x}_{m;\lambda}$, $m = 1, \dots, \mu$

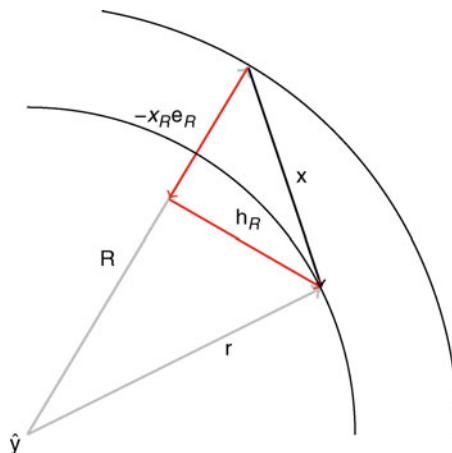
$$\langle \mathbf{x} \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} \mathbf{x}_{m;\lambda} \quad (80)$$

with $\mathbf{x}_{m;\lambda}$ standing for the m th best mutation with respect to the fitness change. The determination of the expected value of r^2 requires the determination of the expectation of the central component $\langle x_R \rangle$, that is, of the mean of the central components of the μ best mutations and of the square of the radial component $\langle \mathbf{h}_R \rangle^2$. Let one first consider $\langle \mathbf{h}_R \rangle^2$. The expectation reads

$$\overline{\langle \mathbf{h}_R \rangle^2} = E \left[\frac{1}{\mu^2} \sum_{m=1}^{\mu} \sum_{k=1}^{\mu} \mathbf{h}_{m;\lambda}^T \mathbf{h}_{k;\lambda} \right] = E \left[\frac{1}{\mu^2} \sum_{m=1}^{\mu} (\mathbf{h}_{m;\lambda})^2 \right] + E \left[\frac{1}{\mu^2} \sum_{m=1}^{\mu} \sum_{k=2, k \neq m}^{\mu} \mathbf{h}_{m;\lambda}^T \mathbf{h}_{k;\lambda} \right] \quad (81)$$

Fig. 3

The decomposition of the mutation vector in the case of the sphere model.



The expectation of the latter part is zero since $E[\mathbf{h}_{m;\lambda}^T \mathbf{h}_{k;\lambda}]$ for $m \neq k$. The perpendicular parts are selectively neutral and thus independent. Therefore,

$$\overline{\langle \mathbf{h}_R \rangle^2} = E\left[\frac{1}{\mu^2} \sum_{m=1}^{\mu} (\mathbf{h}_{m;\lambda})^2\right] = \frac{1}{\mu} \overline{\langle \mathbf{h}_R^2 \rangle} \quad (82)$$

holds. To obtain the expected value of the central and radial components of the μ best mutations, one considers the mutation-induced change of the fitness. The squared distance r_l^2 of an offspring is given by

$$r_l^2 = R^2 - 2Rx_R^l + (x_R^l)^2 + (\mathbf{h}_R^l)^2 \quad (83)$$

Without loss of generality, due to the isotropy of the mutations, one assumes that \mathbf{e}_R aligns with the first component of the coordinate system. The first component is therefore normally distributed with standard deviation σ . The radial component consists of the remaining $(N-1)$ -components – each also normally distributed

$$r_l^2 = R^2 - 2R\sigma z_x + \sigma^2 z_x^2 + \sigma^2 \sum_{i=2}^N \mathcal{N}_i(0, 1)^2 \quad (84)$$

with $z_x \sim \mathcal{N}(0, 1)$. In the following, a normalization for σ is introduced, setting $\sigma^* := \sigma N/R$. Therefore, the equation changes to

$$\begin{aligned} r_l^2 &= R^2 - 2\frac{R^2}{N} \sigma^* z_x + \frac{R^2}{N^2} \sigma^{*2} z_x^2 + \frac{R^2}{N^2} \sigma^{*2} \sum_{i=2}^N \mathcal{N}_i(0, 1)^2 \\ &= R^2 - 2\frac{R^2}{N} \sigma^* z_x + \frac{R^2}{N^2} \sigma^{*2} \sum_{i=1}^N \mathcal{N}_i(0, 1)^2 \end{aligned} \quad (85)$$

The determination of the quality gain is restricted to $N \gg 1$. According to the law of large numbers, $\sum_{i=1}^N \mathcal{N}_i(0, 1)^2/N$ goes to $E[\mathcal{N}(0, 1)^2] = 1$ for $N \rightarrow \infty$. Therefore, $(R^2/N^2) \sigma^{*2} \sum_{i=1}^N \mathcal{N}_i(0, 1)^2$ approaches $(R^2/N)\sigma^{*2}$ for increasing N . Furthermore, according to the central limit theorem, $(1/N) \sum_{i=1}^N \mathcal{N}_i(0, 1)^2$ approaches $\mathcal{N}(1, (2/N))$. Therefore, the contribution of the fluctuations of the squared random variables can be neglected and only the mean must be taken into account

$$r_l^2 = R^2 - 2\frac{R^2}{N} \sigma^* z_x + \frac{R^2}{N} \sigma^{*2} \quad (86)$$

The same holds for the contribution of $\overline{\langle \mathbf{h}_R^2 \rangle}$ which approaches

$$\frac{R^2}{N\mu} \sigma^{*2} \quad (87)$$

for increasing N . Only the contribution of the central component has to be considered. In the case of a single offspring, $z_x \sim \mathcal{N}(0, 1)$ is the only remaining random variable. The m th best offspring or mutation is characterized by having the m th smallest r_l^2 or the m th largest z_x value, out of λ random samples from the standard normal distribution. The expected value of $\langle z_x \rangle$ is thus the mean of the expectation of the first μ order statistics $u_{1;\lambda}, \dots, u_{\mu;\lambda}$ of the standard normal distribution

$$\overline{\langle z_x \rangle} = \frac{1}{\mu} \sum_{m=1}^{\mu} E[u_{m;\lambda}] = \frac{1}{\mu} \sum_{m=1}^{\mu} \int_{-\infty}^{\infty} up_{m;\lambda}(u) du \quad (88)$$

The m th order statistic $u_{m;\lambda}$ is associated with the m th largest outcome. In other words, $(m - 1)$ values must be larger than $u_{m;\lambda}$, and $(\lambda - m)$ values must be smaller. For this constellation, there are $\binom{\lambda - 1}{m - 1}$ different possibilities. Since there are λ trials, it follows

$$p_{m;\lambda}(u) = \lambda \binom{\lambda - 1}{m - 1} \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi}} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} \quad (89)$$

with $\Phi(u)$ the cumulative distribution function (cdf) of the standard normal distribution. The expectation of the mean reads

$$\begin{aligned} \overline{\langle z_x \rangle} &= \frac{1}{\mu} \sum_{m=1}^{\mu} \int_{-\infty}^{\infty} u \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi}} \lambda \binom{\lambda - 1}{m - 1} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} du \\ &= \int_{-\infty}^{\infty} u \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi}} \mu \sum_{m=1}^{\mu} \binom{\lambda - 1}{m - 1} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} du \end{aligned} \quad (90)$$

Let one first consider the sum in $\textcircled{2}$ Eq. 90

$$\begin{aligned} \sum_{m=1}^{\mu} \binom{\lambda - 1}{m - 1} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} &= \sum_{m=0}^{\mu-1} \binom{\lambda - 1}{m} (1 - \Phi(u))^m \Phi(u)^{\lambda-m-1} \\ &= 1 - \sum_{m=\mu}^{\lambda-1} \binom{\lambda - 1}{m} (1 - \Phi(u))^m \Phi(u)^{\lambda-m-1} \end{aligned} \quad (91)$$

This sum can be expressed by an integral. To this end, the regularized incomplete beta function $I_x(a, b)$ is considered

$$I_x(a, b) := \frac{1}{B(a, b)} \int_0^x t^{a-1} (1-t)^{b-1} dt \quad (92)$$

with $B(a, b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ as the complete beta function. The function $\Gamma(a)$ is the Gamma function with $\Gamma(a) = (a-1)!$ for $a \in \mathbb{N}$. The series representation of $\textcircled{2}$ Eq. 92 reads

$$I_x(a, b) := \sum_{j=a}^{a+b-1} \binom{a+b-1}{j} x^j (1-x)^{a+b-1-j} \quad (93)$$

(see, e.g., Abramowitz and Stegun 1984). The sum $\textcircled{2}$ Eq. 91 is therefore

$$\begin{aligned} \sum_{m=1}^{\mu} \binom{\lambda - 1}{m - 1} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} &= \sum_{m=0}^{\mu-1} \binom{\lambda - 1}{m} (1 - \Phi(u))^m \Phi(u)^{\lambda-m-1} \\ &= 1 - \sum_{m=\mu}^{\lambda-1} \binom{\lambda - 1}{m} (1 - \Phi(u))^m \Phi(u)^{\lambda-m-1} \\ &= 1 - I_{1-\Phi(u)}(\mu, \lambda - \mu) \end{aligned} \quad (94)$$

Since $I_x(a, b) = 1 - I_{1-x}(b, a)$, one has

$$\begin{aligned} \sum_{m=1}^{\mu} \binom{\lambda - 1}{m - 1} (1 - \Phi(u))^{m-1} \Phi(u)^{\lambda-m} &= \sum_{m=0}^{\mu-1} \binom{\lambda - 1}{m} (1 - \Phi(u))^m \Phi(u)^{\lambda-m-1} \\ &= I_{\Phi(u)}(\lambda - \mu, \mu) \\ &= \frac{(\lambda - 1)! \int_0^{\Phi(u)} t^{\lambda-\mu-1} (1-t)^{\mu-1} dt}{(\lambda - \mu - 1)! (\mu - 1)!} \end{aligned} \quad (95)$$

The expectation (Eq. 90) therefore reads

$$\overline{\langle z_x \rangle} = \int_{-\infty}^{\infty} \frac{ue^{-\frac{u^2}{2}}}{\sqrt{2\pi}} \left(\frac{\lambda}{\mu} \right) \frac{(\lambda-1)!}{(\lambda-\mu-1)!(\mu-1)!} \int_0^{\Phi(u)} t^{\lambda-\mu-1} (1-t)^{\mu-1} dt du$$

Changing the integration order leads to

$$\overline{\langle z_x \rangle} = \frac{\lambda!}{(\lambda-\mu)!\mu!} \int_0^1 \left(\int_{\Phi^{-1}(t)}^{\infty} \frac{ue^{-\frac{u^2}{2}}}{\sqrt{2\pi}} du \right) t^{\lambda-\mu-1} (1-t)^{\mu-1} dt$$

Setting $t := \Phi(x)$, $dt = \exp(-x^2/2)/\sqrt{2\pi} dx$ leads to

$$\begin{aligned} \overline{\langle z_x \rangle} &= \frac{\lambda!}{(\lambda-\mu-1)!\mu!} \int_{-\infty}^{\infty} \int_x^{\infty} \frac{ue^{-\frac{u^2}{2}}}{\sqrt{2\pi}} du \Phi(x)^{\lambda-\mu-1} (1-\Phi(x))^{\mu-1} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx \\ &= \frac{\lambda!}{(\lambda-\mu-1)!\mu!} \int_{-\infty}^{\infty} \Phi(x)^{\lambda-\mu-1} (1-\Phi(x))^{\mu-1} \frac{e^{-\frac{x^2}{2}}}{2\pi} dx \\ &= (\lambda-\mu) \binom{\lambda}{\mu} \int_{-\infty}^{\infty} \Phi(x)^{\lambda-\mu-1} (1-\Phi(x))^{\mu-1} \frac{e^{-\frac{x^2}{2}}}{2\pi} dx =: c_{\mu/\mu,\lambda} \end{aligned} \quad (96)$$

In general, the integral (Eq. 96) cannot be solved analytically – only numerically. It is a special case of the so-called generalized progress coefficients introduced in Beyer (1995b). Plugging Eqs. 96 and 87 into Eq. 78, the expected value of the squared distance r^2 becomes

$$E[r^2] = R^2 - 2 \frac{R^2}{N} \sigma^* c_{\mu/\mu,\lambda} + \frac{R^2}{N\mu} \sigma^{*2} \quad (97)$$

leading to the quality gain

$$\overline{\Delta Q} = R^2 - E[r^2] = R^2 \left(\frac{2\sigma^*}{N} c_{\mu/\mu,\lambda} - \frac{\sigma^{*2}}{\mu N} \right) \quad (98)$$

Using the normalization $\overline{\Delta Q^*} := N\overline{\Delta Q}/(2R^2)$ gives the final expression

$$\overline{\Delta Q^*} = c_{\mu/\mu,\lambda} \sigma^* - \frac{\sigma^{*2}}{2\mu} \quad (99)$$

Note, due to the normalization, this quality gain is asymptotically equal to the normalized progress rate φ^* .

The result (Eq. 99) can be used to discuss the influence of the normalized mutation strength on the performance, to determine optimal values, and to compare different evolution strategies. Since the derivation assumed a large search space dimensionality, Eq. 99 is asymptotically exact for $N \rightarrow \infty$. In the case of small N , Eq. 99 may be regarded as a simple approximation of the real quality gain. The quality gain Eq. 99 consists of two parts: a linear gain part and a nonlinear loss part. Therefore, Eq. 99 adheres to the *evolutionary progress principle* (EPP) which states that any evolutionary progress is a combination of progress gain and progress loss (see, e.g., Beyer 2001b, p. 20). Another effect that appears in Eq. 99 is the so-called *genetic repair effect* due to intermediate recombination. Recombination reduces the loss part, as indicated by the factor $1/\mu$. As a result, the ES can operate with larger mutation strengths. A more complete discussion of the genetic repair effect can be found in the following section.

2.3 Undisturbed Fitness Environments

This section describes the results obtained so far for undisturbed fitness environments. The overview starts with the sphere model and then proceeds to the discussion of other fitness functions. Afterward, analyses devoted to the step-size adaptation mechanisms are presented.

2.3.1 Sphere Model

The main focus of analyses on the undisturbed sphere concerns the derivation and analysis of the progress rate φ . This progress measure is of great importance because it can be used:

- To derive evolution criteria which have to be fulfilled for a convergence of the strategy
- To derive conditions for optimal performance
- To examine and compare the efficiency of different strategies
- To analyze the effects of recombination and
- To derive recommendations for population sizing

The sphere model was analyzed considering the following evolution strategies:

- The $(1 + 1)$ -ES (Beyer 1989)
- The $(1 + \lambda)$ -ES and the $(1, \lambda)$ -ES (Beyer 1993)
- The (μ, λ) -ES (Beyer 1995b)
- The $(\mu/\mu, \lambda)$ -ES (Beyer 1995a, 1996a) and
- The (λ_{opt}) -ES (Arnold 2006b)

One of the first analyses concerned the $(1 + 1)$ -ES (Beyer 1989). In contrast to other evolution strategies, this single point ES allows for an exact treatment with the caveat that the progress rate can only be obtained by numerical integration. The derivation relied on

- (a) the decomposition of the mutation vector introduced in the previous section and
- (b) on the *success probability*, that is, the probability that the mutation-created offspring substitutes the parent.

The analysis led to the following findings: Firstly, the progress rate rises steeply with the mutation strength to a clearly defined maximal value before it gradually declines. Secondly, the limit curve for $N \rightarrow \infty$ is already usable for search space dimensionalities greater than 30. This justifies the determination and analysis of an analytical expression of the progress rate for infinite-dimensional search spaces.

As first introduced in Beyer (1989), a step-by-step approach can be followed extracting the important characteristics of the process to develop a geometrically motivated model. This approach was applied in Beyer (1993) to derive an asymptotical progress rate for $(1, \lambda)$ - and $(1 + \lambda)$ -ESs.

First consider the $(1 + 1)$ -ES. The approach followed relied again on the decomposition of the mutation vector into two parts: one into the direction of the objective and the other in the perpendicular plane. The length of the mutation vector or of its $(N - 1)$ -dimensional perpendicular part can be replaced with its expected value when considering $N \rightarrow \infty$. This simplified the calculations. Next, normalized quantities were introduced, that is, normalized with respect to the distance to the objective and the search space dimensionality. Further derivations relied on expanding several expressions into their Taylor series. The higher order

parts of these series scale with at least $1/N$. For $N \rightarrow \infty$, their contributions can be neglected. The approach gives the (normalized) asymptotical progress rate formula (Beyer 2001b, p. 67) which was already derived by Rechenberg (1973) following a different approach. The asymptotical progress rate (see [Table 3](#)) is decomposed into a gain and a loss part as indicated by the evolutionary progress principle (EPP). The loss part is weighted by the success probability. Interestingly, the gain part stems from the mutation component in the direction of the objective, whereas the loss part is the result of the perpendicular part. Both parts are zero for zero mutation strength and increase at first with the normalized mutation strength – the gain part linearly and the loss part quadratically. When the mutation strength increases further, both terms are damped more and more so that the progress rate goes to zero for $\sigma^* \rightarrow \infty$, (see [Fig. 4a](#)). This can be explained as follows (Beyer 2001b): Large mutation strengths cause more unsuccessful mutations on average since it becomes harder to hit the region of success (cf. [Fig. 4b](#)). The parent survives for a longer time and the progress is close to zero (cf. [Fig. 4a](#)). The progress rate is significantly positive (>0) only for mutation strengths in the so-called *evolution window* (Rechenberg 1973). There exists an optimal mutation strength of around 1.224, which leads to a maximal progress rate of 0.202 (see [Fig. 4a](#)).

The approach described above can also be used to determine the progress rate for $(1, \lambda)$ -ESs (see [Table 3](#)). And its application to $(\mu/\mu_I, \lambda)$ -ESs has been discussed in the preceding subsection containing the progress rate of the $(1, \lambda)$ -ES as a special case displayed in [Table 3](#). In the progress rate formula, the so-called *progress coefficient* $c_{1,\lambda}$ appears. Mathematically, this is the expectation of the largest $x_{\lambda,i}$ order statistic taken from λ standard, normally and independently distributed random trials (see [Sect. 5](#)).

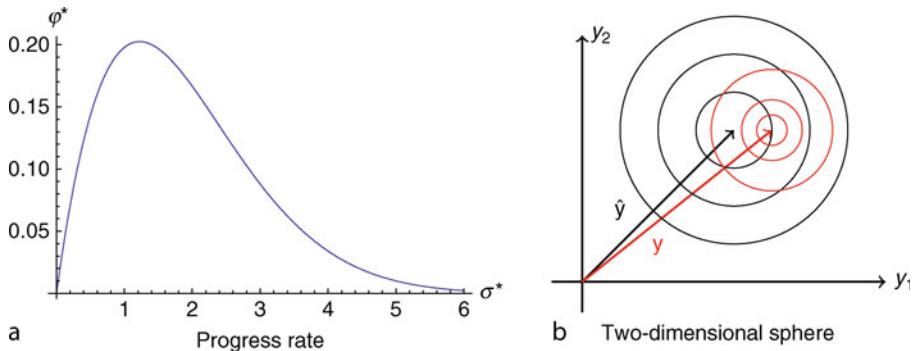
Table 3

Asymptotical progress rates and success probabilities for the sphere model. The normalizations are $\sigma^* = \sigma N/R$ and $\varphi_R^* = \varphi_R N/R$. The $c_{1,\lambda}$ is the progress coefficient (Beyer 2001b, p. 78) whereas $d_{1+\lambda}^{(1)}(x)$ denotes the first order progress function (Beyer 1993) or (Beyer 2001b, p. 78). The progress coefficient, $c_{\mu,\lambda}$, is a function of nine generalized progress coefficients ([Eq. 101](#)) and μ . See [Sect. 5](#) for a definition of the progress coefficients

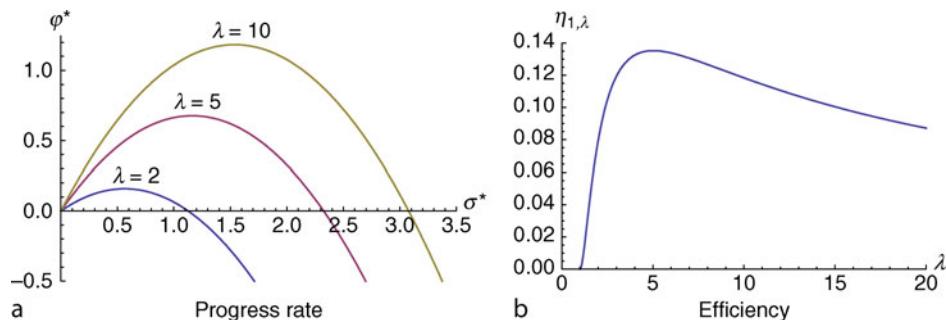
ES	Progress rate	Success probability
$(1 + 1)$ -ES	$\varphi_R^* = \frac{\sigma^*}{\sqrt{2\pi}} e^{-\frac{\sigma^{*2}}{8}} - \frac{\sigma^{*2}}{2} \left(1 - \Phi\left(\frac{\sigma^*}{2}\right)\right)$	$P_s = 1 - \Phi\left(\frac{\sigma^*}{2}\right)$
$(1 + \lambda)$ -ES	$\varphi_R^* = \sigma^* d_{1+\lambda}^{(1)}\left(\frac{\sigma^*}{2}\right) - \frac{\sigma^{*2}}{2} \left(1 - \Phi\left(\frac{\sigma^*}{2}\right)^\lambda\right)$	$P_s = 1 - \Phi\left(\frac{\sigma^*}{2}\right)^\lambda$
$(1, \lambda)$ -ES	$\varphi_R^* = c_{1,\lambda} \sigma^* - \frac{\sigma^{*2}}{2}$	–
(μ, λ) -ES	$\varphi_R^* = c_{\mu,\lambda} \sigma^* - \frac{\sigma^{*2}}{2}$	–
$(\mu/\mu_I, \lambda)$ -ES	$\varphi_R^* = c_{\mu/\mu_I, \lambda} \sigma^* - \frac{\sigma^{*2}}{2\mu}$	–
$(\mu/\mu_D, \lambda)$ -ES	$\varphi_R^* = c_{\mu/\mu_D, \lambda} \sqrt{\mu} \sigma^* - \frac{\sigma^{*2}}{2}$	–

Fig. 4

The asymptotical progress rate φ^* of the $(1 + 1)$ -ES on the sphere, (a) The progress rate is a nonlinear function of the normalized mutation strength, σ^* , with an optimal value of around 0.202 for $\sigma^* \approx 1.224$. (b) shows a two-dimensional sphere. The optimizer is located at \hat{y} . The ES is currently in position y . Operating with higher mutation strengths enlarges the region around y in which mutations most likely will fall.

**Fig. 5**

(a) The asymptotical progress rate φ^* for the $(1, \lambda)$ -ES on the sphere (b) and the efficiency as a function of λ .



The progress rate of the $(1, \lambda)$ -ES reveals a striking difference to that of the $(1 + 1)$ -ES: The progress rate may become negative, which indicates divergence (see [Fig. 5](#)). This leads to the necessary *evolution criterion* which states that the mutation strength must be smaller than a strategy-specific limit (see [Table 4](#)). The limit increases with the population size (see [Fig. 5](#)). Again, the evolutionary progress principle is present in the progress rate expression. The progress rate consists of a (linear) gain part and a (quadratic) loss part. As long as the normalized mutation strength is small, the gain part outweighs the loss part. Increasing the mutation strength changes the relation: Finally, the progress rate becomes smaller than zero. The progress rate has a maximal value and an optimizer. Both depend on the strategy. The maximal value increases with the number of offspring and the location of the optimizer moves upward. While this would indicate that an increase in the number of offspring would be profitable, the increased costs of operating with larger populations have to be considered. These considerations introduce the concept of the progress rate per fitness

Table 4

Several characteristic quantities for ES on the sphere model

ES	Evolution criterion	Maximal progress	Opt. mutation strength	Serial efficiency	Time complexity
$(1, \lambda)$	$\sigma^* < 2c_{1,\lambda}$	$\varphi_{\max}^* = \frac{c_{1,\lambda}^2}{2}$	$\sigma_{\max}^* = c_{1,\lambda}$	$\eta_{1,\lambda} = \frac{c_{1,\lambda}^2}{2\lambda}$	$G_{1,\lambda} = \mathcal{O}\left(\frac{N}{\ln(\lambda)}\right)$
$(\mu/\mu, \lambda)$	$\sigma^* < 2\mu c_{\mu/\mu, \lambda}$	$\varphi_{\max}^* = \frac{\mu c_{\mu/\mu, \lambda}^2}{2}$	$\sigma_{\max}^* = \mu c_{\mu/\mu, \lambda}$	$\eta_{\mu/\mu, \lambda} = \frac{\mu c_{\mu/\mu, \lambda}^2}{2\lambda}$	$G_{\mu/\mu, \lambda} = \mathcal{O}\left(\frac{N}{\lambda}\right)$
(μ, λ)	$\sigma^* < 2c_{\mu, \lambda}$	$\varphi_{\max}^* = \frac{c_{\mu, \lambda}^2}{2}$	$\sigma_{\max}^* = c_{\mu, \lambda}$	$\eta_{\mu, \lambda} = \frac{c_{\mu, \lambda}^2}{2\lambda}$	$G_{\mu, \lambda} = \mathcal{O}\left(\frac{N}{\ln(\lambda)}\right)$

evaluation φ^*/λ (Beyer 2001b, p. 73), which leads to the *serial efficiency* $\eta_{1,\lambda} := \varphi_{\max}^*/\lambda$ with φ_{\max}^* referring to the maximal progress rate for the specific $(1, \lambda)$ -ES. Using numerical maximization, one finds that the efficiency has its maximum at $\lambda \approx 5$. This leads in turn to the conclusion that the $(1, 5)$ -ES is the most efficient $(1, \lambda)$ -ES (see Fig. 5b). The asymptotical behavior of the progress coefficient $c_{1,\lambda}$ can be used to derive expressions for the maximal progress rate and the efficiency. These are $c_{1,\lambda} = \mathcal{O}(\sqrt{\ln(\lambda)})$, $\varphi_{\max}^* = \mathcal{O}(\ln(\lambda))$, and $\eta_{1,\lambda} = \mathcal{O}(\ln(\lambda)/\lambda)$. The progress rate can also be used to estimate the *EA time complexity*

$$G := \frac{N}{\varphi^*(\sigma^*)} \ln\left(\frac{R^{(g_0)}}{R^{(g)}}\right) \quad (100)$$

Since the progress rate scales with $\ln(\lambda)$, the EA time complexity of the $(1, \lambda)$ -ES is of the order $G_{1,\lambda} = \mathcal{O}(N/\ln(\lambda))$.

The concepts developed for the $(1+1)$ -ES and the $(1, \lambda)$ -ES have also been applied to derive the progress rate for the $(1+\lambda)$ -ES (see Table 3). As for the $(1+1)$ -ES, the progress rate approaches zero for increasing normalized mutation strengths. Considering the progress per fitness evaluation, it can be seen that operating with $\lambda > 1$ does not make any sense on the undisturbed sphere unless the algorithm is parallelized. It holds that the progress rate of the $(1+\lambda)$ -ES is greater than (or equal to) the progress rate of the $(1, \lambda)$ -ES. Furthermore, concerning evolution strategies with a single parent on the undisturbed sphere, the $(1+1)$ -ES is the most efficient ES.

In Beyer (1995b), the theory was extended to the (μ, λ) -ES. The progress rate in this case is defined as the expected change of the average distance to the optimizer. The derivation of the progress rate is inherently more difficult since the parent population must be modeled. The distribution of the parents is generation dependent and unknown. Leaving this difficulty aside, the determination of the progress rate requires obtaining the pdf (probability density function) of the distribution of the m th best offspring for which order statistics are to be used. To this end, the density of the offspring population $p(r)$ is needed, which changes from generation to generation. In general, there is no chance to get analytical exact solutions, approximations must be derived instead. Therefore, the random variables are standardized and expanded into a Gram–Charlier series with the normal distribution as reference. The expansions are cut off after the fourth term assuming that the unknown distributions are similar to a normal distribution; however, with a slight skewness.

The approach requires the determination of the mean, the standard deviation, and the third central moment. As a first step, the density of a single offspring $p(r | R_m)$, given a parent

R_m that is randomly drawn from the population, is considered. Any of the parents is drawn with the same probability. The offspring density, given the parent population, is therefore the mixture $p(r|R_1, \dots, R_\mu) = (1/\mu) \sum_{m=1}^\mu p(r|R_m)$ of the μ densities.

In this manner, the required moments can be obtained conditional to the distances of the μ parents. These, however, are random variables themselves produced by the random dynamics from the previous step and were selected as the μ best candidate solutions.

Determining the expectation requires a μ -fold integration and the determination of the joint density of the R_1, \dots, R_μ . In this expression, the pdf and cdf of the offspring of the previous generation appear. The moments of generation $g+1$ depend, therefore, on the distribution of generation g . The next steps aim to express the moments as functions of the previous moments. To this end, the same series expansion for the pdf and cdf at time g is applied.

As a result, one obtains a mapping of the distribution parameters from generation g to $g+1$. Assuming that the distributions approach a time-invariant limit distributions leads to nonlinear fix-point equations for the moments. The nonlinearity concerns primarily the parameter stemming from the third central moment. Assuming that the skewness of the distribution is not large, all higher order terms containing the expectation of this parameter can be dropped. The calculations still require the determination of the expectation of concomitants of order statistics and are quite lengthy. Finally, this leads to the progress rate for the (μ, λ) -ES and to the definition of the *generalized progress coefficients*

$$e_{\mu, \lambda}^{\alpha, \beta} = \frac{\lambda - \mu}{\sqrt{2\pi^{\alpha+1}}} \binom{\lambda}{\mu} \int_0^\infty t^\beta e^{-\frac{\alpha+1}{2}t^2} \Phi(t)^{\lambda-\mu-1} (1 - \Phi(t))^{\mu-\alpha} dt \quad (101)$$

(Beyer 1995b) which reappear in the analysis of other strategies. The progress rate (see ➤ Table 3) itself depends on several generalized progress coefficients, which are finally aggregated in one general $c_{\mu, \lambda}$ -coefficient. It behaves asymptotically as $c_{\mu, \lambda} \sim \sqrt{2 \ln(\lambda/\mu)}$ for $\lambda \gg 1$ (Beyer 1996a). This scaling behavior shows that in the non-noisy case, retaining more than the best candidate solution does not lead to any advantage. As a consequence, see ➤ Table 4, the $(1, \lambda)$ -ES and the (μ, λ) -ES share the same time complexity.

In (Beyer 1995a, 1996a), multi-recombinative evolution strategies were analyzed. Two types of recombinations were considered: dominant or discrete recombination and intermediate recombination. As for dominant recombination, the recombinant is created component-wise by choosing for each coordinate one of the parents randomly and copying the value. Evolution strategies with dominant/discrete recombination are denoted as $(\mu/\mu_D, \lambda)$ -ESs. In the case where intermediate recombination is used, the recombination result is the centroid. These ESs are denoted as $(\mu/\mu_I, \lambda)$ -ESs.

Let one first consider the progress rate of $(\mu/\mu_I, \lambda)$ -ESs: Again, not only the distribution for the largest of λ trials has to be determined but also for all m th largest trials up to μ . For a $(\mu/\mu_I, \lambda)$ -ES, the progress rate is defined as the expected change of the distance of the centroid for two consecutive generations. The new search point is generated by adding the centroid of the μ best mutations to the centroid of the previous generations. As the mutation vector in previous analyses, it can be decomposed into a component pointing toward the optimizer and into a perpendicular part. This decomposition followed by determining the pdf and cdf for the m th best offspring leads, finally, via several Taylor series expansions and neglection of higher order terms to the progress rate of the $(\mu/\mu_I, \lambda)$ -ES (see ➤ Table 3). The derivation of the progress rate revealed an important performance-improving property regarding the perpendicular parts of the mutation vectors of the group of the μ best offspring: The sphere model is

symmetric and the perpendicular parts are therefore selectively neutral when having the same length. The mutations are isotropic and all the directions are equally probable. The conclusion is that the perpendicular parts are statistically independent and the expected value of the product of two different vectors is zero. This results in

$$\overline{\langle \mathbf{h}_R \rangle^2} = \frac{1}{\mu} \langle \overline{\mathbf{h}_R^2} \rangle \quad (102)$$

This will be important when discussing the effects of intermediate recombination.

The progress rate can be used to investigate whether recombinative $(\mu/\mu_b, \lambda)$ -ESs have an advantage over nonrecombinative (μ, λ) -ESs. To ensure a fair comparison, several requirements (Beyer 1995a) should be met:

1. Problem equivalence: This states that the same fitness function and initial conditions should be used. Furthermore, the fitness function should not favor one of the algorithms per se.
2. Resource equivalence: Here, this may be satisfied by using the same number of offspring.
3. Maximal performance: Each algorithm should operate with its maximal performance (Beyer 1995a).

The comparison made for $\lambda = 50$ and $N = 100$ showed that the maximal progress rate of the recombinative strategies exceeds that of nonrecombinative strategies for all choices of μ , except $\mu = 1$. The $(1, \lambda)$ -ES is the best performing nonrecombinative (μ, λ) -ES and is still outperformed by $(\mu/\mu_b, \lambda)$ -ES for a wide range of μ . The maximal performance curve of ES with intermediate recombination indicates that an optimal value of μ exists with the largest maximal progress. Further investigations reveal that the optimal value depends on the search space dimensionality and the population size and increases with both.

The question remains, why recombinative ESs outperform the nonrecombinative comma strategies. This can be traced back to the *genetic repair principle* (GR). Intermediate recombination enables an ES to operate with larger mutation strengths. But why? Comparing the progress rates of the $(\mu/\mu_b, \lambda)$ -ES with that of the (μ, λ) -ES reveals that the main factor to which ESs with intermediate recombination owe their better performance is the reduction of the loss part by $1/\mu$. The loss part of the progress rate stems from the perpendicular parts of the mutation vectors. They enter the derivation of the progress rate over the expected value of the squared length of their average. Intermediate recombination reduces this length to $1/\mu$ th of the average of the squared lengths. This is beneficial: the perpendicular components deteriorate the progress. Let one compare two mutations which result in the same movement in the direction of the optimizer. If the perpendicular parts differ, the mutation with the larger perpendicular part results in a greater distance to the optimizer. Recombination reduces this harmful perpendicular part and offers a kind of *statistical error correction* (Beyer 1995a).

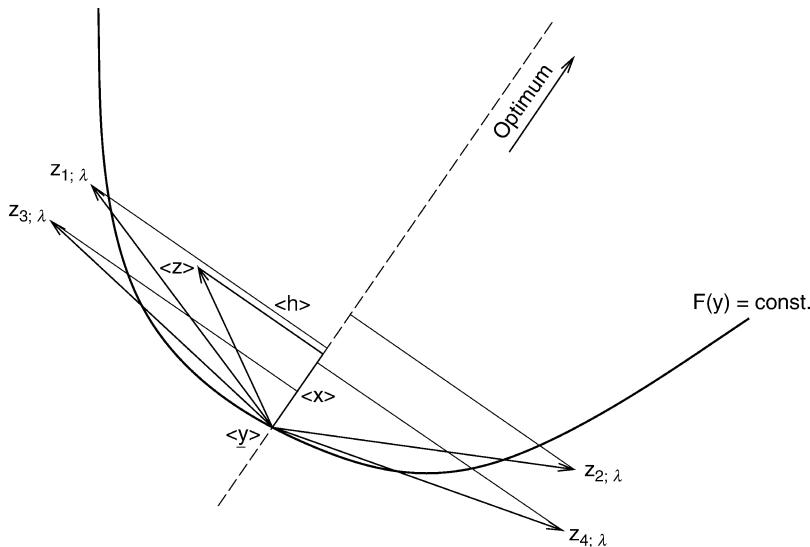
Figure 6 illustrates this mechanism graphically.

Evolution strategies with dominant/discrete recombination remain to be discussed. To enable a comparison with intermediate recombination, the progress rate is defined in the same manner. The calculation of the progress rate appears more difficult than in the previous case. The analysis requires the determination of the distribution of the parent and/or the offspring populations.

The analysis pursued in Beyer (1995a) considers the final effect of recombination and mutation as if it were a result of a *single* random variation. Thus, the concept of *surrogate mutations* was introduced. The surrogate mutations are applied to a virtual parental centroid.

Fig. 6

A visualization of the genetic repair effect for intermediate recombination on the sphere. The ES is a $(4/4, \lambda)$ -ES. The mutations, $z_{1;\lambda}$ to $z_{4;\lambda}$, which correspond to the four best offspring are displayed with their decompositions. As can be seen, intermediate recombination reduces the contribution of the harmful, perpendicular components (see Beyer and Schwefel 2002).



This enables us to decompose the mutation vector in the usual manner. However, the determination of the density of the surrogate mutations still remains. Its calculation appears very demanding and has not been solved for finite N , up until now. However, for $N \rightarrow \infty$ the progress rate can be determined: Within the limit, selection does not have a significant influence on the distribution of the offspring population. To determine the density of the population, the influence of selection can be neglected and only considered again for the progress rate.

The derivation of the distribution for the surrogate mutation vector leads to a very interesting finding: The distribution forgets any initialization effects and shrinks or expands to a limit distribution as $g \rightarrow \infty$. The standard deviation of the components approaches

$$\sigma_s := \sigma\sqrt{\mu} \quad (103)$$

being the “mutation strength” of the surrogate mutations.

The resulting progress rate differs from the progress rate for intermediate recombination in two ways (see [Table 3](#)): The loss part is not reduced by $1/\mu$. Instead, the gain part is increased by the factor $\sqrt{\mu}$. As a result, using intermediate recombination and using discrete recombination should lead to the same maximal performance. A closer look, though, reveals that this is not the case. The observed and better performance of $(\mu/\mu_i, \lambda)$ -ESs is attributed to the explicit genetic repair (GR) present in intermediate recombination (Beyer 1995a).

However, genetic repair can be argued to be present implicitly in dominant recombination – leading to the *GR hypothesis for dominant recombination*. The surrogate mutations combine dominant recombination and mutation. Dominant recombination samples coordinate points

from the μ parents. The mean of this sampling distribution is the respective coordinate point of the centroid. The drawn sample provides an estimate for the sampling distribution – mean included. Afterward mutation is applied. Mutation introduces no bias and therefore leaves the first moment of the distribution unchanged. The outcome of the mutation is again a sample providing an estimate for the distribution, the center of which is still the centroid of the parents – providing a kind of implicit center building and implicit genetic repair. Since it is a statistical estimate with statistical deviations, implicit genetic repair cannot work as well as explicit genetic repair.

Dominant recombination introduces a new hypothesis, the so-called *mutation-induced speciation by recombination* (MISR). In short this is a result of the limit value of the surrogate mutations (Beyer 1995a). Even without selection, recombination results in a stationary distribution with finite standard deviation (☞ Eq. 103): The offspring generated show cohesion. This can be interpreted as a kind of species formation.

The questions remain whether recommendations with regard to population sizing or truncation ratios, $\vartheta = \mu/\lambda$, can be given. Both recombination types achieve the same maximal progress for $N \rightarrow \infty$. The results have been used to determine the optimal truncation ratio $\vartheta_{\text{opt}} = 0.27$ with a resulting serial efficiency of $\eta_{\mu/\mu,\lambda} = 0.202$. Comparing the result with the $(1+1)$ -ES reveals that the recombinative ES achieves the same efficiency as the $(1+1)$ -ES. Interestingly, the optimal success probability of the $(1+1)$ -ES equals the optimal truncation ratio of the $(\mu/\mu, \lambda)$ -ES. The $(\mu/\mu, \lambda)$ -ES can, therefore, be claimed to be the equivalent of the $(1+1)$ -ES for multi-parent strategies. As a consequence, using recombination does not improve the efficiency above that of the $(1+1)$ -ES, unless parallel computation comes into play.

Arnold (2006b) investigated the performance of an ES with intermediate recombination, which uses weighted recombination taking *all* λ offspring into account. The progress vector $\langle \mathbf{z} \rangle$ is obtained by

$$\langle \mathbf{z} \rangle = \sum_{k=1}^{\lambda} w_{k;\lambda} \mathbf{z}_{k;\lambda} \quad (104)$$

The weights, $w_{k;\lambda}$, depend on the rank of the candidate solution. As performance measure, the quality gain was considered. The results were used to determine the optimal mutation strength and quality gain, which depends on the setting of the weights. The best performance is achieved by choosing the weights $w_{k;\lambda}$ proportionally to the expected values of the corresponding orders statistics $z_{\lambda+1-k:\lambda}$ of the standard normal distribution. This (λ_{opt}) -ES is able to outperform the $(\mu/\mu, \lambda)$ -ES and the $(1+1)$ -ES.

2.3.2 General Quadratic and Non-quadratic Functions

The progress rate approach may be difficult for nonspherical fitness functions. Candidate solutions are selected based on the fitness values. Therefore, the induced order statistics appear in the determination of the progress rate: the distribution of μ distances connected with the μ best fitness values has to be obtained. On the sphere both quantities are monotonously related since the fitness is a monotonous function of the distance. But this is rather an exception than a rule. The quality gain, the expected change of the fitness for two consecutive generations, may behave differently and often the related optimizers of the strategy parameter do not agree with each other.

In Beyer (1994), the quality gain was derived for the $(1, \lambda)$ -ES and the $(1 + \lambda)$ -ES. The derivation of the quality gain requires the determination of the density of the fitness of the best offspring. This can be done by using order statistics. One of the tasks remaining is then, to find general expressions for the pdf and cdf of the fitness of a mutation-created offspring. The idea is to standardize the variable as a first step with its mean and standard deviation. Afterward, the distribution is expanded into a Gram–Charlier series with the standard normal distribution as a baseline. To this end, higher order moments or cumulants of the unknown distribution are to be determined.

This can be done for $(1 + \lambda)$ strategies yielding general results which are not specific for a fitness function. An application, however, requires the computation of the mean, standard deviation and higher order moments. The convergence of the series expansion toward the unknown true distribution and the convergence speed depends on the fitness function considered. Therefore, the approximation quality and the point where the expansion can be cut off are function specific. Example applications (Beyer 1994) include the pseudo Boolean function OneMax $F(\mathbf{b}) = \sum_{i=1}^l b_i$ with $b_i \in \{0, 1\}$, biquadratic functions, and general quadratic fitness functions (see \blacktriangleright Table 5).

Arnold (2007) analyzed the performance of the $(\mu/\mu_b, \lambda)$ -ES on a class of positive definite quadratic forms (PDQFs)

$$f(\mathbf{y}) = \xi \sum_{i=1}^{N\vartheta} y_i^2 + \sum_{i=N\vartheta+1}^N y_i^2 \quad (105)$$

with $\vartheta \leq 1$, $N\vartheta$ a natural number, and $\xi \geq 1$. The aim was to study the response to varying degrees of ill-conditioning. The condition number of a positive definite matrix is defined as the ratio of the largest eigenvalue to the smallest. In the case of \blacktriangleright Eq. 105, it is controlled by ξ .

Table 5

The quality gain of the $(1, \lambda)$ -ES for exemplary fitness functions. The symbols S_Q and M_Q denote the standard deviation and mean of the mutation-induced fitness change Q and κ_k the k th cumulant. The normalizations read $\sigma^* = \sigma \text{Tr}[Q]/\|\mathbf{a}\|$ and $\overline{\Delta Q^*} = \overline{\Delta Q \text{Tr}[Q]}/\|\mathbf{a}\|$ for general quadratic functions. In the case of biquadratic functions, they are given by $\sigma^* = \sigma \sqrt[3]{3 \sum_i c_i / \|\mathbf{a}\|}$ and $\overline{\Delta Q^*} = \frac{\overline{\Delta Q}}{\|\mathbf{a}\|} \sqrt[3]{3 \sum_i c_i / \|\mathbf{a}\|}$. For OneMax, they are defined as $\sigma^* = \sqrt{p_m/l} (2^{\frac{F_0}{l}} - 1)$ and $\overline{\Delta Q^*} = 2\overline{\Delta Q} (2^{\frac{F_0}{l}} - 1)$ with l the string length, p_m the bit-flipping probability, and F_0 the function value of the parent

ES	Fitness function	Quality gain
$(1, \lambda)$ -ES	General functions	$\overline{\Delta Q}_{1,\lambda} = M_Q + S_Q \frac{\kappa_3}{6} + c_{1,\lambda} S_Q \left(1 + \frac{5\kappa_3^2}{36} - \frac{\kappa_4}{8} \right) + S_Q d_{1,\lambda}^{(2)} \frac{\kappa_3}{6} + d_{1,\lambda}^{(3)} S_Q \left(\frac{\kappa_4}{24} - \frac{\kappa_3^2}{18} \right) + \dots$
$(1, \lambda)$ -ES	General quadratic functions	$\overline{\Delta Q^*} = c_{1,\lambda} \sigma^* - \frac{\sigma^{*2}}{2}$
$(1, \lambda)$ -ES	Biquadratic functions	$\overline{\Delta Q^*} = c_{1,\lambda} \sigma^* - \sigma^{*4}$
$(1, \lambda)$ -ES	OneMax	$\overline{\Delta Q^*} = c_{1,\lambda} \sigma^* - \frac{\sigma^{*2}}{2}$

Larger values of ξ correspond with higher degrees of ill-conditioning. The performance measure used in Arnold (2007) was

$$\Delta = E \left[-\ln \left(\frac{f(\mathbf{y}^{(g+1)})}{f(\mathbf{y}^{(g)})} \right) \right] \quad (106)$$

a logarithmic quality gain – being asymptotically equal to $E[(f(\mathbf{y}^{(g)}) - f(\mathbf{y}^{(g)}))/f(\mathbf{y}^{(g)})]$ for $N \rightarrow \infty$. Reconsidering [Eq. 105](#) shows that it is the sum of two spheres $f(R_1, R_2) = \xi R_1^2 + R_2^2$ – one $N\vartheta$ -dimensional, the other $N(1-\vartheta)$ -dimensional and evolution takes place in two subspaces. The mutation vector of an offspring can be decomposed into two parts, one for each sphere. Then, the same decomposition as in the case of the sphere can be used with respect to the distance vectors to the centers of the spheres. This allows the determination of the quality change of an offspring as a function of two random components. The same decomposition can be applied to the progress vectors required for the determination of $E[R_1^2]$ and $E[R_2^2]$.

The resulting quality gain

$$\Delta^* = \frac{c_{\mu/\mu,\lambda}\sigma^*}{\sqrt{1 + \left(\frac{R_1}{R_2\xi}\right)^2}} - \frac{\sigma^{*2}}{2\mu} \quad (107)$$

with $\sigma^* := \sigma N\vartheta/R_1$ and $\Delta^* := \Delta N\vartheta/2$ can be used to determine the optimal mutation strength and quality gain as a function of the condition number. For most values of ξ , these values can only be obtained numerically. For large ξ -values, asymptotical expressions can be derived (see [Table 9](#)).

2.3.3 Ridge Functions

Ridge functions ([76](#)) can be considered using the progress rate approach. The concept of the progress rate, however, has to be widened since ridge functions do not have any finite optimum. The form of the ridge functions automatically leads to the definition of two performance measures. One component of the ridge is made up by the position on y_1 or parallel to the axis. The respective progress rate, the *axial progress rate* φ_{y_1} gives the expectation of the change of this component during one generation. The other component can be interpreted as a distance R of the remaining $N-1$ dimensions to the axis. In this case, the *radial progress rate* φ_R measures the expected change of two consecutive distances.

The optimization of ridge functions consists of two subgoals: (a) decreasing the radial distance to the axis – optimizing the sphere component, (b) enlarging the axis position – optimizing the linear part.

Ridge functions without noise were considered in the following publications:

- Oyman et al. (1998, 2000) considered the performance of the $(1, \lambda)$ -ES and $(1 + \lambda)$ -ES on the parabolic ridge.
- Oyman and Beyer (2000) investigated the $(\mu/\mu, \lambda)$ -ES on the parabolic ridge.
- Beyer (2001a) considered the $(1, \lambda)$ -ES on general ridge functions.
- Arnold (2006a) analyzed the $(\mu/\mu_b, \lambda)$ -ES on general ridge functions.

[Table 6](#) shows the progress rates obtained. Oyman et al. (1998) analyzed the performance of the $(1, \lambda)$ -ES and $(1 + \lambda)$ -ES on the parabolic ridge. Not considering any step-size adaptation mechanism, the following results were obtained: The axial progress rate is always nonnegative. Provided that the step-size adaptation mechanism generates positive mutation strengths,

Table 6

Radial and axial progress rates φ_R and φ_{y_1} for ridge functions. Explanation of variables and notation: $c_{\mu/\mu,\lambda}$ progress coefficient (see [Sect. 5](#)), σ^* normalized mutation strength (with respect to distance to optimizer and search space dimensionality N)

ES	Ridge		Progress rate
(1, λ)-ES	Parab.	Asymp.	$\varphi_{y_1}(\sigma) = \frac{c_{1,\lambda}\sigma}{\sqrt{1 + (2dR)^2}}$
(1, λ)-ES	Parab.		$\varphi_{y_1}(\sigma) = \frac{c_{1,\lambda}\sigma}{\sqrt{1 + (2dR)^2 + 2d^2(N - 1)\sigma^2}}$
(μ/μ_r , λ)-ES	Parab.		$\varphi_{y_1}(\sigma) = \frac{c_{\mu/\mu,\lambda}\sigma}{\sqrt{1 + (2dR)^2 + 2d^2(N - 1)\sigma^2}}$
(μ/μ_D , λ)-ES	Parab.		$\varphi_{y_1}(\sigma) = \frac{c_{\mu/\mu,\lambda}\sqrt{\mu}\sigma}{\sqrt{1 + (2dR)^2 + 2\mu d^2(N - 1)\sigma^2}}$
(μ/μ_r , λ)-ES	Gen.	Asymp.	$\varphi_{y_1}^*(\sigma^*) = \frac{c_{\mu/\mu,\lambda}\sigma^*}{\sqrt{1 + \alpha^2 d^2 R^{2x-2}}}$
(μ/μ_r , λ)-ES	Gen.	Asymp.	$\varphi_R^*(\sigma^*) = \frac{\alpha d R^{x-1} c_{\mu/\mu,\lambda} \sigma^*}{\sqrt{1 + \alpha^2 d^2 R^{2x-2}}} - \frac{\sigma^{*2}}{2\mu}$

the ES progresses in axis direction. Evolution strategies generally show a stationary state of the R -dynamics after a transient phase. That is, after a while, the ES fluctuates at a certain distance to the ridge and travels on an average, parallel to the axis. The progress that is achieved depends on two quantities: the residual distance and the mutation strength. The analysis considered large mutation strengths and a large ridge parameter d . Under these conditions, the linear y_1 component can be treated as noise and the ridge appears as a sphere. This assumption enables us to determine the residual distance to the axis in analogy to the sphere model. The axial progress rate for the $(1, \lambda)$ -ES is then obtained by using the following argument: in the steady state of the R -dynamics, the isofitness landscape appears locally as a hyperplane. Progress in gradient direction can therefore be determined as the hyperplane progress rate φ_{hp} multiplied with the gradient direction as $\varphi_{hp} \nabla F / \| \nabla F \|$. The axial progress is then given as the scalar product of the first unit vector with the progress in gradient direction.

The results show an interesting response of the progress rate with respect to increasing mutation strengths: The progress rate reaches a finite strategy-dependent limit.

Comparing experimentally the performance of the $(1, \lambda)$ -ES and the $(1 + \lambda)$ -ES reveals an interesting behavior: The performance of the nonelitist $(1, \lambda)$ -ES increases with the mutation strength to a limit value. In contrast, the progress of the elitist strategy is below the progress of the $(1, \lambda)$ -ES, and decreases with the mutation strength. The $(1 + \lambda)$ -ES emphasizes the short-term goal of minimizing the distance to the axis. This goal is better achieved by elitist strategies. Performing one generation experiments with a static mutation strength and measuring the quality gain starting from various distances show that the optimization behavior of $(1 + \lambda)$ -ES achieves a larger quality gain. The progress in axis direction is a “byproduct of the selection process” (Oyman et al. 1998) and is smaller than that of the $(1, \lambda)$ -ES.

The analysis was extended in Oyman et al. (2000) considering the $(1, \lambda)$ -ES in more detail. The focus lay on obtaining and analyzing the progress rate, the quality gain, the success

probability, and the success rate. In the case of the $(1, \lambda)$ -ES, the success probability denotes the probability that the selected offspring has a better fitness than the parent, whereas the success rate gives the probability of progress by a single mutation (Oyman et al. 2000).

The progress rate was not obtained using a local approximation with a hyperplane. Instead the local quality function and order statistics were used. The so obtained progress rate depends on the distance to the axis and decreases with increasing distance. Therefore, an expression for the stationary distance was obtained by considering the evolution of the distance to the axis and applying a steady state criterion. The experimental comparison of the $(1, \lambda)$ -ES with the $(1 + \lambda)$ -ES again revealed progress deficiencies for the $(1 + \lambda)$ -ES. This is also present in the success probability and success count. Both values deteriorate faster with the normalized mutation strength. This can be traced back to the optimization behavior of the $(1 + \lambda)$ -ES. Further experiments revealed the existence of a stationary distance, which is smaller than for $(1, \lambda)$ -ES. After this stationary state is reached, progress in axis direction is only small since the smaller distance lowers the success probability.

Oyman and Beyer (2000) presented an analysis of the $(\mu/\mu, \lambda)$ -ES on the parabolic ridge. Two types of recombination: intermediate and dominant were considered.

The analysis aimed at providing insights in the progress behavior of these strategies using the axial progress rate and the steady state distance to the ridge axis. The results can be used to give recommendations with regard to the truncation ratio and for a comparison with the progress of the $(1, \lambda)$ -ES on the one hand and for a comparison of the two recombination types on the other.

The analysis approach made use of the methods established for the sphere (Beyer 1995a). The analysis for intermediate recombination required the determination and approximation of the local quality function, the mutation-induced fitness change. First of all, the general function was expanded into its Taylor series and the resulting series was cut off after the quadratic term. In this approximation, the local quality function is a quadratic polynomial in the mutation vectors components. This was used to determine the required cdfs and pdfs in the derivation of the progress rate via a normal approximation of the true distribution.

The case of dominant recombination was treated using the result for intermediate recombination and applying the concept of surrogate mutations (Beyer 1995a), that is, the step size of the surrogate mutation $\sigma_S = \sqrt{\mu}\sigma$ is plugged into the progress rate for the intermediate case.

So far, the steady state dynamics of the distance to the ridge were not included in the analysis. Considering the square of the distance and following a similar approach as for the sphere model leads to an expression for the residual distance for the $(\mu/\mu_D, \lambda)$ -ES. The distance depends on the population parameters μ and λ , on the search space dimensionality, on the mutation strength, and on the ridge parameter d . For $d \rightarrow \infty$, the residual distance approaches the mutation-induced residual distance of the sphere model $(N - 1)\sigma/(2\mu c_{\mu/\mu, \lambda})$. Recombination reduces the distance to the axis. This is similar to the genetic repair effect on the sphere model. Considering the distance to the axis, the mutation vector and the progress vector can be written again as the direct sum of two perpendicular components: one pointing to the axis and the perpendicular part. Averaging effectively reduces the perpendicular parts. The stationary distance for the $(\mu/\mu_D, \lambda)$ -ES is obtained in a similar manner as before. Comparing the distances exemplarily reveals that the intermediate ES achieves smaller distances to the axis than the $(\mu/\mu_D, \lambda)$ -ES.

In order to compare the stationary progress rates, the obtained residual distances are plugged into the respective formulas. In the case of intermediate recombination, the resulting

axial progress rate shows a saturation behavior for increasing mutation strengths. Instead of increasing indefinitely, a strategy-dependent limit value is approached. A similar behavior was found for the $(1, \lambda)$ -ES. Comparing these limits leads to the conjecture that the performance of the $(\mu/\mu, \lambda)$ -ES exceeds that of the $(1, \lambda)$ -ES if the mutation strength is large and $\mu c_{\mu/\mu, \lambda}^2 > c_{1, \lambda}^2$, holds.

The result for the $(\mu/\mu_D, \lambda)$ -ES is interesting since it indicates that using dominant recombination instead of intermediate recombination is beneficial on the parabolic ridge: The progress curve for dominant recombination is above the curve for intermediate recombination, a gap which closes for increasing mutation strengths.

In Beyer (2001a), the performance of the $(1, \lambda)$ -ES on general ridge functions was analyzed for the case of finite N and constant mutation strength. The analysis provided a simple but more accurate approach that allowed a treatment of the radial dynamics. It was shown that the y_1 -dynamics are controlled by the R -dynamics whereas the latter do not depend on the former. First of all, the progress rates were determined. The results were then used to discuss the general properties, and the mean value and steady state behavior of the ES.

The axial progress rate depends on the distance to the ridge. The asymptotical behavior for $R \rightarrow 0$ and $R \rightarrow \infty$ depends on the ridge topology parameter α . For $\alpha > 1$, for instance, the progress rate for the hyperplane is obtained for $R \rightarrow 0$, whereas for $\alpha < 1$, it appears as the limit for $R \rightarrow \infty$.

Any discussion of the mean value dynamics of the y_1 -evolution requires a closer investigation of the R -dynamics. Experiments showed that the R -evolution approaches a steady state, that is, a time-invariant limit distribution, and the mean value converges to a constant. This results in a linear progress of the mean value of the y_1 -dynamics after a transient phase.

The velocity by which the ES travels along the ridge axis depends on the value of the stationary distance R . Using the radial progress rate and setting it to zero gives a general equation to be solved for the stationary distance. However, the mutation strength appears as an unknown variable.

The steady state progress was also considered. For the sharp and the parabolic ridge, the influence of the distance can be eliminated, yielding analytically expressions for the steady state progress as functions of the mutation strength. This allows for an analysis of the influence of the mutation strength.

Other values of α have to be treated numerically. The size of α determines the qualitative behavior of the progress rate with respect to the mutation strength. While the progress rate increases continuously for $\alpha < 2$, it appears to reach a saturation behavior for $\alpha = 2$. Increasing α further leads to a unimodal progress rate with an optimal mutation strength.

Arnold (2006a) considered the general ridge function class. He derived the axial progress rate and a stationary condition for the distance to the ridge axis. The results were used to determine optimal values for the mutation strength, residual distance, and progress rate (see  [Table 10](#)).

2.3.4 Step-Size Adaptation Mechanisms

In the following, the results of applying the dynamical systems approach in the analysis of step-size mechanisms is described. First, some results for self-adaptation are presented, before the cumulative step-size adaptation is considered.

Table 7

Self-adaptation response functions (SARs) for different fitness environments. Explanation of variables and notation: σ^* normalized mutation strength (with respect to distance to optimizer and search space dimensionality). For the coefficients $c_{\mu/\mu,\lambda}$, $e_{\mu,\lambda}^{1,1}$ and $d_{1,\lambda}^{(2)}$ see [Sect. 5](#)

Fitness	ES	SAR
Sphere	$(1, \lambda)$ -ES	$\psi(\sigma^*) = \tau^2 \left(d_{1,\lambda}^{(2)} - \frac{1}{2} - c_{1,\lambda} \sigma^* \right)$
Sphere	$(\mu/\mu, \lambda)$ -ES	$\psi(\sigma^*) = \tau^2 \left(\frac{1}{2} + e_{\mu,\lambda}^{1,1} - c_{\mu/\mu,\lambda} \sigma^* \right)$
Ridge	$(\mu/\mu, \lambda)$ -ES	$\psi(\sigma^*) = \tau^2 \left(\frac{1}{2} + e_{\mu,\lambda}^{1,1} - c_{\mu/\mu,\lambda} \sigma^* \sqrt{\frac{\alpha^2 d^2 R^{2x-2}}{(1 + \alpha^2 d^2 R^{2x-2})}} \right)$
PDQF	$(\mu/\mu, \lambda)$ -ES	$\psi(\sigma^*) = \tau^2 \left(\frac{1}{2} + e_{\mu,\lambda}^{1,1} - c_{\mu/\mu,\lambda} \sigma^* \frac{\vartheta(\xi - 1) + 1}{\vartheta\xi \sqrt{1 + \xi^2}} \right)$

Self-Adaptation

Self-adaptation was considered in the following publications:

- Beyer ([1996b](#)) was the first analysis of the self-adaptation mechanism and introduced the analysis approach also followed in later analyses. The evolution strategy considered was the $(1, \lambda)$ -ES.
- In Meyer-Nieberg and Beyer ([2005](#)), the analysis was extended to the $(\mu/\mu_b, \lambda)$ -ES on the sphere.
- Meyer-Nieberg and Beyer ([2007](#)) and Beyer and Meyer-Nieberg ([2006b](#)) provided an analysis of the self-adaptive $(\mu/\mu_b, \lambda)$ -ES on ridge functions.
- And Beyer and Finck ([2009](#)) extended the method to PDQFs.

An overview of the self-adaptation response functions obtained is given in [Table 7](#). The analyses aimed to provide insights on the functioning of self-adaptation, to analyze the influence of the learning parameter τ , and the effects of recombination. Primarily, the analyses are devoted to the log-normal operator, cf. [Table 1](#), which is a very common choice. The two-point rule was considered in Beyer ([1996b, 2001b](#)). Most of the results for the log-normal operator can be expected to be transferred for the two-point rule rather easily. In the case of the $(1, \lambda)$ -ES, the correspondence between the mutation operator was shown in Beyer ([1996b](#)). In the following, the results for the sphere are described before considering ridge functions.

On the sphere, self-adaptation generally drives the mutation strength to scale with the distance to the optimizer (disregarding fluctuations of course). Using the distance as normalization, leads to the finding that the normalized mutation strength reaches a stationary state. In Beyer ([1996b](#)), the deterministic evolution equations were used to model the ES's time dynamics. To this end, the self-adaptation response function ([68](#)) had to be obtained which can be approximated as a linear function of the mutation strength for small learning parameters. The evolution equations were then used to determine conditions for the stationary state. As it has been shown, self-adaptation steers the normalized mutation strength toward a stationary point between the second zero point of the progress rate and the zero

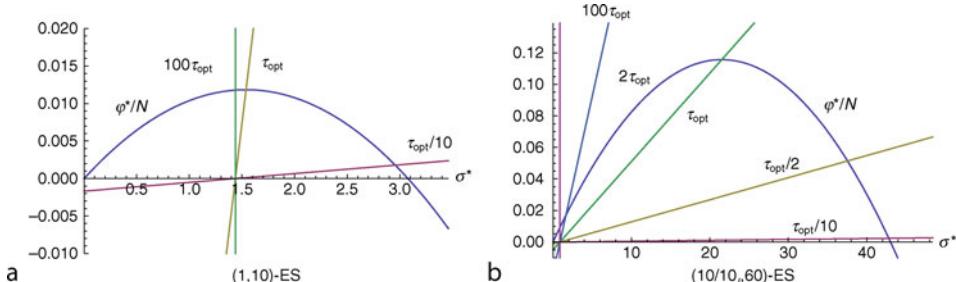
point of the self-adaptation response. On the sphere, it always holds that the zero point of the self-adaptation response is smaller than that of the progress rate. In this process, the learning parameter τ serves as control parameter: For large values of τ , the stationary normalized mutation strength is close to the zero of the SAR. Conversely, it is close to the zero of the progress rate if τ is small. Therefore, for small values of τ , there is nearly no progress at all. The maximum of the progress rate lies in between the two zeros. Therefore, a value of τ exists for which optimal expected progress can be obtained (Beyer 1996b). [Figure 7a](#) illustrates the behavior. It can be shown that choosing $\tau \propto 1/\sqrt{N}$ is roughly optimal which provides a theoretical basis for Schwefel's recommendation for choosing the learning parameter. If τ is increased above this value, the zero of the SAR is approached. This does not deteriorate the performance significantly: The zero of the SAR is close to the optimal point of the progress rate. This provides an explanation as to why $(1, \lambda)$ -ESs are robust with respect to the choice of the learning parameter.

This is not the case anymore if $(\mu/\mu_b, \lambda)$ -ESs are considered: As Grünz and Beyer found experimentally, $(\mu/\mu_b, \lambda)$ -ESs are very sensitive with respect to the choice of τ (Grünz and Beyer 1999). Instead of still showing nearly optimal progress if τ is chosen too large, these evolution strategies slow down: Nearly optimal behavior of an ES is only attainable for a narrow range of learning parameters.

In Meyer-Nieberg and Beyer (2005), an explanation has been provided. The decisive point is that the zero of the SAR is not close to the optimizer of the progress rate anymore. Increasing the number of parents usually shifts the optimizer to larger values. That is, intermediate recombination enables the ES to work with higher mutation strengths which transfers to the optimal point. The zero of the SAR does not behave accordingly, though. It is less sensitive toward variations of the parent number and, even worse, it tends to decrease for the usual choices. There is a widening gap between the zero of the SAR and the optimizer if the number of parents is increased. The consequences of this are twofold: on the one hand, there is still an optimal choice of the learning rate, on the other hand deviating from this choice will soon result in severe progress deterioration. Self-adaptation

Fig. 7

The influence of the learning rate, τ , on the stationary state of the mutation strength σ^* on the sphere. The case of a $(1,10)$ -ES is depicted in (a), (b) shows a multi-parent ES. The stationary state is governed by the equation $-\psi(\sigma^*) = \varphi^*(\sigma^*)/N$, that is, by the intersection of the negative self-adaptation response with the progress rate φ^* (divided by the search space dimensionality N). The SAR is linear in the mutation strength. The slope and the y-axis intercept are scaled with τ^2 . Shown are the progress rate and several SARs for different choices of τ . The search space dimensionality is $N = 100$.



as a mechanism still works, however. The progress remains positive and the ES moves toward the optimizer. However, as Fig. 7b shows exemplarily, it is important to choose τ close to the optimal value. Provided that the parent number is neither too close to one nor too close to the number of offspring, $\tau \approx 1/\sqrt{2N}$ leads to nearly optimal progress for large search space dimensionalities.

One question that remains is, why is the zero of the SAR behaving in this manner? The answer lies in reconsidering the mechanism of self-adaptation. The mutation strength is variated *before* the recombination of the object parameters. No detailed information is passed to the self-adaptation of the mutation strength that – due to intermediate recombination – higher mutation strengths are possible. The feedback is only indirect over the selection of the individuals with higher fitness.

In Beyer (1996b), the dynamical behavior of the $(1, \lambda)$ -ES with self-adaptation was considered. Two approaches have been pursued: One using the deterministic evolution equations, the other using the second-order approach – modeling the perturbation parts of the evolution equations with normally distributed random variables. The induced stochastic process was studied by considering some moments – the mean for most cases.

In the case of the evolution of the distance R , it was shown that the mean converges log-linearly as soon as the normalized mutation strength has reached its stationary state. Using the deterministic approximation, it can be shown that the time an ES needs to approach the steady state depends on $1/\tau^2$. As a result, the τ -scaling law $\tau \propto 1/\sqrt{N}$ causes the ES to have an adaptation time $g_A \propto N$. This might take too long for large search space dimensionalities. A short adaptation time and an optimal performance in the stationary state cannot be reached with a single learning parameter τ . This leads to the recommendation that in the initial phase, an N -independent learning parameter should be used and when approaching the steady state, one should switch to $\tau \propto 1/\sqrt{N}$.

The second-order approach provides an explanation for the observation that the experimentally measured rate is somewhat smaller than the theoretically expected value. The random dynamics of the mutation strength introduce a detrimental variance term to the average progress rate

$$\overline{\varphi_R^*(\sigma^{*(g)})} = c_{1,\lambda} \overline{\sigma^{*(g)}} - \frac{(\overline{\sigma^{*(g)}})^2}{2} - \frac{D^2[\sigma^{*(g)}]}{2} \quad (108)$$

The R -dynamics are driven by the mutation strength. Obtaining expressions for the mean and higher moments proved difficult in the case of the mutation strength because lower order moments depend functionally on higher order moments. Therefore, an *ansatz* using the log-normal distribution can serve as an approximation of the true density. The results were used to show that the stationary mean value of the normalized mutation strength is always less than the deterministic prediction due to the fluctuations. Furthermore, the scaling law of the learning parameter also proved to be valid for this second-order model.

In Beyer and Meyer-Nieberg (2006b) and Meyer-Nieberg and Beyer (2007), ridge functions were considered: the sharp ridge and $(1, \lambda)$ -ESs in Beyer and Meyer-Nieberg (2006b) and $(\mu/\mu_i, \lambda)$ -ESs and additionally the parabolic ridge in Meyer-Nieberg and Beyer (2007). Both ridge functions must be described by two state variables, the distance to the ridge (radial) and the position parallel to the axis direction (axial). Three state variables are therefore decisive: the axial position, the radial position, and the mutation strength. The analyses (Beyer and Meyer-Nieberg 2006b; Meyer-Nieberg and Beyer 2007) revealed that the evolution of the axial position does not influence any other evolution. That is, there is no direct feedback

from the present axial position. Only the mutation strength and the radial distance are the governing forces of the evolution. Therefore, in terms of a self-adaptive ES, a ridge appears as a distorted sphere.

Similar to the sphere, self-adaptation steers the mutation strength to scale on average with the distance to the ridge with the result that a stationary state of the normalized mutation strength is approached in the long run. And as before, self-adaptation drives the mutation strength toward a stationary point between the zero of the self-adaptation response (SAR) and the zero of the radial progress rate.

As for the sharp ridge ($\alpha = 1$) two quantitatively different behaviors can be observed: The ES either converges toward the axis or diverges from it. If it converges, the scaling of the mutation strength with the distance results in mutation strengths that are far too small for any significant axial progress: The search stagnates. Divergence, on the other hand, is coupled with an increasing mutation strength leading on an average to a positive quality or fitness change. The questions remain why the ES shows these different behaviors and what is the factor deciding which one occurs? These questions are answered by taking a closer look at the stationary state behavior.

In the case of the parabolic ridge, the ES attains a stationary state of the mutation strength and the distance (see [Table 8](#)). That is, there are (R, σ^*) -combinations which are stationary states of the R -evolution (zero progress $\varphi_R^* := \varphi_R/N$) and the σ^* -evolution ($\sigma^* = \sigma/N$, zero SAR ψ) (see [Fig. 8](#)). In the case of the sharp ridge, no such states exist. Other than $R = 0$ and $\sigma^* = 0$, no combination fulfills simultaneously both stationary state criteria. The ES either converges or diverges. The parameter d of the ridge function is the decisive parameter which determines the behavior. It can be shown that for a strategy-dependent critical value, the lines for $\varphi_R^* = 0$ and $\psi = 0$ overlap. If d is larger than this critical value, the line for $\varphi_R^* = 0$ is consistently above the line for $\psi = 0$. This is a similarity to the sphere model, where the second zero of the progress rate is always larger than the zero of the SAR. Depending on the choice of the learning rate τ , a self-adaptive ES adapts a normalized stationary mutation strength somewhere between these values. The attained mutation strength is connected with a positive radial progress, that is, the ES moves in expectation toward the axis. In [Fig. 9](#), this indicates that the ES moves into the inner cone delimited by $\varphi_R^* = 0$ and $\psi = 0$. If d is smaller than the critical value, however, the situation is reversed. The line for $\psi = 0$ is above the line for $\varphi_R^* = 0$. Instead of operating with mutation strengths that are connected with positive progress, the ES adapts mutation strengths which lead toward a divergence from the axis.

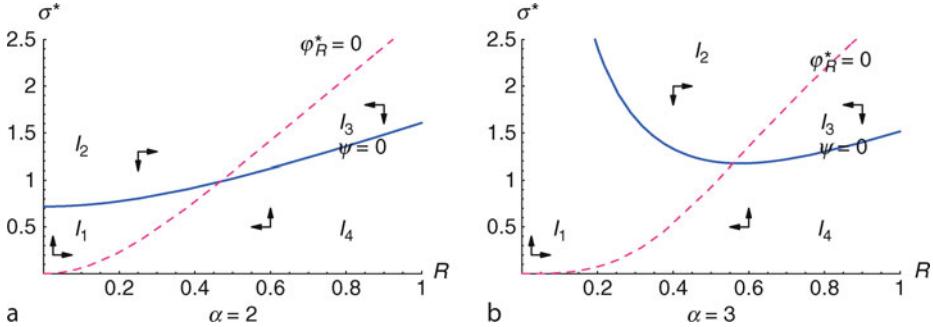
Table 8

Stationary state values of the self-adaptive $(\mu/\mu_i, \lambda)$ -ES on ridge functions. The normalizations are $\sigma^* := \sigma R/N$ with R the distance to the axis, $\varphi_{y_1}^* := \varphi_{y_1} R/N$, and $\varphi_R^* := \varphi_R R/N$

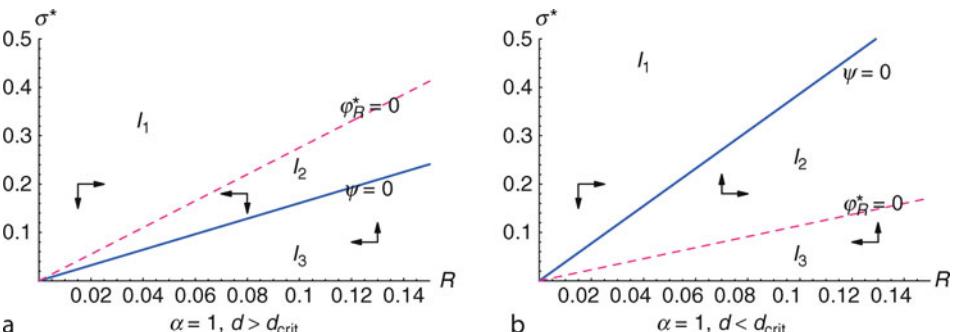
Mutation strength	$\sigma^* = \sqrt{2\mu(1/2 + e_{\mu,\lambda}^{1,1})} \left(\frac{1/2 + e_{\mu,\lambda}^{1,1}}{\alpha^2 d^2 (2\mu c_{\mu/\mu,\lambda}^2 - 1/2 - e_{\mu,\lambda}^{1,1})} \right)^{1/(2\alpha-2)}$
Distance	$R = \left(\frac{1/2 + e_{\mu,\lambda}^{1,1}}{\alpha^2 d^2 (2\mu c_{\mu/\mu,\lambda}^2 - 1/2 - e_{\mu,\lambda}^{1,1})} \right)^{1/(2\alpha-2)}$
Progress	$\varphi_{y_1}^* = \sqrt{(1/2 + e_{\mu,\lambda}^{1,1})(2\mu c_{\mu/\mu,\lambda}^2 - 1/2 - e_{\mu,\lambda}^{1,1})} \left(\frac{1/2 + e_{\mu,\lambda}^{1,1}}{\alpha^2 d^2 (2\mu c_{\mu/\mu,\lambda}^2 - 1/2 - e_{\mu,\lambda}^{1,1})} \right)^{1/(2\alpha-2)}$

Fig. 8

(σ^*, R) -combinations with $\psi = 0$ and $\varphi_R^* = 0$ for (1, 10)-ES with $d = 1$. Shown are the cases for two ridge functions: $\alpha = 2$ in (a) and $\alpha = 3$ in (b). The normalizations are $\sigma^* := \sigma/N$ and $\varphi_R^* := \varphi_R/N$. The intersection indicates the stationary state. The lines divide the (σ^*, R) -phase space into several regions: Region I_1 is characterized by positive expected changes of R , $\Delta R > 0$, and positive changes of σ^* , $\Delta\sigma^* > 0$, I_2 by $\Delta R < 0$, $\Delta\sigma^* > 0$, I_3 by $\Delta R < 0$, $\Delta\sigma^* < 0$, and finally I_4 by $\Delta R > 0$, $\Delta\sigma^* < 0$. The system either leaves every region I_k again, that is, it oscillates, or it converges to the equilibrium point.

**Fig. 9**

(σ^*, R) -combinations for zero progress and zero SAR for (1, 10)-ES on the sharp ridge. The normalizations are $\sigma^* := \sigma/N$ and $\varphi_R^* := \varphi_R/N$. The zero expected change combinations subdivide the (σ^*, R) -phase space. In (a) where $d > d_{\text{crit}}$ region I_1 is characterized by $\Delta R > 0$, $\Delta\sigma^* < 0$, I_2 by $\Delta R < 0$, $\Delta\sigma^* < 0$, and I_3 by $\Delta R < 0$, $\Delta\sigma^* > 0$. Possible movements between the regions are $I_3 \rightarrow I_2$ and $I_1 \rightarrow I_2$. It is easy to see that I_1 and I_3 will be left eventually. The region I_2 cannot be left and the system in σ^* and R approaches the origin. In (b) where $d < d_{\text{crit}}$ region I_1 is characterized by $\Delta R > 0$, $\Delta\sigma^* < 0$, I_2 by $\Delta R > 0$, $\Delta\sigma^* > 0$, and I_3 by $\Delta R < 0$, $\Delta\sigma^* > 0$. Possible movements are $I_1 \rightarrow I_2$ and $I_3 \rightarrow I_2$, but I_2 cannot be left. The system diverges to infinity.



Cumulative Step-Size Adaptation

In Arnold and Beyer (2004), the performance of $(\mu/\mu_b, \lambda)$ -ES on the sphere was investigated for the undisturbed and the noisy case. Concerning the noise-free case, the analysis is aimed at investigating whether the CSA succeeds in adapting a mutation strength that guarantees optimal progress and deriving recommendations for appropriate population sizes.

During an evolutionary run, the ES approaches a steady state: The squared length of the accumulated progress vector, its signed length, and the normalized mutation strength attain a limit distribution. The CSA should strive to increase too small mutation strengths and to decrease too large mutation strengths. In the analysis, the logarithmic adaptation response

$$\Delta_\sigma = \log\left(\frac{\sigma^{(g+1)}}{\sigma^{(g)}}\right) \quad (109)$$

which compares two succeeding mutation strengths is used. The target mutation strength of the CSA is the optimal mutation strength with respect to the progress rate. The ES does not succeed in realizing this objective: Due to the dynamically changing distance to the optimizer, the adaptation of the mutation strength lags behind and the target mutation strength is not reached. The realized normalized mutation strength exceeds the target by a factor of $\sqrt{2}$. In the undisturbed case, the ES with CSA achieves a progress rate of approximately 83% of the optimal progress rate (Arnold and Beyer 2004). Since the optimal mutation strength is not realized, the recommendations (Arnold and Beyer 2002b) for the population sizing with respect to maximal efficiency are slightly to be revised. However, basically the same conclusions as for the sphere can be drawn: The optimal number of offspring is well below the search space dimensionality and increases gradually with it. Furthermore, the recommendation of choosing the truncation ratio about $\mu/\lambda = 0.27$ applies as well.

Arnold (2007) considered the performance of the $(\mu/\mu_b, \lambda)$ -ES on positive quadratic forms (Eq. 73). Being based on the analysis of the expected quality change, the asymptotically optimal mutation strength and the quality gain have been derived as a function of large ξ -values (see Table 9). After a transient phase, the $(\mu/\mu_b, \lambda)$ -ES with CSA yields a stationary mutation strength. This mutation strength and the quality gain depend on the condition number. For $\xi = 1$, the same findings as for the sphere are recovered: The generated mutation strengths are too large by a factor of $\sqrt{2}$. For the asymptotic $\xi \rightarrow \infty$, the ES generates too small mutation strengths (see Table 9). However, the adaptation mechanism still works in that, mutation strengths are realized with result in positive quality gain.

Arnold (2006a) investigated the performance of the $(\mu/\mu_b, \lambda)$ -ES with CSA on the (nearly) complete ridge function class. First, the case of static mutation strengths was considered. After obtaining the axial progress rate and a stationarity condition for the standardized distance to the ridge, optimal values for the progress, distance, and mutation strength can be determined. Optimal refers here to optimal axial progress. The results are given in Table 10 and are applicable for general $\alpha > 2$. As it can be seen, they depend on the value of α . The CSA algorithm would work perfectly on the ridge function class if it succeeded in adopting the optimal mutation strength. This is, however, not the case. An ES using CSA tracks the ridge in unit standardized distance, regardless of α . This in turn results in an α -independent

Table 9

Optimal settings and realized values of the CSA-ES for the quality gain on PDQFs with large condition numbers ξ . The values are normalized quantities $\sigma^* := \sigma N \vartheta / R_1$ and $\Delta^* := \Delta N \vartheta / 2$

Opt. mutation strength	Optimal quality gain	Mut. strength (CSA)	Quality gain (CSA)
$\sigma_{\text{opt}}^* = 2\mu c_{\mu/\mu_b, \lambda}$	$\Delta_{\sigma, \text{opt}}^* = \frac{2\mu c_{\mu/\mu_b, \lambda}^2}{\xi}$	$\sigma_{\text{CSA}}^* = \sqrt{2}\mu c_{\mu/\mu_b, \lambda}$	$\Delta_{\sigma, \text{CSA}}^* = \frac{\mu c_{\mu/\mu_b, \lambda}^2}{\xi}$

Table 10

The $(\mu/\mu_\lambda, \lambda)$ -ES with CSA on the ridge function class. Shown are the values for optimal axial progress and the values achieved by the CSA-mechanism. The following normalizations and abbreviations were used: $\rho = (\alpha d)^{1/(\alpha-1)} R$, $\sigma^* = \sigma N(\alpha d)^{1/(\alpha-1)} / (\mu c_{\mu/\mu_\lambda})$, $\varphi^* = \varphi N(\alpha d)^{1/(\alpha-1)} / (\mu c_{\mu/\mu_\lambda})^2$ (Arnold 2006a)

	Distance	Mutation strength	Progress
Optimal	$\rho = \left(\frac{\alpha}{\alpha - 2} \right)^{\frac{1}{2(\alpha-1)}}$	$\sigma^* = \sqrt{\frac{2\alpha^{\frac{\alpha}{\alpha-1}}}{(\alpha-1)(\alpha-2)^{\frac{1}{\alpha-1}}}}$	$\varphi^* = \frac{\alpha^{\frac{\alpha}{2(\alpha-1)}}(\alpha-2)^{\frac{\alpha-2}{2(\alpha-1)}}}{\alpha-1}$
CSA	$\rho = 1$	$\sigma^* = \sqrt{2}$	$\varphi^* = 1$

normalized mutation strength and progress. For small α , this causes a severe underestimation of the optimal values which gradually diminishes with increasing α .

2.4 Uncertain Environments

This section considers the optimization in uncertain environments, that is, noisy and robust optimization. If noise is involved, the optimizer is often required to show a robustness against perturbations, that is, its quality should decrease only gradually (Beyer and Sendhoff 2006, 2007a; Jin and Branke 2005). Noise is a common problem in practical optimization tasks. Noise may occur for various reasons, for instance, measurement errors or limitations, production tolerances, or incomplete sampling of large search spaces. Noise may also be encountered when optimizing parameters in computer simulations where either numerical errors appear or the simulation technique itself may lead to stochastic results (Arnold and Beyer 2006a; Beyer 2000).

2.4.1 Preliminaries: Accounting for Noise

In the following, several approaches to model noisy fitness measurements are described. Since the noise is modeled by random variable(s) its distribution must capture the main characteristics of real-world situations, for example, an unbiased measurement error. Furthermore, one has to decide how the noise influences the fitness evaluation. For instance, is the noise added afterward, does noise impair the exact localization in the search space or does it even change the functional structure of the fitness function?

Noise Distributions

In order to model practical situations where noise occurs, several noise distributions may be regarded. Each assumes different characteristics of the noise which range from a symmetric distribution without bias over the occurrence of outliers to skewed noise.

The most common noise model is normally distributed noise with zero mean and standard deviation σ_ϵ

$$Z \sim \mathcal{N}(0, \sigma_\epsilon^2), \quad p(z) = \frac{1}{\sqrt{2\pi}\sigma_\epsilon} e^{-\frac{z^2}{2\sigma_\epsilon^2}} \quad (110)$$

The standard deviation σ_ε is usually referred to as the *noise strength*. However, as remarked in Arnold and Beyer (2006a), the Gaussian noise model may not be sufficient to model all practically relevant cases. Not only may the noise be highly non-normal, but also the assumption of a finite variance may be too restrictive. Recommendations obtained for Gaussian noise may not hold, for instance, for noise distributions with heavy tails and frequently occurring outliers. An alternative to using Gaussian noise is to assume Cauchy distributed noise. Its pdf and cdf read

$$p(z) = \frac{\sigma_\varepsilon}{\pi(z^2 + \sigma_\varepsilon^2)} \quad (111)$$

$$P(z) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{z}{\sigma_\varepsilon}\right) \quad (112)$$

The Cauchy distribution has usually two parameters, the *location parameter*, set to zero in [Eqs. 111](#) and [112](#), and the *scale parameter* σ_ε . The Cauchy distribution has a far heavier tail than the normal distribution and can therefore be used as a model for noise with frequently occurring outliers. All moments, including mean and variance, are undefined. The median, however, exists and is equal to the location parameter. Cauchy distributed noise is symmetric around the location parameter.

To model biased and skewed noise, the χ_1^2 distribution may be considered. The χ_1^2 distribution appears when taking the square of a normally distributed random variable. The pdf is

$$p(z) = \frac{e^{-\frac{z}{2\sigma_\varepsilon}}}{\sqrt{2\pi\sigma_\varepsilon} z} \text{ for } z \geq 0 \quad (113)$$

Since the distribution is not symmetric, it matters whether the random term is added or subtracted to the fitness function, or in other words, whether the noise term itself or its negative is χ_1^2 -distributed.

Noise Models

The *standard noise model* assumes that noise is added to the result of the fitness evaluation

$$F_\varepsilon(\mathbf{x}) = F(\mathbf{x}) + z(\varepsilon, g) \quad (114)$$

where the noise term, $z(\varepsilon, g)$, may follow one of the distributions above. As the assumption of Gaussian noise, the additive noise model may be insufficient to capture all characteristics of real-word situations.

The additive noise model may be regarded as a special case of the *general noise model*

$$F_\varepsilon(\mathbf{x}) = F(\mathbf{x}, z(\varepsilon, g)) \quad (115)$$

In this case, the noise may directly influence the fitness function itself. As a result, the fitness function has to be interpreted as a random function. The approach provides more flexibility. For any computations, however, assumptions must be made.

A special case of [Eq. 115](#) is *systematic noise* (or *actuator noise*) which refers to noise on the coordinates of the objective

$$F_\varepsilon(\mathbf{x}) = F(\mathbf{x} + \mathbf{z}(\varepsilon, g)) \quad (116)$$

(Beyer and Sendhoff 2007b). Actuator noise is often considered in the context of *robust optimization*. Robustness means that the obtained solution is relatively stable with respect to perturbations. Several measures can be introduced to evaluate the quality of a solution. In the context of robust optimization, several types of probabilistic criteria can be distinguished, for example: *threshold measures* and *statistical momentum measures*. The former require that the probability of F -realizations below a certain threshold is maximal

$$P(\{F \leq q|\mathbf{y}\}) \rightarrow \max \quad (117)$$

if minimization is considered. The *statistical momentum measures*

$$\mathbb{E}[F^k|\mathbf{y}] \rightarrow \min \quad (118)$$

require minimality with respect to the k th moment (see, e.g., Beyer and Sendhoff 2006, 2007a).

2.4.2 Noise and Progress

In the following, some results from analyses of the influence of noise on the local performance starting with the sphere model are presented. The first task is usually to obtain expressions for the progress rate or the quality gain, which can be used, for instance, to derive necessary or sufficient convergence criteria. Further points include examining the effects of recombination, the efficiency of resampling compared to operating with larger offspring populations, and determining appropriate population sizes.

Sphere Model and Linear Functions

The effects of noise on the progress of an ES were analyzed in the following publications

- Beyer (1993) considered the performance of the $(1, \lambda)$ -ES and the $(1 + \lambda)$ -ES.
- Arnold and Beyer (2002a) analyzed the effects of overvaluation in the $(1 + 1)$ -ES.
- In Arnold and Beyer (2001b), the local performance of the $(\mu / \mu_b, \lambda)$ -ES in infinite-dimensional search spaces was investigated.
- Arnold and Beyer (2002b) extended the analysis of Arnold and Beyer (2001b) to an analysis in N -dimensional search spaces.
- Arnold (2002a) and Arnold and Beyer (2001a, 2003a) considered the behavior of the (μ, λ) -ES.
- In Arnold and Beyer (2006a), several noise distributions were considered. Deriving the progress rates provides a means to investigate the implications of different distributions on the behavior of the ES.

In Beyer (1993), the influence of noise on the performance of evolution strategies with only one parent was analyzed. Two strategies were considered: the $(1, \lambda)$ -ES and the $(1 + \lambda)$ -ES. The first task was to obtain the progress rate (see  Table 11). The analysis used again, the decomposition of the mutation vectors and applied order statistics to derive the final expressions. Due to the noise, induced order statistics, also referred to as noisy order statistics, were used (see Arnold 2002a). Instead of considering the fitness change induced by mutation, the perceived fitness change had to be taken into account. Therefore, two random processes: mutation and noise have to be considered.

The progress rate of the $(1, \lambda)$ -ES consists again of an approximately linear gain part and a quadratic loss part (cf. evolutionary progress principle (EPP)). Only the gain part is

Table 11

Asymptotical progress rates for the ES on the noisy sphere. Here, noise type *standard* refers to the additive noise model (☞ 114) whereas *systematic* is described by ☞ Eq. 116

ES	Noise type	Progress rate
(1, λ)-ES	Standard	$\varphi_R^* = \frac{c_{1,\lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_e^{*2}}} - \frac{\sigma^{*2}}{2}$
(1 + 1)-ES	Standard	$\varphi_R^* = \frac{\sigma^*}{\sqrt{2\pi}} \cdot \frac{\exp\left[-\frac{1}{2}\left(\frac{\sigma^*/2}{\sqrt{1+2\left(\frac{\sigma_e^*}{\sigma^*}\right)^2}}\right)\right]}{\sqrt{1+2\left(\frac{\sigma_e^*}{\sigma^*}\right)^2}}$ $- \frac{\sigma^{*2}}{2} \left(\left(1 - \Phi\left(\frac{\frac{\sigma^*}{2}}{\sqrt{1+2\left(\frac{\sigma_e^*}{\sigma^*}\right)^2}}\right) \right) \right)$
(μ/μ_r , λ)-ES	Standard	$\varphi_R^* = \frac{c_{\mu/\mu_r, \lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_e^{*2}}} - \frac{\sigma^{*2}}{2\mu}$
(μ/μ_r , λ)-ES	Standard	$\varphi_R^* = \frac{c_{\mu/\mu_r, \lambda} \sigma^{*2} \left(1 + \frac{\sigma^{*2}}{2\mu N}\right)}{\sqrt{1 + \frac{\sigma^{*2}}{\mu N}} \sqrt{\sigma^{*2} + \sigma_e^{*2} + \frac{\sigma^{*4}}{2N}}} - N\left(\sqrt{1 + \frac{\sigma^{*2}}{2\mu N}} - 1\right)$
(μ/μ_r , λ)-ES	Systematic	$\varphi_R^* = \frac{c_{\mu/\mu_r, \lambda} \sigma^{*2} \left(1 + \frac{\sigma^{*2}}{2\mu N}\right)}{\sqrt{1 + \frac{\sigma^{*2}}{\mu N}} \sqrt{\sigma^{*2} + \sigma_e^{*2} + \frac{(\sigma^{*2} + \sigma_e^{*2})^2}{2N}}} - N\left(\sqrt{1 + \frac{\sigma^{*2}}{2\mu N}} - 1\right)$

influenced by the noise. Noise decreases the gain part which can be measured by the so-called selection sharpness

$$S := \frac{1}{\sqrt{1 + (\sigma_e^*/\sigma^*)^2}} \quad (119)$$

where σ_e^* is the normalized noise strength which reads for the sphere model (☞ 71)

$$\sigma_e^* = \frac{N\sigma_e}{R \left| \frac{dg}{dR} \right|} \quad (120)$$

For “comma” strategies, there is a certain critical noise strength above which any positive mutation strength causes negative progress and expected divergence. The transition at which this appears is determined by the necessary evolution criterion. Below the critical noise strength, there are regions of (σ^*, σ_e^*) -combinations which allow positive progress. Clearly, the border is defined by zero progress rate. The critical noise strength can be used to determine the *residual location error*, the stationary expected distance to the optimizer. The residual location error depends on σ and σ_e . The mutation strength σ can generally be obtained only by analyzing the step-size adaptation mechanism.

Since the necessary evolution criterion depends on the size of the population, one way to increase the convergence region of the $(1, \lambda)$ -ES would be to increase λ . It can be shown, however, that this is not very efficient. Another way to cope with noise is to reduce the noise

strength itself which can be done by m -times resampling and averaging (provided that the statistical moments of the noise exist). The analysis revealed that except for small values of λ and m , a $(1, \lambda)$ -ES with a lower population size but with repeated resampling and averaging might be preferable.

In the case of the $(1 + \lambda)$ -ES, the perceived fitness of the apparent best offspring is compared with the perceived fitness of the parent. In Beyer (1993), it was assumed that the parent is evaluated anew when comparing it with the offspring. To determine the progress rate, the apparent or perceived success probability had to be obtained. The calculations led to an asymptotical progress rate, which revealed that in contrast to the non-noisy case, the expected progress can be negative – depending on the noise and mutation strength. Due to the nature of the integrals involved, the numerical determination of the progress rate is cumbersome. For the special case of $(1 + 1)$ -ES, the calculations could be performed and analytical results obtained. Interestingly, the perceived success probability increases with the noise strength. That is, the offspring increasingly replaces the parent – without being necessarily better. This can be counteracted to some extent with an increase of the mutation strength – albeit only to a certain point. Depending on the normalized noise strength, there is an optimal success probability that guarantees optimal progress. However, this is only possible up to a certain noise limit. The success probabilities obtained lead to the conclusion that the 1/5th-rule with the standard settings should be applied. Instead of lowering the success probability for noise closer to the limit value, resampling and averaging should be applied to decrease the noise strength (Beyer 2001b, p. 99).

Beyer (1993) considered the case that the fitness of the parent is reevaluated every time it is compared to the offspring. This results in more fitness evaluations than usually necessary. Since fitness evaluations are costly, the question arises whether the $(1 + 1)$ -ES benefits from reevaluations. Arnold and Beyer (2002a) investigated the local performance of the $(1 + 1)$ -ES on the sphere without reevaluation. If the parent is not evaluated anew, the fitness of the parent is systematically overvalued. Accounting for this effect makes the analysis more difficult than before. The degree of overvaluation Ξ of the parent is defined as the difference between the ideal, true fitness, and the perceived fitness. Likewise, ξ denotes the degree of overvaluation of the offspring. For the offspring, ξ is normally distributed. In the case of the parent, selection influences the distribution. Obtaining this distribution proved to be quite difficult; therefore, a Gram–Charlier expansion was used. This also required the determination of several moments of the unknown series. Following Beyer (1995b), self-consistency conditions were imposed, that is, it was assumed that a time-invariant distribution is approached and that the moments do not change over time. Investigations revealed that cutting off the series after the second term, thus, assuming a normal distribution, yields already good results. The remaining task consisted in determining the mean and variance of the distribution of the parent's degree of overvaluation.

The analysis revealed that in the case of the $(1 + 1)$ -ES with reevaluation, noise increases the probability of an offspring replacing the parent. Without reevaluation, noise has the opposite effect. As a result, an ES with reevaluation shows negative quality gains for a wide range of the mutation strength once the noise strength is above a limit. Without reevaluation, the quality gain remains positive.

Comparing the two strategies using their respective optimal mutation strength showed that the ES with reevaluation never outperforms the ES without reevaluation. Its performance is even inferior for noise larger than a limit value. This can be traced back to the reduced success probabilities for ESs without reevaluation, which are caused by the systematic

overvaluation of the parent. Since the parent is overvaluated, an offspring is only accepted if the perceived fitness gain is relatively large. In this case, the perceived better fitness has a good chance to be actually better. The ES is less biased to accept offspring, which have a truly smaller fitness than the parent. However, overvaluation also means that many offspring with true but small fitness improvements may not be accepted.

The question remained what degree of overvaluation may be useful provided that it could be calibrated. The analysis in Arnold and Beyer (2002a) showed that an ES may profit from occasional reevaluations. The frequency of reevaluation, however, depends on the mutation strength and therefore on the step-size adaptation mechanism. Experiments with the 1/5th-rule revealed that this mechanism is not suited for a $(1 + 1)$ -ES without reevaluation, if the noise strengths are larger than a limit: The 1/5th-rule tries to achieve a certain success probability. If the success probabilities are too small, the rule reduces the mutation strength. If the parent is not reevaluated, the target success probability is simply not reachable once the noise is too large. The ES reduces the mutation strength permanently and the search stagnates.

So far the discussion has been on strategies with only one parent. Arnold and Beyer (2001b) considered the performance of the $(\mu/\mu_b, \lambda)$ -ES in infinite-dimensional search spaces. The analysis led to an evolution criterion, that is, a maximal noise strength for a nonnegative quality gain, which depends on the population parameters. Keeping the truncation ratio fixed, it can be seen that the effects of any noise strength can be counteracted by increasing the population size. A further result was that the $(\mu/\mu_b, \lambda)$ -ES showed an improved performance in comparison with the $(1, \lambda)$ -ES. This can be traced back to the genetic repair effect and the resulting larger mutation strengths. Larger mutation strengths decrease the noise-to-signal ratio, which improves the progress.

Arnold and Beyer (2002b) carried the analysis over to finite-dimensional search spaces and derived an asymptotically correct expression for the progress rate (see Table 11). This could be used to give more accurate estimates for the (σ^*, σ_e^*) -combinations guaranteeing positive progress. It was found that increasing the number of parents widens the convergence region. The efficiency $\eta = \max_{\sigma^*} \varphi^*(\sigma^*)/\lambda$ is also influenced by the parent number with an optimal efficiency for a truncation ratio of 0.27 (Arnold and Beyer 2002b). In contrast to the finding in Arnold and Beyer (2001b), the efficiency does not increase indefinitely with the number of offspring. Using optimal values for the mutation strength and parent number, it was shown that an optimal number of offspring exists which depends on the noise strength and search space dimensionality.

In the case of the undisturbed sphere, the $(1 + 1)$ -ES is the most efficient of the standard types of evolution strategies. Using common multi-parent strategies does not have any advantage unless the algorithm makes use of parallel computation. This changes in the presence of noise. The efficiency of multi-parent strategies exceeds the efficiency of point-based strategies – except for low noise levels. On the one hand, this can be traced back to the genetic repair effect, which is caused by intermediate recombination. On the other hand, the question remains whether evolution strategies without recombination, but with populations, may also achieve a better performance than the $(1 + 1)$ -ES. Experiments had already shown that retaining more than the best candidate solution can lead to an improved performance.

In Arnold (2002a) and Arnold and Beyer (2001a, 2003a) the behavior of the (μ, λ) -ES was analyzed in more detail. As in the non-noisy case, the determination of the progress rate or quality gain is a demanding task. The distribution of the apparently μ best offspring has to be derived or approximated. This not only requires the use of noisy order statistics, but also

the treatment of the parental states as random variables. The time-variant distributions of succeeding populations under the influence of mutation, noise, and selection have to be taken into account and modeled. Usually, however, as in the non-noisy case, the assumption can be made that a time-invariant limiting distribution is approached. The ES populations are samples drawn from this distribution. Therefore, the moments of the population also approach a limit distribution.

In Arnold (2002a) a first moment-based analysis was presented. In a first step, infinite noise strength was assumed. In this case, selection becomes purely random and can be neglected. This allows for an exact determination of the expectation of central moments. Extending the analysis to the case of finite noise, the determination of the joint distributions of noisy order statistics is necessary, which complicates the calculations. An approximation of the density of the offspring distribution from which the candidates are drawn can be obtained using the Gram–Charlier series. This yields the expectation of the central moments of the population at generation $g + 1$ as a function of this expansion and the moments at generation g . The latter are of course random variables. In Arnold (2002a), a simple approach was followed by ignoring all fluctuations of these terms and considering only their expected values. This leads to a quality gain expression for linear fitness functions which agrees well with the result of experiments.

Arnold and Beyer (2001a, 2003a) analyzed the (μ, λ) -ES on a noisy linear function and the sphere model with respect to whether the use of populations is beneficial and how this comes to pass. Therefore, the (μ, λ) -ES was compared to the $(\mu/\mu_0, \lambda)$ -ES. The analysis followed the approach introduced in Beyer (1995b). The distribution of the offspring cannot be determined exactly. Therefore, it was expanded into derivatives of a normal distribution. To this end, the moments had to be obtained. The approach presented in Arnold (2002a) differs from Beyer (1995b). Instead of considering a skewed distribution, it was assumed that the derived random variables are normally distributed. Therefore, only the mean and variance of the population have been calculated using the respective values of the parent population and taking the effect of mutation into account.

The linear function, $f(\mathbf{y}) = \mathbf{a}^T \mathbf{y}$, effectively projects candidate solutions onto a line defined by the direction of \mathbf{a} . The problem is, therefore, basically one-dimensional and the quality gain and progress rate can be defined straightforwardly. In analogy to the axial progress rate for ridge functions, the progress rate measures the expected change of the position. First, a general expression for the mean and variance of the offspring population must be derived. Due to the assumption that the distribution of the offspring is normal, a result obtained in Arnold (2002a) for noisy order statistics can be directly applied, leading to the quality gain and the progress rate. Both depend on the progress coefficient

$$c_{\mu/\lambda}(\theta) = \frac{1 + \kappa_2}{\sqrt{1 + \kappa_2 + \theta^2}} e_{\mu,\lambda}^{1,0} \quad (121)$$

with $\theta = \sigma_e/\sigma$ the noise-to-signal ratio and $\kappa_2 = D^2/\sigma^2$. The parameter D^2 denotes the variance of the parent population. The result can be used to discuss the effects of operating with a population of $\mu > 1$. Without noise, the increase of the population variance with its size μ is counterbalanced by the decrease of the progress coefficient $e_{\mu,\lambda}^{1,0}$. Noise increases the population variance additionally. Furthermore, evolution strategies with populations operate with a different noise-to-signal ratio. While this value is $\theta = \sigma_e/\sigma$ for the $(1, \lambda)$ -ES, it is $\theta/\sqrt{1 + \kappa_2}$ for the (μ, λ) -ES. That is, (μ, λ) -ESs algorithmically decrease the noise-to-signal ratio.

As for the calculation of the parental population variance and mean, the approach of Beyer (1995b) was applied. This led to fix-point or self-consistent equations to be solved. The result showed that the population variance increases with the size of the population and with the noise. A comparison with experiments showed that the normality assumption for the offspring distribution is usually not sufficient for higher noise strengths and population sizes.

A similar approach can be applied to the sphere model using the decomposition technique for the mutation vectors. The analysis is based on the assumption that the distance to the objective exceeds the population variance. To this end, large search space dimensionalities and population sizes μ , not too large, must be assumed in conjunction with a normal approximation of the offspring distribution. Experiments showed that the resulting progress rate formula is an accurate predictor.

The progress rate can be used to compare several strategies using the efficiency as comparison measure. It was found that for each λ and noise strength σ_e^* an optimal μ exists, which is usually between 0.1λ and 0.3λ . Using this optimal μ , the efficiency of the $(\mu_{\text{opt}}, \lambda)$ -ES and the $(1, \lambda)$ -ES can be compared. The efficiency of the $(\mu_{\text{opt}}, \lambda)$ -ES is greater than that of the $(1, \lambda)$ -ES except for small noise strengths. Retaining more than one offspring enables the ES to operate with smaller offspring populations in the presence of noise. Considering a $(1 + 1)$ -ES showed that the single point strategy is only superior for very small normalized noise strengths and shows worse performance for larger noise strengths.

Regarding the efficiency of the (μ, λ) and the $(\mu/\mu_b, \lambda)$ -ES, it can be stated that both strategies are able to operate with a reduced noise-to-signal ratio. The efficiency formulas read

$$\eta_{\mu, \lambda} = \frac{1}{\lambda} \left(\frac{1 + \kappa_2}{\sqrt{1 + \kappa_2 + \theta^2}} \sigma^* e_{\mu, \lambda}^{1,0} - \frac{\sigma^{*2}}{2} \right) \quad (122)$$

$$\eta_{\mu/\mu_b, \lambda} = \frac{1}{\lambda} \left(\frac{\sigma^* e_{\mu, \lambda}^{1,0}}{\sqrt{1 + \theta^2}} - \frac{\sigma^{*2}}{2\mu} \right) \quad (123)$$

The mechanisms by which the improved efficiency is obtained differ: The (μ, λ) -ES reduces the noise-to-signal ratio by an increase of the population variance which enhances the effect of the mutations. The $(\mu/\mu_b, \lambda)$ -ES reduces the ratio by operating with higher mutation strengths made possible by means of the genetic repair effect.

Arnold (2006b) investigated the (λ_{opt}) -ES on the noisy sphere. This ES-type uses a weighted multi-recombination taking the whole offspring population into account. Optimal weights $w_{k, \lambda}$ are given by the expected value of the $(\lambda + 1 - k)$ th order statistic of the standard normal distribution. It was shown that the (λ_{opt}) -ES does not benefit from the genetic repair effect. However, a rescaling of the weights $w_{k, \lambda} \rightarrow w_{k, \lambda}/\kappa$, with $\kappa > 1$ can be applied. This leaves the optimal quality gain unchanged in the noise-free case (for $N \rightarrow \infty$). The optimal mutation strength, however, increases with the scaling factor κ . An increase of the mutation strength lowers the noise-to-signal ratio. Obtaining the quality gain for the (λ_{opt}) -ES on the noisy sphere, it could be shown that the ES is capable of maintaining a positive quality change up to a normalized noise strength of $\sigma_e^* = 2\kappa$. The analysis indicated that large values of κ should be preferred.

Arnold and Beyer (2003b) analyzed the effects of outliers on the outcome of optimization. To simulate a noise distribution with frequently occurring outliers, the Cauchy distribution was chosen. The analysis was extended in Arnold and Beyer (2006a). Three noise distributions were compared for the $(\mu/\mu_b, \lambda)$ -ES: normally distributed noise, Cauchy noise, and χ_1^2

distributed noise. In the latter case, two different cases were considered: adding the noise term to the fitness function, which results in a random variable with a mean greater than the actual value, and subtracting the noise term leading to a random variable with a smaller mean.

One aim was to investigate whether the assumption of a Gaussian noise model is too restrictive to allow a transfer of the obtained results to other situations. For each model, the first task was to derive the progress rate. The results can be used to investigate:

- The effects of varying noise levels
- The influence of population sizing, the truncation ratio, and the number of offspring
- The efficiency of reevaluation/resampling of search points compared to upgrading offspring population sizes

and to compare the results with respect to transferability from one noise model to another.

As shown in Arnold and Beyer (2006a), Cauchy noise leads to similar qualitative responses of the ES as Gaussian noise. It should be noted that in the case of Cauchy noise, the role of the noise strength is different. Instead of being the standard deviation of the noise, it should be regarded as a scale parameter. (Actually, this interpretation can also be applied to the Gaussian noise case, thus, allowing for a direct comparison of the performances.) Therefore, a comparison of the scaling behavior has been made (Arnold and Beyer 2006a). As for Gaussian noise, progress is only possible up to a maximal normalized noise strength. This quantity depends on the truncation ratio and on the size of the offspring population. Strategies with a ratio of 0.5 achieve the best results. Increasing offspring population sizes appears to increase the maximal admissible noise strength. To achieve optimal performance, the truncation ratio has to be variated with respect to the noise level starting with 0.27 for zero noise and going up to 0.5 for the upper noise limit. The difference between Gaussian and Cauchy noise lies only in the shape of the response curves.

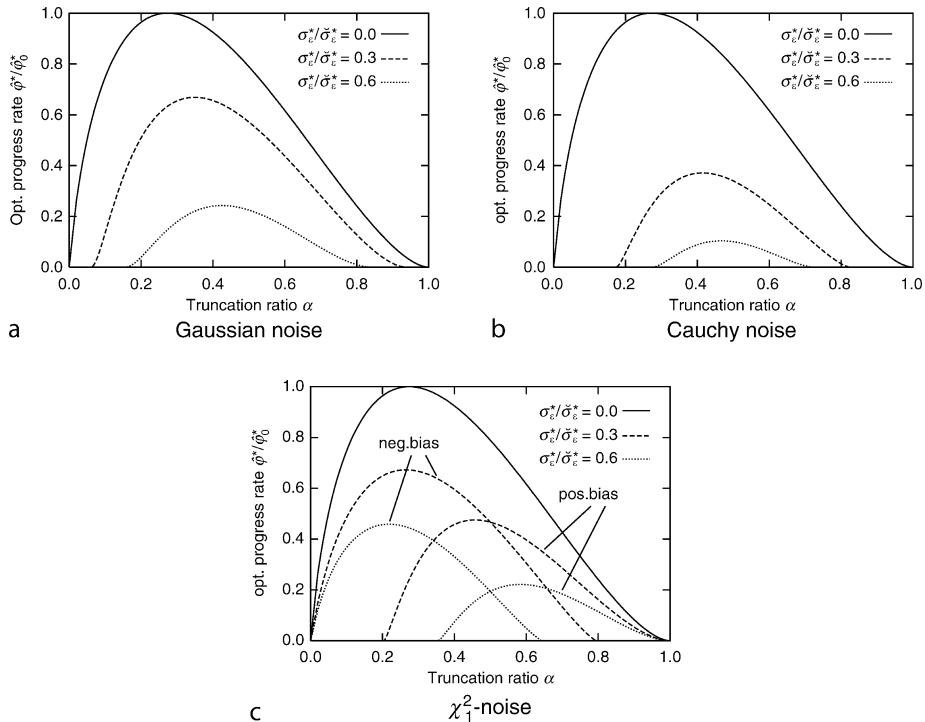
The question remains for which ranges of the truncation ratios positive progress occurs. For both distributions, the range narrows with increasing noise levels to smaller and smaller intervals, which encompass 0.5 (see Fig. 10a, b). The similarities between the two noise models transfers to the question whether resampling is more efficient than using an ES with a larger offspring population. For both noise models, this can be negated. Comparing ESs with the same computational costs and varying resampling sizes showed that larger populations have to be preferred. The difference between the approaches is more pronounced for Cauchy noise, however (Arnold and Beyer 2006a). For Gaussian noise, the ES benefits from averaging over the reevaluations of search points since the variance of the noise is reduced. For Cauchy noise, averaging has no effect since the resulting distribution is exactly the same.

One major difference between Gaussian and Cauchy noise exists. This is caused by a different response toward vanishing noise-to-signal ratios. In the case of Gaussian noise, the effects of the noise may be nearly eliminated, if the population size and the mutation strength are sufficiently large. In the case of Cauchy noise, this does not hold anymore. Increasing the population sizes will not allow the strategy to operate as if noise were not present.

Noise with a χ_1^2 -distribution leads to two possible cases: adding (positively biased noise) or subtracting the random term (negatively biased noise). The ES accordingly responds in different ways. First of all, increasing the offspring population size still increases the limit noise strength. However, the first difference appears in the optimal truncation ratio and its response to varying noise levels (see Fig. 10c). For neither case of χ_1^2 -distributed noise, does the truncation ratio approach 0.5. As for Cauchy and Gaussian noise, increasing noise narrows the band of truncation ratios for which positive progress is possible. Unlike the previous cases, however,

Fig. 10

The progress rate $\hat{\varphi}^*$ for optimal mutation strengths as a function of the truncation ratio α (Arnold and Beyer 2006a). The y-axis is scaled with the maximal progress rate $\hat{\varphi}_0^*$ obtained for the noise-free case and a truncation ratio of 0.27. Shown are the curves for three different noise strength σ_e^* to limit noise strength $\check{\sigma}_e^*$ ratios. Figure (a) depicts the case of Gaussian noise, (b) the case of Cauchy noise, and (c) shows the case of a χ_1^2 distributed noise.



only one boundary of the interval moves: Positively biased noise extends upward to a ratio of 1.0, whereas negatively biased noise stretches downward to zero (Arnold and Beyer 2006a).

Comparing the efficiency of strategies with resampling and strategies with larger population sizes leads to the same finding as before: An increased population size is preferable. Noise with a χ_1^2 -distribution has a similarity with Gaussian noise: The effects of noise can be eliminated with vanishing noise-to-signal ratio.

General Quadratic Functions and Other Functions

In Beyer and Meyer-Nieberg (2004), the quality gain of the $(1, \lambda)$ -ES was derived under the assumption of additive, normally distributed noise. The approach used the concept of noisy order statistics and a Gram–Charlier expansion for the pdf of the fitness values. The result

$$\overline{AQ_{1,\lambda}} = \frac{S_Q^2}{\sqrt{S_Q^2 + \sigma_e^2}} c_{1,\lambda} + M_Q \quad (124)$$

with mean M_Q and standard deviation S_Q of the local quality change, that is, of the fitness change induced by a mutation – strongly resembles the progress rate equation for the noisy

sphere. Instead of the mutation strength, the standard deviation of the local quality change appears. Noise only influences the first part in [Eq. 124](#) and not the mean M_Q . A sufficient evolution criterion is obtained by demanding that $\overline{\Delta Q_{1,\lambda}} \geq 0$, leading to $S_Q^2 / \sqrt{S_Q^2 + \sigma_\varepsilon^2} c_{1,\lambda} \geq -M_Q$ (Beyer and Meyer-Nieberg [2004](#)).

The remaining task for this kind of analysis is to determine the mean and standard deviation for the respective fitness function. This has been done for general quadratic, biquadratic, the L_1 -norm, and – to test the ranges of the applicability – for the bit-counting function OneMax (see [Table 12](#)). For all these functions, the mean of the fitness corresponds with a negative, nonlinear loss term.

Ridge Functions

As in the undisturbed case, the ES is faced with two subgoals in the optimization process. Accordingly, there are two variables that have to be considered: The position on the y_1 axis and the distance R to the y_1 axis. In the long run, the latter (usually) approaches a stationary distribution, whereas the y_1 -evolution continues to progress. The question remains how noise influences the stationary distribution of the distance and what the consequences are for the axial progress. The first question is similar to determining the localization error for the sphere model. The result, however, determines the axial progress, that is, the performance of the ES.

Table 12

Quality change formulae for different fitness environments. Explanation of variables and notation: $c_{\mu/\mu,\lambda}$ progress coefficient ([Eq. 132](#), see Beyer [\(2001b, p. 172\)](#)), σ^* normalized mutation strength, σ_ε^* standard deviation of noise distribution (normalized), higher order progress coefficients $d_{1,\lambda}^{(k)}$ ([Eq. 129](#)) or see Beyer [\(2001b p. 119\)](#), S_Q and M_Q standard deviation and mean of the mutation-induced fitness change Q , κ_k k th cumulant. The normalizations are as follows:

(a) **sphere** $\sigma^* = \sigma N/R$, $\sigma_\varepsilon^* = \sigma_\varepsilon N/R$, $\overline{\Delta Q^*} = \overline{\Delta QN}/(Rg'(R))$ (b) **quadratic functions**

$\sigma^* = \sigma \text{Tr}[Q]/\|Q(y - \hat{y})\|$, $\sigma_\varepsilon^* = \sigma_\varepsilon \text{Tr}[Q]/(2\|Q(y - \hat{y})\|^2)$, $\overline{\Delta Q^*} = \overline{\Delta Q}/(2\|Q(y - \hat{y})\|^2)$ (c) **biquadratic functions**

$\sigma^* = \sigma^3 \sqrt{3 \sum_i c_i / \|a\|^4}$, $\sigma_\varepsilon^* = \sigma_\varepsilon^3 \sqrt{3 \sum_i c_i / \|a\|^4}$, $\overline{\Delta Q^*} = \overline{\Delta Q} \sqrt[3]{3 \sum_i c_i / \|a\|^4}$, and

(d) **OneMax** $\sigma^* = \sqrt{lp_m} 2(2F_0/I - 1)$, $\sigma_\varepsilon^* = \sigma_\varepsilon 2(2F_0/I - 1)$, $\overline{\Delta Q^*} = \overline{\Delta Q} 2(2F_0/I - 1)$ with F_0 the fitness of the parent, p_m the bit-flipping probability, and I the string length

Fitness	ES	Quality gain
General functions	$(1, \lambda)$ -ES	$\overline{\Delta Q_{1,\lambda}} = \frac{c_{1,\lambda} S_Q}{\sqrt{S_Q^2 + \sigma_\varepsilon^2}} + M_Q$
Quadratic functions	$(1, \lambda)$ -ES	$\overline{\Delta Q^*} = \frac{c_{1,\lambda} \sigma^*}{\sqrt{\sigma^{*2} + \sigma_\varepsilon^{*2}}} - \frac{\sigma^{*2}}{2}$
Biquadratic functions	$(1, \lambda)$ -ES	$\overline{\Delta Q^*} = \frac{c_{1,\lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_\varepsilon^{*2}}} - \sigma^{*4}$
OneMax	$(1, \lambda)$ -ES	$\overline{\Delta Q^*} = \frac{c_{1,\lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_\varepsilon^{*2}}} - \frac{\sigma^{*2}}{2}$
Sphere	$(\mu/\mu, \lambda)$ -ES	$\overline{\Delta Q^*} = \frac{c_{\mu/\mu,\lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_\varepsilon^{*2}}} - \frac{\sigma^{*2}}{2\mu}$

Arnold and Beyer (2008) analyzed evolution strategies on the noisy parabolic ridge. The noise model considered was an additive, normally distributed noise model with different functional dependencies of the noise strength: constant noise strength, noise strength scaling quadratically with the distance, and noise scaling cubically with the distance.

Using the same decomposition of the progress vector as in the non-noisy case, the axial progress rate could be obtained (see \blacktriangleright Table 13). Instead of determining the radial progress rate, the expected change of the square of the distance was determined leading to an expression for the standardized distance to the ridge which is a polynomial of order four and influenced by the noise strength and the mutation strength.

Let one first consider the case of constant noise strength. In this case, increasing the mutation strength increases the performance as in the non-noisy case. Large mutation strengths achieve a better noise-to-signal ratio $\theta = \sigma_e/\sigma$. Increasing the mutation strength eventually eliminates virtually the influence of noise on the progress. Furthermore, the ES achieves a stationary distance of $R_\infty = \sigma N / (\mu c_{\mu/\mu,\lambda})$ and accordingly an axial progress rate of nearly $\varphi_{\max} = \mu c_{\mu/\mu,\lambda}^2 / (dN)$. By increasing μ and λ , the location error to the axis can be decreased – if σ is constant. This is coupled with an increase of the performance. As on the sphere, it is optimal to choose a truncation ratio of $\mu / \lambda = 0.27$, since this maximizes $\mu c_{\mu/\mu,\lambda}^2$ (Arnold and Beyer 2008).

For quadratic noise strengths, an increase of the mutation strength is also beneficial and connected with an increased performance. While for the constant noise case the increase of the mutation strength eventually enabled the ES to reach nearly the same progress as in the undisturbed case, this is not the case for quadratic noise strengths. Increasing the mutation strength also increases the distance to the axis. Since the noise scales quadratically, the noise-to-signal ratio cannot be driven to zero by an increase of the mutation strength. Instead, a positive limit value is reached and the progress rate approaches a saturation value. The maximal achievable progress rate decreases with the noise. Furthermore, positive progress is only possible up to a certain noise level $\mu c_{\mu/\mu,\lambda} / (dN)$, with d the scaling parameter of the squared distance in \blacktriangleright Eq. 76. The noise level depends on the ES.

The case of cubic noise strengths is more involved. The stationary condition leads to two potential stationary solutions. For small mutation strengths, the smaller solution represents a stable stationary point, whereas the larger is unstable. Increasing the mutation strength increases the stable solution until it finally coincides with the unstable point. Thus, for

Table 13

Asymptotical axial progress rates for the noisy parabolic ridge in the stationary case. The normalizations read $\varphi_{y_1}^* := \varphi_{y_1} dN / (\mu c_{\mu/\mu,\lambda}^2)$, $\sigma^* := \sigma dN / (\mu c_{\mu/\mu,\lambda})$, and $\sigma_e^* := \sigma_e dN / (\mu c_{\mu/\mu,\lambda})$ for constant noise. In the case of quadratic noise, $\sigma_e^* = \zeta \rho^2$, $\zeta^* := \zeta dN / (\mu c_{\mu/\mu,\lambda})$ with ρ the standardized distance to the axis $\rho := 2Rd$

Noisy type	Progress rate
Constant noise	$\varphi_{y_1}^* = \frac{\sigma^{*2}}{\frac{\sigma^{*2}}{2} + \sqrt{\frac{\sigma^{*4}}{4} + \sigma^{*2} + \sigma_e^{*2}}}$
Quadratic noise	$\varphi_{y_1}^* = \frac{\sigma^*(1 - \zeta^{*2})}{\frac{\sigma^{*2}}{2} + \sqrt{\frac{\sigma^{*4}}{4} + 1 - \zeta^{*2}}}$

mutation strengths above a noise-dependent limit value, no (stable) stationary solution exists. The ES diverges from the axis. The case of cubic noise is interesting, since it results in a new behavior of the noise-to-signal ratio. Here, increasing the mutation strength increases the ratio indefinitely. The influence of noise cannot be counteracted with an increase of the mutations strength as usual, but only with a decrease: the larger the noise, the smaller the mutation strength has to be. Depending on the noise, an increase of the mutation strength is only beneficial up to a certain limit. This limit decreases with the noise.

2.4.3 Residual Location Error and Fitness Error

Noise does not only influence the local performance of an ES but also the final solution quality. Provided that the noise strength is constant, the ES is unable to locate the optimizer. Instead, it typically converges to a time-invariant limit distribution. On an average, a residual location error, the expected distance to the optimizer and a residual fitness error, the expected difference between outcome and the optimum, are observed. In the following, how the local progress measures may be used to derive these values is described. The expressions can then be used to derive recommendations for population sizing with respect to the solution quality. As it will be shown, optimal settings with respect to the solution quality usually correspond to nonoptimal progress. Therefore, a compromise between convergence speed and solution quality must be made.

Sphere Model

The analyses for the sphere model aim to derive lower bounds for the residual location and fitness error and to investigate the influence of the truncation ratio. An important tool is the evolution criterion gained by the progress rate which gives the maximal noise strength (normalized). The limit noise strength corresponds with zero progress and therefore with a stationary state of the evolution of the distance. Undoing the normalization leads to an inequality in which the distance appears as the unknown. Therefore, solving the distance gives a lower bound for the residual location error.

In Arnold and Beyer (2002b), the simple sphere $f(R) = \beta R^\alpha$ was considered yielding lower bounds for the residual location error (see \circledast Table 14). The lower bound increases linearly with the noise strength and search space dimensionality. Further influences are the parent and offspring population sizes. Minimal values are obtained for truncation ratios of 0.5. The lower

\blacksquare Table 14

Noise and residual location errors for the sphere. See \circledast Table 11 for the definition of the noise type. The progress coefficient is given by \circledast Eq. 132

Fitness function	Noise type	Residual location error
Sphere	Systematic	$R_\infty \geq \frac{N\sigma_e}{\sqrt{8\mu c_{\mu/\mu,\lambda}}} \sqrt{1 + \sqrt{1 + \frac{8\mu^2 c_{\mu/\mu,\lambda}^2}{N}}}$
Sphere	Standard	$R_\infty = \sqrt{\frac{N\sigma_e}{4\mu c_{\mu/\mu,\lambda}}}$

bound approximates the residual location error very closely. This is caused by the step-size adaptation mechanisms which operate with comparatively low mutation strengths as experiments showed.

General Quadratic Fitness Model

The analysis of general quadratic fitness models (☞ 73) is more difficult than the simple sphere model. While the residual location error can be obtained rather easily if the function is of the form $f(R) = \beta R^2$, the general model requires more effort. Usually, the residual fitness error is derived instead of the location error. One approach, followed in Beyer and Arnold (1999, 2003a), is analogous to the sphere model case. The same evolution criterion is used substituting the radius with the mean radius to be obtained using differential geometry methods. This leads finally to a stationarity condition for general quadratic fitness functions. Switching to the eigen space of the \mathbf{Q} -matrix simplifies the calculations.

In Beyer and Arnold (2003a) steady state conditions for the $(\mu/\mu_b, \lambda)$ -ES were derived (see ☞ Table 15). Using the sphere model evolution criterion for this ES (Arnold and Beyer 2002b), an inequality for the square of the transformed location error $\|\mathbf{Q}(\mathbf{y} - \hat{\mathbf{y}})\|$ can be derived. Taking the expectation of the transformed location error leads to an expression that can be interpreted as the variance $E[(y_i - \hat{y}_i)^2]$ of the object vector component y_i in the eigen space of \mathbf{Q} , centered about the optimizer \hat{y}_i . Using the *equipartition assumption* which states that in expectation, every weighted component $q_i(y_i - \hat{y}_i)^2$ contributes the same amount to the fitness error, a lower bound for the fitness error could be derived. The equipartition assumption is similar to the equipartition assumption in statistical thermodynamics. Up to now, it has not been proven formally, but the following plausibility argument can be made (Beyer and Arnold 2003a): The ES is in the steady state. The mutations which generate the object vector components are not directed. Selection is not influenced by a single component since it only experiences the fitness as a whole and thus does not prefer a single weighted component. If one component dominated the rest, the corresponding individual would go extinct since this results in lower fitness values. The components fluctuate independently around the optimizer.

Table 15

Noise and residual fitness errors for some fitness functions. The trace of the matrix \mathbf{Q} is defined as $\text{Tr}[\mathbf{Q}] = \sum_{i=1}^N (\mathbf{Q})_{ii}$, where $(\mathbf{Q})_{ii}$ are the diagonal elements of \mathbf{Q}

Fitness function	Noise type	Residual fitness error
Biquadratic functions	Standard	$E[4F] \geq \frac{3\sigma_e N}{40\mu c_{\mu/\mu,\lambda}}$
L_1 -norm	Standard	$E[4F] \geq \frac{\sigma_e N}{\pi\mu c_{\mu/\mu,\lambda}}$
General quadratic functions	Standard	$E[4F] \geq \frac{\sigma_e N}{4\mu c_{\mu/\mu,\lambda}}$
General quadratic functions	Actuator	$E[4F] \geq \sigma_e^2 \text{Tr}[\mathbf{Q}] + \frac{N\sigma_e^2 \text{Tr}[\mathbf{Q}]}{8\mu^2 c_{\mu/\mu,\lambda}^2} \times \left(1 + \sqrt{1 + \frac{8\mu^2 c_{\mu/\mu,\lambda}^2 \text{Tr}[\mathbf{Q}^2]}{\text{Tr}[\mathbf{Q}]^2}} \right)$

For this reason, selection does not prefer any particular search direction. If there were such a preference, the ES would still exhibit a directed movement and would not have arrived at the steady state.

The lower bound obtained by the equipartition assumption depends linearly on the noise strength, on the search space dimensionality, and on the population parameters (see  [Table 15](#)). Interestingly, there is no influence of the matrix \mathbf{Q} . That is, the result holds for all ellipsoidal quadratic success domains. The agreement of the lower bound with experimentally determined residual fitness errors is quite good (Beyer and Arnold [2003a](#)).

Other Fitness Models

Beyer and Meyer-Nieberg ([2005](#)) considered the residual fitness error of the biquadratic functions, and the L_1 -norm. The expressions were obtained for the $(1, \lambda)$ -ES. With a simple analogy argument, the results were transferred to the $(\mu/\mu_t, \lambda)$ -ES which enabled a determination of the optimal truncation ratio.

The quality gain obtained in Beyer and Meyer-Nieberg ([2004](#)) can be used to derive a necessary evolution criterion $S_F^2 \geq \sigma_e |M_F| / c_{1,\lambda}$. Under certain conditions, this lower bound can be assumed to be sharp. To obtain the fitness error, the mean and the standard deviation of the fitness value must be determined. For the fitness functions considered, the stationary state criterion is a nonlinear expression of the object vector components and the mutation strength. The determination of the fitness error requires some simplifications and assumptions, especially the equipartition assumption. All the function classes considered have a final fitness error of the form $E[\Delta F] = \sigma_e N/(Cc_{1,\lambda})$, with C depending on the fitness function. Substituting $c_{1,\lambda}$ with $\mu c_{\mu/\mu_t, \lambda}$ gives the result for the intermediate ES. Owing to the form of the fitness error, the results obtained for the sphere can be transferred to biquadratic functions and the L_1 -norm: A truncation ratio of $1/2$ is optimal with respect to the fitness error. Since working with $\mu/\lambda = 0.27$ does not significantly degrade the solution quality, this truncation ratio can still be used.

2.4.4 General and Systematic Noise

In the case of general noise, the fitness function itself is a random function F_ϵ which makes a general treatment difficult. In Beyer et al. ([2003](#)), however, the authors present a decomposition technique. Instead of using the exact random function, an approximate model is utilized. The random function F_ϵ is decomposed into two parts: a deterministic part, that is, the expectation $E[F_\epsilon|\mathbf{y}]$, and a stochastic part. The stochastic part is expanded into an Edgeworth or Gram–Charlier series. In a first approximation, this leads to a normal distribution

$$F_a(\mathbf{y}) = E[F_\epsilon|\mathbf{y}] + \mathcal{N}(0, \text{Var}[F_\epsilon|\mathbf{y}]) \quad (125)$$

Thus, on the one hand, the conditional expectation and variance of F_ϵ given the state \mathbf{y} have to be obtained. On the other hand, it is possible to apply the analysis techniques developed for the additive noise model.

Systematic noise was considered under the following aspects:

1. Beyer et al. ([2003](#)) introduced the decomposition approach.
2. Sendhoff et al. ([2002](#)) analyzed functions with noise-induced multi-modality (FNIMs).
3. Beyer et al. ([2004](#)) considered systematic noise on the quadratic sphere.

4. Beyer (2004) addressed actuator noise on general quadratic functions.
5. Beyer and Sendhoff (2006) introduced and analyzed a class of test functions for evolutionary robust optimization.

Beyer et al. (2003) considered the behavior of the $(\mu/\mu_r, \lambda)$ -ES in two uncertain environment types, the sphere with actuator noise and a mixed noise model

$$F(\mathbf{y}) = a - (z + 1)\|\mathbf{y}\|^x + bz \quad \text{with} \quad z \sim \mathcal{N}(0, \sigma_e^2) \quad (126)$$

The task in this case is to maximize the fitness values. Two classes of probabilistic performance criteria, threshold measures (Eq. 117) and statistical momentum measures (Eq. 118), were applied in the analysis. In the case of the sphere model, it followed that the robust optimizer with respect to both measures equals $\mathbf{y} = \mathbf{0}$. A lower bound for the residual location error was obtained with a quite accurate predictive power. Again, the minimal location error is obtained for a truncation ratio of 1/2. The standard recommendation $\mu / \lambda = 0.27$ still applies as a reasonable setting since the solution quality is only slightly degraded and the convergence velocity to the steady state is larger.

A further point of the analysis concerned the online fitness behavior and the robustness with respect to the statistical momentum measures. In the stationary state, the fitness values fluctuate around a mean value. In Beyer et al. (2003), the mean and the standard deviation of the fitness values were obtained. Using an *ansatz* for the variance of a single component of the centroid in object parameter space, it was possible to derive estimates for the expected value and standard deviation. Again, an ES with a truncation ratio of 1/2 achieves the smallest mean and standard deviation.

Concerning Eq. 126, the analysis showed that threshold measures are not suitable as conflicting information from a distance increase or decrease. Consequently, there is no unique value of an optimal distance. Therefore, only momentum measures should be taken into account. However, this approach also leads to ambiguities. While the first moment can be decreased by a decrease in the distance, this results in an increased variance. The variance has a minimal value for $R^x = b$. The analysis revealed that the ES shows a bifurcation behavior depending on the size of f

$$\frac{2\alpha\mu c_{\mu/\mu_r, \lambda}}{\sigma_e N} \quad (127)$$

If Eq. 127 is larger than one, only one stationary point exists and the ES shows a stable behavior with a well-defined lower bound for the residual location error. For values of Eq. 127 smaller than one, the system switches from having only one steady state to two steady states. Whether the ES converges or diverges depends on the initial distance. If the ES does not diverge, it compromises, in a sense, between a fitness maximal distance and a variance minimal distance: The residual location error lying somewhere in between and the population parameters can be used to bias the ES toward either of the extremes.

In Beyer et al. (2004), the influence of systematic noise on the performance and steady state behavior of ESs on the quadratic sphere was considered. The analysis was focused on the following questions:

1. Efficiency.
2. What are sufficient evolution criteria considering the dynamical behavior of the ES? That is, under which conditions do the ES *not* diverge?

3. What does this imply for σ control rules? And are there differences between CSA and σ SA control?

Using the decomposition technique from above, it is possible to derive the progress rate showing close resemblance to the case of additive noise.

The progress rate obtained, the question of efficiency defined by $\eta = \max_{\sigma} \varphi(\sigma) / \lambda$ could be discussed. However, due to the form of the progress rate, the maximal efficiency can only be determined for cases where the noise strength scales with the distance to the optimizer. This special case leads to the recommendation of a truncation ratio $\mu/\lambda = 0.27$.

Considering general noise again, the progress rate allows for a derivation of a sufficient evolution criterion which leads to two main findings:

1. To increase the convergence probability in cases with high noise levels, any σ control mechanism should produce small step sizes. This, however, comes at the cost of reduced convergence speed. Again, there is a trade-off between convergence reliability and convergence speed.
2. Problems with higher search space dimensionality are evolutionarily more stable (Beyer et al. 2004).

The analysis then considered the case of constant noise strength. A steady state, that is zero progress, is attained if the equal sign in the evolution criterion is exactly fulfilled. As investigated empirically, both CSA and σ SA eventually steer the ES toward the zero progress region. The criterion of zero progress can then be used to derive lower bounds for the residual location error yielding two results: First, the residual location error scales linearly with the noise strength. Second, recommendations concerning μ and λ can be given. It is optimal to have a truncation ratio of $\mu/\lambda = 1/2$. The region around the optimal truncation ratio is rather flat, allowing the use of smaller ratios with better efficiency. Furthermore, keeping the truncation ratio fixed, increasing the population size decreases the location error.

In Beyer (2004), the effects of actuator noise on the stationary state behavior of intermediate evolution strategies were considered using the general quadratic fitness model (🔗 73). The analysis showed that the expected deviation from the actual optimum consists of two parts: one part which is independent of the ES and depends only on the function and on the noise strength, and one part that can be influenced by the ES. Using a truncation ratio of 1/2 yields a minimal deviation from the actual optimum. This ratio is valid as long as the assumption of (at least asymptotical) normality is guaranteed.

In Beyer and Sendhoff (2006), a new test function class, functions with noise-induced multi-modality (FNIMs), was introduced and analyzed. The criteria for the analysis were moment-based measures and threshold measures. The authors succeeded in developing a test function that captures the important properties of general FNIMs while being amenable to a theoretical analysis. The steady state performance of the $(\mu/\mu_b, \lambda)$ -ES was investigated yielding the following findings:

- The resampling size should be kept as small as possible.
- A truncation ratio of about 0.3 is a good compromise between convergence speed and solution quality.
- Comparing σ SA and CSA showed that each mechanism has its advantages and disadvantages: While using CSA leads to a faster convergence, σ SA provides a better exploration behavior in the stationary state, albeit, with the risk of oscillatory behavior.

2.4.5 Noise and Step-Size Adaptation

This section considers the behavior of step-size adaptation mechanisms in the case of noisy fitness evaluations. Two mechanisms were primarily considered, self-adaptation and cumulative step-size adaptation. First, the results for self-adaptation are described.

Self-Adaptation

In Beyer and Meyer-Nieberg (2006a), the self-adaptive $(1, \lambda)$ -ES on the noisy sphere was considered. The first task was to determine the self-adaptation response (see [Table 18](#)). Afterwards, the deterministic evolution equations were used in the analysis. The evolution equations model the dynamical behavior of the normalized mutation strength, the distance to the optimizer, and the normalized noise strength. It can be shown that the evolution of the normalized mutation strength and noise strength do not depend on the distance R . Therefore, the analysis can be restricted to the two-dimensional system of σ^* and σ_ε^* .

Three phases of the evolution can be distinguished for constant noise strength σ_ε .

- A first phase, characterized by large distances to the optimizer in which the ES nearly behaves as if the fitness evaluations were exact.
- A transient phase, in which the ES is more and more influenced by the noise as it moves toward the optimizer.
- A stationary state in which no progress on average occurs. In this case, the dynamics of the mutation strength σ^* and σ_ε^* approach stationary distributions.

The progress rate and the self-adaptation response can be used to determine the $(\sigma^*, \sigma_\varepsilon^*)$ -combinations, which equal zero progress and zero expected change of the mutation strength. This leads to expressions for the stationary mutation strength, noise strength, and the residual location error, displayed in [Table 16](#). In contrast to the result obtained in Beyer (1993), the nonzero mutation strength introduces a correction factor which increases the residual location error. However, the deviation is not large, that is, self-adaptation does not lead to a significant degradation of the solution quality.

There is a peculiarity in the behavior of the $(1, \lambda)$ -ES: if the noise strength becomes too large, the ES loses step-size control. The mutation strength exhibits a random walk-like behavior biased toward small mutation strengths. This behavior cannot be predicted by the deterministic approach used in Beyer and Meyer-Nieberg (2006a). In spite of that, an explanation for this behavior can be given by considering the probability of a decrease in the mutation strength under random selection. This probability equals $1/2$. Selection changes the situation. The $(1, \lambda)$ -ES accepts a decrease of the mutation strength with high probability.

Table 16

The stationary state values of the $(1, \lambda)$ -ES with σ -self-adaptation on the sphere $g(R) := -cR^\alpha$. The normalizations read $\sigma^* := \sigma N / R$ and $\sigma_\varepsilon^* := \sigma_\varepsilon N / (c\alpha R^\alpha)$

Mutation strength	Noise strength	Location error
$\sigma^* = \frac{2c_{1,\lambda}}{\sqrt{2(2c_{1,\lambda}^2 + 1 - d_{1,\lambda}^{(2)})}}$	$\sigma_\varepsilon^* = 2c_{1,\lambda} \sqrt{1 - \frac{1}{2(2c_{1,\lambda}^2 + 1 - d_{1,\lambda}^{(2)})}}$	$R = \sqrt{\frac{\sigma_\varepsilon N}{c\alpha 2c_{1,\lambda}}} \sqrt{\frac{2(2c_{1,\lambda}^2 + 1 - d_{1,\lambda}^{(2)})}{2(2c_{1,\lambda}^2 + 1 - d_{1,\lambda}^{(2)}) - 1}}$

Large mutation strengths coincide more often with a selective disadvantage and are punished on average. This results in a tendency of the $(1, \lambda)$ -ES toward smaller mutation strengths: The ES performs some kind of biased random walk.

The aforementioned deficiency of $(1, \lambda)$ -ESs leads to the question whether intermediate recombination ESs exhibit similar erratic behaviors. Interestingly, this is not the case. These strategies are biased (in probability) toward an increase of the mutation strength resulting in a more stable σ -adaptation behavior.

In Meyer-Nieberg and Beyer (2008) the performance of the $(\mu/\mu_b, \lambda)$ -ES on the noisy sharp ridge was investigated (see [Tables 17](#) and [18](#)). The analysis provided an unexpected result: on the sharp ridge, noise can increase the performance. In order to understand this interesting property, one has to recall that the optimization consists of two subgoals:

1. Reducing the distance to the axis, that is, optimizing the embedded sphere model
2. Enlarging the linear axis component

Given a fixed strategy, the ridge parameter d decides which subgoal dominates the other. If d is large, the ES tries to optimize the sphere model which ultimately results in stagnation, since the mutation strength is decreased accordingly. Additive noise prevents the ES from achieving this subgoal. It attains a positive stationary $\sigma > 0$ and a “residual location error” (fluctuating at a certain distance from the ridge axis). Due to the positive mutation strength, the ES can serve the second subgoal: the enlargement of the linear part. Putting it another way, larger noise strengths result in worse achievements of the sphere subgoal which correspond to greater axial progress. Axial progress, mutation strength, and distance scale linearly with the noise. A remark concerning recombination is to be added. Recombination reduces the residual location error. This goes along with reduced mutation strengths and axial progress rates.

Table 17

The stationary state values of the $(\mu/\mu_b, \lambda)$ -ES with σ -self-adaptation on the sharp ridge. Shown are the approximate values obtained for large population sizes λ and appropriate choices of μ . The normalizations read $\sigma^* := \sigma N / R$ and $\varphi^* := \varphi N / R$

Mutation strength	Distance	Progress
$\sigma^* = \sqrt{\mu}$	$R = \frac{\sigma_e N}{2d\mu c_{\mu/\mu_b, \lambda}}$	$\varphi^* = \frac{1}{2d}$

Table 18

Self-adaptation response functions (SARs) for different fitness environments. Explanation of variables and notation: $c_{\mu/\mu_b, \lambda}$ progress coefficient (see [Sect. 5](#)), σ^* normalized mutation strength (with respect to distance to optimizer and search space dimensionality), σ_e^* standard deviation of noise distribution (normalized), d and α parameters of the ridge function

Fitness	SAR
Sphere	$\psi(\sigma^*) = \tau^2 \left(\frac{1}{2} + \frac{e_{\mu/\mu_b, \lambda}^{1,1} \sigma^{*2}}{\sigma^{*2} + \sigma_e^{*2}} - \frac{c_{\mu/\mu_b, \lambda} \sigma^{*2}}{\sqrt{\sigma^{*2} + \sigma_e^{*2}}} \right)$
Ridge	$\psi(\sigma^*) = \tau^2 \left(\frac{1}{2} + \frac{e_{\mu/\mu_b, \lambda}^{1,1} (1 + \alpha^2 d^2 R^{2\alpha-2}) \sigma^{*2}}{(1 + \alpha^2 d^2 R^{2\alpha-2}) \sigma^{*2} + \sigma_e^{*2}} - \frac{\alpha d R^{\alpha-1} c_{\mu/\mu_b, \lambda} \sigma^{*2}}{\sqrt{(1 + \alpha^2 d^2 R^{2\alpha-2}) \sigma^{*2} + \sigma_e^{*2}}} \right)$

However, recombination is necessary, since the $(1, \lambda)$ -ES experiences a loss of step-size control. Therefore, the use of small truncation ratios $> 1/\lambda$ is recommended.

Cumulative Step-Size Adaptation (CSA)

Arnold and Beyer (2000, 2004, 2008) and Beyer and Arnold (2003b) analyzed the performance of the $(\mu/\mu_b, \lambda)$ -ES with cumulative step-size adaptation (CSA). As far as the noise-free case is concerned, the reader is referred to [Sect. 2.3.4](#). The work done focused on a deeper discussion of the underlying philosophy of CSA, progress rate optimality, and population sizing. Several fitness environments were considered, starting with the sphere (Beyer and Arnold 2003b; Arnold and Beyer 2004) and continuing to ridge functions Arnold and Beyer (2008).

This paragraph starts with the sphere model and the additive Gaussian noise model. In Arnold and Beyer (2000) a first analysis of the performance of adaptive $(\mu/\mu, \lambda)$ -ESs was provided. Two major step-size adaptation mechanisms were considered: self-adaptive ESs (with arithmetic and geometric recombination of the mutation strength) and ESs with cumulative step-size adaptation on the noisy sphere. The noise strength was set to scale with the distance to the optimizer, that is, the normalized noise strength is constant. Demanding positiveness of the progress rate yields the necessary evolution criterion, that is, a criterion that must be fulfilled to have convergence to the optimizer at all. As a consequence, there is a maximal normalized noise strength above which the strategy diverges. This maximal noise strength depends on the population parameters and the fitness function. Experiments showed that the adaptation mechanisms encounter problems when noise comes into play. In the absence of noise, the algorithms achieve linear convergence order. For lower levels of noise, nearly optimal mutation strengths are generated. Increasing the noise level degrades the performance. For high noise-levels, the CSA generates decreasing normalized mutation strengths which first results in slower convergence and finally stagnation, if the noise-levels become too high. In contrast to the CSA-ES, the self-adaptive $(\mu/\mu_b, \lambda)$ -ES achieves a stationary normalized mutation strength which results in divergent behavior.

In Arnold and Beyer (2004), the authors performed an in-depth analysis of the $(\mu/\mu_b, \lambda)$ -CSA-ES on the sphere using the Gaussian noise model. The ES approaches a steady state which means that the progress vector, its length, and the normalized mutation strength have a time-invariant limit distribution. Calculating the target mutation strength, that is, the normalized mutation strength the value of which the CSA mechanism does not change in expectation, leads to a value smaller than the optimal mutation strength needed to maximize the progress rate. Furthermore, the logarithmic adaptation response indicates that for mutation strengths, which are significantly below the optimal value, there is only a small tendency toward an increase. This can be explained by the way the cumulative step-size adaptation works: It tries to adapt σ as if consecutive progress vectors were uncorrelated. Since noise overshadows the information gained from smaller step sizes, steps for small mutation strengths become nearly random and correlations disappear. This is in accordance with the CSA design philosophy, but the resulting search behavior is not always that what users desire when optimizing in noisy environments.

As in the non-noisy case, the ES cannot realize the target mutation strength in the steady state and it is usually not optimal. The mutation strength is too small for large noise strengths and too large for small noise strengths. The resulting convergence velocity is about 20–30% smaller than the optimum. Concerning the question of the population size, keeping the best 25–30% of the offspring population is a good choice.

In Beyer and Arnold (2003b), the fundamental working premises of the CSA algorithm were examined in detail. The analysis concerned the $(\mu/\mu_b, \lambda)$ -ES on the *static* noisy sphere and tried to shed light on the question whether the perpendicularity or uncorrelation condition of the CSA guarantees optimal progress and if not to identify the reasons. As a first step in the analysis, the perpendicularity condition (Hansen 1998), stating that the expectation of the scalar product of two consecutive step vectors $\mathbf{z}^{(g)}$ (Eq. 2) is zero, was revisited. Considering two consecutive generations and decomposing the second search step, the angle between the two vectors can be calculated. Perpendicularity is achieved for a certain mutation strength which depends on the population parameters and the noise strength: In the absence of noise, the perpendicularity condition leads to an asymptotically optimal adaptation of the mutation strength. In the presence of noise, however, the optimal mutation strength is not reached. The perpendicularity condition leads to a CSA evolution criterion (see Table 19) resulting in a maximally admissible noise strength, which is only half the limit value necessitated by the *ES evolution criterion*. If the noise strength is in between, the CSA realizes angles below 90° . To “cope” with the apparently too large mutation strength, the CSA responds by reducing the mutation strength continuously to zero. This explains the observations made in experiments.

In Arnold (2006b), the performance of the (λ_{opt}) -ES with CSA and weight rescaling was investigated on the noisy sphere. The realized mutation strength of the CSA differs from the optimal mutation strength. Again, there is a certain performance loss. However, it can be reduced by working with a large rescaling factor κ .

In Arnold and Beyer (2008), the performance of the $(\mu/\mu_b, \lambda)$ -ES with CSA was analyzed for the noisy parabolic ridge. Three model functions describing different scalings of the standard deviation σ_e (also referred to as noise strength) of the normally distributed noise have been investigated: (a) constant, (b) quadratically scaled, and (c) cubically scaled noise strengths. In the case of (b) and (c), the noise strength increases with the distance to the ridge axis. The steady state distance to the ridge axis, the mutation strength, and the axial progress rate were determined. Interestingly, it was found that the ES with CSA tracks the axis with a distance that is independent of the noise type. In the noise-free case, the $(\mu/\mu_b, \lambda)$ -ES with CSA achieves a progress rate that is half the optimum. In the presence of noise, only for low levels of noise, a stationary distance and positive progress can be achieved.

For constant noise strength, see Table 20, this means that above a certain strategy-dependent noise strength, progress is not possible. Note that the analysis part, without considering step-size adaptation mechanisms, indicates that an ES could cope with any noise strength by increasing the mutation strength. However, this property cannot be

Table 19

The perpendicularity condition of the CSA mechanism. According to the CSA philosophy, the expected angle, $\bar{\beta}$, between the parental change vector, and the local vector to the optimizer should be 90° . That is, $\cos(\bar{\beta}) = 0$ leads to the second column and in turn to the third

Angle	Mutation strength	CSA evolution criterion	ES evolution criterion
$\cos(\bar{\beta}) = \sigma^* \sqrt{\frac{\mu}{N} \left(\frac{1}{\mu} - \frac{c_{\mu/\mu,\lambda}}{\sqrt{\sigma^*{}^2 + \sigma_e^*{}^2}} \right)}$	$\sigma^* = \sqrt{\mu^2 c_{\mu/\mu,\lambda}^2 - \sigma_e^*{}^2}$	$\sigma_e^* < \mu c_{\mu/\mu,\lambda}$	$\sigma_e^* < 2\mu c_{\mu/\mu,\lambda}$

Table 20

The $(\mu/\mu_b, \lambda)$ -ES with CSA on the noisy parabolic ridge in the case of constant noise $\sigma_e = \text{const}$

Distance	Mutation strength	Progress
$R = \frac{1}{2d}$	$\sigma = \sqrt{\frac{\mu^2 c_{\mu/\mu_b, \lambda}^2}{2d^2 N^2} - \frac{\sigma_e^2}{2}}$	$\varphi = \frac{\mu c_{\mu/\mu_b, \lambda}^2}{2dN} - \frac{\sigma_e^2 dN}{2\mu}$

conserved when switching the CSA part on. Increasing the strength of the noise results in a decrease of the mutation strength. Therefore, the CSA-ES fails to track the axis for higher noise strengths. Similar findings hold for the quadratically and cubically scaled noise cases. Above a certain limit of the scaling parameter, the CSA-ES fails to generate useful mutation strengths. This limit can be shifted to larger values by increasing the population sizes accordingly.

2.5 **Dynamical Optimization: Optimum Tracking**

This section considers *dynamical optimization* problems which arise in many practical optimization situations. In contrast to conventional, static optimization where the problem does not change with time, dynamical optimization considers problems where the position of the optimizer changes – either randomly or deterministically. While in static optimization, the goal is to locate the optimizer fast and reliably, the task in dynamical optimization is to track the moving optimizer closely. That is, the task is not to converge to a fixed state, but to adapt fast to dynamically changing environments. Examples of dynamical optimization problems include online job scheduling with new jobs arriving in the course of optimization or dynamical routing problems.

In Arnold and Beyer (2002c, 2006b), the dynamical systems approach was applied to two examples of dynamical optimization problems: a sphere model with a randomly moving target (Arnold and Beyer 2002c) and a sphere model with linearly moving target (Arnold and Beyer 2006b). The analyses aimed at answering the following questions:

1. Are ES with cumulative step length adaptation (CSA) capable of dealing with dynamical problems – although the CSA rule was designed for static optimization?
2. What are the effects of using populations and recombination?

In Arnold and Beyer (2002c), the performance of the $(\mu/\mu_b, \lambda)$ -ES with CSA was investigated considering a sphere with random movements of the optimizer. Assuming that there exists a steady state of the ES once initialization effects have faded, an expression for the stationary tracking distance can be derived. The tracking distance depends on the mutation strength. Computing the derivative of the distance (with respect to the mutation strength) and setting it to zero, an optimal mutation strength and the minimal tracking error can be calculated (see [Table 21](#)). The question arises whether the CSA mechanism is able to tune the mutation strength to its optimal value. This is indeed the case (Arnold and Beyer 2002c). Therefore, it can be concluded that the CSA mechanism works perfectly for the $(\mu/\mu_b, \lambda)$ -ES for tracking a randomly moving target.

In Arnold and Beyer (2006b), the analysis was extended to the tracking of a linearly moving target. The tracking distance of the stationary state of the ES as a function of the

Table 21

Tracking behavior of the $(\mu/\mu_\lambda, \lambda)$ -CSA-ES on a randomly moving target (Arnold and Beyer 2002c) and a linearly moving target (Arnold and Beyer 2006b). The parameters μ and λ denote the sizes of the parent and offspring populations, N the search space dimensionality, c_{μ/μ_λ} is the progress coefficient (see [Sect. 5](#)), and δ denotes the speed of the dislocation of the objective. As for the random case, δ is the standard deviation of the random change of a single component of the optimizer, that is, $\hat{y}_i^{(g+1)} = \hat{y}_i^{(g)} + \mathcal{N}(0, \delta^2)$. In the linear case, δ is simply the velocity of change, that is, $\hat{y}^{(g+1)} = \hat{y} + \delta v$ with v the unit vector

Dynamics	Tracking distance	Optimal track. distance	Optimal mutation strength	Mutation strength realized
Random	$R = \frac{N}{2c_{\mu/\mu_\lambda}} \left(\frac{\sigma}{\mu} + \frac{\delta^2}{\sigma} \right)$	$R = \frac{N\delta}{\sqrt{\mu} c_{\mu/\mu_\lambda}}$	$\sigma = \sqrt{\mu}\delta$	$\sigma = \sqrt{\mu}\delta$
Linear	$R = \frac{N\sigma^3 c_{\mu/\mu_\lambda}}{2\mu(\sigma^2 c_{\mu/\mu_\lambda}^2 - \delta^2)}$	$R = \frac{3\sqrt{3}N\delta}{4\mu c_{\mu/\mu_\lambda}^2}$	$\sigma = \frac{\sqrt{3}\delta}{c_{\mu/\mu_\lambda}}$	$\sigma = \frac{\sqrt{2}\delta}{c_{\mu/\mu_\lambda}}$

mutation strength, the optimal mutation strength, and distance were determined. Considering the CSA-rule, it was found that an ES with CSA is able to track the linearly moving target. However, the mutation strength adapted is not optimal. The ES realizes the same mutation strength as in the static case.

Summarizing, it was found that ESs with CSA are capable of tracking dynamically moving targets. In the case of randomly moving targets, the CSA even succeeds in adapting the optimal mutation strength. In both cases, recombination improves the behavior of the ES: Evolution strategies that make use of parent populations with size greater than one and of recombination track the target more closely than $(1, \lambda)$ -ESs. These theoretical results are also of certain interest for other EAs, such as GAs, where it has been reported that dynamical problems often degrade the performance severely. As to the cases investigated so far, the ES does not experience such problems.

3 The Dynamical Systems Approach in Comparison

The approach presented so far used local progress measures to obtain analytical solutions, which can be used to discuss the dynamical behavior and convergence properties of the EAs. On the one hand, this enables us to consider the process in more detail: It is possible to analyze the relationship between parameter settings and the performance of the evolution strategy. On the other hand, due to the simplifications applied and assumptions made, ultimately, a “model” of the actual ES behavior is analyzed. The derivation of the model aims to take all important characteristics of the true ES into account and to transfer them to the final result. The most important assumption is the consideration of large search space dimensionalities N . This allows for the derivation of asymptotical progress measures, that is, measures obtained for infinite-dimensional search spaces. The so derived conclusions are valid as long as N is large. Considering small N will lead to deviations and usually the error made cannot be quantified analytically. That is why, the theoretical results must be supplemented with experiments.

Most analyses of self-adaptation consider only the expected changes. The ES dynamics are treated as if they were stemming from a deterministic model which may not be sufficient to capture all important characteristics of the process as the $(1, \lambda)$ -ES on the noisy sphere has revealed.

Auger and Hansen (2006) presented a discussion on the limitations of the progress rate theory on the sphere model for the $(1, \lambda)$ -ES. As they pointed out, for finite search space dimensionalities the usual definition of the progress rate theory corresponds to a *convergence in mean*. Connected with *almost sure convergence* is the *logarithmic progress rate* $\varphi_{\ln} = E[\ln(R^{(g)}/R^{(g+1)})]$, instead. However, the discrepancy between both measures decreases with increasing search space dimensionality, and in the asymptotical limit, both measures are the same. While, using the logarithmic progress rate has a certain appeal (and it is implicitly used when plotting fitness dynamics in logarithmic scale), up until now, there is no method to calculate it analytically for finite N sphere. Monte Carlo simulations must be used for a quantification. This diminishes the usefulness of that progress measure.

While using local performance measures and treating the ES as a dynamical system has proven as a very fruitful approach yielding results of practical relevance, it is not the only one. Analysis methods from the field of stochastic systems have been tried as well. These either used the theory of Markov processes (Bienvenüe and François 2003; Auger 2005) or studied the induced supermartingales (Semenov 2002; Semenov and Terkel 2003; Hart et al. 2003). However, nearly all attempts had to resort to numerical Monte Carlo methods at some point. Therefore, also those results are not exact in that, the correctness of their conclusions can only be guaranteed probabilistically, that is, with a probability not equal to one.

Bienvenüe and François (2003) examined the global convergence of adaptive and self-adaptive $(1, \lambda)$ -evolution strategies on the sphere. They showed that $(z_g)_{g \geq 1} = (\|x_g\|/\sigma_g)$ is a homogeneous Markov chain, that is, z_g only depends on z_{g-1} . This also confirms an early result obtained in Beyer (1996b) that the evolution of the mutation strength can be decoupled from the evolution of $\|x_g\|$. Furthermore, they showed that (x_g) converges or diverges log-linearly if the process has certain characteristics.

Auger (2005) followed this line of research. She analyzed a general model of an $(1, \lambda)$ -ES on the sphere. Auger proved that the induced Markov process fulfills the required conditions for log-linear convergence or divergence if the offspring number λ is chosen appropriately. However, her final results for the $(1, \lambda)$ -ES again depend on a Monte Carlo simulation, thus providing only a “probabilistic proof”.

Semenov (2002) and Semenov and Terkel (2003) examined the convergence and the convergence velocity of evolution strategies using the theory of supermartingales. Because of the complicated nature of the underlying stochastic process, the authors did not succeed in a rigorous mathematical treatment of the stochastic process. Similar to the Markov chain approach, the authors had to resort to Monte Carlo simulations in order to show that the necessary conditions are fulfilled. In Semenov (2002) and Semenov and Terkel (2003), the $(1, \lambda)$ -ES was considered. Offspring are generated according to

$$\begin{aligned}\sigma_{g,l} &= \sigma_g e^{\vartheta_{g,l}} \\ x_{g,l} &= x_g + \sigma_{g,l} \zeta_{g,l}\end{aligned}\tag{128}$$

The fitness function considered is given by $f(x) = -|x|$. The random variables $\vartheta_{g,l}$ and $\zeta_{g,l}$ are uniformly distributed with $\vartheta_{g,l}$ assuming values in $[-2, 2]$ whereas $\zeta_{g,l}$ is defined on $[-1, 1]$. For this problem, it could be shown that the object variable and the mutation strength

converge almost surely to zero – provided that there are at least three offspring. Additionally, the convergence velocity of the mutation strength and the distance to the optimizer is bounded from above by a function of the form $\exp(-ag)$, which holds asymptotically almost surely.

Another viewpoint is to consider the evolutionary algorithm as a randomized algorithm and to conduct runtime analyses. The goal is to obtain upper and lower bounds for the runtime, bypassing the dynamical processes.

Concerning continuous search spaces, so far results for the $(1 + \lambda)$ -ES, the $(\mu + 1)$ -ES, and the $(1, \lambda)$ -ES have been reported (see, e.g., Jägersküpper 2003, 2006b, 2007; Jägersküpper and Witt 2005). Some of the Jägersküpper (2006a) results are reproduced here:

- The $(1 + 1)$ -ES performs with overwhelming probability $\mathcal{O}(N)$ steps to halve the approximation error in the search space.
- The $(1 + \lambda)$ -ES as well as the $(1, \lambda)$ -ES get along with $\mathcal{O}(N/\sqrt{\ln(1 + \lambda)})$ steps with overwhelming probability—when the 1/5-rule is based on the number of successful mutations.
- The $(1 + \lambda)$ -ES using a modified 1/5-rule, which is based on the number of successful steps, is proved to be indeed capable of getting along with $\mathcal{O}(N/\sqrt{\ln(1 + \lambda)})$ steps with overwhelming probability, which is asymptotically optimal.
- The $(\mu + 1)$ -ES using Gaussian mutations adapted by the 1/5-rule needs $\mathcal{O}(\mu N)$ steps with overwhelming probability to halve the approximation error in the search space, which is also asymptotically optimal. (An *overwhelming probability* is defined as follows: An event occurs with overwhelming probability with respect to N if the probability of nonoccurrence is exponentially small in N (see Jägersküpper 2006a, p. 15).)

The analyzed fitness functions are the sphere and positive definite quadratic forms. The analyses also considered the adaptation mechanism of the mutation strength. So far, the 1/5th-rule could be analyzed. Up to now, no results have been obtained considering self-adaptation or the cumulative step-size adaptation.

One general drawback of the approach is that the constants hidden in the \mathcal{O} -notation usually cannot be quantified.

So far, it can be stated that exact provable results can be obtained using runtime analysis or the theory of stochastic processes. However, this goes along with a coarser image of the dynamics. As a result, questions regarding the optimal population sizing or the influence of the mutation strength cannot be answered. In contrast, the progress rate theory and the dynamical systems approach can be used to obtain analytical results and to derive answers for these questions. These, however, are usually based on models of the real evolutionary processes and are valid as long as the assumptions made in the derivation hold. Each approach aims to obtain a specific type of result. Therefore, the different methodologies should be seen as complementary and not as conflictive approaches allowing for a gradually improving understanding of the convergence and optimization behavior of evolutionary algorithms.

4 Conclusions

This chapter presented an overview over the progress rate and dynamical systems approach mainly applied to the analysis of evolution strategies in real-valued search spaces. The approach aims to provide quantitative analyses of evolutionary algorithms. To this end, local progress measures, that is, expected changes of important quantities from one generation

to the next, are introduced and analyzed. The results can be used to investigate the performance of different algorithms and its dependencies on the strategy parameter settings.

While the local progress measures may be regarded as the “microscopic forces” driving the EA as a dynamical system, the “macroscopic behavior” appears as the consequence of these quantities: The (self-)adaptive behavior of the EA is modeled as a dynamical system. The step-size adaptation mechanisms considered include mutative self-adaptation and the cumulative step-size adaptation which lies at the core of the covariance matrix adaptation. As a result, one gains insights into the functioning of the adaptation methods.

So far, the approach was mainly applied to evolution strategies. Further applications include real-coded genetic algorithms (Deb and Beyer 1999). However, the analysis techniques developed bear the potential to also gain insights into other natural computing paradigms, such as estimation of distribution algorithms (EDAs) and particle swarm optimization (PSO) algorithms.

5 The Progress Coefficients

For the readers’ convenience, the definition of the progress coefficients are listed below:

$$d_{1,\lambda}^{(k)} = \frac{\lambda}{\sqrt{2\pi}} \int_{-\infty}^{\infty} t^k e^{-\frac{t^2}{2}} \Phi(t)^{\lambda-1} dt \quad (129)$$

$$d_{1+\lambda}^{(k)}(x) = \frac{\lambda}{\sqrt{2\pi}} \int_x^{\infty} t^k e^{-\frac{t^2}{2}} \Phi(t)^{\lambda-1} dt \quad (130)$$

$$e_{\mu,\lambda}^{\alpha,\beta} = \frac{\lambda - \mu}{\sqrt{2\pi}^{\alpha+1}} \binom{\lambda}{\mu} \int_0^{\infty} t^{\beta} e^{-\frac{\alpha+1}{2}t^2} \Phi(t)^{\lambda-\mu-1} (1 - \Phi(t))^{\mu-\alpha} dt \quad (131)$$

$$c_{\mu,\lambda} = \frac{e_{\mu,\lambda}^{1,0} - \gamma_{\mu,\lambda}(1)e_{\mu,\lambda}^{1,1}}{\sqrt{1 - \frac{\mu-1}{\mu} [1 + e_{\mu,\lambda}^{1,1} - e_{\mu,\lambda}^{2,0} - 2\gamma_{\mu,\lambda}(1)(e_{\mu,\lambda}^{1,0} + e_{\mu,\lambda}^{1,2} - e_{\mu,\lambda}^{2,1})]}} \quad (132)$$

$$\gamma_{\mu,\lambda}(1) = \frac{e_{\mu,\lambda}^{1,0} - e_{\mu,\lambda}^{1,2} + 3e_{\mu,\lambda}^{2,1} - 2e_{\mu,\lambda}^{3,0}}{\frac{6\mu^2}{(\mu-1)(\mu-2)} - (6 + 3e_{\mu,\lambda}^{1,1} + 3e_{\mu,\lambda}^{1,3} - 6e_{\mu,\lambda}^{2,0} - 9e_{\mu,\lambda}^{2,2} + 6e_{\mu,\lambda}^{3,1})}. \quad (133)$$

The progress coefficient $c_{1,\lambda}$ is a special case of the progress coefficients $d_{1,\lambda}^{(k)}$ with $c_{1,\lambda} := d_{1,\lambda}^{(1)}$. The progress coefficient $c_{\mu/\mu,\lambda}$ is given as $c_{\mu/\mu,\lambda} = e_{\mu,\lambda}^{1,0}$.

References

- Abramowitz M, Stegun IA (1984) Pocketbook of mathematical functions. Harri Deutsch, Thun
- Arnold BC, Balakrishnan N, Nagaraja HN (1992) A first course in order statistics. Wiley, New York
- Arnold DV (2002a) Noisy optimization with evolution strategies. Kluwer, Dordrecht
- Arnold DV (2006a) Cumulative step length adaptation on ridge functions. In: Runarsson TP et al. (eds) Parallel problem solving from nature PPSN IX. Springer, Heidelberg, pp 11–20
- Arnold DV (2006b) Weighted multirecombination evolution strategies. Theor Comput Sci 361(1):18–37.
- Foundations of Genetic Algorithms
- Arnold DV (2007) On the use of evolution strategies for optimising certain positive definite quadratic forms. In: Proceedings of the 9th annual conference on Foundations of Genetic Algorithms

- genetic and evolutionary computation: GECCO'07: London, July 7–11, 2007. ACM New York, pp 634–641
- Arnold DV, Beyer H-G (2000) Efficiency and mutation strength adaptation of the $(\mu/\mu_b, \lambda)$ -ES in a noisy environment. In: Schoenauer M (ed) Parallel problem solving from nature, vol 6. Springer, Heidelberg, pp 39–48
- Arnold DV, Beyer H-G (2001a) Investigation of the (μ, λ) -ES in the presence of noise. In: Proceedings of the CEC'01 conference, Seoul, May 27–30, 2001. IEEE Piscataway, NJ, pp 332–339
- Arnold DV, Beyer H-G (2001b) Local performance of the $(\mu/\mu_b, \lambda)$ -ES in a noisy environment. In: Martin W, Spears W (eds) Foundations of genetic algorithms, vol 6. Morgan Kaufmann, San Francisco, CA, pp 127–141
- Arnold DV, Beyer HG (2002a) Local performance of the $(1+1)$ -ES in a noisy environment. *IEEE Trans Evol Comput* 6(1):30–41
- Arnold DV, Beyer H-G (2002b) Performance analysis of evolution strategies with multi-recombination in high-dimensional \mathbb{R}^n -search spaces disturbed by noise. *Theor Comput Sci* 289:629–647
- Arnold DV, Beyer H-G (2002c) Random dynamics optimum tracking with evolution strategies. In: Merelo Guervós JJ et al. (eds) Parallel problem solving from nature, vol 7. Springer, Heidelberg, pp 3–12
- Arnold DV, Beyer H-G (2003a) On the benefits of populations for noisy optimization. *Evolutionary Computation*, 11(2):111–127
- Arnold DV, Beyer H-G (2003b) On the effects of outliers on evolutionary optimization. In: Liu J, Cheung Y-M, Yin H (eds) IDEAL 2003: Fourth international conference on intelligent data engineering and automated learning, Hong Kong, March 21–23, 2003. Springer, Heidelberg, pp 151–160
- Arnold DV, Beyer H-G (2004) Performance analysis of evolutionary optimization with cumulative step length adaptation. *IEEE Trans Automatic Control*, 49(4):617–622
- Arnold DV, Beyer H-G (2006a) A general noise model and its effect on evolution strategy performance. *IEEE Trans Evol Comput* 10(4):380–391
- Arnold DV, Beyer H-G (2006b) Optimum tracking with evolution strategies. *Evol Comput* 14:291–308
- Arnold DV, Beyer H-G (2008) Evolution strategies with cumulative step length adaptation on the noisy parabolic ridge. *Nat Comput* 4(7):555–587
- Arnold L (2002b) Random dynamical systems, 2nd printing. Springer, New York
- Auger A (2005) Convergence results for the $(1, \lambda)$ -SA-ES using the theory of ϕ -irreducible Markov chains. *Theor Comput Sci* 334:35–69
- Auger A, Hansen N (2006) Reconsidering the progress rate theory for evolution strategies in finite dimensions. Seattle, WA, July 2006. Proceedings of the 8th annual conference on genetic and evolutionary computation: GECCO'06: ACM, New York, pp 445–452
- Bäck T (1997) Self-adaptation. In: Bäck T, Fogel D, Michalewicz Z (eds) *Handbook of evolutionary computation*. Oxford University Press, New York, pp C7.1:1–C7.1:15
- Beyer H-G (1989) Ein Evolutionsverfahren zur mathematischen Modellierung stationärer Zustände in dynamischen Systemen. Dissertation, Hochschule für Architektur und Bauwesen, Weimar, Reihe: HAB-Dissertationen, Nr. 16
- Beyer H-G (1993) Toward a theory of evolution strategies: some asymptotical results from the $(1+\lambda)$ -theory. *Evol Comput* 1(2):165–188
- Beyer H-G (1994) Towards a theory of ‘evolution strategies’: progress rates and quality gain for $(1+\lambda)$ -strategies on (nearly) arbitrary fitness functions. In: Davidor Y, Männer R, Schwefel H-P (eds) Parallel problem solving from nature, vol 3. Springer, Heidelberg, pp 58–67
- Beyer H-G (1995a) Toward a theory of evolution strategies: on the benefit of sex – the $(\mu/\mu, \lambda)$ -theory. *Evol Comput* 3(1):81–111
- Beyer H-G (1995b) Toward a theory of evolution strategies: the (μ, λ) -theory. *Evol Comput* 2(4):381–407
- Beyer H-G (1996a) On the asymptotic behavior of multi-recombinant evolution strategies. In: Voigt H-M, Ebeling W, Rechenberg I, Schwefel H-P (eds) Parallel problem solving from nature, vol 4. Springer, Heidelberg, pp 122–133
- Beyer H-G (1996b) Toward a theory of evolution strategies: self-adaptation. *Evol Comput* 3(3):311–347
- Beyer H-G (2000) Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Comput Methods Appl Mech Eng* 186 (2–4):239–267
- Beyer H-G (2001a) On the performance of $(1, \lambda)$ -evolution strategies for the ridge function class. *IEEE Trans Evol Comput* 5(3):218–235
- Beyer H-G (2001b) The theory of evolution strategies. Natural computing series. Springer, Heidelberg
- Beyer H-G (2004) Actuator noise in recombinant evolution strategies on general quadratic fitness models. In: Deb K et al. (eds) GECCO 2004: Proceedings of the genetic and evolutionary computation conference, Seattle, WA, June 26–30, 2004. Springer, Heidelberg, pp 654–665
- Beyer H-G, Arnold DV (1999) Fitness noise and localization errors of the optimum in general quadratic fitness models. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakela M, Smith RE (eds), GECCO-99: Proceedings of the genetic and evolutionary computation conference, Orlando, FL, July 1999. Morgan Kaufmann, San Francisco, CA, pp 817–824

- Beyer H-G, Arnold DV (2003a) The steady state behavior of $(\mu/\mu_t, \lambda)$ -ES on ellipsoidal fitness models disturbed by noise. In: Cantú-Paz E et al. (eds) Proceedings of the genetic and evolutionary computation conference, Chicago, IL, July 2003. Springer, Berlin, pp 525–536
- Beyer H-G, Arnold DV (2003b) Qualms regarding the optimality of cumulative path length control in CSA/CMA-evolution strategies. *Evol Comput* 11(1):19–28
- Beyer H-G, Finck S (2009) Performance of the $(\mu/\mu_t, \lambda)$ - σ SA-ES on PDQFs. *IEEE Trans Evol Comput* (accepted) <http://dx.doi.org/10.1109/TEVC.2009.2033581>
- Beyer H-G, Meyer-Nieberg S (2004) On the quality gain of $(1, \lambda)$ -ES under fitness noise. In: Yao X, Schwefel H-P et al. (eds) PPSN VIII: Proceedings of the 8th international conference on parallel problem solving from nature, Birmingham. Springer, Berlin, pp 1–10
- Beyer H-G, Meyer-Nieberg S (2005) On the prediction of the solution quality in noisy optimization. In: Wright AH, Vose MD, De Jong KA, Schmitt LM (eds), FOGA 2005: Foundations of genetic algorithms 8. Lecture notes in computer science, vol 3469. Springer, Springer, Berlin, pp 238–259
- Beyer H-G, Meyer-Nieberg S (2006a) Self-adaptation of evolution strategies under noisy fitness evaluations. *Genetic Programming Evolvable Mach* 7(4):295–328
- Beyer H-G, Meyer-Nieberg S (2006b) Self-adaptation on the ridge function class: first results for the sharp ridge. In: Runarsson TP et al. (eds) Parallel problem solving from nature, PPSN IX. Springer, Heidelberg, pp 71–80
- Beyer H-G, Schwefel H-P (2002) Evolution strategies: a comprehensive introduction. *Nat Comput* 1(1):3–52
- Beyer H-G, Sendhoff B (2006) Functions with noise-induced multimodality: a test for evolutionary robust optimization – properties and performance analysis. *IEEE Trans Evol Comput* 10(5):507–526
- Beyer H-G, Sendhoff B (2007a) Evolutionary algorithms in the presence of noise: to sample or not to sample. In: McKay B et al. (eds) Proceedings of the 2007 IEEE symposium on foundations of computational intelligence, Honolulu, HI, April 2007. IEEE Press, Piscataway, NJ, pp 17–23
- Beyer H-G, Sendhoff B (2007b) Robust optimization – a comprehensive survey. *Comput Methods Appl Mech Eng* 197:3190–3218
- Beyer H-G, Olhofer M, Sendhoff B (2003) On the behavior of $(\mu/\mu_t, \lambda)$ -ES optimizing functions disturbed by generalized noise. In: De Jong K, Poli R, Rowe J, (eds) Foundations of genetic algorithms, vol 7. Morgan Kaufmann, San Francisco, CA, pp 307–328
- Beyer H-G, Olhofer M, Sendhoff B (2004) On the impact of systematic noise on the evolutionary optimization performance – a sphere model analysis. *Genetic Programming Evolvable Mach* 5:327–360
- Bienvenüe A, François O (2003) Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. *Theor Comput Sci* 308:269–289
- Braun M (1998) Differential equations and their applications. Springer, Berlin
- Deb K, Beyer H-G (1999) Self-adaptation in real-parameter genetic algorithms with simulated binary crossover. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakela M, Smith RE (eds) Proceedings of the genetic and evolutionary computation conference (GECCO-99), Orlando, FL, July 1999. Morgan Kaufmann, San Francisco, CA, pp 172–179
- Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Natural computing series. Springer, Berlin
- Grünz L, Beyer H-G (1999) Some observations on the interaction of recombination and self-adaptation in evolution strategies. In: Angeline PJ (ed) Proceedings of the CEC'99 conference, Washington, DC, July 1999. IEEE, Piscataway, NJ, pp 639–645
- Hansen N (1998) Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Doctoral thesis, Technical University of Berlin
- Hansen N (2006) The CMA evolution strategy: a comparing review. In: Lozano JA, Larrañaga P, Inza I, Bengoechea E (eds) Towards a new evolutionary computation. Advances in estimation of distribution algorithms. Springer, Berlin, Heidelberg, pp 75–102
- Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: ICEC'96: Proceedings of 1996 IEEE international conference on evolutionary computation, Japan, May 1996. IEEE Press, New York, pp 312–317
- Hansen N, Ostermeier A (1997) Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: the $(\mu/\mu_t, \lambda)$ -CMA-ES. In: Zimmermann H-J (ed) EUFIT'97: 5th European congress on intelligent techniques and soft computing. Aachen, Germany, September, Mainz, pp 650–654
- Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evol Comput* 9(2):159–195
- Hart WE, DeLaurentis JM, Ferguson LA (2003) On the convergence of an implicitly self-adaptive evolutionary algorithm on one-dimensional unimodal problems. *IEEE Trans Evol Comput*
- Hofbauer J, Sigmund K (2002) Evolutionary games and population dynamics. Cambridge University Press, Cambridge
- Jägersküpper J (2003) Analysis of a simple evolutionary algorithm for minimization in euclidean spaces. In: Baeten J et al. (eds) ICALP 2003: Proceedings of the 30th international colloquium on automata,

- languages, and programming, Eindhoven, 2003. Lecture notes in computer science, vol 2719. Springer, Berlin, pp 1068–1079
- Jägersküpper J (2006a) Probabilistic analysis of evolution strategies using isotropic mutations. Ph.D. thesis, Dortmund University
- Jägersküpper J (2006b) Probabilistic runtime analysis of $(1+\lambda)$ -ES using isotropic mutations. In: GECCO'06: Proceedings of the 8th annual conference on genetic and evolutionary computation Seattle, WA, July 2006. ACM, New York, pp 461–468
- Jägersküpper J (2007) Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theor Comput Sci* 379(3):329–347
- Jägersküpper J, Witt C (2005) Rigorous runtime analysis of a $(\mu + 1)$ ES for the sphere function. In: Beyer H-G et al. (eds) GECCO 2005: Proceedings of the genetic and evolutionary computation conference Washington, DC, June 2005. ACM Press, New York, pp 849–856
- Jin Y, Branke J (June 2005) Evolutionary optimization in uncertain environments—a survey. *IEEE Trans Evol Comput* 9(3):303–317
- Kolossa JK (2006) Series approximation methods in statistics. Springer, New York
- Meyer-Nieberg S, Beyer H-G (2005) On the analysis of self-adaptive recombination strategies: first results. In: McKay B et al. (eds) Proceedings of 2005 congress on evolutionary computation (CEC'05), Edinburgh, UK. IEEE Press, Piscataway, NJ, pp 2341–2348
- Meyer-Nieberg S, Beyer H-G (2007) Mutative self-adaptation on the sharp and parabolic ridge. In: Stephens Ch et al. (eds) FOGA 2007: Foundations of genetic algorithms IX, Mexico City, January 2007. Springer Heidelberg, pp 70–96
- Meyer-Nieberg S, Beyer H-G (2008) Why noise may be good: additive noise on the sharp ridge. In: Keijzer M et al. (eds) GECCO 2008: Proceedings of the 10th annual conference on genetic and evolutionary computation, Atlanta, GA, July 2008. ACM, New York, pp 511–518
- Ostermeier A, Gawelczyk A, Hansen N (1995) A randomized approach to self-adaptation of evolution strategies. *Evol Comput* 2(4):369–380
- Oyman AI (1999) Convergence behavior of evolution strategies on ridge functions. Ph.D. Thesis, University of Dortmund
- Oyman AI, Beyer H-G (2000) Analysis of the $(\mu/\mu, \lambda)$ -ES on the parabolic ridge. *Evol Comput* 8(3):267–289
- Oyman AI, Beyer H-G, Schwefel H-P (1998) Where elitists start limping: evolution strategies at ridge functions. In: Eiben AE, Bäck T, Schoenauer M, Schwefel H-P (eds) Parallel problem solving from nature, vol 5. Springer, Heidelberg, pp 34–43
- Oyman AI, Beyer H-G, Schwefel H-P (2000) Analysis of a simple ES on the “parabolic ridge.” *Evol Comput*, 8(3):249–265
- Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart
- Schwefel H-P (1977) Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Interdisciplinary systems research; 26. Birkhäuser, Basel
- Semenov MA (2002) Convergence velocity of evolutionary algorithms with self-adaptation. In: Langdon WB et al. (eds), GECCO 2002: Proceeding of the genetic and evolutionary computation conference. Morgan Kaufmann, New York City, July 2002, pp 210–213
- Semenov MA, Terkel DA (2003) Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic Lyapunov function. *Evol Comput* 11(4):363–379
- Sendhoff B, Beyer H-G, Olhofer M (2002) On noise induced multi-modality in evolutionary algorithms. In: Wang L, Tan KC, Furuhashi T, Kim J-H, Sattar F (eds), Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning – SEAL, vol 1, Singapore, November 18–22, pp 219–224
- Wiggins S (1990) Introduction to applied nonlinear dynamical systems and chaos. Springer, New York

26 Computational Complexity of Evolutionary Algorithms

Thomas Jansen

Department of Computer Science, University College Cork, Ireland
t.jansen@cs.ucc.ie

1	<i>Introduction and Motivation</i>	816
2	<i>Evolutionary Algorithms</i>	817
3	<i>Fundamental Limits</i>	820
4	<i>Methods for the Analysis of Evolutionary Algorithms</i>	825
5	<i>Summary</i>	842

Abstract

When applying evolutionary algorithms to the task of optimization it is important to have a clear understanding of their capabilities and limitations. By analyzing the optimization time of various variants of evolutionary algorithms for classes of concrete optimization problems, important insights can be gained about what makes problems easy or hard for these heuristics. Still more important than the derivation of such specific results is the development of methods that facilitate rigorous analysis and enable researchers to derive such results for new variants of evolutionary algorithms and more complex problems. The development of such methods and analytical tools is a significant and very active area of research. An overview of important methods and their foundations is presented together with exemplary applications. This enables one to apply these methods to concrete problems and participate in the theoretical foundation of evolutionary computing.

1 Introduction and Motivation

Evolutionary algorithms form a large and loosely defined class of algorithms, they can be and are used for many different purposes. Even though it can be argued that they are not really suited for optimization in the strong sense (De Jong 1992), optimization remains one of the main fields of application. There they compete with other optimization methods and can be classified as randomized optimization heuristics, while in the more general sense the label randomized search heuristic is more appropriate. (It can be argued that it would be more appropriate to speak of probabilistic heuristics since heuristics is not dealt with where randomization is introduced later but with heuristics where randomization is a central concept. However, since the notion of randomized algorithms is well established, this notion is adopted here, too.) When the performance of evolutionary algorithms is to be assessed in the context of optimization, it makes most sense to consider evolutionary algorithms as randomized *algorithms*, adopting the perspective that is common in the community concerned with the design and analysis of (randomized) algorithms (Cormen et al. 2001; Mitzenmacher and Upfal 2005; Motwani and Raghavan 1995). This implies that we perform rigorous analysis and present strictly proven results based on no unproven assumptions or simplifications. While such a rigorous approach makes it difficult to obtain results for complex evolutionary algorithms and complex optimization problems, it has several advantages that in summation make it much more attractive than a purely experimental approach or heuristic analyses based on reasonable yet unproven assumptions.

The most prominent and important advantage of the rigorous analytical approach is that it provides foundations, a firm ground of knowledge that can serve as a trustworthy basis for design and analysis of evolutionary algorithms. In most cases, the proofs of analytical results provide a level of insight and give leverage to understand the working principles of evolutionary algorithms that is difficult if not impossible to obtain in other ways. It advances common beliefs about evolutionary algorithms to proven facts and knowledge, advancing the field to a well-founded science. The results derived with the methods presented here come with a degree of generality that is impossible to achieve with experiments alone. Moreover, the methods themselves turn out to be applicable to a wide range of evolutionary algorithms and many different fitness functions. Another important aspect that is dealt with briefly is oriented toward complexity theory, that is, the derivation of fundamental lower bounds that hold for

all algorithms operating within the considered scenario. This reveals fundamental limitations and can in practice save a lot of time since no effort is wasted for provably impossible tasks. Finally, a completely different aspect of equal importance is the usefulness of rigorously proven results in teaching. By learning analytical tools and methods for the analysis of evolutionary algorithms, students can develop a deep understanding and the ability to design and analyze their own algorithms in their own contexts.

It can be remarked that computational complexity of evolutionary algorithms can be taken to mean something entirely different, namely, aspects of the efficient implementation of evolutionary algorithms. While it is true that for some modules of evolutionary algorithms there are implementations known that are considerably more efficient than the obvious ones this is not at all the focus of this chapter. (One example is implementing standard bit mutations where each of n bits flips with probability $1/n$ by randomly determining the position of the next bit to be flipped. This decreases the (expected) number of random experiments from n to just 1.) While in concrete cases efficient implementations can decrease the run time considerably, in general, evolutionary algorithms are rather easy and straightforward to implement. In practical applications, often most time is spent in the evaluation of a complex fitness function so that the computational effort spent on the evolutionary algorithm's modules becomes negligible. This motivates that the analysis can be simplified by considering the number of evaluations of the fitness functions as an appropriate measure for the actual run time. This is more accurate than the number of generations, another measure that is often used, where the effect of the population size on the run time cannot be assessed. One has to be cautious, however, when dealing with complex evolutionary algorithms or hybridizations of evolutionary algorithms where the computational effort of the algorithm itself, apart from evaluations of the fitness function, becomes considerable or even dominant.

In the next section, we give a brief outline of the kind of evolutionary algorithms we consider and we describe the perspective adopted for analysis. Fundamental limitations that are inherent to the specific scenario that usually governs the application of evolutionary algorithms are discussed in [Sect. 3](#). There are two ways of finding and describing such limits that are fundamentally different, namely, “no free lunch” arguments and black-box complexity. A brief introduction to both concepts will be given. [Section 4](#) is devoted to the description of methods for the analysis of evolutionary algorithms. Clearly, this section contains the most important and hopefully useful parts of this chapter. Finally, this chapter concludes with a summary in [Sect. 5](#).

2 Evolutionary Algorithms

While there is a large amount of different variants of evolutionary algorithms with new and related paradigms such as particle swarm optimization or ant colony emerging, here we restrict ourselves to simplified and rather basic algorithms. This restriction facilitates the analysis and allows for a clearer identification of causes for the effects that are observed. Yet, it is not too restrictive: it turns out that the methods, which are developed with a specific and quite restricted evolutionary algorithm in mind, can often be generalized and extended to more complex evolutionary algorithms.

The most basic evolutionary algorithm considered is the so-called $(1 + 1)$ EA. Its extreme simplicity makes it attractive for formal analysis, hence it shares important typical properties with much more complex evolutionary algorithms. Here we present it without any stopping

criterion as this subject is typically not dealt with in the context of computational complexity of evolutionary algorithms.

Algorithm 1 (the $(1 + 1)$ EA)

1. Initialization
 $t := 0$

Choose $x_t \in \{0, 1\}^n$ uniformly at random.

2. Repeat forever
3. Mutation

Create $y := x_t$. Independently for each $i \in \{1, 2, \dots, n\}$, flip the bit $y[i]$ with probability $1/n$.

4. Selection

If $f(y) \geq f(x_t)$ then $x_{t+1} := y$ else $x_t := y$.

 $t := t + 1$

The mutation operator employed in line 3 of the $(1 + 1)$ EA (Algorithm 1) is called standard bit mutation with mutation probability $1/n$. While $1/n$ is the most recommended choice, it is known that it can be far from optimal for some fitness functions (Jansen and Wegener 2000). When this mutation operator is replaced by a local operator that flips exactly one bit, we obtain an algorithm that is known as *randomized local search* (RLS). While being close to the $(1+1)$ EA we consider RLS not to be an evolutionary algorithm since evolutionary algorithms are supposed to be able to perform a global search in the search space and randomized local search, by definition, is restricted to a local search. Since the $(1 + 1)$ EA with mutation probability $1/n$ flips on average one bit in each mutation one may

Algorithm 2 (the $(\mu + \lambda)$ EA)

1. Initialization
 $t := 0$

Choose $x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)} \in \{0, 1\}^n$ independently uniformly at random.

2. Repeat forever
3. For $i \in \{1, 2, \dots, \lambda\}$ **do**
4. Selection

Select $z \in \{x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)}\}$ uniformly at random.

5. Crossover

With probability p_c

 $z' := z$

Select $z'' \in \{x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)}\}$ uniformly at random.

 $z := \text{crossover}(z', z'')$
6. Mutation

Create $y^{(i)} := z$. Independently for each $j \in \{1, 2, \dots, n\}$, flip the bit $y^{(i)}[j]$ with probability $1/n$.

7. Selection

Sort $x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)}, y^{(1)}, y^{(2)}, \dots, y^{(\lambda)}$ with respect to fitness in decreasing order, in case of equal fitness sort offspring y before parents x . Let the first μ points be the sequence $x_{t+1}^{(1)}, x_{t+1}^{(2)}, \dots, x_{t+1}^{(\mu)}$.

speculate that the two algorithms behave similarly. It is known that this is indeed the case for many problems but that they differ in extreme ways on other problems and that it is difficult to give a precise description of problems where the two algorithms provably perform similarly (Doerr et al. 2008).

In general, evolutionary algorithms make use of a larger population size and produce more than one offspring per generation. A still very simple evolutionary algorithm that has both properties is described. Since the population size is larger than one, it can also make use of crossover.

It can be seen that the $(1 + 1)$ EA is the special case of the $(\mu + \lambda)$ EA that we obtain by setting $\mu = \lambda = 1$. As crossover operator, any operator acting on binary strings may be used. Uniform crossover will be considered. It is also possible to consider the $(\mu + \lambda)$ EA without crossover, that is, set the probability for crossover to 0, $p_c = 0$.

Finally, a third kind of evolutionary algorithm called (μ, λ) EA is introduced. It is similar to the $(\mu + \lambda)$ EA but employs a different selection at the end of each generation. The new population is selected from the offspring alone, the old generation is discarded in any case. Clearly, this implies that

$$\max \left\{ x_t^{(i)} \mid i \in \{1, 2, \dots, \mu\} \right\}$$

may decrease with increasing t . A formal description is given just as was done for the other algorithms. Note that $\lambda \geq \mu$ is needed here.

Algorithm 3 (the (μ, λ) EA)

1. Initialization

$t := 0$

Choose $x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)} \in \{0, 1\}^n$ independently uniformly at random.

2. Repeat forever

3. For $i \in \{1, 2, \dots, \lambda\}$ do

4. Selection

Select $z \in \{x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)}\}$ uniformly at random.

5. Crossover

With probability p_c

$z' := z$

Select $z'' \in \{x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(\mu)}\}$ uniformly at random.

$z := \text{crossover}(z', z'')$

6. Mutation

Create $y^{(i)} := z$. Independently for each $j \in \{1, 2, \dots, n\}$,

flip the bit $y^{(i)}[j]$ with probability $1/n$.

7. Selection

Sort $y^{(1)}, y^{(2)}, \dots, y^{(\lambda)}$ with respect to fitness in decreasing order.

Let the first μ points be the sequence $x_{t+1}^{(1)}, x_{t+1}^{(2)}, \dots, x_{t+1}^{(\mu)}$.

$t := t + 1$

Since in our description of evolutionary algorithms they never stop it is not obvious how we want to measure their computational complexity. We decide to consider the first point of time (measured by the number of evaluations of the fitness function) when a global optimum is sampled. We call this the *optimization time*. Note that it is usually not

difficult to change this into other notions of achieving a goal like approximation. For the $(1+1)$ EA, the optimization time on a fitness function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ is given by the following equation.

$$T_{(1+1) \text{ EA}, f} = \min\{t \mid f(x_t) = \max\{f(x') \mid x' \in \{0, 1\}^n\}\}$$

For the $(\mu + \lambda)$ EA, we are slightly less accurate when defining the optimization time. We define it based on the number of generations (but still close to the actual number of f -evaluations) in the following way.

$$T_{(\mu+\lambda) \text{ EA}, f} = \min\left\{\mu + \lambda t \mid \max\left\{f\left(x_t^{(i)}\right) \mid i \in \{1, \dots, \mu\}\right\} = \max\{f(x) \mid x \in \{0, 1\}^n\}\right\}$$

For the (μ, λ) EA we use the same definition, $T_{(\mu, \lambda) \text{ EA}, f} = T_{(\mu+\lambda) \text{ EA}, f}$.

Clearly, $T_{A,f}$ is a random variable and we are mostly interested in its expectation $E(T_{A,f})$. In cases where this expected value is misleading due to exceptionally large values that occur only with small probabilities other properties are of interest. In particular, bounds on $\text{Prob}(T_{A,f} \leq t_u(n))$ and $\text{Prob}(T_{A,f} \geq t_l(n))$ for some upper bound t_u and some lower bound t_l are often of interest. One can consider an algorithm to be efficient if either $E(T_{(1+1) \text{ EA}, f})$ is polynomially bounded from above or $\text{Prob}(T_{A,f} \leq t_u(n)) \geq 1/p(n)$ holds for some polynomials $t_u(n)$ and $p(n)$. The reason for this rather generous notion of efficiency can be found in restarts: If one knows that $\text{Prob}(T_{A,f} \leq t_u(n)) \geq 1/p(n)$ holds for some evolutionary algorithm A , one can turn it into an evolutionary algorithm with polynomial expected optimization time at most $t_u(n)p(n)$ by stopping A after $t_u(n)$ fitness evaluations and restarting it.

In addition to considering simple evolutionary algorithms, the analysis is further simplified by not analyzing the optimization time with arbitrary precision but by restricting ourselves to an asymptotic analysis taking into account the dominating factors. One can make use of the well-known notions for the description of the order of growth of functions (see, e.g., Cormen et al. 2001) and perform analysis for a growing dimension of the search space n . Thus, the dimension of the search space n plays the same role as the length of the input in the analysis of problem-specific algorithms.

3 Fundamental Limits

Evolutionary algorithms are considered in the context of optimizing some unknown fitness function $f: S \rightarrow R$ where S is some finite search space and R typically is a subset of \mathbb{R} . Since S is finite it usually does not hurt to assume that R is finite, too. When looking for fundamental limits on the performance of evolutionary algorithms (and all other algorithms working in the same scenario) one has to be careful about what exactly the algorithm can make use of. It is assumed that the algorithm designer knows that the algorithm is going to be applied to some unknown function $f \in \mathcal{F} \subseteq \{g: S \rightarrow R\}$. So, any problem-specific knowledge the algorithm designer may have is in the framework expressed in this knowledge about the set of potential fitness functions \mathcal{F} . An algorithm is required to find some optimal point $s \in S$, that is, some $s \in S$ with $f(s) = \max\{f(x) \mid x \in S\}$. One can assume that the algorithm can make any use of the fact that the concrete fitness function f belongs to \mathcal{F} , $f \in \mathcal{F}$, but has no other way of obtaining knowledge about the concrete fitness function f but to query the function value $f(x)$ for arbitrary $x \in S$. The number of such queries (we call f -evaluations) are counted until an optimum is found for the first time. Note that we do not expect the algorithm to know or to

notice that it found an optimum. This is the scenario that we consider and call *black-box optimization*. We call an algorithm A a *black-box algorithm* for \mathcal{F} if for each function $f \in \mathcal{F}$ it finds an optimum of f within at most t f -evaluations with probability converging to 1 with t going to infinity, $\forall f \in \mathcal{F} : \lim_{t \rightarrow \infty} \text{Prob}(T_{A,f} \leq t) = 1$.

Adopting the usual worst-case perspective of computer science, one is interested in the best performance any algorithm can achieve for such a set of potential fitness functions \mathcal{F} where performance is taken to be the worst-case performance. Formally, one can define for a class of functions \mathcal{F} and a black-box algorithm for \mathcal{F} the *worst case expected optimization time*

$$T_{A,\mathcal{F}} = \max \{ E(T_{A,f}) \mid f \in \mathcal{F} \}$$

Finally, one can define for $\mathcal{F} \subseteq \{f : S \rightarrow R\}$ the *black-box complexity* $B_{\mathcal{F}}$ of \mathcal{F} :

$$B_{\mathcal{F}} = \min \{ T_{A,\mathcal{F}} \mid A \text{ is a black-box algorithm for } \mathcal{F} \}$$

It is not difficult to see that restricting the attention to f -evaluations alone can lead to cases where NP-hard function classes \mathcal{F} have only polynomial black-box complexity (Droste et al. 2006): if a polynomial number of f -evaluations is sufficient to reconstruct the concrete problem instance at hand an optimal solution can be computed (in exponential time) without any further f -evaluations. This seeming weakness is more than compensated by the fact that in contrast to classical complexity theory one is able to derive absolute lower bounds that are not based on any complexity theoretical assumption like $P \neq NP$.

Another noteworthy consequence from the definition is that for any class of functions \mathcal{F} we have $B_{\mathcal{F}} \leq |\mathcal{F}|$: a black-box algorithm knows about all $f \in \mathcal{F}$, this includes knowledge about the global optima. Thus, it can sample for each function one of its global optima one by one sampling an optimum of the unknown objective function $f \in \mathcal{F}$ after at most $|\mathcal{F}|$ steps. Moreover, we have

$$\mathcal{F} \subseteq \mathcal{F}' \Rightarrow B_{\mathcal{F}} \leq B_{\mathcal{F}'}$$

since for the subset \mathcal{F} the maximum in the definition of $T_{A,\mathcal{F}}$ is taken over a subset of functions.

Attention is restricted to deterministic black-box algorithms for a moment. Such algorithms can be described as a tree: The first point in the search space $x \in S$ they sample is the root of the tree. The second point the algorithm decides to sample is in general based on its f -value $f(x)$ so that there are up to $|R|$ possible second points. We represent this in the tree by attaching each second point as child to the root. We mark the corresponding edge with the f -value that led to this choice. Clearly, this way the root can become a node of degree at most $|R|$. Continuing this way by adding the third points the algorithm samples (depending on the first two points together with their function values) to the appropriate node we see that any deterministic black-box algorithm can be described as an $|R|$ -ary tree. If we are interested in optimal black-box algorithms, it is safe to assume that on each path from the root to a leaf each point from S is encountered at most once. Thus, the trees have finite size. If necessary, one can enlarge the trees such that each path from the root to a leaf contains exactly $|S|$ nodes. If the criterion is not changed for assessing the performance of an algorithm, this formal change does not change its performance and may make a formal treatment simpler.

Considering deterministic black-box algorithms as trees we recognize that due to their finite size and finite possible markings at the nodes and edges there is only a finite number

of such algorithms. In such cases, we describe any randomized black-box algorithm as a probability distribution over the deterministic black-box algorithms. From an algorithmic perspective, the idea is to modify randomized black-box algorithms in a way that they begin with making a sufficiently large number of random coin tosses, storing the outcomes in memory. After that, they work deterministically and every time the original randomized algorithm needed to make a random decision the modified version acts deterministically using the stored outcome of the random experiment.

3.1 “No Free Lunch”

For a fixed set of functions \mathcal{F} , in general $T_{A,\mathcal{F}}$ depends on the black-box algorithm A under consideration. There are, however, classes of functions, where this value is equal regardless of the choice of A – provided that we consider only distinct f -evaluations or, equivalently, black-box algorithms that do not revisit any point in the search space. Thinking of constant fitness functions this is clearly not surprising at all. But it is quite remarkable that this also holds for the class of all functions $\mathcal{F} = \{f : S \rightarrow R\}$. When this was first published (Wolpert and Macready 1997), using the colorful label “no free lunch theorem,” it caused enormous discussions about its scope and meaning. If on average over all problems all algorithms perform equal, what sense does it make to design any specific randomized search heuristic? The result can even be strengthened. It is not necessary that one averages only over constant functions or over all functions. The same result can be proven if the set of functions \mathcal{F} has the property to be closed under permutations of the search space. For a function $f: S \rightarrow R$ we can define the function $\sigma f: S \rightarrow R$ by $\sigma f(x) = f(\sigma^{-1}(x))$ for a permutation σ of S . A class of functions \mathcal{F} is called *closed under permutations* of the search space if $f \in \mathcal{F}$ implies $\sigma f \in \mathcal{F}$ for all functions $f \in \mathcal{F}$ and all permutations σ . Moreover, it can be shown that being closed under permutations is not only a sufficient but also a necessary condition for a set \mathcal{F} to allow for a “no free lunch” result. Note that it is not necessary to consider the average over the class of functions \mathcal{F} . One can make precise statements over the existence of “no free lunch” results for probability distributions over \mathcal{F} different from the uniform distribution leading to the most general “no free lunch theorem” known (Igel and Toussaint 2004). Since the generalization is not too surprising, one can stick to the simpler result (Schumacher et al. 2001). In order to formulate the result precisely, the notion of a *performance measure* is needed.

Remember that any randomized black-box algorithm can be described as a probability distribution over the set of all deterministic black-box algorithms. Moreover, remember that any deterministic black-box algorithm can be described as an $|R|$ -ary tree where each path from the root to a leaf contains exactly $|S|$ nodes. When one such deterministic black-box algorithm A and some function $f \in \mathcal{F}$ are fixed, this defines a path in the tree that one can call a *trace* $T(A, f) = ((x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_{|S|}, f(x_{|S|})))$. This trace can be projected onto a vector of function values $V(A, f) = (f(x_1), f(x_2), \dots, f(x_{|S|}))$. If it is agreed that reasonable performance measures should only depend on the function values and not on the search points themselves then one can give a formal definition of a performance measure as a function mapping such vectors of function values to \mathbb{R} . Clearly, important performance measures, in particular the performance measure one can consider (the first point of time when a global maximum is found), can be defined in this way. Using this formal definition of a performance measure one can obtain the following result that is cited without a proof.

Theorem 1 (NFL Theorem (Schumacher et al. 2001)) *Let S, R be two finite sets. On average over all functions $f \in \mathcal{F} \subseteq \{g: S \rightarrow R\}$ all black-box algorithms perform equal for all performance measures $M: \{V|V = V(A, h)\} \rightarrow \mathbb{R}$ if and only if the set of functions \mathcal{F} is closed under permutations of the search space.*

While this result has the obvious consequence that there is no way one can develop meaningful algorithms for function classes \mathcal{F} that are closed under permutations, it is not so clear what the relevance of this result is. We mention that it is unusual for a class of functions to be closed under permutations. The fraction of classes where this is the case is exponentially small (Igel and Toussaint 2003). Moreover, classes of functions that are defined via some nontrivial notion of a neighborhood cannot be closed under permutations since neighborhoods are not preserved under permutations of the search space. On the other hand, it is not difficult to see that each black-box algorithm that performs well on a specific function performs very poorly on an exponentially large number of similar functions with probability very close to one (Droste et al. 2002b).

3.2 Black-Box Complexity

Much more interesting than the fundamental but in their relevance severely limited “no free lunch” results are concrete results on the black-box complexity $B_{\mathcal{F}}$ for some relevant class of functions \mathcal{F} . One can obtain upper bounds on $B_{\mathcal{F}}$ by proving that some specific black-box algorithms optimizes on average each function $f \in \mathcal{F}$ within some bounded number of f -evaluations. Lower bounds, however, are much more difficult to obtain. They require a formal proof that no algorithm at all is able to optimize all functions from \mathcal{F} with less than a certain number of f -evaluations. If one sticks to this assumption that S and R are finite sets, though, Yao’s minimax principle is a very useful tool to prove such lower bounds.

Theorem 2 (Yao’s Minimax Principle (Motwani and Raghavan 1995; Yao 1977)) *If a problem consists of a finite set of instances of a fixed size and allows a finite set of deterministic algorithms, the minimal worst-case instance expected optimization time of a randomized algorithm is bounded below for each probability distribution on the instances by the expected optimization time of an optimal deterministic algorithm.*

Yao’s minimax principle simplifies the proofs of lower bounds in two ways: First, there is no need to analyze a randomized algorithm but we can restrict our attention to deterministic algorithms. This, typically, is much simpler. Second, one is free to define a distribution over the class of functions \mathcal{F} any way one likes. Two concrete examples will be considered to see how such results can be derived. To this end, two example functions and a general way of creating a class of functions based on a single fitness function are discussed.

Definition 1 Let $n \in \mathbb{N}$ be given. The function $\text{ONEMAX}: \{0, 1\}^n \rightarrow \mathbb{R}$ is defined by $\text{OneMAX}(x) = \sum_{i=1}^n x[i]$. The function $\text{NEEDLE}: \{0, 1\}^n \rightarrow \mathbb{R}$ is defined by $\text{NEEDLE}(x) = \prod_{i=1}^n x[i]$.

For any function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ and any $a \in \{0, 1\}^n$ let $f_a: \{0, 1\}^n \rightarrow \mathbb{R}$ be defined by $f_a(x) = f(x \oplus a)$, where $x \oplus a$ denotes the bit-wise exclusive or of x and a . For any function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ let $f^* := \{f_a | a \in \{0, 1\}^n\}$.

The very well-known example function ONEMAX yields as function value the number of ones in the bit string. Its generalization ONEMAX_a can be described as maximizing the Hamming distance to the bit-wise complement of the global optimum \bar{a} , where the fitness value exactly equals this Hamming distance. Clearly, ONEMAX is rather easy to optimize since it gives clear hints into the direction of the unique global optimum. On the other, the example function NEEDLE is a flat plateau of fitness 0 with a single peek at the all ones bit string 1^n . Its generalization NEEDLE_a has this unique global optimum at \bar{a} . Since NEEDLE_a gives no hints at all with respect to the global optimum, it is rather difficult to optimize.

It is remarked that NEEDLE^* is closed under permutations of the search space. It follows from the NFL theorem (Theorem 1) that all algorithms that do not revisit any points make the same number of f -evaluations on NEEDLE^* . We consider a deterministic algorithm that enumerates the search space $\{0, 1\}^n$ in some fixed order. Clearly, for each $i \in \{1, 2, \dots, 2^n\}$, there is exactly one function NEEDLE_a that is optimized by this algorithm with the i th f -evaluation. Thus, the average number of f -evaluations this algorithm (and due to \blacktriangleright Theorem 1 any algorithm) makes on NEEDLE^* equals

$$\sum_{i=1}^{2^n} \frac{i}{2^n} = \frac{2^n(2^n + 1)}{2 \cdot 2^n} = 2^{n-1} + \frac{1}{2}$$

and we obtain $B_{\mathcal{F}} = 2^{n-1} + \frac{1}{2}$. Nevertheless, the lower bound will be derived in the proof of the next theorem to see a very simple application of Yao's minimax principle (\blacktriangleright Theorem 2).

Theorem 3 (Droste et al. 2006) $B_{\text{NEEDLE}^*} = 2^{n-1} + 1/2$. $B_{\text{ONEMAX}^*} = \Omega(n/\log n)$

Proof The upper bound for B_{NEEDLE^*} follows from the analysis of the deterministic algorithm enumerating the search space $\{0, 1\}^n$. For the lower bound, we use the uniform distribution over NEEDLE^* and consider a deterministic black-box algorithm A for NEEDLE^* . Clearly, such an algorithm is equivalent to a tree with at least 2^n nodes since otherwise there is some $x \in \{0, 1\}^n$ that is not a label of any node in this tree. Any function that has this x as its unique global optimum cannot be optimized by A . Since x is the unique global optimum of $\text{NEEDLE}_{\bar{x}}$, we see that we have at least 2^n nodes as claimed. Each node in the tree of an optimal black-box algorithm has at most one child. The only two possible function values are 0 and 1. After making an f -evaluation for some x with function value 1 the algorithm terminates since the global optimum is found. Thus, the tree is actually a list of 2^n nodes. Clearly, the average number of function evaluations made by this algorithm equals

$$\sum_{i=1}^{2^n} \frac{i}{2^n} = \frac{2^n(2^n + 1)}{2 \cdot 2^n} = 2^{n-1} + \frac{1}{2}$$

and the lower bound follows.

For ONEMAX^* one can argue in a similar way, again using the uniform distribution over the class of functions under consideration. Any black-box algorithm for ONEMAX^* needs to contain at least 2^n nodes since otherwise there is some function ONEMAX_a that cannot be optimized by this algorithm. Now there are $n+1$ different function values for ONEMAX_a . After sampling a point with function value n any optimal deterministic black-box algorithm for

ONEMAX* can terminate. Thus, the degree of each node in the tree of such an algorithm is bounded above by n . Since an n -ary tree of height h contains at most

$$\sum_{i=0}^h n^i = n^{h+1} - 1$$

nodes we have $n^{h+1} - 1$ and $h = \Omega(n/\log n)$ follows. Since the average height of this tree is still $\Omega(n/\log n)$, $\Omega(n/\log n)$ is obtained as lower bound on B_{ONEMAX^*} .

Results on black-box complexity are not restricted to results for example functions and classes of functions obtained by generalizing such example functions. It is possible to derive nontrivial lower bounds on the black-box complexity of combinatorial optimization problems such as sorting and shortest paths. Moreover, for the class of unimodal problems an exponential lower bound of 2^{n^ε} for any positive constant $\varepsilon < 1$ is known (Droste et al. 2006).

4 Methods for the Analysis of Evolutionary Algorithms

The analysis of evolutionary algorithms is not so much concerned with the derivation of general lower bounds on the difficulty of problems as in black-box complexity but with proving concrete bounds on the (expected) optimization time of specific evolutionary algorithms for specific problems. While at the end of the day one is interested in results for one's own evolutionary algorithm on one's own specific problem or, from a broader perspective, in results on classes of evolutionary algorithms on relevant practical problems, it is worthwhile to start with very simple evolutionary algorithms on very simple example functions. The aim of this section is not to present the most advanced results obtained by clever applications of the most advanced methods. Here we aim at presenting the most important and useful methods in an accessible way following a hands-on kind of approach. The goal is to enable the reader of this section to apply these methods himself or herself and participate in the development of more advanced results and methods.

4.1 Fitness-Based Partitions

We begin with a very simple and basic method for the derivation of upper bounds on the expected optimization time. In spite of its simplicity, this method, called the method of fitness-based partitions, is very helpful and delivers strong bounds in many cases. Consider the $(1 + 1)$ EA (Algorithm 1) on some arbitrary fitness function $f: \{0, 1\}^n \rightarrow \mathbb{R}$. Due to the strict selection employed, the fitness value $f(x_t)$ is increasing with t , though not strictly, of course. Clearly, $E(T_{(1+1)\text{EA},f})$ equals the expected number of f -evaluations until the function value of the current search point x_t is increased to the maximum value for the first time. This together with the increasing nature of $f(x_t)$ leads one to the idea to structure the optimization of f by the $(1 + 1)$ EA in the following way. The search space is partitioned into disjoint sets of points with similar fitness values. We obtain an upper bound on $E(T_{(1+1)\text{EA},f})$ by summing up the average waiting times for leaving these sets. This idea is made more precise in the following definition and theorem.

Definition 2 Let $f: \{0, 1\}^n \rightarrow \mathbb{R}$ be some fitness function. L_0, L_1, \dots, L_k (for $k \in \mathbb{N}$) is called an f -based partition if the following properties hold.

1. $L_0 \cup L_1 \cup \dots \cup L_k = \{0, 1\}^n$
2. $\forall i < j \in \{0, 1, \dots, k\}: \forall x \in L_i, y \in L_j: f(x) < f(y)$
3. $L_k = \{x \in \{0, 1\}^n \mid f(x) = \max\{f(y) \mid y \in \{0, 1\}^n\}\}$

Due to the strict selection, if the $(1 + 1)$ EA leaves some set L_i it can only reach a point in some set L_j with $j > i$. Moreover, with reaching L_k a global optimum is found. These observations lead directly to the following theorem.

Theorem 4 *For $f: \{0, 1\}^n \rightarrow \mathbb{R}$ be some fitness function. Let L_0, L_1, \dots, L_k be some f -based partition. The probability of leaving L_i is defined as*

$$s_i := \min \left\{ \sum_{j=i+1}^k \sum_{y \in L_j} \left(\frac{1}{n}\right)^{H(x,y)} \left(1 - \frac{1}{n}\right)^{n-H(x,y)} \mid x \in L_i \right\}$$

where $H(x, y)$ denotes the Hamming distance of $x, y \in \{0, 1\}^n$

For the expected optimization time of the $(1 + 1)$ EA on f , the following holds:

$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq \sum_{i=0}^{k-1} \frac{1}{s_i}$$

Proof We observe that for any $x \in L_i$

$$\sum_{j=i+1}^k \sum_{y \in L_j} \left(\frac{1}{n}\right)^{H(x,y)} \left(1 - \frac{1}{n}\right)^{n-H(x,y)}$$

equals the probability to mutate $x \in L_i$ to some $y \in L_j$ with $j > i$. Thus, $f(y) > f(x)$ holds and such an event implies that the $(1 + 1)$ EA leaves L_i . Thus, s_i is a lower bound on the actual probability to leave the current set L_i by means of the next mutation. The expected waiting time for such an event is geometrically distributed with expectation $1/s_i$. Since each fitness layer needs to be left at most once

$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq \sum_{i=0}^{k-1} \frac{1}{s_i}$$

follows.

This method will be applied to the analysis of OneMax. Note that since the $(1 + 1)$ EA is completely symmetric with respect to 0- and 1-bits, the same result holds for any OneMax_a . We use the trivial f -based partition, namely, defining

$$L_i = \{x \in \{0, 1\}^n \mid \text{ONEMAX}(x) = i\}, i \in \{0, 1, \dots, n\}$$

to have one fitness layer for each of the $n + 1$ fitness values. It is observed that in order to leave L_i , it suffices to flip exactly one of the $n - i$ 0-bits in one mutation. Thus,

$$s_i \geq \binom{n-i}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{en}$$

follows and we obtain

$$\mathbb{E}(T_{(1+1) \text{ EA}, \text{OneMAX}}) \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^n \frac{1}{i} = O(n \log n)$$

as an upper bound. In comparison with the lower bound obtained by means of black-box complexity, it can be seen that this upper bound is off by a factor of at most $O(\log^2 n)$. It will be seen in [Sect. 4.2](#) that it is even tight.

We consider the function $\text{BinVal}: \{0, 1\}^n \rightarrow \mathbb{R}$ defined by

$$\text{BINVAL}(x) = \sum_{i=1}^n x[i] 2^{n-i-1}$$

We observe that $x \in \{0, 1\}^n$ yields as function value the integer that has x as its binary representation. Using the trivial f -based partition, namely

$$L_i = \{x \in \{0, 1\}^n \mid \text{BINVAL}(x) = i\}, i \in \{0, 1, \dots, 2^n - 1\}$$

we obtain 2^n fitness layers. Since it is obvious that at least one generation is needed to leave each layer, the best one can hope for is $2^n - 1$ as upper bound. We learn that a large number of fitness layers cannot be used if small upper bounds are to be proved.

We try to be a bit less naive and define

$$L_i = \left\{ x \in \{0, 1\}^n \setminus \bigcup_{j=0}^{i-1} L_j \mid \text{BINVAL}(x) < \sum_{j=0}^i 2^{n-1-j} \right\}, i \in \{0, 1, \dots, n\}$$

as f -based partition for BINVAL . We observe that L_i contains all $x \in \{0, 1\}^n$ that start with a sequence of i consecutive 1-bits followed by a 0-bit, so L_0 contains exactly all $x \in \{0, 1\}^n$ where the leftmost bit is 0 and $L_n = \{1^n\}$. Thus, for each L_i with $i < n$, it suffices to flip the leftmost 0-bit in order to leave L_i . This implies

$$s_i \geq \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$$

as lower bound on s_i and we obtain

$$\mathbb{E}(T_{(1+1) \text{ EA}, \text{BINVAL}}) \leq \sum_{i=0}^{n-1} en = en^2 = O(n^2)$$

as upper bound for the expected optimization time of the $(1 + 1)$ EA on BINVAL . While being much improved over $2^n - 1$ it will be seen in [Sect. 4.4](#) that this is still not tight.

Fitness-based partitions do also work for families of functions with similar structure. To see an example, we consider $\text{JUMP}_k: \{0, 1\}^n \rightarrow \mathbb{R}$ (Droste et al. 2002a) with

$$\text{JUMP}_k(x) = \begin{cases} n - \text{OneMax}(x) & \text{if } n - k < \text{OneMax}(x) < n \\ k + \text{OneMax}(x) & \text{otherwise} \end{cases}$$

where $k \in \{1, 2, \dots, n\}$ is a parameter. These n different fitness functions all have the same main properties. The unique global optimum is the all one string 1^n , all points with exactly $n - k$ 1-bits are local optima (except for $k = 1$, since $\text{JUMP}_1 = 1 + \text{OneMax}$ and there is no local optimum that is not also a global one). For $x \in \{0, 1\}^n$ with less than $n - k$ 1-bits the function

equals $k + \text{ONEMAX}(x)$ so in that region, the local optima can be found in time $O(n \log n)$. There a direct mutation to the global optimum is necessary; such a mutation flips exactly all k remaining 0-bits. An upper bound on $E(T_{(1+1) \text{ EA}, \text{JUMP}_k})$ is obtained using the trivial f -based partition where each fitness value gets its own fitness layer L_i by

$$L_i = \{x \in \{0, 1\}^n \mid \text{JUMP}_k(x) = i\}, i \in \{1, 2, \dots, n\}$$

and $L_{n+1} = \{1^n\}$. For all $i < k$ we have $n - k < \text{ONEMAX}(x) < n$ for all $x \in L_i$. Thus, for these i it suffices to flip one of the $n - i$ 1-bits in a mutation to leave L_i . For all i with $k \leq i < n$ the situation is similar to **ONEMAX**, it suffices to flip one of the $k + n - i$ 0-bits to leave L_i . Finally, for L_n , it is necessary to flip exactly the k remaining 0-bits. These insights yield

$$s_i \geq \begin{cases} \binom{n-i}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{en} & \text{for } i < k \\ \binom{n-i+k}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i+k}{en} & \text{for } k \leq i < n \\ \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \geq \frac{1}{en^k} & \text{for } k = n \end{cases}$$

leading to

$$E(T_{(1+1) \text{ EA}, \text{JUMP}_k}) \leq \sum_{i=1}^{k-1} \frac{en}{n-i} + \sum_{i=k}^n \frac{en}{n-i+k} + en^k = O(n \log n + n^k)$$

as upper bound on the expected optimization time.

We have already seen for **BinVal** that different f -based partitions can lead to different upper bounds on $E(T_{(1+1) \text{ EA}, f})$ for the same fitness function f . In some situations this may turn out to be useful. We consider the class of functions **LONGPATH** $_k$: $\{0, 1\}^n \rightarrow \mathbb{R}$ (Horn et al. 1994) with parameter $k \in \mathbb{N}$, $k > 1$ and $(n-1)/k \in \mathbb{N}$, given by

$$\text{LONGPATH}_k(x) = \begin{cases} n^2 + i & \text{for } x = x^{(i)} \\ n^2 - n \sum_{i=1}^k x[i] - \sum_{i=k+1}^n x[i] & \text{otherwise} \end{cases}$$

based on a path $P_k^n = (x^{(1)}, x^{(2)}, \dots, x^{(l(k,n))})$ with $x^{(i)} \in \{0, 1\}^n$ and $H(x^{(i)}, x^{(i+1)}) = 1$ for all i . The path P_k^n is recursively defined by $P_1^1 = (0, 1)$ for $n = 1$ and

$$P_k^n = \left(0^k x^{(1)}, 0^k x^{(2)}, \dots, 0^k x^{(l(k,n-k))}, 0^{k-1} 1 x^{(l(k,n-k))}, 0^{k-2} 11 x^{(l(k,n-k))}, \dots, \right. \\ \left. 1^k x^{(l(k,n-k))}, 1^k x^{(l(k,n-k)-1)}, \dots, 1^k x^{(1)} \right)$$

for $n \in \mathbb{N}$ where $x^{(i)} \in \{0, 1\}^{n-k}$ is the i th point in the path P_k^{n-k} . The path P_k^n is called the long- k path of dimension n , it has length $l(k, n) = (k+1)2^{(n-1)/k} - k + 1$ and the following interesting property. For each $x^{(i)} \in P_k^n$ with at least s successors on the path we have that $H(x^{(i)}, x^{(i+j)}) = j$ for all $j \in \{1, 2, \dots, \min\{s, k-1\}\}$ and no other successor of $x^{(i)}$ on the path has Hamming distance j to $x^{(i)}$ (Rudolph 1997).

Clearly, **LONGPATH** $_k$ does not have any local optima. Either a point is the unique global optimum or it has a Hamming neighbor with strictly larger function value. Thus, it is possible to optimize **LONGPATH** $_k$ by means of single-bit mutations. On the other hand, due to its recursive definition, the long k -path of dimension n consists of k segments where for each path point in any segment, it suffices to flip at most k specific bits to leave the current segment.

Thus, there are two different f -based partitions that align with the properties of LONGPATH_k . The trivial one has one fitness layer for each fitness value. Since the mutations of single bits are sufficient to leave a fitness layer, this yields $E(T_{(1+1) \text{ EA}, \text{LONGPATH}_k}) = O(n|P_k^n|)$ as upper bound. The other f -based partition uses the segments of the long k -path as fitness layers leading to $E(T_{(1+1) \text{ EA}, \text{LONGPATH}_k}) = O(n^{k+1}/k)$ as upper bound. Since both upper bounds are valid we have the following result.

Theorem 5 (Rudolph 1997) $E(T_{(1+1) \text{ EA}, \text{LONGPATH}_k}) = O(\min\{n|P_k^n|, n^{k+1}/k\})$

As a final example for the wide applicability of the method of fitness-based partitions the class of linear functions is considered. A fitness function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ is called *linear* if there exist weights $x_0, x_1, \dots, x_n \in \mathbb{R}$ such that $f(x) = w_0 + \sum_{i=1}^n x[i]w_i$ holds. We observe that ONEMAX as well as BINVAL are linear functions.

Since the $(1+1)$ EA is completely symmetric with respect to the roles of 0-bits and 1-bits one can change any linear function with some weights $w_i < 0$ to another linear function with no negative weights simply by replacing w_i by $-w_i$. This changes the roles of 0 and 1 at the i th position and does not influence the optimization time in any way. Moreover, one can assume $w_0 = 0$ since a constant summand has no influence on the selection employed in the $(1+1)$ EA. Since the $(1+1)$ EA is also completely symmetric with respect to bit positions, one can reorder the bits such that $w_1 \geq w_2 \geq \dots \geq w_n$ holds. Finally, if one is concerned with upper bounds on the expected optimization time, it can be assumed that even $w_1 \geq w_2 \geq \dots \geq w_n > 0$ holds, since $w_i = 0$ implies that the i th bit has no influence on the function value. Clearly, this doubles the number of global optima and can only decrease the expected optimization time in comparison to the case where $w_i \neq 0$ holds.

Now, considering any linear function $f(x) = \sum_{i=1}^n w_i x[i]$ with $w_1 \geq w_2 \geq \dots \geq w_n > 0$ one defines the following f -based partition:

$$L_i = \left\{ x \in \{0, 1\}^n \mid \sum_{j=1}^{i-1} w_j x[j] \leq f(x) < \sum_{j=1}^i w_j x[j] \right\}, i \in \{1, 2, \dots, n\}$$

and $L_{n+1} = \{1^n\}$. It is observed that now the situation is similar to the situation for BINVAL . For any $x \in L_i$ there is at least one bit among the leftmost i bits such that flipping this bits leads to an offspring $y \in L_j$ with $j > i$. This yields $s_i \geq (1/n)(1 - 1/n)^{n-1} \geq 1/(en)$ and we obtain

$$E(T_{(1+1) \text{ EA}, f}) \leq \sum_{i=1}^n en = O(n^2)$$

as upper bound. Note that this upper bound holds for any linear function f .

The method of fitness-based partitions is not necessarily restricted to the simple $(1+1)$ EA, but it is tightly linked to the strict plus-selection that does not accept any decrease in function values. It is not too difficult to generalize it to other evolutionary algorithms with such a selection. For example, it can be shown that for the $(1+\lambda)$ EA, that is, the $(\mu+\lambda)$ EA with population size $\mu = 1$, the expected optimization time on ONEMAX is bounded above by $E(T_{1+\lambda \text{ EA}, \text{ONEMAX}}) = O(n \log n + n\lambda)$ using the method of fitness-based partitions (Jansen et al. 2005). But it does not work for the (μ, λ) EA due to the missing monotonicity in fitness of the current best member of the population.

4.2 General Lower Bound

In order to assess the strength of an upper bound on $E(T_{A,f})$, a lower bound is needed for comparison. One way of obtaining lower bounds is resorting to fundamental limits. This, however, will not lead to strong lower bounds since “no free lunch” results do not apply to function classes of practical interest and black-box complexity most often yields relatively weak lower bounds since it takes into account the performance of optimal algorithms, not the specific evolutionary algorithm one is dealing with. In order to obtain stronger lower bounds on $E(T_{A,f})$ one needs to take into account specific properties of evolutionary algorithm A .

We consider an evolutionary algorithm A that uses standard bit mutations as the only way of creating new offspring. This holds for the $(1+1)$ EA and the $(\mu+\lambda)$ -EA without crossover. If A has a population of size $\mu > n \log n$ it makes $\Omega(n \log n)$ f -evaluations already in the initialization. It can be claimed that if the fitness function has a unique global optimum $x^* \in \{0, 1\}^n$, we have $E(T_{A,f}) = \Omega(n \log n)$ in any case, that is, for $\mu \leq n \log n$, too (Jansen et al. 2005).

We consider some $x \in \{0, 1\}^n$ from the initial population, that is, chosen uniformly at random. Each bit in x differs from the corresponding bit in the unique global optimum x^* with probability $1/2$. We can introduce random variables X_i with $i \in \{1, 2, \dots, n\}$, one for each bit, that assumes the value 1 if $x[i] \neq x^*[i]$ and the value 0 otherwise. Clearly, the expected Hamming distance between x and x^* equals $E(H(x, x^*)) = \sum_{i=1}^n X_i = n/2$. Since the random variables X_i are independent, one is in a situation where Chernoff bounds can be applied (Motwani and Raghavan 1995). We have a random variable $X = \sum_{i=1}^n X_i$ with $0 < \text{Prob}(X_i = 1) < 1$ for all $i \in \{1, 2, \dots, n\}$. Chernoff bounds yield strong bounds on the probability that X deviates considerably from its expected value. For example, for δ with $0 < \delta < 1$

$$\text{Prob}(X \geq (1 + \delta)E(X)) = e^{-E(X)\delta^2/3}$$

$$\text{Prob}(X \leq (1 - \delta)E(X)) = e^{-E(X)\delta^2/2}$$

hold. In this situation, we have that with probability $1 - e^{-\Omega(n)}$ the Hamming distance of x and x^* is bounded below by $n/3$. With a population of size $\mu \leq n \log n$ this holds for each member of the population. We get a bound on the probability that any member of the population has a Hamming distance of less than $n/3$ to the unique global optimum x^* by taking the union bound, that is, by summing up the probabilities for each member of the initial population. Thus, with probability $1 - \sum_{i=1}^{\mu} e^{-\Omega(n)} = 1 - e^{-\Omega(n)}$, each member of the initial population has Hamming distance at least $n/3$ to x^* . One can denote this event as B and have $\text{Prob}(B) = 1 - e^{-\Omega(n)}$.

If one wants to reach x^* in this situation within at most t f -evaluations, there has to be at least one member of the initial population where all those differing bits flip at least once within at most t mutations. Remember that the number of differing bits is bounded below by $n/3$. The probability for this event can be estimated in a very direct way. A single bit flips with probability $1/n$, thus it does not flip at all in t mutations with probability $(1 - 1/n)^t$. One can see that it flips at least once within these t mutations with probability $1 - (1 - 1/n)^t$. Since the bits are independent we have that with probability at most $(1 - (1 - 1/n)^t)^{n/3}$ the at

least $n/3$ bits differing from x^* are all flipped at least once in these t mutations. Thus, finally, one can bound the probability that among those at least $n/3$ bits there is at least one that was never flipped in t mutations from below by $1 - (1 - (1 - 1/n)^t)^{n/3}$. We consider $t = (n-1) \ln n$ generations and see that $(1 - 1/n)^{(n-1) \ln n} \geq e^{-\ln n} = 1/n$ holds. Thus, in $t = (n-1) \ln n$ generations the probability that there is one bit that is needed to flip in order to reach x^* is not flipped at all is bounded below by $1 - (1 - 1/n)^{n/3} \geq 1 - e^{-1/3} = \Omega(1)$. We denote this event as C and have $\text{Prob } C = \Omega(1)$. For the optimization time we thus have

$$\begin{aligned} E(T_{A,f}) &\geq E(T_{A,f} | B \wedge C) \text{ Prob}(B \wedge C) = E(T_{A,f} | B \wedge C) \text{ Prob}(B) \text{ Prob}(C) \\ &= (n-1) \ln(n) \left(1 - e^{-\Omega(n)}\right) \Omega(1) = \Omega(n \log n) \end{aligned}$$

as claimed. It is not difficult to see that the proof can be generalized to fitness functions where the number of global optima is larger than 1 but polynomially bounded. It can be remarked that this proof is similar to the classical result on the coupon collector's problem (Mitzenmacher and Upfal 2005; Motwani and Raghavan 1995).

Clearly, this general lower bound $E(T_{A,f}) = \Omega(n \log n)$ that holds for any mutation-based evolutionary algorithm and any fitness function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ is still rather general and as an immediate consequence rather weak. Nevertheless, it proves that the upper bound on ONEMAX is in fact asymptotically tight and we have $E(T_{(1+1) \text{ EA}, \text{ONEMAX}_a}) = \Theta(n \log n)$ for all $a \in \{0, 1\}^n$.

4.3 Expected Multiplicative Distance Decrease

We return to the often simpler task of proving upper bounds on the expected optimization time. The idea of fitness-based partitions is to measure the progress of $(1 + 1)$ EA by means of the increase in fitness values. To achieve this, the fitness values are grouped in appropriate layers. Considering BINVAL as the example, we have already seen that finding an appropriate grouping can make the difference between a good upper bound and a completely useless upper bound. While this method has the advantage that it is easy to apply and often yields very useful and sometimes even tight upper bounds, it has the intrinsic disadvantage that one needs to define in advance how the algorithm under consideration will most likely increase the function values. This ordering is in some sense implicit in the definition of the fitness layers. One can see this by considering BINVAL. The definition of L_i made it necessary to have the bits in the current bit string flip from left to right. Clearly, the real random process under consideration is less strict: any 0-bits can be flipped to increase the function value. It may well be the case that due to this greater degree of freedom, the real random process that governs the $(1 + 1)$ EA on BINVAL is faster in finding the optimum than the upper bound of $O(n^2)$ predicts.

We reconsider the main idea – measuring the progress the algorithm makes by means of an increase in function value – and capture it in a more general form. To be concrete, attention should be again restricted to the $(1 + 1)$ EA and it should be remarked that generalizations to other evolutionary algorithms like the $(1 + \lambda)$ EA are again not too difficult.

Consider the $(1 + 1)$ EA optimizing some fitness function $f: \{0, 1\}^n \rightarrow \mathbb{Z}$ with $f_{\text{opt}} = \max\{f(x) | x \in \{0, 1\}^n\}$. Note that the assumption that the function values are integers will be important in the following. When the current search point is x_t , we know that a global

optimum is reached if the function value is increased by $f_{\text{opt}} - f(x_t)$. If, for any $x_t \in \{0, 1\}^n$, one is able to describe a sequence of at most l operations that can all be applied to x_t and that when applied simultaneously lead to the global optimum, then each of these operations causes an average increase of function value of at least $(f_{\text{opt}} - f(x_t))/l$. In different terms, each operation decreases the distance to the maximal fitness value $f_{\text{opt}} - f(x_t)$ on average by a factor of at least $1 - 1/l$. Since different potential points $x_t \in \{0, 1\}^n$ are dealt with, it does not really make sense to talk about $f(x_t)$; the average increase in function value $(f_{\text{opt}} - f(x_t))/l$ clearly depends on the x_t under consideration. Thus, we consider $f_{\text{max}} = \max\{f_{\text{opt}} - f(x) \mid x \in \{0, 1\}^n\}$. Now we have the distance to the maximal function value after t such operations is decreased in expectation by a factor of at least $(1 - 1/l)^t$. Since the function values are integers, we know that the global optimum is reached if the difference is decreased to less than 1. Thus, if $(1 - 1/l)^t f_{\text{max}} < 1$ holds, we know that in expectation the optimum is reached. So, a number of $l \ln f_{\text{max}}$ such local operations is on average sufficient to reach the optimum. If it is additionally assumed that the expected waiting time for each such local operation is bounded above by w we obtain

$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(wl \ln f_{\text{max}})$$

as upper bound on the expected optimization time.

This method of the expected multiplicative distance decrease (Neumann and Wegener 2004) is applied to the class of linear functions with integer weights. Consider some $f: \{0, 1\}^n \rightarrow \mathbb{Z}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x[i]$ with all $w_i \in \mathbb{Z}$. As we did when using the method of fitness-based partitions, we assume without loss of generality that $w_0 = 0$ and $w_1 \geq w_2 \geq w_3 \geq \dots \geq w_n > 0$ holds since one wants to derive an upper bound on $\mathbb{E}(T_{(1+1) \text{ EA}, f})$. Remember that we had $\mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(n^2)$ using fitness-based partitions.

Due to the assumptions on f , the unique global optimum is 1^* and we have

$$f_{\text{max}} = \max\{f_{\text{opt}} - f(x) \mid x \in \{0, 1\}^n\} = \sum_{i=1}^n w_i \leq nw_{\text{max}}$$

where $w_{\text{max}} = \max\{w_i \mid i \in \{1, 2, \dots, n\}\}$ denotes the maximal weight of a single bit. For each $x \in \{0, 1\}^n$ we consider the at most n mutations that flip single bits for positions i where $x[i] = 0$ holds. Clearly, applying these up to n mutations simultaneously leads to the unique global optimum 1^n . Moreover, the probability to have a mutation flipping a single bit equals

$$\binom{n}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} = \left(\frac{1}{n}\right)^{n-1} > e^{-1}$$

so we have $w < e$ in the notation. Since $l = n$ we have

$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(wl \ln f_{\text{max}}) = O(n \ln(nw_{\text{max}}))$$

for any linear functions with integer weights. For **BINVAL**, we have $w_{\text{max}} = 2^{n-1}$ and $\mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(n^2)$ follows. This is no improvement over the bound of the same size obtained using f -based partitions. But this result is more powerful. If one restricts oneself to linear functions with polynomially bounded integer weights, one can obtain $\mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(n \log n)$, which is asymptotically tight since it matches the general lower bound $\Omega(n \log n)$.

While being more flexible than the method of fitness-based partitions, the method of the expected multiplicative distance decrease has the inherent disadvantage that the upper bounds one can obtain directly depend on the size of the function values. Since for example functions there are no bounds on the size of the function values, one may be tempted to believe that at least for some example functions, one cannot obtain any useful result at all using this method. This, however, is too pessimistic. It turns out that function values can be unnecessarily large when considering evolutionary algorithms or other randomized search heuristics where selection depends only on the ordering of the function values and not on the function values themselves. Another way to say this is the following. Given a fitness function $f: \{0, 1\}^n \rightarrow \mathbb{Z}$ with arbitrary function values, it is possible to come up with another fitness function $f': \{0, 1\}^n \rightarrow \mathbb{Z}$ such that the $(1 + 1)$ EA behaves the same on f and f' but the function values of f' are limited in size (Reichel and Skutella 2009). This way, much tighter bounds can be obtained, not only for example functions but also for combinatorial optimization problems.

4.4 Drift Analysis for Upper Bounds

The idea of both methods, fitness-based partitions as well as the method of the expected multiplicative distance decrease, is to measure the progress of the evolutionary algorithm A in some way. When using fitness-based partitions this progress has to be achieved in a pre-defined way that is implicit in the definition of the fitness layers. When using the method of the expected multiplicative distance decrease, the progress can be obtained by operations in any way but it is always measured in function values. We gain even more flexibility if we allow for different measures of progress. Consider some evolutionary algorithm A where Z denotes the set of all possible populations of A . We call a function $d: Z \rightarrow \mathbb{R}_0^+$ a *distance measure* if $d(P) = 0$ holds if and only if the population $P \in Z$ contains a point with maximal function value. Calling a unary operator a distance measure appears to be strange at first sight. However, what is really measured is the distance to the set of global optima. Thus, we can omit the obligatory second part in distance measuring because it is fixed for a fixed fitness function f . Note that this notion of distance is very weak, in particular, d need not be a metric. Again, we concentrate on the $(1 + 1)$ EA in order to simplify the considerations. It has to be remarked, though, that the method taken into consideration is not at all restricted to such simple evolutionary algorithms and that it can be easily applied to much more complex evolutionary algorithms.

For the $(1 + 1)$ EA we have $Z = \{0, 1\}^n$ since the current population consists of a single search point $x_t \in \{0, 1\}^n$. Clearly, we have $T_{(1+1) \text{ EA}, f} = \min\{t \mid d(x_t) = 0\}$ for any fitness function f . We consider the maximum distance M and want to bound the average time until initial distance that is at most M is decreased to 0. Clearly, $M = \max\{d(x) \mid x \in Z\}$ holds. We concentrate on a single generation and consider the decrease in distance in a single generation, that is, $D_t = d(x_{t-1}) - d(x_t)$. Note that D_t is a random variable that may take negative values with positive probability, actually indicating an increase in distance. One is interested in the expected decrease in distance in a single generation, $E(D_t \mid T_{(1+1) \text{ EA}, f} > t)$, and call this *drift*. Since one wants to derive an upper bound on the expected optimization time one can assume a pessimistic point of view and consider the worst-case drift Δ , that is, the smallest drift possible, $\Delta = \min\{E(D_t \mid T_{(1+1) \text{ EA}, f} > t) \mid t \in \mathbb{N}\}$. Even if this worst-case drift Δ is strictly positive, one can prove the drift theorem $E(T_{(1+1) \text{ EA}, f}) \leq M/\Delta$ (He and Yao 2004),

here for the $(1+1)$ EA. The result is intuitively plausible. If one wants to overcome a distance of at most M and can travel at a speed of at least Δ , the average travel time is bounded above by M/Δ . Since the proof will turn out to be helpful when looking for a similar method for lower bounds, the theorem is stated together with its proof here. Again, we restrict ourselves to the $(1+1)$ EA even though much more general results can be proved with essentially the same proof.

Theorem 6 (Drift Theorem for the $(1+1)$ EA (He and Yao 2004)) *Consider the $(1+1)$ EA on some function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ and some distance measure $d: \{0, 1\}^n \rightarrow \mathbb{R}_0^+$ with $d(x) = 0 \Leftrightarrow f(x) = \max\{f(y) \mid y \in \{0, 1\}^n\}$. Let $M = \max\{d(x) \mid x \in \{0, 1\}^n\}$, $D_t = d(x_{t-1}) - d(x_t)$, and $\Delta = \min\{\mathbb{E}(D_t \mid T_{(1+1)\text{EA},f} > t) \mid t \in \mathbb{N}\}$.*

$$\Delta > 0 \Rightarrow \mathbb{E}(T_{(1+1)\text{EA},f}) \leq M/\Delta$$

Proof In order to simplify notation we define $T := T_{(1+1)\text{EA},f}$. If we consider $\sum_{t=1}^T D_t$ we see that

$$\sum_{i=1}^T D_i = \sum_{t=1}^T d(x_{t-1}) - d(x_t) = d(x_0) - d(x_T) = d(x_0)$$

holds since most of the summands cancel out and we have $d(x_T) = 0$ by assumption on d and definition of T . Since M is an upper bound on $d(x)$ for all $x \in \{0, 1\}^n$, we have $M \geq d(x_0)$ and

$$M \geq \sum_{i=1}^T D_i$$

follows. Since M is not a random variable, we have $M = \mathbb{E}(M)$. By the definition of the expected value and the law of total probability, we get

$$M \geq \mathbb{E}\left(\sum_{i=1}^T D_i\right) = \sum_{t=1}^{\infty} \text{Prob}(T = t) \mathbb{E}\left(\sum_{i=1}^T D_i \mid T = t\right)$$

Making use of the linearity of the expectation and changing the ordering in the sums, we get

$$\sum_{t=1}^{\infty} \text{Prob}(T = t) \mathbb{E}\left(\sum_{i=1}^T D_i \mid T = t\right) = \sum_{t=1}^{\infty} \sum_{i=1}^T \text{Prob}(T = t) \mathbb{E}(D_i \mid T = t)$$

Making use of the total law of probability a second time and reordering again, we get

$$\begin{aligned} & \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \text{Prob}(T = t) \mathbb{E}(D_i \mid T = t) \\ &= \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \text{Prob}(T \geq i) \text{Prob}(T = t \mid T \geq i) \mathbb{E}(D_i \mid T = t \wedge T \geq i) \\ &= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \sum_{t=i}^{\infty} \text{Prob}(T = t \mid T \geq i) \mathbb{E}(D_i \mid T = t \wedge T \geq i) \\ &= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \sum_{t=1}^{\infty} \text{Prob}(T = t \mid T \geq i) \mathbb{E}(D_i \mid T = t \wedge T \geq i) \end{aligned}$$

where the last equation makes use of the fact that $\text{Prob}(T = t \mid T \geq i) = 0$ for $t < i$ holds.

Another application of the law of total probability (this time in the reverse direction) yields

$$\begin{aligned} \sum_{i=1}^{\infty} \text{Prob } (T \geq i) \sum_{t=1}^{\infty} \text{Prob } (T = t \mid T \geq i) E(D_i \mid T = t \wedge T \geq i) \\ = \sum_{i=1}^{\infty} \text{Prob } (T \geq i) E(D_i \mid T \geq i) \end{aligned}$$

Clearly, $\Delta \leq E(D_i \mid T \geq i)$ holds, so we have

$$\sum_{i=1}^{\infty} \text{Prob } (T \geq i) E(D_i \mid T \geq i) \geq \Delta \sum_{i=1}^{\infty} \text{Prob } (T \geq i) = \Delta E(T)$$

by the definition of the expectation. Putting things together we observe that we proved $M \geq \Delta E(T)$ and $E(T) \leq M/\Delta$ follows for $\Delta > 0$.

One of the advantages of the drift theorem is that it allows for the definition of almost arbitrary distance measures. It was known that the expected optimization time of the (1+1) EA on linear functions is $O(n \log n)$ (Droste et al. 2002a) (different from the bound $O(n^2)$ that was proven here using f -based partitions) before the method of drift analysis was introduced to the field of evolutionary computation, but the proof is long, tedious, and difficult to follow. Applying drift analysis, the same result can be shown in a much simpler way making use of an appropriate distance measure.

As usual, we consider a linear function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ with $f(x) = \sum_{i=1}^n w_i x[i]$ where $w_1 \geq w_2 \geq \dots \geq w_n > 0$ holds. Clearly, the bits with smaller index have at least the potential to be more important than the other bits. In one extreme case, **BinVal** is an example, any bit has a weight w_i that is so large that it dominates all bits to its right, that is, $w_i > \sum_{j=i+1}^n w_j$. In the other extreme case, **OneMax** is an example, all weights are equal. We try to prove an upper bound for all these cases simultaneously by defining a distance measure (similar to a potential function) that in some way expresses the greater importance of bits that are closer to the left of the bit string. We define (He and Yao 2004)

$$d(x) = \ln \left(1 + 2 \sum_{i=1}^{n/2} (1 - x[i]) + \sum_{i=(n/2)+1}^n (1 - x[i]) \right)$$

making bits in the left half twice as important as bits in the right half. It can be observed that

$$\begin{aligned} M &= \max \left\{ \ln \left(1 + 2 \sum_{i=1}^{n/2} (1 - x[i]) + \sum_{i=(n/2)+1}^n (1 - x[i]) \right) \mid x \in \{0, 1\}^n \right\} \\ &= \ln \left(1 + n + \frac{n}{2} \right) = \Theta(\log n) \end{aligned}$$

holds. For an upper bound, one needs now a lower bound on

$$E(D_t \mid T_{(1+1) \text{ EA}, f} > t) = E(d(x_{t-1}) - d(x_t) \mid T_{(1+1) \text{ EA}, f} > t)$$

It is easy to observe that this expectation is minimal either for $x_{t-1} = 011 \dots 1 = 01^{n-1}$ or $x_{t-1} = 11 \dots 1011 \dots 1 = 1^{n/2} 01^{(n/2)-1}$. One can skip the still somewhat tedious calculations

for these two cases and remark that $\Omega(1/n)$ can be proved as the lower bound in both cases. So, we have $\Delta = \Omega(1/n)$ and $M = \Theta(\log n)$ and this leads to

$$E(T_{(1+1) EA,f}) = \frac{M}{\Delta} = \frac{\Theta(\log n)}{\Omega(1/n)} = O(n \log n)$$

as claimed.

Another advantage of this method is that it is not a problem if the distance is not monotonically decreased during the run. It completely suffices if the expected decrease in distance, that is, the drift, is positive. This can be illustrated with a small example that makes use of the (μ, λ) EA (Algorithm 3).

Consider the $(1, n)$ EA, that is, the (μ, λ) EA with $\mu = 1$ and $\lambda = n$ on the function LEADINGONES: $\{0, 1\}^n \rightarrow \mathbb{R}$ that is defined by $\text{LEADINGONES}(x) = \sum_{i=1}^n \prod_{j=1}^i x[j]$. The function value

$\text{LEADINGONES}(x)$ equals the number of consecutive 1-bits in x counting from left to right. A simple application of fitness-based partitions yields $E(T_{(1+1)EA,LEADINGONES}) = O(n^2)$ (Droste et al. 2002a) and $E(T_{(1+\lambda) EA,LEADINGONES}) = O(n^2 + n\lambda)$ (Jansen et al. 2005). For the $(1, \lambda)$ EA, however, this method cannot be applied. With drift analysis, however, an upper bound can be derived with ease.

We define $d(x) = n - \text{LEADINGONES}(x)$ as distance and observe that $M = n$ holds. Applying the drift theorem, as presented in Theorem 6, yields an upper bound on the number of expected *generations* the evolutionary algorithm needs to locate an optimum. For the $(1 + 1)$ EA this simply equals the number of f -evaluations. For the $(1, \lambda)$ EA, this number needs to be multiplied by λ in order to get the expected number of f -evaluations. For the estimation of $E(d(x_{t-1}) - d(x_t) | T > t)$ the following two simple observations are made. If among the $\lambda = n$ mutations generating offspring there is at least one where exactly the left most 0-bit is flipped, the distance is decreased by exactly 1. Since such a mutation has probability $(1/n)(1 - 1/n)^{n-1} \geq 1/(en)$ we have that this happens with probability at least $1 - (1 - (1/(en)))^n = \Omega(1)$. Second, the distance can only increase if in one generation there is no mutation that does not change a bit at all. This holds since such a mutation simply creates a copy of its parent. In this case, the parent belongs to the offspring and the fitness of the parent in the next generation cannot be worse. Such an event happens with probability at most $(1 - (1 - 1/n))^n \leq (1 - 1/e)^n = e^{-\Theta(n)}$. Moreover, in this case, the distance cannot increase by more than n . Together this yields

$$\begin{aligned} E(d(x_{t-1}) - d(x_t) | T > t) &\geq 1 \cdot \left(1 - \left(1 - \left(\frac{1}{en}\right)\right)\right) - n \cdot \left(\left(1 - \frac{1}{n}\right)^n\right)^n \\ &\geq 1 \cdot \Omega(1) - n \cdot e^{-\Theta(n)} = \Omega(1) \end{aligned}$$

and proves that $\Delta = \Omega(1)$ holds. Thus, we have $E(T_{(1, n) EA, LEADINGONES}) = O(n)^2$.

4.5 Drift Analysis for Lower Bounds

It would be nice to have such a general and powerful method like drift analysis for lower bounds, too. We reconsider the proof of Theorem 6 and observe that this is not so hard to have. In the calculations in this proof, almost all steps are exact equalities, only at two places

we made estimations and use of bounds. One is $\Delta \leq E(D_i | T \geq i)$ and this stems from the definition of Δ :

$$\Delta = \min\{E(D_t | T_{(1+1) EA,f} > t) | t \in \mathbb{N}\}$$

Clearly, for a lower bound on the expected optimization time we need a different definition that takes into account the maximal drift, not the minimal as for the upper bound. Thus, we replace Δ by

$$\Delta_l = \max\{E(D_t | T_{(1+1) EA,f} > t) | t \in \mathbb{N}\}$$

and have $\Delta_l \geq E(D_i | T \geq i)$ as needed.

The second place one can make use of a bound is $M \geq \sum_{i=1}^T D_i$. This, again is true due to the definition of M

$$M = \max \{d(x) | x \in \{0, 1\}^n\}$$

Clearly, one could again replace M by $M_l = \min \{d(x) | x \in \{0, 1\}^n\}$, but this, unfortunately, is pointless: $M_l = 0$ holds and only the trivial bound $E(T_{A,f}) \geq 0$ can be achieved. Reconsidering the proof of (❷ [Theorem 6](#)) a bit closer, we observe that actually the first step reads

$$E(M) = M \geq E\left(\sum_{i=1}^T D_i\right)$$

and we see that $E(M)$ could be replaced by $E(d(P_0))$, the expected distance of the initial population P_0 . This yields $E(d(P_0)) = E\left(\sum_{i=1}^T D_i\right) \leq E(T_{A,f})\Delta_l$ and we obtain $E(T_{A,f}) \geq E(d(P_0))/\Delta_l$ as drift theorem for lower bounds.

We apply this lower bound technique at a simple example and derive a lower bound on $E(T_{(1+1) EA, LEADINGONES})$. Like for the upper bound for the $(1, n)$ EA on **LEADINGONES** we use the rather trivial distance measure

$$d(x) = n - \text{LEADINGONES}(x)$$

Now we need to bound the decrease in distance in one generation from above. To this end, it is observed that it is necessary that the leftmost 0-bit is flipped in a mutation to decrease d . This happens with probability $1/n$. In this case, clearly, the decrease can be bounded by n . This, however, does not yield a useful bound. It should be remembered that one only needs to bound the *expected* decrease and try to estimate this in a more careful way.

The decrease in distance stems from two sources. The leftmost 0-bits that is flipped decreases the distance by 1. Moreover, any consecutive bits right of this leftmost 0-bit that all happen to be 1-bits after the mutation decrease this distance further, it is decreased by the length of the sequence of consecutive 1-bits. One needs to estimate this random length.

The crucial observation for the estimation of the random length of this block of 1-bits is that all bits that are right of the leftmost 0-bit never had any influence on the selection. This is the case since the number of leading ones never decreases and the function value

$\text{LEADINGONES}(x) = \sum_{i=1}^n \prod_{j=1}^i x[j]$ does not depend on any bit right of the leftmost 0-bit. It is claimed that this implies that these bits are distributed according to the uniform distribution. This can be proved by induction on the number of steps. Clearly, it holds in the beginning

since this is the way the initial bit string x_0 is generated. For any $t > 0$ we have for an arbitrary $x \in \{0, 1\}^n$

$$\begin{aligned}\text{Prob } (x_t = x) &= \sum_{y \in \{0,1\}^n} \text{Prob } (x_{t-1} = y) \text{ Prob } (\text{mut}(y) = x) \\ &= 2^{-n} \sum_{y \in \{0,1\}^n} \text{Prob } (\text{mut}(y) = x)\end{aligned}$$

making use of the induction hypothesis. The crucial observation is that

$$\text{Prob } (\text{mut}(y) = x) = \text{Prob } (\text{mut}(x) = y)$$

holds for standard bit mutations. This is the only property of the mutation operator that is needed here. Note, however, that there are mutation operators known where this does not hold (Jansen and Sudholt 2010). Now we have

$$\begin{aligned}\text{Prob } (x_t = x) &= 2^{-n} \sum_{y \in \{0,1\}^n} \text{Prob } (\text{mut}(y) = x) \\ &= 2^{-n} \sum_{y \in \{0,1\}^n} \text{Prob } (\text{mut}(x) = y) = 2^{-n}\end{aligned}$$

and see that these bits are indeed uniformly distributed.

Thus, the expected decrease in distance in a single mutation is

$$\sum_{i=1}^n i \text{Prob } (\text{decrease distance by } i) \leq \sum_{i=1}^n i \left(\frac{1}{n}\right) \left(\frac{1}{2^{i-1}}\right) < \frac{1}{n} \sum_{i=1}^{\infty} \frac{i}{2^{i-1}} = \frac{4}{n}$$

and we have $\Delta_l < 4/n$.

Since $E(d(x_0))/\Delta_l = \Theta(nE(d(x_0)))$, all one has to do is to estimate the expected initial distance. Clearly,

$$E(d(x_0)) < n - \sum_{i=1}^n \frac{i}{2^i} = n - 2$$

and we have $E(T_{(1+1)} \text{ EA, LEADINGONES}) = \Theta(n^2)$. It can be seen that this implies that the upper bound on $E(T_{(1+1)} \text{ EA, LEADINGONES})$ derived by means of f -based partitions is indeed tight.

4.6 Typical Events

Sometimes, a statement about the expected optimization time can be made and proved based on a good understanding of the evolutionary algorithm and the fitness function. Such proofs tend to be *ad hoc* and difficult to generalize. Sometimes, however, there is a common element that is worth pointing out. Here, we consider situations where the optimization time is dominated by something that typically happens in a run of the evolutionary algorithm, we call this a typical event.

To make things more concrete, an example is considered. As usual, we use the $(1 + 1)$ EA and reconsider a fitness function where we already proved an upper bound on the expected optimization time, JUMP_k . Remember that using the method of fitness-based partitions, we were able to prove that

$$E(T_{(1+1) \text{ EA, JUMP}_k}) = O(n^k + n \log n)$$

holds. The dominating term n^k stems from the fact that in order to reach the unique global optimum starting in one of the local optima, a mutation of k specific bits is necessary. Such a mutation has probability $((1/n)^k(1 - 1/n)^{n-k})$ and leads to an expected waiting time of $\Theta(n^k)$. Remembering the structure of JUMP_k

$$\text{JUMP}_k(x) = \begin{cases} n - \text{ONEMAX}(x) & \text{if } n - k < \text{ONEMAX}(x) < n \\ k + \text{ONEMAX}(x) & \text{otherwise} \end{cases}$$

the following observations are made. If the current x_t contains at most $n - k$ 1-bits, in order to reach the global optimum by means of a direct mutation, such a mutation needs to flip exactly all 0-bits in x_t . In this case the expected optimization time is bounded below by $\Omega(n^k)$. Since for all $x \in \{0, 1\}^n$ with more than $n - k$ but less than n 1-bits the function value equals $n - \text{ONEMAX}(x)$ leading toward the local optima, we believe that it is quite likely that a local optimum will be reached.

We make this reasoning more concrete and define the typical event B that there exists some $t \in \mathbb{N}$, such that $\text{ONEMAX}(x_t) \leq n - k$ holds. It follows from the law of total probability that

$$E(T_{(1+1)\text{EA}, \text{JUMP}_k}) \geq \text{Prob}(B)E(T_{(1+1)\text{EA}, \text{JUMP}_k} \mid B)$$

holds. As already argued above, $E(T_{(1+1)\text{EA}, \text{JUMP}_k} \mid B) = \Omega(n^k)$ holds. Since $E(T_{(1+1)\text{EA}, \text{JUMP}_k}) = \Omega(n \log n)$ follows from the general lower bound (Sect. 4.2), it suffices to prove that $\text{Prob}(B) = \Omega(1)$ holds.

We make a case distinction on the value of k . For $k < n/3$ we have

$$\text{Prob}(\text{ONEMAX}(x_0) \leq n - k) = 1 - 2^{-\Omega(n)}$$

as can easily be shown using Chernoff bounds. Clearly, $\text{Prob}(B) = 1 - 2^{-\Omega(n)}$ follows in this case. For the case $k > (n/3)$ one can first observe that still

$$\text{Prob}(\text{ONEMAX}(x_0) \leq n - k) = 1 - 2^{-\Omega(n)}$$

holds. It can be seen that $\text{Prob}(B)$ is bounded below the probability that the unique global optimum is reached before a local optimum is reached. It can be observed that after an initialization with at most $(2/3)n$ 1-bits the global optimum can only be reached via a mutation of all 0-bits with at least $n/3$ 0-bits in each current x_t . Such a mutation has probability at most $1/n^{n/3}$. Since the function is similar to $-\text{ONEMAX}$ it can be concluded that some local optimum is reached in expectation within $O(n \log n)$ steps. More precisely, there is some constant c such that the expected time to reach a local optimum is bounded above by $cn \log n$. By Markov's inequality, it follows that with probability at least $1/2$, we reach a local optimum in at most $2cn \log n$ steps if the global optimum is not reached before. We consider $2cn^2 \log n$ steps as n independent trials to reach a local optimum, each consisting of $2cn \log n$ steps. It can be concluded that, given that the global optimum is not reached before, a local optimum is reached within $O(n^2 \log n)$ steps with probability $1 - 2^{-\Omega(n)}$. Since the probability to reach the optimum in a single mutation is bounded above by $1/n^{n/3}$, the probability to reach the optimum within $O(n^2 \log n)$ steps is bounded above by $O((n^2 \log n)/n^{n/3}) = 2^{-\Omega(n)}$. Together, we have $\text{Prob}(B) = 1 - 2^{-\Omega(n)}$ in this case, too. So we have

$$E(T_{(1+1)\text{EA}, \text{JUMP}_k}) = \Theta(n^k + n \log n)$$

and observe that the upper bound derived by means of f -based partitions is indeed tight.

4.7 Typical Runs

The situation described in the previous section was in some sense exceptionally simple. The optimization time is governed by one specific event that needs to occur: Some point x with at most $n - k$ 1-bits needs to become x_t before the global optimum is found. Often the situation is less clear but still there is a rather clear idea how an evolutionary algorithm may optimize a specific fitness function. Often, it is not even necessary that the evolutionary algorithm does in fact optimize this fitness function in this way. Proving that one specific way is possible within some number of f -evaluations and with some probability is often sufficient to get at least an upper bound on the expected optimization time. This can be made clear by again considering JUMP _{k} but now a more complex evolutionary algorithm. Attention is restricted to JUMP _{k} for relatively small values of k , $k = O(\log n)$.

We consider the $(\mu + \lambda)$ EA with $\lambda = 1$, $p_c > 0$ and uniform crossover. Note that this is the first time that an evolutionary algorithm with crossover is dealt with. It will be seen that this makes the analysis considerably more difficult. It is assumed that the population size μ is polynomially bounded from above and not too small. It is assumed $\mu \geq k \log^2 n$ holds. Since $k = O(\log n)$, this allows for relatively small population sizes. Finally, it is assumed that the probability for crossover p_c is rather small. A proof that works for $p_c \leq 1/(ckn)$ where $c > 0$ is a sufficiently large positive constant is discussed.

We start by describing what we consider to be a typical run of the $(\mu + 1)$ EA with uniform crossover. We do so by describing different *phases* and what one expects in these phases to happen. For each phase, we define entry conditions that need to be fulfilled at the beginning of each phase and exit conditions that define the end of each phase. Here, three phases are described. For the first phase, the entry condition is empty. This has the advantage that we can assume any time to start with phase 1. In particular, if anything goes wrong in any phase, we can assume the algorithm to be restarted. This would be different if we had entry conditions that assume that the current population is actually randomly distributed according to the uniform distribution. The exit condition of the first phase is that each member of the current population contains exactly $n - k$ 1-bits. This implies that the population contains only local optima.

The entry condition for the second phase is equal to the exit condition of the first phase. This is usually the case since the exit of one phase should be the entry to the next phase. The exit condition of the second phase is that still all members of the current population contain exactly $n - k$ 1-bits each; additionally, we demand that

$$\forall i \in \{1, 2, \dots, n\}: \sum_{j=1}^{\mu} 1 - x^{(j)}[i] \leq \mu/(4k) \quad (1)$$

holds where $x^{(j)}$ denotes the j th member of the current population. It is observed that \bullet Eq. 1 ensures that at each of the n position, the number of 0-bits summed over the complete population is bounded above by $\mu/(4k)$. Note that this is a relatively large upper bound. The complete population contains exactly μk 0-bits, since each member contains exactly k 0-bits in the second phase. Thus, if these bits were evenly distributed, one would expect to see $\mu k/n = O(\mu \log(n)/n)$ 0-bits at each position. The bound $\mu/(4k)$ is by a factor of $\Theta(n/k^2) = \Omega(n/\log^2 n)$ larger.

The entry condition for the third and final phase is the exit condition of the second case. Its exit condition is that an optimal search point is member of the current population.

In the first phase, the situation is similar to ONE MAX. It is not difficult to see that on average after $O(\mu \log n)$ f -evaluations the exit condition of phase 1 is met (Jansen and Wegener 2002). Moreover, by considering longer runs as restarts like we did in the previous section, one can obtain an upper bound of $p_1 \leq \varepsilon$ for the probability not to complete the first phase within $O(\mu \log n)$ steps for any positive constant ε .

The second phase is much more difficult to analyze. When the phase begins, each member of the population contains exactly k 0-bits. We have to prove that these 0-bits become somewhat distributed among the different positions. To this end, we consider some specific bit position, say the first. Let z denote the number of 0-bits at this position summed over the whole population. We want to show that $z \leq \mu/(4k)$ holds after some time.

Clearly, in one generation, this number of 0-bits z can change by at most by 1. Let A_z^+ resp. A_z^- be the event that (given z) the number of zeros (at position 1) increases resp. decreases in one round. One wants to estimate the probabilities of A_z^+ and A_z^- . Since one wants to prove an upper bound on the number of 0-bits, one needs an upper bound on $\text{Prob}(A_z^+)$ and a lower bound on $\text{Prob}(A_z^-)$. Since this seems to be rather complicated, one can consider six (classes of) events that have probabilities that one can estimate directly. From now on, the index z will be omitted to simplify the notation.

- B : crossover is chosen as operator, $\text{Prob}(B) = p_c$.
- C : an object with a one at position 1 is chosen for replacement, $\text{Prob}(C) = (\mu - z)/\mu$.
- D : an object with a zero at position 1 is chosen for mutation, $\text{Prob}(D) = z/\mu$.
- E : the bit at position 1 does not flip, $\text{Prob}(E) = 1 - 1/n$.
- F_i^+ : there are exactly i positions among the $(k-1)$ 0-positions $j \neq 1$ that flip and exactly i positions among the $(n-k)$ 1-positions that flip, $\text{Prob}(F_i^+) = \binom{k-1}{i} \binom{n-k}{i} (1/n)^{2i} (1 - 1/n)^{n-2i}$.
- G_i^+ : there are exactly i positions among the k 0-positions that flip and exactly $i-1$ positions among the $(n-k-1)$ 1-positions $j \neq 1$ that flip, $\text{Prob}(G_i^+) = \binom{k}{i} \binom{n-k-1}{i-1} (1/n)^{2i-1} (1 - 1/n)^{n-2i}$.

Using this event, one can observe that

$$A^+ \subseteq B \cup \left(\overline{B} \cap C \cap \left[\left(D \cap E \cap \bigcup_{0 \leq i \leq k-1} F_i^+ \right) \cup \left(\overline{D} \cap \overline{E} \cap \bigcup_{1 \leq i \leq k} G_i^+ \right) \right] \right)$$

holds. Similarly, we get

$$A^- \supseteq \overline{B} \cap \overline{C} \cap \left[\left(D \cap \overline{E} \cap \bigcup_{1 \leq i \leq k} F_i^- \right) \cup \left(\overline{D} \cap E \cap \bigcup_{0 \leq i \leq k} G_i^- \right) \right]$$

where

- F_i^- : there are exactly $i-1$ positions among the $(k-1)$ 0-positions $j \neq 1$ that flip and exactly i positions among the $(n-k)$ 1-positions that flip, $\text{Prob}(F_i^-) = \binom{k-1}{i-1} \binom{n-k}{i} (1/n)^{2i-1} (1 - 1/n)^{n-2i}$.

- G_i^- : there are exactly i positions among the k 0-positions that flip and exactly i positions among the $(n - k - 1)$ 1-positions $j \neq 1$ that flip, $\text{Prob}(G_i^-) = \binom{k}{i} \binom{n - k - 1}{i} (1/n)^{2i} (1 - 1/n)^{n-2i-1}$.

Since we are interested in the situation where the number of 0-bits is too large, we assume $z \geq \mu/(8k)$ in the following. It is tedious but not difficult to see that $\text{Prob}(A^-) - \text{Prob}(A^+) = \Omega(1/(nk))$ holds (Jansen and Wegener 2002). We consider the random walk defined by the random changes of z and prove that after $\Theta(\mu n^2 k^3)$ steps, the number of 0-bits at the current position is sufficiently decreased with probability $1 - O(1/n)$. By increasing the constant factor hidden in the number of generations $\Theta(\mu n^2 k^3)$ we can decrease the probability not to decrease the number of 0-bits. We obtain a bound on the probability that this happens at least one of the n positions by simply taking the union bound.

Unfortunately, this line of reasoning is valid only if $z \geq \mu/(8k)$ holds during the complete phase. If this is not the case, the probabilities $\text{Prob}(A^+)$ and $\text{Prob}(A^-)$ change. We cope with this additional difficulty in the following way. Consider the last time point of time when $z < \mu/(8k)$ holds. If at this point of time there are at most $\mu/(8k)$ generations left, the number of 0-bits can at most increase to $\mu/(8k) + \mu/(8k) = \mu/(4k)$, which is the bound that we wanted to prove. If there are more than $\mu/(8k)$ steps left, we can repeat the line of reasoning and now know that $z \geq \mu/(8k)$ holds all the time.

Finally, in the third phase, the probability that the global optimum is found is not too small. Note that this is the only phase where we actually take beneficial effects of uniform crossover into account. We can control the number of 0-bits at each position in a way that is similar to the second phase. We ensure that the number of 0-bits does not exceed $\mu/(2k)$ at each position. So we can concentrate on the creation of the global optimum. This is achieved if the following happens. We perform crossover (with probability p_c), we select two parents that do not share a 0-bit at any position (with some probability p), in uniform crossover the optimum 1^n is created (with probability $(1/2)^{2n}$), and this optimum is not destroyed by mutation (with probability $(1 - 1/n)^n$). We need to estimate the probability p of selecting two parents without a common 0-bit. After the first parent is selected, there are exactly k positions where the second parent may “collide” with this parent. Since at each position one has at most $\mu/(2k)$ 0-bits, the probability for this event is bounded below by $k \cdot (\mu/(2k))/\mu = 1/2$. Together, we have a probability of $\Omega(p_c 2^{-2k})$ to create the global optimum for each generation of the third phase. Combining the results for the different phases, we obtain $O(\mu n^2 k^3 + 2^{2k}/p_c)$ as upper bound on the expected optimization time.

5 Summary

Evolutionary algorithms can be described as randomized search heuristics that are often applied as randomized optimization heuristics. This motivates the investigation of their efficiency as is done for other optimization algorithms, too. Placing evolutionary algorithms in the context of design and analysis of randomized algorithms in this way makes the question of their computational complexity one of immense importance.

The aim of this chapter is to demonstrate that the analysis of the expected optimization time of evolutionary algorithms is something that can be done. More importantly, it aims at

demonstrating how it can be done. This encompasses rather general approaches like “no free lunch” theorems and results on the black-box complexity. For the latter, Yao’s minimax principle is introduced and its application is demonstrated in practical examples. On the other hand, concrete tools and methods for the analysis of evolutionary algorithms’ expected optimization times are discussed.

The method of fitness-based partitions is the conceptually simplest of the analytical tools introduced. It measures the progress of an evolutionary algorithm by means of fitness values in a predefined way. For a successful analysis, the definition of an appropriate grouping of fitness values into fitness layers is crucial. For many functions, tight upper bounds on the expected optimization time can be achieved this way. The method, however, is restricted to algorithms employing a kind of elitist selection, it cannot deal with decreases in function value.

The method of the expected multiplicative distance decrease also establishes upper bounds on the expected optimization time by measuring the progress of an evolutionary algorithm by means of fitness values. It does so, however, in a more flexible way. Instead of defining static fitness layers, one identifies ways of increasing the fitness in all situations. This yields an average multiplicative decrease of the difference between the current fitness and the fitness of an optimal solution that finally leads to an upper bound on the average time needed to decrease the difference to 0. This method is restricted to functions with integer function values. Since a finite search space is dealt with, this is not a fundamental limitation. Multiplication can turn rational function values into integer ones. More fundamental is that the proven bounds depend on the size of the function values. While this can be removed to some extent (Reichel and Skutella 2009), it often leads to bounds that are not asymptotically tight.

An even more flexible way of measuring the progress of an evolutionary algorithm is drift analysis. Here, progress can be measured in an almost arbitrary way. Moreover, it suffices to have bounds on the expected progress in a single generation for the proof of bounds on the expected optimization time. This allows for the analysis of algorithms that do not guarantee monotone progress as long as the expected progress is positive.

These three general methods for establishing upper bounds on the expected optimization time are complemented by lower bound techniques. There is a very general lower bound for all mutation-based algorithms of size $\Omega(n \log n)$ that holds for almost all fitness functions of practical interest. While being useful, it does not clearly establish a technique. The most powerful technique for proving lower bounds on the expected optimization time is drift analysis. The differences between the drift techniques for establishing upper and lower bounds are rather small, and drift analysis is equally flexible and powerful for both upper and lower bounds. We concentrated on proving upper and lower bounds for expected optimization times that are polynomial. For proving exponential lower bounds, better-suited drift theorems are available that allow us to establish strong negative results at ease (Oliveto and Witt 2008).

Basing analysis on either typical events or typical runs are not upper or lower bound techniques per se. Here, a lower bound proven with the method of typical events and an upper bound using the method of typical runs are presented. Both methods require a good idea of how the evolutionary algorithms actually works on the concrete fitness function under consideration. It is noteworthy that using the method of typical runs one could derive proven results for an evolutionary algorithm with crossover. However, it is not the case that the other methods do not allow for such results.

All methods using example functions that are in some sense artificial are introduced. This has the advantage that these fitness functions have a very clear structure. They can easily be used to demonstrate certain effects. Here they allowed for a clear presentation of the analytical

tools employed. Most methods described here most often have been developed using such example functions that facilitate the analysis. It is worth mentioning that the method of the expected multiplicative distance decrease is an exception. It was introduced not using any specific example function but for a combinatorial optimization problem, namely the minimum spanning tree problem (Neumann and Wegener 2004).

Clearly, these methods for the analysis of evolutionary algorithms are in no way restricted to the analysis of such example functions. They have been successfully applied to a large number of combinatorial optimization problems and a variety of different variants of evolutionary algorithms, see Oliveto et al. (2007) for an overview.

We concentrated on the presentation of analytical tools in a way that allows the reader to employ these methods for his or her own problems. This leads to a presentation using practical examples instead of presenting the methods in their most general form. Moreover, a complete overview of all methods ever employed in the analysis of evolutionary algorithms is not presented. Two methods not covered here worth mentioning are the method of delay sequences (Dietzfelbinger et al. 2003) that allows for very tight bounds and the method of family trees that facilitates the analysis of an evolutionary algorithm with a population size $\mu > 1$ (Witt 2006). Moreover, the various mathematical tools that are useful in the analysis of algorithms are not covered in detail, in particular the application of martingales is not discussed (Jansen and Sudholt 2010; Williams 1991).

While aiming at the analysis of evolutionary algorithms no method is actually restricted to this class of algorithms. It is possible to apply these methods to other randomized algorithms, successful examples include ant colony optimization (Doerr et al. 2007; Sudholt and Witt 2008a), memetic algorithms (Sudholt 2008), and particle swarm optimization (Sudholt and Witt 2008b; Witt 2009). We can be sure to see these methods presented here further applied to more algorithms and more problems and to see more powerful methods developed in the future.

Acknowledgment

This material is based upon works supported by Science Foundation Ireland under Grant No. 07/SK/I1205.

References

- Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, 2nd edn. MIT Press, Cambridge, MA
- De Jong KA (1992) Genetic algorithms are NOT function optimizers. In: Whitley LD (ed) Proceedings of the second workshop on foundations of genetic algorithms (FOGA), Morgan Kaufmann, San Francisco, CA, pp 5–17
- Dietzfelbinger M, Naudts B, Hoyweghen CV, Wegener I (2003) The analysis of a recombinative hill-climber on H-IFF. *IEEE Trans Evolut Comput* 7(5): 417–423
- Doerr B, Neumann F, Sudholt D, Witt C (2007) On the runtime analysis of the 1-ANT ACO algorithm. In: Proceedings of the genetic and evolutionary computation conference (GECCO), ACM, New York, pp 33–40
- Doerr B, Jansen T, Klein C (2008) Comparing global and local mutations on bit strings. In: Proceedings of the genetic and evolutionary computation conference (GECCO), ACM, New York, pp 929–936
- Droste S, Jansen T, Wegener I (2002a) On the analysis of the (1+1) evolutionary algorithm. *Theor Comput Sci* 276:51–81

- Droste S, Jansen T, Wegener I (2002b) Optimization with randomized search heuristics – the (A)NFL theorem, realistic scenarios, and difficult functions. *Theor Comput Sci* 287(1):131–144
- Droste S, Jansen T, Wegener I (2006) Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory Comput Syst* 39(4): 525–544
- He J, Yao X (2004) A study of drift analysis for estimating computation time of evolutionary algorithms. *Nat Comput* 3(1):21–35
- Horn J, Goldberg DE, Deb K (1994) Long path problems. In: Davidor Y, Schwefel HP, Männer R (eds) *Proceedings of the 3rd international conference on parallel problem solving from nature (PPSN III)*, Springer, Berlin, Germany, LNCS 866, pp 149–158
- Igel C, Toussaint M (2003) On classes of functions for which no free lunch results hold. *Inf Process Lett* 86:317–321
- Igel C, Toussaint M (2004) A no-free-lunch theorem for non-uniform distributions of target functions. *J Math Model Algorithms* 3:313–322
- Jansen T, Sudholt D (2010) Analysis of an asymmetric mutation operator. *Evolut Comput* 18(1):1–26
- Jansen T, Wegener I (2000) On the choice of the mutation probability for the (1+1) EA. In: Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo-Guervós J, Schwefel HP (eds) *Proceedings of the 6th international conference on parallel problem solving from nature (PPSN VI)*, Springer, New York, LNCS 1917, pp 89–98
- Jansen T, Wegener I (2002) On the analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica* 34(1):47–66
- Jansen T, De Jong KA, Wegener I (2005) On the choice of the offspring population size in evolutionary algorithms. *Evolut Comput* 13(4):413–440
- Mitzenmacher M, Upfal E (2005) Probability and computing. Cambridge University Press, Cambridge, MA
- Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge, MA
- Neumann F, Wegener I (2004) Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In: *Proceedings of the genetic and evolutionary computation conference (GECCO)*, Springer, Berlin, Germany, LNCS 3102, pp 713–724
- Oliveto PS, Witt C (2008) Simplified drift analysis for proving lower bounds in evolutionary computation. In: *Proceedings of the 10th international conference on parallel problem solving from nature (PPSN X)*, Springer, Berlin, Germany, LNCS 5199, pp 82–91
- Oliveto PS, He J, Yao X (2007) Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. *Int J Automation Comput* 4(3):281–293
- Reichel J, Skutella M (2009) On the size of weights in randomized search heuristics. In: Garibay I, Jansen T, Wiegand RP, Wu A (eds) *Proceedings of the tenth workshop on foundations of genetic algorithms (FOGA)*, ACM, New York, pp 21–28
- Rudolph G (1997) How mutation and selection solve long path problems polynomial expected time. *Evolut Comput* 4(2):195–205
- Schumacher C, Vose MD, Whitley LD (2001) The no free lunch and problem description length. In: *Proceedings of the genetic and evolutionary computation conference (GECCO)*, Morgan Kaufmann, San Francisco, CA, pp 565–570
- Sudholt D (2008) Memetic algorithms with variable-depth search to overcome local optima. In: *Proceedings of the genetic and evolutionary computation conference (GECCO)*, ACM, New York, pp 787–794
- Sudholt D, Witt C (2008a) Rigorous analyses for the combination of ant colony optimization and local search. In: *Proceedings of ant colony and swarm intelligence (ANTS)*, Springer, Berlin, Germany, LNCS 5217, pp 132–143
- Sudholt D, Witt C (2008b) Runtime analysis of binary PSO. In: *Proceedings of the genetic and evolutionary computation conference (GECCO)*, ACM, New York, pp 135–142
- Williams D (1991) Probability with martingales. Cambridge University Press, Cambridge
- Witt C (2006) Runtime analysis of the $(\mu+1)$ EA on simple pseudo-boolean functions. *Evolut Comput* 14(1):65–86
- Witt C (2009) Why standard particle swarm optimizers elude a theoretical runtime analysis. In: Garibay I, Jansen T, Wiegand RP, Wu A (eds) *Proceedings of the tenth workshop on foundations of genetic algorithms (FOGA)*, ACM, New York, pp 13–20
- Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evolut Comput* 1(1):67–82
- Yao AC (1977) Probabilistic computations: towards a unified measure of complexity. In: *Proceedings of the 17th IEEE symposium on foundations of computer science (FOCS)*, New York, pp 222–227

27 Stochastic Convergence

Günter Rudolph

Department of Computer Science, TU Dortmund,
Dortmund, Germany
guenter.rudolph@tu-dortmund.de

1	<i>Introduction</i>	848
2	<i>Stochastic Process Models of Evolutionary Algorithms</i>	848
3	<i>Stochastic Convergence</i>	851
4	<i>Convergence Results for Evolutionary Algorithms</i>	855
5	<i>Further Reading and Open Questions</i>	866

Abstract

Since the state transitions of an evolutionary algorithm (EA) are of stochastic nature, the deterministic concept of the “convergence to the optimum” is not appropriate. In order to clarify the exact semantic of a phrase like “the EA converges to the global optimum” one has to, at first, establish the connection between EAs and stochastic processes before distinguishing between the various modes of stochastic convergence of stochastic processes. Subsequently, this powerful framework is applied to derive convergence results for EAs.

1 Introduction

Broadly speaking, the notion of “convergence” of an evolutionary algorithm (EA) simply means that the EA should approach some “limit” in the course of its evolutionary sequence of populations. Needless to say, this limit should represent something like the optimum of the optimization problem that one intends to solve. In this case, the “convergence” of an EA to a “limit” is a desirable property.

In order to endow the terms “convergence” and “limit” with a precise meaning in the context of EAs, it is necessary to embed EAs in the framework of “stochastic processes” (☞ Sect. 2) before the different modes of “stochastic convergence” and their basic results are introduced in ☞ Sect. 3. Subsequently, these basic results are applied, refined, and extended to EAs in ☞ Sect. 4, before ☞ Sect. 5 provides annotated pointers to the first results regarding the convergence theory of multi-objective EAs.

2 Stochastic Process Models of Evolutionary Algorithms

The analysis of the dynamic behavior of EAs benefits from the fact that it is always possible to establish a bijective mapping between an EA and a particular stochastic process. As a consequence, one can exploit the results and techniques of the well-founded theory of stochastic processes (Doob 1967).

Definition 1 Let $(X_t)_{t \in T}$ be a family of random variables (r.v.s) on a joint probability space (Ω, \mathcal{F}, P) with values in a set E of a measurable space (E, \mathcal{B}) and index set T . Then $(X_t)_{t \in T}$ is called a *stochastic process* with index set T . □

In general, there is no mathematical reason for restricting index set T to be a set of numerical values. But here the index set T is identical with \mathbb{N}_0 and the indices $t \in T$ will be interpreted as points of time.

Definition 2 A stochastic process $(X_t)_{t \in T}$ with index set $T = \mathbb{N}_0$ is called a *stochastic process with discrete time*. The image space E of $(X_t)_{t \in T}$ is called the *state space* of the process. The *transition probability*

$$P\{X_{t+1} \in A | X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_1 = x_1, X_0 = x_0\} \quad (1)$$

describes the probability of the event of transitioning to some state in set $A \subseteq E$ in step $(t+1)$ subject to previous steps though the state space E . □

Evidently, random variable X_t will represent the population at generation $t \geq 0$ and the function in $\textcircled{1}$ Eq. 1 will indicate the likeliness that the population at generation $t + 1$ will be some $x \in A$ conditioned by all previous populations. Since the transition probability $\textcircled{1}$ Eq. 1 must somehow capture the stochastic nature of the genetic operators and selection methods, it is clear that the modeling of the transition probability function is the main task when establishing a link between EA and stochastic process. Due to combinatorial complexity, the more difficult this task becomes the larger is the set of preceding populations on which the transition probabilities are conditioned. Fortunately, the vast majority of EAs are designed in a manner that does not require the most general formulation of the stochastic process model, although there are tools and results supporting an analysis (Iosifescu and Grigorescu 1990). But typically, the genetic operators and selection methods only act on the current population of individuals without taking into account the genomes of the antecedent populations of individuals. In other words: The phylogenetic trees of the current individuals do not affect the future. As a consequence, the transition probabilities only depend on the current population. Stochastic processes with this property are endowed with an extensive theory (see, e.g., Iosifescu (1980), Seneta (1981), Nummelin (1984), and Meyn and Tweedie (1993)) and they bear their own name:

Definition 3 A stochastic process $(X_t)_{t \in T}$ with discrete time whose transition probability satisfies

$$\mathbb{P}\{X_{t+1} \in A | X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_1 = x_1, X_0 = x_0\} = P_t\{X_{t+1} \in A | X_t = x_t\}$$

for all $t \geq 0$ is called a *Markov process with discrete time*. As for notation, often $P_t(x, A) := P_t\{X_{t+1} \in A | X_t = x\}$. If the transition probability of a Markov process with discrete time does not explicitly depend on the time parameter $t \geq 0$, that is,

$$\forall t \geq 0: P_t\{X_{t+1} = x | X_t = x_t\} = P\{X_{t+1} = x | X_t = x_t\},$$

then the Markov process is *time-homogeneous*, and *time-inhomogeneous* otherwise. A Markov process with countable state space is termed a *Markov chain*. \square

Since EAs work with discrete time, their stochastic models also use discrete time. Therefore, the phrase “with discrete time” will be left out hereinafter when talking about Markov processes and Markov chains. Next, suppose that an EA is deployed for solving the optimization problem

$$f(x) \rightarrow \min! \quad \text{subject to } x \in \mathcal{X} \tag{2}$$

where the objective function $f: \mathcal{X} \rightarrow \mathbb{R}$ is bounded from below. Now, solely, the type of the search space \mathcal{X} determines which type of Markov model and which part of Markov theory can be used.

2.1 Finite Search Space

If the search space \mathcal{X} is finite, so is the state space of the Markov chain. There are at least two approaches for modeling the state space. This will be exemplified by the binary search space $\mathcal{X} = \mathbb{B}^n$ with $\mathbb{B} = \{0, 1\}$.

2.1.1 Generic State Space Model

A straightforward generic model for the state space E of a Markov chain representing an EA with finite population size μ and search space $\mathcal{X} = \mathbb{B}^n$ is the μ -fold cartesian product $E = \mathcal{X}^\mu = (\mathbb{B}^n)^\mu = \mathbb{B}^{n \times \mu}$ that has been used in the early 1990s (see Eiben et al. (1991), Fogel (1994), Rudolph (1994b), and others). As a consequence, there are $|E| = 2^{n \times \mu}$ possible states and one needs $|E| \times |E|$ transition probabilities to characterize the stochastic effects caused by variation and selection operators. Evidently, these transition probabilities can be gathered in a square matrix $P = (p_{ij})$ of size $|E| \times |E|$, termed the *transition matrix*. Finally, one has to specify the *initial distribution* of the Markov chain to obtain a complete stochastic model of the EA. The initial distribution models the initialization of the EA.

The advantages of the generic state space model are that it also works for EAs with spatial structure and that it has a natural analogon for denumerable and innumerable search spaces.

2.1.2 Davis/Vose State Space Model

An alternative state space model was introduced independently by Davis (1991) and Vose (Nix and Vose 1992) also in the early 1990s. Its advantage is a smaller state space but its disadvantage is the fact that only panmictic EAs (every individual may mate with every other individual) without spatial structure can be modeled in this manner. Moreover, it has no analogon in case of innumerable search spaces.

Suppose that the position of the individuals in the population is of no importance and that the EA has a panmictic mating policy. Then, one only needs to know how many individuals of a certain type are present in the current population for deriving the transition probabilities. Since there are $|\mathcal{X}| = |\mathbb{B}^n| = 2^n$ possible individuals one needs a 2^n -tuple of nonnegative integers whose components count the number of occurrences of each type of individuals. Of course, the sum over all components must be exactly μ . As a consequence, the cardinality of the state space reduces to

$$|E| = \binom{2^n + \mu - 1}{\mu}$$

which is considerably smaller than the value $2^{n \times \mu}$ of the generic model. The transition matrix is of considerably smaller size as well.

Both approaches are thoroughly justified. But it is important to keep in mind that every EA that can be modeled by this approach can also be modeled in the generic setting whereas the converse is wrong in general. For example, see the efforts and insuperable difficulties in Muhammad et al. (1999) when trying to model a spatially structured EA with a Davis/Vose state space model.

2.2 Denumerable Search Space

An example of a denumerable but not finite search space is the integer search space $\mathcal{X} = \mathbb{Z}^n$. Apparently, this case has not been analyzed yet in the context of Markov chains although there exists a theory (see, e.g., part II in Seneta (1981)) using infinite-dimensional transition matrices. The reason might be due to the fact that in practice, the search space is almost

always equipped with box constraints leading to a finitely sized subset. Exceptions without box constraints are rare (see, e.g., Rudolph 1994c).

The generic state space model for a population of μ individuals is $E = \mathcal{X}^\mu = \mathbb{Z}^{n \times \mu}$ whereas the analogon of the Davis/Vose state space model $E = \{e \in \mathbb{N}_0^\infty : \sum_{i=1}^{\infty} e_i = \mu\}$ requires an infinite-dimensional tuple of nonnegative integers whose components count the number of occurrences of each type of individual and must add up to μ . Since this model has the same limitations as in the finite case and both models are now of infinite cardinality, it is quite obvious that there is no advantage of the Davis/Vose model in the denumerable case.

2.3 Innumerable Search Space

The typical representative for an innumerable search space is $\mathcal{X} = \mathbb{R}^n$. Here, only the generic state space model is expedient: $E = \mathcal{X}^\mu = \mathbb{R}^{n \times \mu}$. Since the probability to sample a specific point in \mathbb{R}^n from a continuous distribution is zero, the transition probabilities are specified by transition probability functions (in lieu of matrices) describing transitions from a state to a set of states with nonzero measure (see, e.g., Rudolph 1996).

3 Stochastic Convergence

The limit behavior of stochastic sequences requires a concept of convergence that captures the random nature of the sequence. This can be done in quite a different manner (Lukacs 1975). Here, only the most frequently used concepts are presented.

Definition 4 Let Z, Z_0, Z_1, \dots be random variables defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. The sequence $(Z_t : t \geq 0)$ is said

(a) to *converge completely* to random variable Z , denoted $Z_t \xrightarrow{c} Z$, if

$$\sum_{t=0}^{\infty} \mathbb{P}\{|Z_t - Z| > \varepsilon\} < \infty \text{ for every } \varepsilon > 0,$$

(b) to *converge with probability 1*, denoted $Z_t \xrightarrow{w.p.1} Z$, if

$$\mathbb{P}\left\{\lim_{t \rightarrow \infty} |Z_t - Z| = 0\right\} = 1,$$

(c) to *converge in probability* to Z , denoted $Z_t \xrightarrow{p} Z$, if

$$\lim_{t \rightarrow \infty} \mathbb{P}\{|Z_t - Z| > \varepsilon\} = 0 \text{ for every } \varepsilon > 0 \text{ and}$$

(d) to *converge in mean* to Z , denoted $Z_t \xrightarrow{m} Z$, if

$$\lim_{t \rightarrow \infty} \mathbb{E}[|Z_t - Z|] = 0. \quad \square$$

Basic relationships between these concepts of stochastic convergence are summarized in Theorem 1 below.

Theorem 1 (see Lukacs (1975), pp. 33–36 and 51–52) $Z_t \xrightarrow{c} Z \Rightarrow Z_t \xrightarrow{w.p.1} Z \Rightarrow Z_t \xrightarrow{p} Z$ and $Z_t \xrightarrow{m} Z \Rightarrow Z_t \xrightarrow{p} Z$. The converse is wrong in general. \square

Under additional conditions, some implications may be reversed. Here, the last implication is of special interest:

Theorem 2 (see Williams 1991, pp. 127–130) *If $|Z_t| \leq Y$ with $E[Y] < \infty$ for all $t \geq 0$ and $Z_t \xrightarrow{P} Z$ then also $Z_t \xrightarrow{m} Z$. \square*

Notice that the so-called *dominated convergence theorem* above also holds for the special case, where random variable Y is replaced by a finite constant $K \in (0, \infty)$. The example below is intended to provide a first impression about the differences between the modes of stochastic convergence introduced previously.

Example 1 (Modes of stochastic convergence) Let $(Z_t)_{t \geq 1}$ be a sequence of independent random variables. Depending on their probability distributions, the sequences converge to zero in different modes.

- $P\{Z_t = 0\} = 1 - \frac{1}{t}$ and $P\{Z_t = 1\} = \frac{1}{t}$

At first, one can confirm convergence to zero in probability since $P\{Z_t > \varepsilon\} = P\{Z_t = 1\} = \frac{1}{t} \rightarrow 0$ for $t \rightarrow \infty$. But there is no complete convergence since the probability mass does not move quickly enough to zero: $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = 1\} = \sum_{t=1}^{\infty} \frac{1}{t} = \infty$. Note that $0 \leq Z_t \leq 1$ for all $t \geq 0$. Evidently, the sequence is bounded from above by constant $K = 1$. Thanks to $\textcircled{1}$ **Theorem 1**, one has convergence in mean.

- $P\{Z_t = 0\} = 1 - \frac{1}{t^2}$ and $P\{Z_t = 1\} = \frac{1}{t^2}$

As can be seen from $P\{Z_t > \varepsilon\} = P\{Z_t = 1\} = \frac{1}{t^2} \rightarrow 0$ for $t \rightarrow \infty$ the sequence converges to zero in probability. Here, the convergence is quick enough to guarantee even complete convergence: $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = 1\} = \sum_{t=1}^{\infty} \frac{1}{t^2} < \infty$. For the same reasons as above, the sequence also converges in mean.

- $P\{Z_t = 0\} = 1 - \frac{1}{t}$ and $P\{Z_t = t\} = \frac{1}{t}$

This sequence also converges to zero in probability: $P\{Z_t > \varepsilon\} = P\{Z_t = t\} = \frac{1}{t} \rightarrow 0$ for $t \rightarrow \infty$. But the convergence is not quick enough to establish complete convergence: $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = t\} = \sum_{t=1}^{\infty} \frac{1}{t} = \infty$. Note that the sequence is not bounded from above. Moreover, the fact that $E[Z_t] = 0 \cdot P\{Z_t = 0\} + t \cdot P\{Z_t = t\} = t \cdot \frac{1}{t} = 1$ for all $t \geq 1$ reveals that the sequence does not converge to zero in mean.

- $P\{Z_t = 0\} = 1 - \frac{1}{t^2}$ and $P\{Z_t = t\} = \frac{1}{t^2}$

Thanks to $P\{Z_t > \varepsilon\} = P\{Z_t = t\} = \frac{1}{t^2} \rightarrow 0$ for $t \rightarrow \infty$ one may confirm not only convergence in probability but also complete convergence: $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = t\} = \sum_{t=1}^{\infty} \frac{1}{t^2} < \infty$. Although the sequence is not bounded from above, one can ensure convergence in mean via $E[Z_t] = 0 \cdot P\{Z_t = 0\} + t \cdot P\{Z_t = t\} = t \cdot \frac{1}{t^2} = \frac{1}{t} \rightarrow 0$ for $t \rightarrow \infty$.

- $P\{Z_t = 0\} = 1 - \frac{1}{t}$ and $P\{Z_t = t^2\} = \frac{1}{t}$

This sequence converges to zero in probability because of $P\{Z_t > \varepsilon\} = P\{Z_t = t^2\} = \frac{1}{t} \rightarrow 0$ for $t \rightarrow \infty$, but owing to $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = t^2\} = \sum_{t=1}^{\infty} \frac{1}{t} = \infty$ and $E[Z_t] = 0 \cdot P\{Z_t = 0\} + t^2 \cdot P\{Z_t = t^2\} = t^2 \cdot \frac{1}{t} = t \rightarrow \infty$ for $t \rightarrow \infty$ there is neither complete convergence nor convergence in mean.

- $P\{Z_t = 0\} = 1 - \frac{1}{t^2}$ and $P\{Z_t = t^2\} = \frac{1}{t^2}$

Finally, this sequence converges to zero in probability owing to $P\{Z_t > \varepsilon\} = P\{Z_t = t^2\} = \frac{1}{t^2} \rightarrow 0$ for $t \rightarrow \infty$ and it actually converges completely to zero on account

Table 1

The modes of stochastic convergence that are realized by the random sequences of Example 1 depending on their probability distributions

Probability distribution		\xrightarrow{c}	\xrightarrow{p}	\xrightarrow{m}
$P\{Z_t = 0\} = 1 - \frac{1}{t}$	$P\{Z_t = 1\} = \frac{1}{t}$	-	+	+
$P\{Z_t = 0\} = 1 - \frac{1}{t^2}$	$P\{Z_t = 1\} = \frac{1}{t^2}$	+	+	+
$P\{Z_t = 0\} = 1 - \frac{1}{t}$	$P\{Z_t = t\} = \frac{1}{t}$	-	+	-
$P\{Z_t = 0\} = 1 - \frac{1}{t^2}$	$P\{Z_t = t\} = \frac{1}{t^2}$	+	+	+
$P\{Z_t = 0\} = 1 - \frac{1}{t}$	$P\{Z_t = t^2\} = \frac{1}{t}$	-	+	-
$P\{Z_t = 0\} = 1 - \frac{1}{t^2}$	$P\{Z_t = t^2\} = \frac{1}{t^2}$	-	+	+

of $\sum_{t=1}^{\infty} P\{Z_t > \varepsilon\} = \sum_{t=1}^{\infty} P\{Z_t = t^2\} = \sum_{t=1}^{\infty} \frac{1}{t^2} < \infty$. Convergence to zero in mean, however, cannot be attested: $E[Z_t] = 0 \cdot P\{Z_t = 0\} + t^2 \cdot P\{Z_t = t^2\} = t^2 \cdot \frac{1}{t^2} = 1$ for all $t \geq 1$. \square

• **Table 1** summarizes the results obtained from this example.

Stochastic processes with special properties can be useful devices in the analysis of the limit behavior of evolutionary algorithms.

Definition 5 Let (Ω, \mathcal{F}, P) be a probability space and $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \dots \subseteq \mathcal{F}$ be an increasing family of sub- σ -algebras of \mathcal{F} and $\mathcal{F}_{\infty} := \sigma(\bigcup_t \mathcal{F}_t) \subseteq \mathcal{F}$. A stochastic process $(Z_t)_{t \geq 0}$ that is \mathcal{F}_t -measurable for each t is termed a *supermartingale* if

$$E[|Z_t|] < \infty \text{ and } E[Z_{t+1} | \mathcal{F}_t] \leq Z_t \text{ w.p.1}$$

for all $t \in \mathbb{N}_0$. \square

Nonnegative supermartingales, which means that additionally $Z_t \geq 0$ for all $t \geq 0$ have the following remarkable property:

Theorem 3 (see Neveu (1975, p. 26)) If $(Z_t)_{t \geq 0}$ is a nonnegative supermartingale then $Z_t \xrightarrow{w.p.1} Z < \infty$. \square

Although nonnegative supermartingales do converge to a finite limit with probability 1, nothing can be said about the limit itself unless additional conditions are imposed. The example below reveals that even a strict inequality, that is, $E[Z_{t+1} | \mathcal{F}_t] < Z_t$, does not necessarily imply a zero limit for nonnegative supermartingales.

Example 2 (Monotone decrease does not imply zero limit) Let $Z_t = 1 + 2^{-t} \cdot X_t^2 \geq 0$ for $t \geq 1$ where the sequence X_1, X_2, \dots consists of independent and identically distributed (i.i.d.) random variables with standard normal distribution $N(0, 1)$. Since $E[X_t^2] = 1$ for all $t \geq 1$ one has

$$E[E[Z_{t+1} | \mathcal{F}_t]] = 1 + 2^{-(t+1)} < E[Z_t] = 1 + 2^{-t}$$

for all $t \geq 1$ but $Z_t \xrightarrow{m} 1$ as $t \rightarrow \infty$. \square

For later purposes, it is of interest under which conditions the limit is the constant zero.

Theorem 4 (see Rudolph (1997a, p. 52)) *If $(Z_t)_{t \geq 0}$ is a nonnegative supermartingale satisfying*

$$\mathbb{E}[Z_{t+1} | \mathcal{F}_t] \leq c_t Z_t \text{ w.p.1}$$

for all $t \geq 0$ with $c_t \geq 0$ and

$$\sum_{t=1}^{\infty} \left(\prod_{k=0}^{t-1} c_k \right) < \infty \quad (3)$$

then $Z_t \xrightarrow{m} 0$ and $Z_t \xrightarrow{c} 0$.

It remains to provide some simple conditions that imply inequality (► Eq. 3).

Lemma 1 (see Rudolph (1997a, p. 53)) *The infinite series in (► Eq. 3) converges to a finite limit if*

- (a) $\limsup_{t \geq 0} c_t < 1$ or
- (b) $c_t \leq 1 - a/t$ for some $a > 1$ and almost all $t \geq 1$. □

The limit properties and stability of fixed points of deterministic dynamical systems can be analyzed via so-called Lyapunov functions. Bucy (1965) has shown that supermartingales play the role of Lyapunov functions in the stochastic setting. In fact, both concepts are closely related.

Theorem 5 (see Bucy (1965, pp. 153–154), and Bucy and Joseph (1968, pp. 83–84)) *Let $h: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ be a continuous function with $h(0, \cdot) = 0$. Consider the stochastic dynamical system*

$$X_{t+1} = h(X_t, Y_t) \quad (4)$$

for $t \geq 0$ with $X_0 \in \mathbb{R}^n$ and where $(Y_t)_{t \geq 0}$ is a sequence of random vectors. If there exists a continuous nonnegative function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ with the properties

- (a) $V(0) = 0$,
- (b) $V(x) \rightarrow \infty$ as $\|x\| \rightarrow \infty$,
- (c) $V(X_t)$ is a supermartingale along the motion of ► Eq. 4 and
- (d) there exists a continuous function $\gamma: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ vanishing only at the origin and along the motion of ► Eq. 4 holds

$$\mathbb{E}[V(X_{t+1}) | \mathcal{F}_t] \leq V(X_t) - \gamma(\|X_t\|) \quad (5)$$

then $\|X_t\| \xrightarrow{w.p.1} 0$ as $t \rightarrow \infty$. □

In the theorem above, the function $V(\cdot)$ is a Lyapunov function. This result can be used to obtain a slightly weaker result than ► Theorem 4 without any effort: Set $Z_t = V(X_t)$ such that $(Z_t)_{t \geq 0}$ is a nonnegative supermartingale and choose $\gamma(z) = (1 - c)z$ with $c \in (0, 1)$. Then all conditions of ► Theorem 5 including ► Eq. 5 with

$$\mathbb{E}[Z_{t+1} | \mathcal{F}_t] \leq Z_t - \gamma(Z_t) = Z_t - (1 - c)Z_t = cZ_t$$

are fulfilled and one may conclude that $Z_t \xrightarrow{w.p.1} 0$ as $t \rightarrow \infty$.

4 Convergence Results for Evolutionary Algorithms

With the definitions of the previous section, one can assign a rigorous meaning to the notion of the convergence of an evolutionary algorithm.

Definition 6 Let $(X_t : t \geq 0)$ be the sequence of populations generated by some evolutionary algorithm and let $F_t^* = \min\{f(X_{t,1}), \dots, f(X_{t,\mu})\}$ denote the best objective function value of the population of size $\mu < \infty$ at generation $t \geq 0$. An evolutionary algorithm is said to *converge in mean (in probability, with probability 1, completely) to the global minimum* $f^* = \min\{f(x) : x \in \mathcal{X}\}$ of objective function $f: \mathcal{X} \rightarrow \mathbb{R}$ if the nonnegative random sequence $(Z_t : t \geq 0)$ with $Z_t = F_t^* - f^*$ converges in mean (in probability, with probability 1, completely) to zero. \square

The convergence results for evolutionary algorithms depend on the type of the transitions and the search space.

4.1 Time-Homogeneous Transitions

4.1.1 Finite State Space

If the state space is finite and the transition matrix does not depend on the iteration counter, then the limit behavior of the Markov chain and its associated EA solely depend on the structure of the transitions matrix: Let $p^{(0)}$ denote the initial distribution and P the transition matrix. The *state distribution* $p^{(t)}$ of the Markov chain at step $t \geq 0$ is given by $p^{(t)} = p^{(t-1)} \cdot P$ for $t \geq 1$ and $p^{(0)}$ for $t = 0$. The Chapman–Kolmogorov equations (see, e.g., Iosifescu 1980, p. 65) reveal that actually $p^{(t)} = p^{(0)} \cdot P^t$ for $t \geq 0$, where P^t denotes the t th power of transition matrix P . Clearly, the structural properties of P determine the shape of the limit matrix P^∞ , if any, for $t \rightarrow \infty$ and, therefore, also the limit behavior of the EA. In order to exploit these facts, some terminology is required first:

Definition 7 A square matrix $P: m \times m$ is called a *permutation matrix* if each row and each column contain exactly one 1 and $m - 1$ zeros. A matrix A is said to be *cogredient* to a matrix B if there exists a permutation matrix P , such that $A = P'BP$. A square matrix A is said to be *diagonal-positive* if $a_{ii} > 0$ for all diagonal elements and it is said to be *nonnegative (positive)*, denoted $A \geq 0$ (> 0), if $a_{ij} \geq 0$ (> 0) for each entry a_{ij} of A . A nonnegative matrix is called *reducible* if it is cogredient to a matrix of the form

$$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}$$

where C and T are square matrices. Otherwise, the matrix is called *irreducible*. An irreducible matrix is called *primitive* if there exists a finite constant $k \in \mathbb{N}$, such that its k th power is positive. A nonnegative matrix is said to be *stochastic* if all its row sums are 1. A stochastic matrix is termed *stable* if it has identical rows and it is called *column allowable* if each column contains at least one positive entry. \square

Theorem 6 (Iosifescu 1980, p. 95) *Each transition matrix of a homogeneous finite Markov chain is cogredient to one of the following normal forms:*

$$P_1 = \begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_r \end{pmatrix} \quad \text{or} \quad P_2 = \begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ R_1 & R_2 & \cdots & R_r & T \end{pmatrix}$$

where submatrices C_1, \dots, C_r with $r \geq 1$ are irreducible and at least one of the submatrices R_i is nonzero. \square

Submatrix T in matrix P_2 above is associated with the *transient set* of the state space, whereas the matrices C_i are associated with the non-transient or *recurrent states*. Each matrix C_i represents a *recurrent set* of states that cannot be left once it is entered – as is evident from the structure of the normal forms above. In contrast, transient states can be visited several times but as soon as there is a transition to a recurrent state, the transient state will never be visited again. Thus, a transient state will be left forever with probability 1 after a finite number of iterations. Not surprisingly, recurrent and transient sets will play a different role in the limit behavior of a homogeneous finite Markov chain. Before presenting basic limit theorems, some terms have to be introduced:

Definition 8 Let P be the transition matrix of a homogeneous finite Markov chain. A distribution p on the states of the Markov chain is called a *stationary distribution* if $pP = p$ and a *limit distribution* if the limit $p = p^{(0)} \lim_{t \rightarrow \infty} P^t$ does exists. \square

Definition 9 Let $E = \mathcal{X}^\mu$ be the state space of a Markov chain representing some EA with population size $\mu \in \mathbb{N}$, finite search space \mathcal{X} and objective function $f: \mathcal{X} \rightarrow \mathbb{R}$ to be minimized. A state $x^* \in E$ is said to be *optimal* if there exists an $i = 1, \dots, \mu$ such that $f(x_i^*) = f^*$. The set E^* of all optimal states is termed the *optimal state set*. \square

Now some limit theorems may be stated:

Theorem 7 (Iosifescu 1980, p. 126; Seneta 1981, p. 127) *Let P be a reducible stochastic matrix, where $C: m \times m$ is a primitive stochastic matrix and $R, T \neq 0$. Then*

$$P^\infty = \lim_{k \rightarrow \infty} P^k = \lim_{k \rightarrow \infty} \begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{pmatrix} = \begin{pmatrix} C^\infty & 0 \\ R_\infty & 0 \end{pmatrix}$$

is a stable stochastic matrix with $P^\infty = 1' p^{(\infty)}$, where $p^{(\infty)} = p^{(0)} P^\infty$ is unique regardless of the initial distribution, and $p^{(\infty)}$ satisfies: $p_i^{(\infty)} > 0$ for $1 \leq i \leq m$ and $p_i^{(\infty)} = 0$ for $m < i \leq n$. The rate of approach to the limit is geometric. \square

Thanks to this general result, it is a straightforward exercise to formulate general limit theorems for EAs: Suppose that the transition matrix for some EA is reducible and that all associated recurrent states are optimal states, that is, $E_r \subseteq E^*$. In this case

$$\mathbb{P}\{Z_t \leq \varepsilon\} = \mathbb{P}\{F_t^* - f^* \leq \varepsilon\} = \mathbb{P}\{X_t \in E^*\} \geq \mathbb{P}\{X_t \in E_r\} = \sum_{i \in E_r} p_i^{(t)}$$

holds for sufficiently small $\varepsilon > 0$. Since $\mathbb{P}\{Z_t > \varepsilon\} = 1 - \mathbb{P}\{Z_t \leq \varepsilon\}$ one obtains

$$\mathbb{P}\{Z_t > \varepsilon\} \leq 1 - \mathbb{P}\{X_t \in E_r\} = 1 - \sum_{i \in E_r} p_i^{(t)} \quad (6)$$

so that the limit of \bullet Eq. 6 is

$$\lim_{t \rightarrow \infty} \mathbb{P}\{Z_t > \varepsilon\} \leq 1 - \sum_{i \in E_r} p_i^{(\infty)} = 1 - 1 = 0 \quad (7)$$

which is the defining condition for convergence to the optimum in probability. Thanks to \bullet Theorem 7, it is known that $p_i^{(\infty)} > 0$ if and only if $i \in E_r$. Therefore, the sum in \bullet Eq. 7 accumulates to 1. Moreover, \bullet Theorem 7 states that the approach to the limit is geometric. Since a geometric series (of probabilities < 1) converges to a finite limit, one obtains the stronger mode of complete convergence to the optimum. Convergence in mean follows from \bullet Theorem 2 since $Z_t \xrightarrow{P} 0$ and the value of Z_t must be bounded in a finite state space. As a result, we have proven:

Theorem 8 (Rudolph 1997a, p. 119) *If the transition matrix of an EA is reducible and the set of recurrent states is a subset of the set E^* of optimal states, then the EA converges completely and in mean to the global optimum regardless of the initial distribution.* \square

The above theorem is formulated as a sufficient criterion. Actually, it is also a necessary criterion as shown in Agapie (1998a) and it also holds for stochastic processes more general than Markov chains (“random systems with complete connections”).

Theorem 9 (Necessary and sufficient condition; see Agapie (1998b, 2007)) *A time-homogeneous EA with finite state space converges completely and in mean to the global optimum if and only if all recurrent states are optimal or, equivalently, if and only if all nonoptimal states are transient.* \square

Another general result from Markov chain theory can be exploited for a different limit result regarding EAs.

Theorem 10 (Iosifescu 1980, p. 123; Seneta 1981, p. 119) *Let P be a primitive stochastic matrix. Then P^k converges as $k \rightarrow \infty$ to a positive stable stochastic matrix $P^\infty = 1' p^{(\infty)}$, where the limit distribution $p^{(\infty)} = p^{(0)} \cdot \lim_{k \rightarrow \infty} P^k = p^{(0)} P^\infty$ has nonzero entries and is unique regardless of the initial distribution. The rate of approach to the limit is geometric. Moreover, the limit distribution is identical with the unique stationary distribution and is given by the solution $p^{(\infty)}$ of the system of linear equations $p^{(\infty)} P = p^{(\infty)}$, $p^{(\infty)} 1' = 1$.* \square

The weakest version of stochastic convergence considered here is convergence in probability. If this mode of convergence cannot be verified, then all stronger modes are precluded automatically. Now, suppose that the transition matrix for some EA is primitive. Owing to \bullet Theorem 10, one obtains

$$\mathbb{P}\{Z_t > \varepsilon\} = 1 - \mathbb{P}\{X_t \in E^*\} = 1 - \sum_{i \in E^*} p_i^{(t)} \quad (8)$$

for sufficiently small $\varepsilon > 0$. Note that $E^* \subset E$ unless the objective function is constant. Therefore

$$\sum_{i \in E^*} p_i^{(\infty)} < 1 \quad \text{and finally} \quad \lim_{t \rightarrow \infty} \mathbb{P}\{Z_t > \varepsilon\} = 1 - \sum_{i \in E^*} p_i^{(\infty)} > 0$$

falsifying convergence in probability. Thus, we have proven:

Theorem 11 (Rudolph 1997a, p. 120) *If the transition matrix of an EA is primitive, then the EA does not converge in probability to the global optimum regardless of the initial distribution.* \square

At this point, it should be noted that the property of visiting an optimal state with probability one is a precondition for convergence but that the additional property of convergence itself does not automatically indicate any advantage with respect to finding the global solution. Since all practical implementations of EAs store the best solution they have ever seen, it suffices to visit some optimal state only once. Therefore, EAs with primitive transition matrix also can be useful optimization algorithms.

Theorem 12 (Rudolph 1997a, p. 120) *Let $(X_t)_{t \geq 0}$ be the sequence of populations generated by an EA with primitive transition matrix. Then the stochastic sequence $(V_t)_{t \geq 0}$ of best objective function values ever found defined by $V_t = \min\{F_\tau^*: \tau = 0, 1, \dots, t\}$ converges completely and in mean to the global optimum.* \square

As shown above, the convergence properties of EAs can be determined by a closer look at their transition matrices. Since the transition matrix can be decomposed into a product of intermediate transition matrices, each of them describing the probabilistic behavior of a single evolutionary operator (like crossover, mutation, and selection), it is sufficient to map each evolutionary operator on a transition matrix and to look at their products. For this purpose, the following result is useful.

Lemma 2 (Rudolph 1994b, p. 97; Agapie 1998a, p. 188) *Let I, D, C, P, A be stochastic matrices where I is irreducible, D is diagonal-positive, C column-allowable, P positive, and A arbitrary. Then the products*

- (a) *AP and PC are positive,*
- (b) *ID and DI are irreducible.*

\square

After initialization, the typical cycle of an EA consists of selection for reproduction (transition matrix R), recombination / crossover (transition matrix C), mutation (transition matrix M), and selection for survival (transition matrix S). As a consequence, the transition matrix P of the EA can be decomposed via $P = R \cdot C \cdot M \cdot S$.

For example, the transition matrix M of standard bit-flipping mutation on binary strings has the entries $m_{ij} = p_m^{\|\bar{i}-\bar{j}\|}(1 - p_m^{n\mu - \|\bar{i}-\bar{j}\|})$, where $\|\cdot\|$ denotes the Hamming norm on binary strings. The entries m_{ij} are all positive if the mutation probability p_m is greater than 0 and less than 1. In this case, transition matrix M for mutation is positive. Now, \bullet Lemma 2(a) reveals, that regardless of the structure of the ‘crossover matrix’ C , the product $C \cdot M$ is positive. The same \bullet Lemma 2(a) implies that no structure of the transition matrix of selection for reproduction R can avoid that the product $R \cdot C \cdot M$ is positive. Finally, \bullet Lemma 2(a) also

ensures that the entire transition matrix P is positive, if the transition matrix of selection for survival S is column allowable. Since every positive matrix is also primitive, one may invoke \bullet Theorem 11 to conclude that the EA will not converge to the optimum. Results of this type have been proven in Davis and Principe (1993) and Rudolph (1994b) for proportional survival selection: the corresponding transition matrix S is in fact column-allowable. The same result can be achieved for other popular selection operations.

Lemma 3 (Rudolph 1997a, p. 122) *If the selection operation chooses from offspring only then the associated transition matrix is column-allowable for proportional selection, q -ary tournament selection, q -fold binary tournament selection, and truncation selection.* \square

The structure of matrix S is decisive for convergence if the preceding operations are represented by a positive matrix. If the selection operation is made *elitist* then the product of all transition matrices becomes a reducible matrix as required by \bullet Theorem 8. A proof of this result specialized to proportional selection was sketched in Eiben et al. (1991) and rigorously elaborated in Suzuki (1993, 1995). Elitism can be realized by at least two different mechanisms:

1. Select from parents and offspring and ensure that the best individual is selected with probability 1.
2. Select from offspring only; if the best selected individual is worse than the best parent, then replace the worst selected individual with the best parent.

Selection operations that can be used for elitism of the first kind are stochastic universal sampling, q -fold binary tournament selection, and truncation selection. Selection operations with elitism of the second kind can use almost any selection method in the first phase since the reinjection of the best parent, if necessary, in the second phase guarantees convergence once an optimum has been found for the first time.

Theorem 13 (Rudolph 1997a, p. 125) *If the transition matrix for mutation is positive and the selection operation for survival is elitist, then the EA converges completely and in mean to the optimum regardless of the initial distribution.* \square

There are at least two reasons for nonconvergence: The optimum cannot be found with probability 1 or the optimum is as often lost as it is found. The second case can be eliminated by elitism. Then it remains to make sure that the optimum is found with probability 1.

Without elitism, the mutation operator usually causes loss of an optimum previously found. Therefore, early research (prior to 1990) recommended to switch off mutation. But this maneuver does not help, since an EA without mutation may converge to an arbitrary (possibly nonoptimal) uniform population (Fogel 1994) that cannot be altered by crossover / recombination and selection. Nevertheless, for special cases (Rudolph 2005), it is possible to specify the probability (< 1) of finding the optimum depending on the population and problem size.

But elitism is not a CONDITIO SINE QUA NON to achieve convergence to the optimum for Markovian optimization algorithms with time-homogeneous transition matrices and finite state space (Agapie 1998b) as demonstrated in the next example.

Example 3 (Convergence to optimum without elitism) The idea for the example is taken from Agapie (1998b). Consider the threshold accepting (TA) algorithm (Dueck and

Algorithm 1 Threshold accepting

```

initialize individual  $X_0 \in \mathbb{B}^n$ 
set  $t = 0$ 
repeat
     $Y_t = X_t \oplus B_t$ 
    if  $f(Y_t) \leq f(X_t) + T$  then
         $X_{t+1} = Y_t$ 
    else
         $X_{t+1} = X_t$ 
    end if
    increment  $t$ 

```

Scheuer 1990) as shown in [Algorithm 1](#), where B_0, B_1, \dots is an i.i.d. sequence of Bernoulli random vectors (i.e., each component is an independent Bernoulli random variable with parameter $0 < p_m < 1$), $T > 0$ is a positive threshold and \oplus denotes the bitwise exclusive-or operation. Notice that TA would be exactly an *elitist* (1+1)-EA if $T = 0$.

Evidently, TA can reach every point in the search space in one step, it accepts every improvement and also worse points, provided the difference does not exceed the threshold $T > 0$. Since worse points may be accepted, the method is not elitist. If this algorithm is applied to the minimization of the objective function

$$f(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1 + \sum_{i=1}^n x_i, & \text{otherwise} \end{cases}$$

with threshold $T = 1$, then it is easily seen that TA may move to states with worse objective function value. TA may cycle freely through all states/solutions, but as soon as it hits the optimum at $x = 0$, then this state cannot be left since the difference in objective function values to all other solutions is larger than $T = 1$. Thus, only the optimal state 0 is recurrent whereas all other states are transient. Now, [Theorem 9](#) ensures convergence to the optimum without elitism. \square

The rigorous analysis in terms of Markov chain theory and the insight gained thereby has led to the conclusion that powerful sufficient conditions concerning properties and combinations of variation and selection operators can be derived already via simple basic probabilistic arguments (Rudolph 1998b) without using results from Markov chain theory: Let $(x_1, x_2, \dots, x_\mu) \in \mathcal{X}^\mu$ denote the population of μ parents. An offspring is produced as follows: At first, ρ parents are selected to serve as mates for the recombination process. This operation is denoted by

$$\text{mat} : \mathcal{X}^\mu \rightarrow \mathcal{X}^\rho$$

where $2 \leq \rho \leq \mu$. These individuals are then recombined by the procedure

$$\text{reco} : \mathcal{X}^\rho \rightarrow \mathcal{X}$$

yielding a preliminary offspring. Finally, a mutation via

$$\text{mut} : \mathcal{X} \rightarrow \mathcal{X}$$

yields the complete offspring. After all λ offspring have been produced in this manner, the selection procedure

$$\text{sel} : \mathcal{X}^k \rightarrow X^\mu$$

decides which offspring and possibly parents ($k \geq \mu$) will serve as the new parents in the next iteration. Thus, a single iteration of the evolutionary algorithm can be described as follows:

$$\forall i \in \{1, \dots, \lambda\} : x'_i = \text{mut}(\text{reco}(\text{mat}(x_1, \dots, x_\lambda)))$$

$$(y_1, \dots, y_\mu) = \begin{cases} \text{sel}(x_{\pi(1)}, \dots, x_{\pi(q)}, x'_1, \dots, x'_\lambda) & (\text{parents and offspring}) \\ \text{sel}(x'_1, \dots, x'_\lambda) & (\text{only offspring}) \end{cases}$$

where $1 \leq q \leq \mu$ and $\pi(1), \dots, \pi(\mu)$ is a permutation of the indices $1, \dots, n$ such that $f(x_{\pi(1)}) \leq f(x_{\pi(2)}) \leq \dots \leq f(x_{\pi(\mu)})$. This formulation includes selection methods that choose from the offspring and a subset of parents under the restriction that the best parent is a member of this subset.

After this operational description of evolutionary algorithms, one is in the position of defining some assumptions about the properties of the variation and selection operators:

- (A1) $\forall x \in (x_1, \dots, x_\mu) : P\{x \in \text{reco}(\text{mat}(x_1, \dots, x_\mu))\} \geq \delta_r > 0$.
- (A2) For every pair $x, y \in \mathcal{X}$ there exists a finite path x_1, x_2, \dots, x_τ of pairwise distinct points with $x_1 = x$ and $x_\tau = y$ such that $P\{x_{i+1} = \text{mut}(x_i)\} \geq \delta_m > 0$ for all $i = 1, \dots, \tau - 1$.
- (A2') For every pair $x, y \in \mathcal{X}$ holds $P\{y = \text{mut}(x)\} \geq \delta_m > 0$.
- (A3) $\forall x \in (x_1, \dots, x_k) : P\{x \in \text{sel}(x_1, \dots, x_k)\} \geq \delta_s > 0$.
- (A4) Let $v_k^*(x_1, \dots, x_k) = \max\{f(x_i) : i = 1, \dots, k\}$ denote the best fitness value within a population of k individuals ($k \geq \mu$). The selection method fulfills the condition

$$P\{v_\mu^*(\text{sel}(x_1, \dots, x_k)) = v_k^*(x_1, \dots, x_k)\} = 1.$$

Assumption (A₁) means that every parent may be selected for mating and is not altered by recombination with minimum probability $\delta_r > 0$. Assumption (A₂) ensures that every individual can be changed to an arbitrary other individual by a finite number of successive mutations, whereas assumption (A_{2'}) asserts the same but within a single mutation. Assumption (A₃) guarantees that every individual competing for survival may survive with minimum probability $\delta_s > 0$, whereas assumption (A₄) makes sure that the best individual among the competitors in the selection process will survive with probability one.

These assumptions lead to a nonzero lower bound $\delta_r \cdot \delta_m \cdot \delta_s > 0$ on the probability to move from some nonoptimal state to some other state on the finite path of length τ to the optimum. Repetitions of this argument lead to the lower bound $\delta = (\delta_r \cdot \delta_m \cdot \delta_s)^\tau > 0$ to reach an optimal state within τ iterations. Consequently, the probability that the optimal state has not been found after $t \geq \tau$ iterations is at most $(1 - \delta)^{\lfloor t/\tau \rfloor}$ converging to zero geometrically fast as $t \rightarrow \infty$. This proves complete and mean convergence to the optimum.

Theorem 14 (Rudolph 1998b) *If either assumption (A_{2'}) or the assumptions (A₁), (A₂), and (A₃) are valid then the evolutionary algorithm visits the global optimum after a finite number of iterations with probability one, regardless of the initialization. If assumption (A₄) is valid additionally and the selection method chooses from parents as well as offspring then the evolutionary algorithm converges completely and in mean to the global optimum regardless of the initialization.* \square

Finally, note that there are a number of general convergence conditions for EAs (see, e.g., He and Kang 1999 or He and Yu 2001) that are only of limited value since they are not further refined to properties of the variation and selection operations. It is not clear if such a refinement can lead to an extension of the results known currently.

4.1.2 Innumerable State Space

The convergence theory of probabilistic optimization methods resembling a $(1+1)$ -EA with $\mathcal{X} = \mathbb{R}^n$ and time-homogeneous transitions was established in Devroye (1976), Oppel and Hohenbichler (1978), Born (1978), Solis and Wets (1981), Pintér (1984), and others between the mid-1970s and mid-1980s. The proofs in each of these publications exploited the algorithms' property that the parent of the next generation cannot be worse than the current one, that is, it is guaranteed by the construction of the algorithms that the stochastic sequence $(Z_t : t \geq 0)$ is *monotonically* decreasing. As a consequence, $(Z_t : t \geq 0)$ is a nonnegative supermartingale that converges w.p.1 to a finite limit. Before summarizing the convergence results, some regularity conditions must be proclaimed: For all objective functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ holds

- (R₁) $f^* > -\infty$ and
- (R₂) the set $\mathcal{X}_\varepsilon^* = \bigcup_{x^* \in \mathcal{X}^*} \mathcal{V}_\varepsilon(x^*)$ has nonzero measure for all $\varepsilon > 0$,

where $\mathcal{V}_\varepsilon(x^*) = \{x \in \mathcal{X}: \|x - x^*\| \leq \varepsilon\}$ denotes the ε -vicinity around $x^* \in \mathcal{X}$. The first condition ensures the existence of an optimum whereas the second condition makes sure that at least one optimum is not an isolated point. The set $\mathcal{X}_\varepsilon^*$ is the set of ε -optimal solutions whereas $E_\varepsilon^* = \{x \in E \mid \exists i = 1, \dots, \mu : x_i \in \mathcal{X}_\varepsilon^*\}$ is the set of ε -optimal states.

Let $F^*(x) = \min\{f(x_1), f(x_2), \dots, f(x_\mu)\}$ denote the best objective function value in a state / population $x \in E$ consisting of μ individuals. Then the set $B(y) = \{x \in E : F^*(x) < F^*(y)\}$ is simply the set of states (populations) that contain an individual better than the best individual in population $y \in E$.

Theorem 15 (Rudolph 1997a, p. 201) *Let $X_0 \in E$ be the initial population of some elitist EA and let $P_c(x, A)$, $P_m(x, A)$ and $P_s(x, A)$ denote the transition probability functions of the crossover, mutation and selection operator, respectively, with $x \in E$ and $A \subseteq E$. If the conditions*

- (a) $\exists \delta_c > 0 : \forall x \in E : P_c(x, B(x)) \geq \delta_c$,
- (b) $\exists \delta_m > 0 : \forall x \in B(X_0) : P_m(x, E_\varepsilon^*) \geq \delta_m$ and
- (c) $\exists \delta_s > 0 : \forall x \in E : P_s(x, B(x)) \geq \delta_s$

hold simultaneously, then for every $\varepsilon > 0$, there exists a $\delta > 0$ such that $P_{cms}(x, A_\varepsilon) \geq \delta > 0$ for every $x \in B(X_0)$ and the EA converges completely and in mean to the optimum. \square

The next result refines \bullet Theorem 15 to specific variation and selection operators.

Theorem 16 (Rudolph 1997a, p. 204f) *A population-based EA with elitism that uses*

- (a) *multipoint, parametrized uniform, parametrized intermediate, averaging, gene pool, or parametrized intermediate gene pool recombination (with replacement);*

- (b) a mutation distribution with support \mathbb{R}^n and (possibly varying) positive definite covariance matrix whose spectrum stays in a fixed interval $[a, b] \subset \mathbb{R}_+$;
- (c) standard proportional, proportional SUS, q -ary tournament, q -fold binary tournament or top μ selection

converges completely and in mean to the global minimum of an objective function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ from the set $\{f \in F : f(x) \rightarrow \infty \text{ as } \|x\| \rightarrow \infty\}$. \square

EAs with search space $\mathcal{X} = \mathbb{R}^n$ can converge to the optimum even without elitism – but only for special classes of problems (Rudolph 1994a, 1997b,a) (e.g., (L, Q) -convex functions) and where the mutation distribution is scaled proportionally to the length of the gradient at the current position. If this kind of information is available, then it is possible to apply general convergence results for supermartingales like \bullet Theorem 4, which is specialized to EAs below.

Theorem 17 (Rudolph 1997b) Let $(X_t : t \geq 0)$ be the sequence of populations generated by some evolutionary algorithm and let F_t^* denote the best objective function value of the population at generation $t \geq 0$. If $E[Z_t] < \infty$ and

$$E[Z_{t+1} | X_t, X_{t-1}, \dots, X_0] \leq c_t Z_t \quad w.p.1 \quad (9)$$

where $Z_t = F_t^* - f^*$ and $c_t \in [0, 1]$ for all $t \geq 0$ such that the infinite product of the c_t converges to zero, then the evolutionary algorithm converges in mean and with probability 1 to the global minimum of the objective function $f(\cdot)$. \square

Convergence properties of EAs applied to constrained “corridor model” function in the context of Markov chains can be found in Agapie and Agapie (2007).

4.2 Time-Inhomogeneous Transitions

4.2.1 Finite Search Space

Apparently, the development of EAs with time-inhomogeneous transitions was motivated by the observation that, owing to \bullet Theorem 11, many popular EAs do not converge but there existed many convergence proofs for the simulated annealing (SA) optimization algorithm (Hajek 1988; Aarts and Korst 1989; Haario and Saksman 1991) that is endowed with time-inhomogeneous transitions. As a consequence, the EAs were modified to meet the convergence conditions of the SA convergence theory (Davis 1991; Mahfoud and Goldberg 1992, 1995; Adler 1993). A more general point of view was the starting point in Suzuki (1998) and Schmitt et al. (1998) and Schmitt (2001, 2004). They exploited the theory of time-inhomogeneous Markov chains. Before summarizing their results, the case of elitist selection and time-dependent variation operators is considered.

Theorem 18 Let $(X_t)_{t \geq 0}$ be the stochastic sequence generated by an EA with elitist selection and a transition probability function $P_t(x, A)$ with $A \subseteq E$, that only describes the sequence of variation operations. If $P_t(x, E^*) \geq \delta_t$ for all $x \in E \setminus E^*$ and

$$\sum_{t=0}^{\infty} \delta_t = \infty$$

then the EA converges w.p.1 and in mean to the optimum as $t \rightarrow \infty$.

Proof Since the selection operation is elitist, it suffices to show that the probability of not transitioning to some optimal state in E^* converges to zero as $t \rightarrow \infty$. This probability can be bounded via

$$\mathbb{P}\{X_t \notin E^*\} \leq \prod_{k=1}^t (1 - \delta_k) \quad (10)$$

In order to prove that the product above converges to zero, the following equivalence (see \blacklozenge Sect. 2.3.1 in Rudolph (1999)) is useful:

$$\prod_{t=1}^{\infty} (1 - \delta_t) \rightarrow 0 \Leftrightarrow \sum_{t=1}^{\infty} \log\left(\frac{1}{1 - \delta_t}\right) \rightarrow \infty$$

From the series expansion of the logarithm, it is clear that

$$\log\left(\frac{1}{1 - z}\right) > z$$

for all $z \in (0,1)$. As a consequence, since

$$\sum_{t=1}^{\infty} \log\left(\frac{1}{1 - \delta_t}\right) > \sum_{t=1}^{\infty} \delta_t = \infty$$

is fulfilled by the precondition of the theorem, it has been proven that the product in \blacklozenge Eq. 10 converges to zero. This proves convergence in probability. Since selection is elitist, the associated sequence of best function values $(Z_t)_{t \geq 0}$ is a nonnegative supermartingale that converges w.p.1 to some finite limit. Since the limits of w.p.1 convergence and convergence in probability must be unique, one obtains w.p.1 convergence to the optimum. Moreover, elitism implies that $\mathbb{E}[Z_t] \leq \mathbb{E}[Z_0]$ for all $t \geq 0$, which in turn implies convergence in mean. \square

It remains to show how the preconditions of the theorem can be fulfilled. For example, use an arbitrary crossover operation and the usual bit-flipping mutation with time-variant mutation probability $p_m(t)$ with $p_m(t) \rightarrow 0$ as $t \rightarrow \infty$. It suffices to bound the probability that just one individual becomes optimal. Evidently, if $p_m(t) \leq 1/2$ then

$$\delta_t \geq \min\{p_m^h(t) \cdot (1 - p_m(t))^{n-h} : h = 0, 1, \dots, n\} = p_m^n(t)$$

The inequality below

$$\sum_{t=1}^{\infty} \delta_t \geq \sum_{t=1}^{\infty} p_m^n(t) = \infty$$

is fulfilled if, for example, $p_m(t) = (t + 1)^{-1/n}$. Thus, the mutation probability must not decrease too fast.

If elitism is dropped from the list of preconditions, then at least the selection operation must become time-dependent. The detailed studies in Schmitt et al. (1998) and Schmitt (2001, 2004) contain numerous results in this direction, which are dared to be summarized as follows:

Theorem 19 (Schmitt 2004) *An EA that uses*

- (a) *scaled mutations with $p_m(t) \sim t^{-\kappa_m/n}$ where $\kappa_m \in (0,1]$,*

- (b) gene-lottery or regular pair-wise crossover with $p_c(t) \sim p_m^{\kappa_c}(t)$ where $\kappa_c \in (0,1]$,
 (c) proportional selection using exponentiation $f^{g(t)}$ with $g(t) \sim \log(t+1)$

converges w.p.1 and in mean to the optimum. \square

4.2.2 Innumerable Search Space

Time-inhomogeneous transitions in innumerable search spaces have not been studied for EAs, apparently. If crossover is arbitrary, mutation is chosen as in [Theorem 16\(b\)](#), and selection is done as in simulated annealing, then the SA convergence theory can be applied directly ([Bélisle 1992](#)). Moreover, there is a simple analogon to the result with elitist selection in finite state space.

Corollary 1 (to [Theorem 18](#)) Let $(X_t)_{t \geq 0}$ be the stochastic sequence generated by an EA with elitist selection and a transition probability function $P_t(x,A)$ with $A \subseteq E$, that only describes the sequence of variation operations. If $P_t(x, E_e^*) \geq \delta_t$ for all $x \in E \setminus E_e^*$ and

$$\sum_{t=0}^{\infty} \delta_t = \infty$$

then the EA converges w.p.1 and in mean to the optimum as $t \rightarrow \infty$.

Proof Set $\mathcal{X} = \mathbb{R}^n$, $E = \mathbb{R}^{n \times \mu}$, replace E by E_e^* in the statement and proof of [Theorem 18](#). \square

4.3 Self-Adaptive Transitions

Self-adaptation in EAs may appear in various forms. Here, the focus is on the special case where each individual consists of the position in the search space $\mathcal{X} = \mathbb{R}^n$ and a number of (strategy) parameters that determine the shape of the mutation distribution.

In the simplest case, an individual might be described by the pair (x, σ) with $x \in \mathcal{X}$ and $\sigma > 0$. A mutation of an individual can be realized by $y = x + \sigma Z$ where Z is a standard multinormally distributed random vector. If σ is altered depending on the outcome y , then σ has been self-adapted. Evidently, in the context of Markov chains, this is the time-homogeneous case – but the state space becomes larger since the strategy parameters also must be represented in the state space. This fact makes the analysis considerably more complicated. Therefore, the presentation of results for self-adaptive transitions have been separated from the case with fixed, externally set strategy parameters (like mutation and crossover probability).

General results known by now are rare. One may apply [Theorem 16](#) to ensure convergence to the optimum for any kind of self-adaptive mutation mechanisms as long as it is guaranteed that the support is \mathbb{R}^n and that the eigenvalues of the covariance matrices stay in a fixed compact interval in \mathbb{R}_+ . In this case, one can derive a lower positive bound on the probability to hit \mathcal{X}_e^* in every iteration, which implies convergence to the optimum. The situation changes if the spectrum of the eigenvalues has no restrictions, that is, the smallest eigenvalue may approach zero and the largest eigenvalue may grow arbitrarily large. This may happen for Rechenberg's $\frac{1}{2}$ -rule. It has been shown ([Rudolph 1999, 2001b](#)) that the probability

to escape from local, non-global optima under the $\frac{1}{5}$ -rule is strictly less than 1. As a consequence, convergence to the optimum is not guaranteed. But if the objective function only has a single local and therefore global optimum, then the $\frac{1}{5}$ -rule leads to convergence w. p.1 where the approximation error is decreased geometrically fast (Jägersküpper 2006, 2007). If selection only accepts points whose objective function value is at least $\varepsilon > 0$ better than the parent, and if the $\frac{1}{5}$ -rule is changed such that the variance is increased for success probabilities far below $\frac{1}{5}$ (say, below $\frac{1}{20}$), then convergence to the optimum can be asserted (Greenwood and Zhu 2001).

In case of mutative self-adaptation of the variance of the mutation distributions, only unimodal or spherical objective functions have been analyzed. Whereas the proof in Semenov and Terkel (2003) via stochastic Lyapunov functions/supermartingales has a theoretical gap (a lemma is “proven” by experiments), the proof in Bienvenüe and Francois (2003) and Auger (2004) via Markov processes ensure convergence to the optimum for objective function $f(x) = x^2$ with $x \in \mathbb{R}$.

5 Further Reading and Open Questions

The previous section tacitly assumed that EAs are applied to optimization problems with a single objective function. But optimization under multiple objectives is the *de facto* standard for practical applications nowadays. Unfortunately, the convergence theory cannot easily be transferred from single- to multi-objective EAs. First, the solution in multi-objective optimization (via the *a posteriori* approach) is not a single point but a set of points (the Pareto set) whose objective function values (the Pareto front) are only partially ordered. Second, it is unclear how to assess the quality of the Pareto front approximation: all points should be close to the true Pareto front, the points should represent the entire Pareto front, and they should be “somehow uniformly distributed” over the Pareto front approximation. Early work (Rudolph 1998a, c) proved that at least a single individual approaches the Pareto front (convergence of the scalar Euclidean distance between point and set). Subsequent work (Hanne 1999; Rudolph and Agapie 2000; Rudolph 2001a) proved that the entire population approaches the Pareto front (convergence of the distance between sets). But nothing was shown with respect to the spread of the individuals and their distribution: it was possible that the entire population converges to a single point of the Pareto front. The problem was that the quality of an approximation is described by three objectives. As a consequence, achieving a good Pareto front approximation is itself a multi-objective problem with possibly incomparable approximations. A good compromise was a scalar measure termed *dominated hypervolume* or *S-metric* (Zitzler and Thiele 1998), as it rewards closeness to the Pareto front, it rewards a good spread, and it rewards a certain distributions of solutions (the more curved the Pareto front the more solutions). Although this scalar measure is appealing for formulating convergence results, it seems to be quite difficult to accomplish this goal. Moreover, it is quite likely that only those multi-objective EAs can reach an approximation with best S-metric value that deploy the S-metric indicator in the selection process for deciding which individuals should be integrated or discarded from the approximation. For such an EA, it was shown that it can converge to an approximation with best S-metric value if the true Pareto front is linear (Beume et al. 2009) but there are counter-examples for nonlinear Pareto fronts (Zitzler et al. 2008). Another approach was initiated in Schütze et al. (2007) by proving approximation properties of the Pareto archive of multi-objective EAs.

Summing up, the convergence theory for self-adaptive transitions and for multi-objective EAs is not complete yet. Challenging open questions that await answers are the convergence properties of model-assisted EAs, the covariance matrix adaptation evolution strategy (CMA-ES), and other related methods.

References

- Aarts EHL, Korst J (1989) Simulated annealing and Boltzman machines: a stochastic approach to combinatorial optimization and neural computing. Wiley, Chichester
- Adler D (1993) Genetic algorithms and simulated annealing: A marriage proposal. In: IEEE international conference on neural networks, San Francisco, CA, March–April 1993. IEEE Press, Piscataway, NJ, pp 1104–1109
- Agapie A (1998a) Genetic algorithms: minimal conditions for convergence. In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (eds) AF'97: Artificial evolution: Third European conference; selected papers, Nimes, France, October 1997. Springer, Berlin, pp 183–193
- Agapie A (1998b) Modelling genetic algorithms: from Markov chains to dependance with complete connections. In: Bäck T, Eiben AE, Schoenauer M, Schwefel HP (eds) Parallel problem solving from nature – PPSN V. Springer, Berlin, pp 3–12
- Agapie A (2007) Evolutionary algorithms: modeling and convergence. Editura Academiei Române, Bucarest, Romania
- Agapie A, Agapie M (2007) Transition functions for evolutionary algorithms on continuous state-space. *J Math Model Algorithms* 6(2):297–315
- Auger A (2004) Convergence results for the $(1,\lambda)$ -ES using the theory of ϕ -irreducible Markov chains. *Theor Comput Sci* 334(1–3):181–231
- Bélisle CJP (1992) Convergence theorems for a class of simulated annealing algorithms on \mathbb{R}^d . *J Appl Probability* 29:885–895
- Beume N, Naujoks B, Preuss M, Rudolph G, Wagner T (2009) Effects of 1-greedy S-metric-selection on innumerably large pareto fronts. In: Ehrgott M et al. (eds) Proceedings of 5th international conference on evolutionary multi-criterion optimization (EMO 2009), Nantes, France, April 2009. Springer, Berlin, pp 21–35
- Bienvenüe A, Francois O (2003) Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. *Theor Comput Sci* 306(1–3):269–289
- Born J (1978) Evolutionsstrategien zur numerischen Lösung von Adaptionsaufgaben. Dissertation A, Humboldt-Universität, Berlin
- Bucz R (1965) Stability and positive supermartingales. *J Differ Equations* 1(2):151–155
- Bucz R, Joseph P (1968) Filtering for stochastic processes with applications to guidance. Interscience Publishers, New York
- Davis T (1991) Toward an extrapolation of the simulated annealing convergence theory onto the simple genetic algorithm. PhD thesis, University of Florida, Gainesville
- Davis T, Principe J (1993) A Markov chain framework for the simple genetic algorithm. *Evol Comput* 1(3): 269–288
- Devroye LP (1976) On the convergence of statistical search. *IEEE Trans Syst Man Cybern* 6(1):46–56
- Doob J (1967) Stochastic processes, 7th edn. Wiley, New York
- Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm superior to simulated annealing. *J Comput Phys* 90(1):161–175
- Eiben AE, Aarts EHL, van Hee KM (1991) Global convergence of genetic algorithms: a Markov chain analysis. In: Schwefel HP, Männer R (eds) Parallel problem solving from nature. Springer, Berlin, pp 4–12
- Fogel D (1994) Asymptotic convergence properties of genetic algorithms and evolutionary programming: analysis and experiments. *Cybern Syst* 25(3): 389–407
- Greenwood G, Zhu Q (2001) Convergence in evolutionary programs with self-adaptation. *Evol Comput* 9(2): 147–157
- Haario H, Saksman E (1991) Simulated annealing process in general state space. *Adv Appl Probability* 23:866–893
- Hajek B (1988) Cooling schedules for optimal annealing. *Math Oper Res* 13(2):311–329
- Hanne T (1999) On the convergence of multiobjective evolutionary algorithms. *Eur J Oper Res* 117(3): 553–564
- He J, Kang K (1999) On the convergence rates of genetic algorithms. *Theor Comput Sci* 229(1–2):23–39
- He J, Yu X (2001) Conditions for the convergence of evolutionary algorithms. *J Syst Archit* 47(7): 601–612
- Iosifescu M (1980) Finite Markov processes and their applications. Wiley, Chichester

- Iosifescu M, Grigorescu S (1990) Dependence with complete connections and its applications. Cambridge University Press, Cambridge
- Jägersküpper J (2006) How the (1+1) ES using isotropic mutations minimizes positive definite quadratic forms. *Theor Comput Sci* 361(1):38–56
- Jägersküpper J (2007) Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theor Comput Sci* 379(3):329–347
- Lukacs E (1975) Stochastic convergence, 2nd edn. Academic, New York
- Mahfoud SW, Goldberg DE (1992) A genetic algorithm for parallel simulated annealing. In: Männer R, Manderick B (eds) Parallel problem solving from nature, vol. 2. North Holland, Amsterdam, pp 301–310
- Mahfoud SW, Goldberg DE (1995) Parallel recombinative simulated annealing: a genetic algorithm. *Parallel Comput* 21:1–28
- Meyn S, Tweedie R (1993) Markov chains and stochastic stability. Springer, London
- Muhammad A, Bargiela A, King G (1999) Fine-grained parallel genetic algorithm: a global convergence criterion. *Int J Comput Math* 73(2):139–155
- Neveu J (1975) Discrete-parameter martingales. North Holland, Amsterdam
- Nix A, Vose M (1992) Modeling genetic algorithms with Markov chains. *Ann Math Artif Intell* 5:79–88
- Nummelin E (1984) General irreducible Markov chains and non-negative operators. Cambridge University Press, Cambridge
- Oppel U, Hohenbichler M (1978) Auf der Zufallssuche basierende Evolutionsprozesse. In: Schneider B, Ranft U (eds) Simulationsmethoden in der Medizin und Biologie. Springer, Berlin, pp 130–155
- Pintér J (1984) Convergence properties of stochastic optimization procedures. *Math Oper Stat Ser Optimization* 15:405–427
- Rudolph G (1994a) Convergence of non-elitist strategies. In: Proceedings of the first IEEE conference on evolutionary computation, vol. 1. Orlando, FL, June 1994. IEEE Press, Piscataway, NJ, pp 63–66
- Rudolph G (1994b) Convergence properties of canonical genetic algorithms. *IEEE Trans Neural Netw* 5(1): 96–101
- Rudolph G (1994c) An evolutionary algorithm for integer programming. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel problem solving from nature, vol. 3. Springer, Berlin, pp 139–148
- Rudolph G (1996) Convergence of evolutionary algorithms in general search spaces. In: Proceedings of the third IEEE conference on evolutionary computation, Nagoya, Japan, May 1996. IEEE Press, Piscataway, NJ, pp 50–54
- Rudolph G (1997a) Convergence properties of evolutionary algorithms. Kovač, Hamburg
- Rudolph G (1997b) Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control Cybern* 26(3):375–390
- Rudolph G (1998a) Evolutionary search for minimal elements in partially ordered finite sets. In: Porto VW, Saravanan N, Waagen D, Eiben AE (eds) Evolutionary programming VII, Proceedings of the 7th annual conference on evolutionary programming, San Diego, CA, March 1998. Springer, Berlin, pp 345–353
- Rudolph G (1998b) Finite Markov chain results in evolutionary computation: a tour d'horizon. *Fund Inform* 35(1–4):67–89
- Rudolph G (1998c) On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In: Proceedings of the 1998 IEEE international conference on evolutionary computation, Anchorage, AK, May 1998. IEEE Press, Piscataway, NJ, pp 511–516
- Rudolph G (1999) Global convergence and self-adaptation: A counter-example. In: CEC'99: Proceedings of the 1999 congress of evolutionary computation, vol. 1. Washington, DC, July 1999. IEEE Press, Piscataway, NJ, pp 646–651
- Rudolph G (2001a) Evolutionary search under partially ordered fitness sets. In: Sebaaly MF (ed) ISI 2001: Proceedings of the international NAISO congress on information science innovations, Dubai, UAE, March 2001. ICSC Academic Press, Millet and Sliedrecht, pp 818–822
- Rudolph G (2001b) Self-adaptive mutations may lead to premature convergence. *IEEE Trans Evol Comput* 5(4):410–414
- Rudolph G (2005) Analysis of a non-generational mutationless evolutionary algorithm for separable fitness functions. *Int J Comput Intell Res* 1(1):77–84
- Rudolph G, Agapie A (2000) Convergence properties of some multi-objective evolutionary algorithms. In: Zalzala A, Fonseca C, Kim JH, Smith A, Yao X (eds) CEC 2000: Proceedings of the 2000 congress on evolutionary computation, vol. 2. La Jolla, CA, July 2000. IEEE Press, Piscataway, NJ, pp 1010–1016
- Schmitt L (2001) Theory of genetic algorithms. *Theor Comput Sci* 259(1–2):1–61
- Schmitt L (2004) Theory of genetic algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness functions under scaling. *Theor Comput Sci* 310(1–3):181–231
- Schmitt LM, Nehaniv CL, Fujii RH (1998) Linear analysis of genetic algorithms. *Theor Comput Sci* 200 (1–2):101–134
- Schütze O, Laumanns M, Tantar E, Coello Coello C, Talbi EG (2007) Convergence of stochastic search algorithms to gap-free Pareto front approximations. In: GECCO 2007: Proceedings of the 9th annual conference on genetic and evolutionary computation, London, July 2007. ACM, New York, pp 892–901

- Semenov M, Terkel D (2003) Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic Lyapunov function. *Evol Comput* 11(4):363–379
- Seneta E (1981) Non-negative matrices and Markov chains, 2nd edn. Springer, New York
- Solis FJ, Wets RJB (1981) Minimization by random search techniques. *Math Oper Res* 6(1):19–30
- Suzuki J (1993) A Markov chain analysis on a genetic algorithm. In: Forrest S (ed) Proceedings of the fifth international conference on genetic algorithms, Urbana-Champaign, IL, June 1993. Morgan Kaufmann, San Mateo, CA, pp 146–153
- Suzuki J (1995) A Markov chain analysis on simple genetic algorithms. *IEEE Trans Syst Man Cybern* 25(4):655–659
- Suzuki J (1998) A further result on the Markov chain model of genetic algorithm and its application to a simulated annealing-like strategy. *IEEE Trans Syst Man Cybern B* 28(1):95–102
- Williams D (1991) Probability with martingales. Cambridge University Press, Cambridge
- Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms: comparative case study. In: Eiben A et al. (eds) Parallel problem solving from nature (PPSN V). Springer, Berlin, pp 292–301
- Zitzler E, Thiele L, Bader J (2008) On set-based multi-objective optimization. Technical Report 300, Computer Engineering and Networks Laboratory, ETH Zurich

28 Evolutionary Multiobjective Optimization

Eckart Zitzler

PHBern – University of Teacher Education, Institute for Continuing Professional Education, Weltistrasse 40, CH-3006, Bern, Switzerland
eckart.zitzler@phbern.ch
eckart.zitzler@tik.ee.ethz.ch

1	<i>Introduction</i>	872
2	<i>Fundamental Concepts</i>	873
3	<i>Approaches to Multiobjective Optimization</i>	879
4	<i>Search Algorithm Design</i>	884
5	<i>Applications</i>	895

Abstract

This chapter provides an overview of the branch of evolutionary computation that is dedicated to solving optimization problems with multiple objective functions. On the one hand, it sketches the foundations of multiobjective optimization and discusses general approaches to deal with multiple optimization criteria. On the other hand, it summarizes algorithmic concepts that are employed when designing corresponding search methods and briefly comments on the issue of performance assessment. This chapter concludes with a summary of the main application areas of evolutionary multiobjective optimization, namely, learning/decision making and multiobjectivization.

1 Introduction

The term *evolutionary multiobjective optimization* – EMO for short – refers to the employment of evolutionary algorithms to search problems involving multiple optimization criteria. This type of problem arises naturally in practical applications, simply because there is usually no such thing as a ‘free lunch’: in embedded system design high performance comes along with high cost, in structural engineering the level of stability is related to the weight of the construction, and in machine learning the accuracy of a model needs to be traded off against the model complexity. There exist countless other examples, many of which are not restricted to a pair of counteracting criteria but a multitude of competing objectives. The design of a car, for instance, requires many aspects to be taken into account, be it driving characteristics, cost, environmental sustainability, or more subjective criteria such as comfort and appearance. One may argue that the multiobjective case represents the *normal* situation in reality, while the single-objective case is rather an exception.

Example 1 Consider the following knapsack problem, a classical combinatorial optimization problem. Given is a set $\{1, \dots, l\}$ of items and with each item i a corresponding weight $w_i \in \mathbb{R}^+$ and a corresponding profit $p_i \in \mathbb{R}^+$ is associated. The goal is to identify a subset S of the items that (i) maximizes the overall profit $f_p(S)$ and (ii) minimizes the overall weight $f_w(S)$ where

$$f_p(S) = \sum_{i \in S} p_i \quad (1)$$

$$f_w(S) = \sum_{i \in S} w_i \quad (2)$$

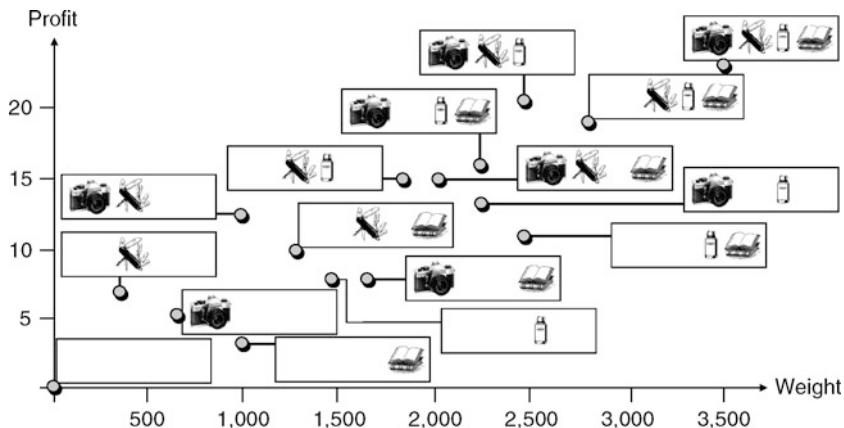
are the corresponding objective functions.  [Figure 1](#) illustrates a problem instance with four items.

It is obvious that with the knapsack problem, in general, both objectives cannot be achieved at the same time: maximum profit means all items need to be selected, minimum weight is obtained by choosing none of the items. In between these extremes, several compromise solutions may emerge, each of them representing a particular trade-off of profit versus weight. Classically, this problem is approached by transforming the weight objective into a constraint, that is,

$$\begin{aligned} & \text{argmax}_{S \subseteq \{1, \dots, l\}} f_p(S) \\ & \text{subject to} \quad f_w(S) \leq w_{\max} \end{aligned} \quad (3)$$

Fig. 1

Example of a knapsack problem instance with four items: (1) a camera with profit $p_1 = 5$ and weight $w_1 = 750$, (2) a pocket knife with profit $p_2 = 7$ and weight $w_2 = 300$, (3) a thermos flask with profit $p_3 = 8$ and weight $w_3 = 1,500$, and (4) a book with profit $p_4 = 3$ and weight $w_4 = 1,000$. All possible solutions are plotted with respect to their objective function values.



where w_{\max} is a fixed weight limit. This is one way to handle scenarios with multiple objectives, but various others exist.

The field of *Multiple Criteria Decision Making (MCDM)* explicitly addresses these issues and can be defined as (Steuer and Miettinen 2008)

- ▶ the study of methods and procedures by which concerns about multiple conflicting criteria can be formally incorporated into the management planning process.

In this sense, EMO can be regarded as a branch of MCDM (as well as it can be considered a branch of evolutionary computation) where evolutionary algorithms and other randomized search heuristics are the methods of choice. From a historical perspective, though, MCDM has been – until recently – mainly devoted to exact optimization techniques such as linear programming, while the first EMO studies pursued a specific MCDM approach, namely to find a *set* of suitable compromise solutions. Meanwhile, the boundaries between these communities have blurred, but still the large majority of papers dedicated to EMO is related to approximating the set of optimal compromise solutions, as will be detailed later.

This chapter is an attempt to summarize 25 years of EMO research from a unified point of view. The focus is on general concepts, and therefore the following overview does not lay claim to completeness, with respect to both specific research topics and publications in the field. Readers interested in a detailed discussion of specific algorithms, a complete historical overview, and an extensive collection of references may consult, for example, the textbooks by Deb (2001) and Coello Coello et al. (2007).

2 Fundamental Concepts

Before different ways on how to approach scenarios with multiple optimization criteria and how to design evolutionary algorithms for this purpose are discussed, some basic notions and reflections on multiobjective optimization will be presented in general.

2.1 Multiobjective Optimization Problems

Roughly speaking, optimization stands for the task of identifying within a set of solutions one solution that is best (or close to best). The set of potential solutions is denoted as *decision space*, in the following represented by the symbol X , and its elements are denoted as decision vectors – or simply solutions. Sometimes, the terms “decision space” and “search space” are used interchangeably, although here a subtle differentiation will be made: the search space is the space a concrete evolutionary algorithm operates on; it can be different from the decision space, which is part of the formal description of the optimization problem.

In order to define what “best” means, all solutions are evaluated by means of objective functions on the basis of which they can be compared. Without loss of generality, here it is assumed that an objective function f returns a real value, that is, $f: X \mapsto \mathbb{R}$, although the codomain could be any set. In single-objective optimization, there is just one objective function, while in the multiobjective case there are several ones, say f_1, \dots, f_n . These functions form a vector function $\mathbf{f} = (f_1, \dots, f_n)$ that assigns each decision vector a real vector, or *objective vector* in general. The objective vector contains the function values for all n objective functions and is an element of the so-called objective space Z , here $Z = \mathbb{R}^n$. Therefore, $\mathbf{f}: X \mapsto Z$ provides the mapping from the decision space to the objective space.

Finally, a binary relation \leqslant specifies an order on the objective space, which completes the formal description of the optimization problem. The minimum requirement for this relation is usually that it is a partial order, that is, reflexive, antisymmetric, and transitive. (A binary relation \leq over a set Y is (1) reflexive iff $a \leq a$ for all $a \in Y$, (2) antisymmetric iff $(a \leq b \wedge b \leq a) \Rightarrow (a = b)$ for all $a, b \in Y$, and (3) transitive iff $(a \leq b \wedge b \leq c) \Rightarrow (a \leq c)$ holds for all a, b , and c in Y .) In the single-objective case, usually the relations \leq (minimization) or \geq (maximization) are employed, in the multiobjective case, the choice is less obvious and will be discussed later in [Sect. 2.2](#). Now given \leqslant , two solutions $\mathbf{a}, \mathbf{b} \in X$ can be compared by first determining the objective vectors $\mathbf{f}(\mathbf{a}), \mathbf{f}(\mathbf{b})$ and then checking whether $\mathbf{f}(\mathbf{a}) \leqslant \mathbf{f}(\mathbf{b})$ or vice versa. The goal is to find a solution \mathbf{a} that is best, that is, there is no better solution; formally

$$\forall \mathbf{b} \in X : (\mathbf{f}(\mathbf{b}) \leqslant \mathbf{f}(\mathbf{a})) \Rightarrow (\mathbf{f}(\mathbf{a}) \leqslant \mathbf{f}(\mathbf{b})) \quad (4)$$

Furthermore, constraints may be involved, each of which can be regarded as a special case of an objective function in combination with a prespecified threshold. Fulfilling a constraint means the given threshold must not be exceeded; here it is assumed that a constraint is given by a constraint function $g: X \mapsto \mathbb{R}$, which needs to be equal to or less than 0 to be met. Several such constraint functions may be considered simultaneously, and a decision vector fulfilling the constraints is called a *feasible solution*; the entirety of the feasible solutions constitutes the *feasible set*. Hence, the optimization goal can be restated as finding the best among the feasible solutions.

Example 2 In the classical formulation of the knapsack problem according to [Eq. 3](#), the decision space is $X = \{0, 1\}^l$, the objective space is $Z = \mathbb{R}$, the objective function is $\mathbf{f} = (f_1)$ with $f_1(a_1, \dots, a_l) = \sum_{1 \leq i \leq l} a_i \cdot p_b$, the considered relation is \geq , and the weight is included using a constraint function $g(a_1, \dots, a_l) = (\sum_{1 \leq i \leq l} a_i \cdot w_i) - w_{\max}$ where w_{\max} is the weight limit. For the example in [Fig. 1](#), the feasible set is $X_f = \{(0, 0, 0, 0), (0, 1, 0, 0), (1, 0, 0, 0)\}$ when setting the weight limit to $w_{\max} = 800$.

In summary, a (multiobjective) optimization scenario can be described as follows:

- A *multiobjective optimization problem* is defined by a 5-tuple $(X, Z, \mathbf{f}, \mathbf{g}, \leq)$ where
 - X is the decision space,
 - $Z = \mathbb{R}^n$ is the objective space,
 - $\mathbf{f} = (f_1, \dots, f_n)$ is a vector-valued function consisting of n objective functions $f_i: X \mapsto \mathbb{R}$,
 - $\mathbf{g} = (g_1, \dots, g_m)$ is a vector-valued function consisting of m constraint functions $g_j: X \mapsto \mathbb{R}$, and
 - $\leq \subseteq Z \times Z$ is a binary relation on the objective space.

The goal is to identify a decision vector $\mathbf{a} \in X$ such that (i) for all $1 \leq i \leq m$ holds $g_i(\mathbf{a}) \leq 0$ and (ii) for all $\mathbf{b} \in X$ holds $\mathbf{f}(\mathbf{b}) \leq \mathbf{f}(\mathbf{a}) \Rightarrow \mathbf{f}(\mathbf{a}) \leq \mathbf{f}(\mathbf{b})$.

The question of how to define the relation \leq in a multiobjective context has remained open so far – it will be addressed next.

2.2 Preference Relations and Optimality

Consider the biobjective knapsack problem from ◉ [Example 1](#). Given two item sets S and T and the corresponding objective vectors (p_S, w_S) and (p_T, w_T) , when can S be said to be superior to T ? Clearly, if S is better in both objectives, that is, it provides higher profit at lower weight, then it can be called overall superior. Even if S is only better in one objective, while there is no difference in the other, for example, $p_S > p_T \wedge w_S = w_T$ then one may make the same statement. That means, in general, that a solution is better than another if the former is (1) not worse in any objective and (2) better in at least one objective. These considerations lead to a concept known as *(weak) Pareto dominance*.

- An objective vector $\mathbf{u} = (u_1, \dots, u_n)$ *weakly Pareto dominates* an objective vector $\mathbf{v} = (v_1, \dots, v_n)$, written as $\mathbf{u} \leq_{par} \mathbf{v}$, iff

$$\forall 1 \leq i \leq n : u_i \leq v_i \quad (5)$$

assuming without loss of generality that all objectives are to be minimized. Furthermore, iff

$$\mathbf{u} \leq_{par} \mathbf{v} \wedge \mathbf{v} \not\leq_{par} \mathbf{u} \quad (6)$$

then \mathbf{u} is said to *(Pareto) dominate* \mathbf{v} .

Using this concept, an objective vector is said to be better than another if the former weakly Pareto dominates the latter, while the latter does not weakly Pareto dominate the former. Pareto dominance represents the most commonly accepted notion of superiority in a multiobjective setting since it is a canonical generalization of the single-objective case. However, dominance may be defined in other ways as well, see ◉ [Fig. 2](#). One example is the so-called (additive) epsilon dominance (Helbig and Pateva 1994; Laumanns et al. 2002; Zitzler et al. 2003) defined as

$$\mathbf{u} \leq_{\varepsilon} \mathbf{v} : \Leftrightarrow \forall 1 \leq i \leq n : u_i \leq v_i + \varepsilon \quad (7)$$

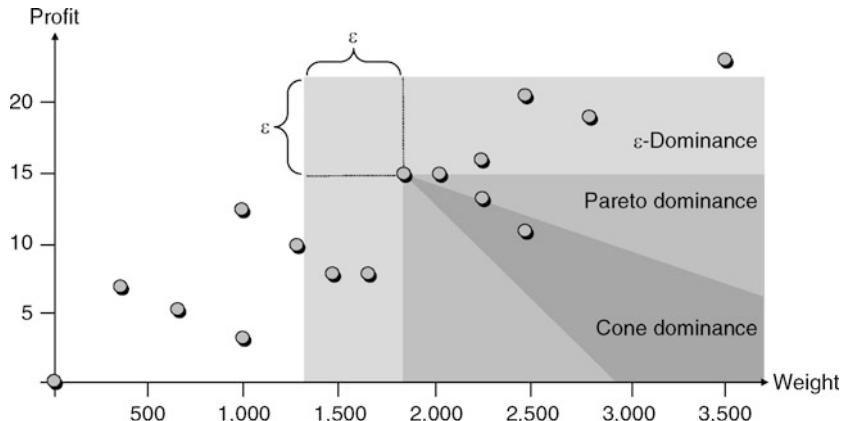
which is a relaxation of Pareto dominance using an additive term $\varepsilon \in \mathbb{R}$; another possibility is to use ordering cones (Miettinen 1999; Ehrgott 2005).

The dominance relation \leq structures the objective space and thereby implicitly induces a corresponding *preference relation* \preccurlyeq on the decision space with

$$\mathbf{a} \preccurlyeq \mathbf{b} : \Leftrightarrow \mathbf{f}(\mathbf{a}) \leq \mathbf{f}(\mathbf{b}) \quad (8)$$

Fig. 2

Illustration of three dominance concepts based on the example shown in [Fig. 1](#): Pareto dominance, epsilon dominance, and dominance using a pointed convex cone. The figure shows for a particular objective vector the area that is dominated in the objective space. Note that the considered knapsack problem is a mixed minimization/maximization problem and that, therefore, the definitions of the dominance relations change accordingly.



In this sense, the triple (f, Z, \leq) can be regarded as an indirect means to construct an order on X . In principle, there are also other ways to define this order and therefore the term preference relation will be used in a general notion:

- A *preference relation* $\preccurlyeq \subseteq X \times X$ is a binary relation on the decision space that is both reflexive and transitive, that is, \preccurlyeq is a preorder.

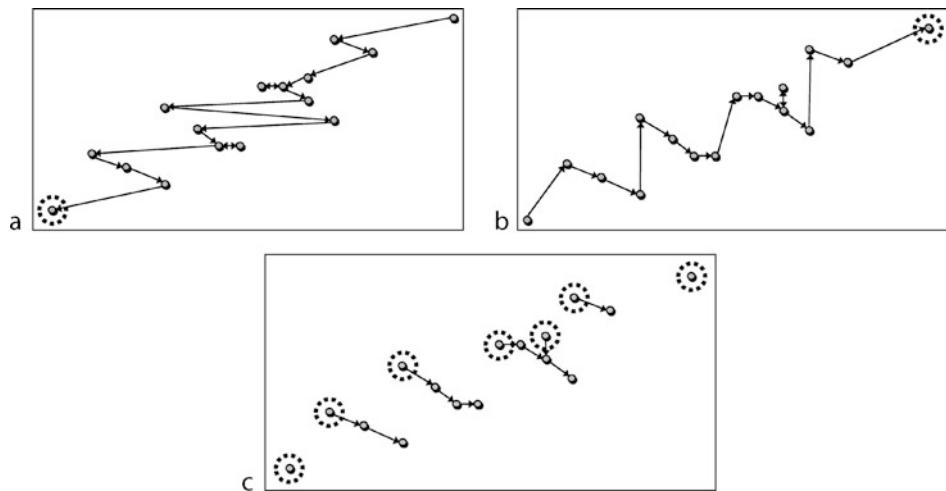
Note that there is an important difference between the relation $\leq \subseteq Z \times Z$ and the corresponding preference relation $\preccurlyeq \subseteq X \times X$. Since two different decision vectors may be mapped to the same objective vector, antisymmetry is not guaranteed, that is, from $\mathbf{a} \preccurlyeq \mathbf{b} \wedge \mathbf{b} \preccurlyeq \mathbf{a}$ it cannot be inferred that $\mathbf{a} = \mathbf{b}$. Hence, \preccurlyeq is usually only a preorder, while \leq is a partial order.

The above considerations hold for the single-objective as well as the multiobjective case. What is now the difference between these cases? To illustrate this, a graphical representation of a preference relation is helpful. The idea is to interpret the pair (X, \preccurlyeq) as a graph where X stands for the nodes and $\preccurlyeq \subseteq X \times X$ defines the edges: there is a directed edge from node $\mathbf{a} \in X$ to node $\mathbf{b} \in X$ iff $\mathbf{a} \preccurlyeq \mathbf{b}$. Such a graph will be denoted as a *relation graph*. For better visualization and to make the representation more compact, both reflexive edges and edges that can be inferred through the transitivity property will be omitted in the graphical rendering in the following.

Example 3 [Figure 3](#) depicts three relation graphs for the knapsack problem instance defined by [Fig. 1](#). For all graphs, weak Pareto dominance is the underlying dominance relation, but each time a different set of objectives is considered; the resulting preference relations are denoted as $\preccurlyeq_{par}^{f_w}$ (only weight is taken into account), $\preccurlyeq_{par}^{f_p}$ (only profit is taken into account), and $\preccurlyeq_{par}^{f_p, f_w}$ (biobjective case).

Fig. 3

Relation graphs for the knapsack problem considering (a) $(X, \preccurlyeq_{par}^{f_w})$, (b) $(X, \preccurlyeq_{par}^{f_p})$, and (c) $(X, \preccurlyeq_{par}^{f_p, f_w})$. The optimal solutions are marked by dotted circles.



As can be seen from Fig. 3, in the case of $n = 1$ (only a single objective is considered), the resulting graph represents a chain – a chain means that any two solutions $\mathbf{a}, \mathbf{b} \in X$ are *comparable*, that is, it holds either $\mathbf{a} \preccurlyeq \mathbf{b}$ or $\mathbf{b} \preccurlyeq \mathbf{a}$ (or both). In other words: the preference relation \preccurlyeq is a total preorder. (A relation $\leq \subseteq Y \times Y$ is total if for all $a, b \in Y$ holds $(a \leq b) \vee (b \leq a)$.) When $n > 1$, then there may be some solution pairs \mathbf{a}, \mathbf{b} for which neither $\mathbf{a} \preccurlyeq \mathbf{b}$ nor $\mathbf{b} \preccurlyeq \mathbf{a}$ holds, see Fig. 3c; in such a situation, \mathbf{a} and \mathbf{b} are denoted as *incomparable*. Among a set of incomparable solutions – a so-called *antichain* – no solution is preferable a priori unless further preference information is provided.

If several optimal solutions exist that are incomparable to each other, then multiple optima emerge in the objective space – a situation that is particular to multiobjective settings. In general, every solution for which no better solution exists is an optimal solution, cf. Sect. 2.1; in order theory, such a solution is denoted as a minimal element.

- The *minimal set* of a preordered set (Y, \leq) is defined as

$$\text{Min}(Y, \leq) := \{a \in Y \mid \forall b \in Y : b \leq a \Rightarrow a \leq b\} \quad (9)$$

The elements of $\text{Min}(Y, \leq)$ are denoted as *minimal elements*.

If we denote the situation $\mathbf{a} \preccurlyeq \mathbf{b} \wedge \mathbf{b} \not\preccurlyeq \mathbf{a}$ as “ \mathbf{a} dominates \mathbf{b} ,” then the minimal elements are those not dominated by others, that is, the *nondominated* ones. Whenever weak Pareto dominance is the underlying dominance relation, then the minimal elements of (X, \preccurlyeq_{par}) are called *Pareto optimal* and the entirety of the minimal elements is also known as *Pareto-optimal set*. The corresponding elements of the objective space are also called *Pareto-optimal*, while the set of Pareto-optimal objective vectors constitutes the *Pareto-optimal front*. The Pareto-optimal front may contain fewer elements than the Pareto-optimal set since decision vectors can be mapped to identical objective vectors.

In the example shown in Fig. 3, the Pareto-optimal sets for the single-objective scenarios (a) and (b) contain one solution, while in the biobjective setting (c) seven incomparable

solutions are included in the Pareto-optimal set. Note that, in principle, also in the case of a single objective, multiple Pareto optima may exist; however, they would not be incomparable and all Pareto-optimal solutions would be assigned the same objective function value. In the presence of multiple objectives, the Pareto-optimal set usually contains an antichain that reflects different trade-offs among the objectives.

2.3 Properties of the Pareto-Optimal Set

The Pareto-optimal set has different characteristics, each of which can have an impact on the optimization and the decision-making process. The most important aspects are the following:

Range: The ideal and the nadir objective vectors represent lower and upper bounds, respectively, on the objective function values of the Pareto-optimal set; knowledge of these points is not only helpful for decision making, but can also be important for interactive optimization procedures (Deb et al. 2006).

- ▶ Let X_p be the Pareto-optimal set of a multiobjective optimization problem. The *ideal point* $\underline{\mathbf{z}} = (\underline{z}_1, \dots, \underline{z}_n) \in \mathbb{R}^n$ of this problem is defined as

$$\underline{z}_i := \min_{\mathbf{a} \in X_p} f_i(\mathbf{a}) \quad (10)$$

and the *nadir point* $\bar{\mathbf{z}} = (\bar{z}_1, \dots, \bar{z}_n) \in \mathbb{R}^n$ is given by

$$\bar{z}_i := \max_{\mathbf{a} \in X_p} f_i(\mathbf{a}) \quad (11)$$

where all objectives are without loss of generality to be minimized.

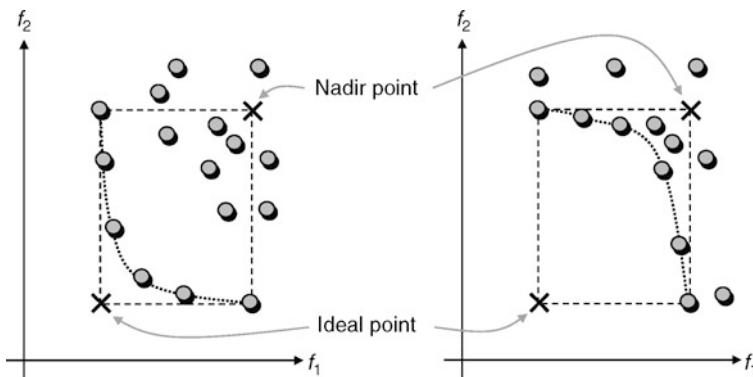
- [Figure 4](#) illustrates these concepts. If ideal point and nadir point are identical, then there is no conflict among the objectives and the Pareto-optimal front consists solely of the ideal point.

Shape: The image of the Pareto optima in the objective space can have different shapes that reflect how strongly objectives are in agreement or disagreement. For instance, the curvature of the front may be convex ([Fig. 4](#), left), concave ([Fig. 4](#), right), or contain convex and non-convex parts. If the decision space is continuous, then connectedness is another property that characterizes the front shape, see Ehrgott (2005). Roughly speaking, connectedness means that there are no gaps in the Pareto-optimal front; in the biobjective case, this implies that the Pareto-optimal objective vectors constitute a line. Pareto-optimal sets that are not contiguous in the objective space can cause difficulties with respect to both decision making and search.

Size: Whenever the underlying optimization problem is discrete, that is, the feasible set is finite as with the knapsack problem from [Example 1](#), the number of the Pareto-optimal solutions can become a crucial issue. If the size of the minimal set is rather small, one may aim at generating the entire Pareto-optimal set using appropriate optimization methods. The larger the set of Pareto optima becomes, the less feasible is this approach. If the number of Pareto-optimal solutions is even exponential in the input size, for example, exponential in the number of items with the knapsack problem, then the problem of identifying the Pareto-optimal set becomes intractable in general. This is the case with many multiobjective variants of combinatorial optimization problems like the multiobjective shortest path problem and the traveling salesperson problem (Ehrgott 2005).

Fig. 4

Two hypothetical minimization problems with two objectives each. The ideal points and the nadir points are identical, whereas the shapes of the Pareto-optimal fronts – visually emphasized by the auxiliary dotted lines – are different. The conflict between the objectives is weaker on the left side.



It is interesting to note that the size is related to the number of objectives involved. Winkler (1985) proved that the number of incomparable solution pairs increases, if further randomly generated objectives are added. Thereby, on the one hand the Pareto-optimal front may become larger and on the other hand the power of the dominance relation to guide the search may diminish – these are the main arguments that various researchers (Fonseca and Fleming 1995; Horn 1997; Deb 2001; Coello Coello et al. 2002; Fleming et al. 2005; Wagner et al. 2007), list in favor of the assumption that the search becomes harder the more objectives are involved. In the extreme case, when all solution pairs are incomparable, the Pareto-optimal set equals the decision space.

Computational complexity: The difficulty of combinatorial optimization problems can be expressed in terms of complexity classes. For instance, an NP-hard problem is assumed to be not solvable generally in polynomial time. In this context, it is remarkable that a polynomially solvable single-objective problem can become NP-hard when considering the multiobjective variant of it. One example is the shortest path problem: already in the bicriterion case it is, in general, not possible to identify *any* Pareto-optimal solution in polynomial time, while the extremes of the Pareto-optimal front can be computed efficiently, see Ehrhart (2005).

Several other properties could be listed here, for example, necessary and sufficient conditions for Pareto optimality. Optimality conditions are not discussed here, since the focus is on black-box optimization; the interested reader is referred to Miettinen (1999) instead.

3 Approaches to Multiobjective Optimization

In solving an optimization problem, one is interested in choosing the best solution for the application at hand. However, in the presence of multiple objectives the problem is usually underspecified. Whenever there is a conflict between the objectives and different trade-offs emerge, the choice of a Pareto-optimal decision vector is not arbitrary. Otherwise, it would be

sufficient to consider one of the objectives only and to generate an extreme point. It matters which Pareto-optimum is selected, and what is missing is the specification of this choice. But what is then the use of formulating a problem in terms of multiple optimization criteria?

The answer is that a multiobjective model can be regarded as an intermediate step. A *decision-making process* is necessary to add the information that is not contained in the model yet. To this end, a *decision maker*, a person or a group of persons responsible for the selection, expresses his or her preferences based on experience, expert knowledge, or further insights into the application. The preference information is then used to carry out a problem transformation that – usually – leads to a fully specified model where a total preorder on the decision space is induced. In principle, two types of problem transformations can be distinguished: (1) solution-oriented ones that leave the decision space unchanged, and (2) set-oriented ones where the approximation of the entire Pareto-optimal set is the focus.

3.1 Solution-Oriented Problem Transformations

Taking the sum of the objective function values is a commonly known method to reformulate a multiobjective problem in terms of a single objective problem. It is one example for handling multiple objectives by means of *scalarization* where a scalarizing function represents the new optimization criterion.

- ▶ A *scalarizing function* s is a function $s : Z \mapsto \mathbb{R}$ that maps each objective vector $(u_1, \dots, u_n) \in Z$ to a real value $s(u_1, \dots, u_n) \in \mathbb{R}$.

Typically, each scalarization relies on individual parameters by which the trade-off between the original objectives can be specified. The transformed problem is then given by $(X, \mathbb{R}, s \circ f, g, \leq)$, assuming that the scalarizing function is to be minimized. Three selected functions will be briefly presented here, more detailed discussions and other techniques can be found in Miettinen (1999) and Ehrgott (2005).

Weighted Sum Method: The scalarizing function is given by

$$s_W(u_1, \dots, u_n) := \sum_{1 \leq i \leq n} w_i u_i \quad (12)$$

where $w_1, \dots, w_n \in \mathbb{R}^+$ are the scalarization parameters that assign each objective a positive weight; typically, the weights sum up to 1. If all weights are greater than 0, then each optimal solution of the transformed problem is also a Pareto-optimal solution of the original multiobjective problem. The opposite does not hold in general, that is, in certain cases, there exists no weight combination such that a specific Pareto-optimal solution is also an optimal solution of the transformed problem (Ehrgott 2005).

Tchebycheff Method: This technique considers the distance to a reference point $(z_1, \dots, z_k) \in Z$ using an l_∞ -norm

$$s_T(u_1, \dots, u_n) := \max_{1 \leq i \leq n} w_i (u_i - z_i) \quad (13)$$

where $w_1, \dots, w_n \in \mathbb{R}^+$ are the weights as before with $\sum_{1 \leq i \leq n} w_i = 1$. Here, for every Pareto-optimal solution a corresponding weight combination exists, but not all minimal elements of the transformed problem are Pareto-optimal in the common sense. The reference point stands for an objective vector that the decision maker would like to achieve and that may represent an ideal choice.

ϵ -Constraint Method: Here, one of the objective functions is selected for optimization, without loss of generality the last, leading to the scalarizing function

$$s_C(u_1, \dots, u_n) := u_n \quad (14)$$

while the remaining objectives are converted into constraints, that is, $n - 1$ additional constraints $g_{m+i}(\mathbf{a}) = f_i(\mathbf{a}) - \varepsilon_i$ for $1 \leq i < n$ are created where $\varepsilon_i \in \mathbb{R}$ represents the threshold for the i th objective. (The constraints are added to the existing ones, that is, $\mathbf{g} = (g_1, \dots, g_m, g_{m+1}, \dots, g_{m+n-1})$.) The optimal solution to an ϵ -constraint problem is not necessarily Pareto-optimal with respect to the original problem, but any Pareto-optimal solution can be found by solving multiple ϵ -constraint transformations (Miettinen 1999; Laumanns et al. 2006).

Besides scalarization techniques, another possibility to totally order the decision space is to exchange the relation \leqslant of the multiobjective problem $(X, Z, \mathbf{f}, \mathbf{g}, \leqslant)$ under consideration. A lexicographic order \leqslant_{lex} , for instance, is a total order that encodes a ranking of the objectives

$$(u_1, \dots, u_n) \leqslant_{lex} (v_1, \dots, v_n) : \Leftrightarrow \exists 1 \leq i \leq n : (\forall 1 \leq j \leq i : u_j = v_j) \wedge (i = n \vee u_{i+1} < v_{i+1}) \quad (15)$$

with $(u_1, \dots, u_n), (v_1, \dots, v_n) \in \mathbb{R}^n$. The principle is that the objectives are considered one by one according to the ranking: the first objective where the objective values differ between two solutions decides which solution is better. Note that by renumbering the objectives, different rankings can be generated.

Independently of how exactly a solution-oriented problem transformation is carried out, the general idea is to refine the preference relation \preccurlyeq on the decision space such that all incomparable solutions become comparable, respectively, that \preccurlyeq becomes total. When considering the relation graph for (X, \preccurlyeq) , this mainly means that edges are inserted until a complete chain emerges; the edges represent additional knowledge or preferences that are specified by weights, constraint thresholds, objective rankings, etc., as sketched above. The resulting chain again should be a preference relation, \preccurlyeq_{ref} which should, to a certain extent, preserve the original preference relation. This can be formulated as follows:

- A relation $\preccurlyeq_{ref} \subseteq Y \times Y$ is a *refinement* of another relation $\preccurlyeq \subseteq Y \times Y$ iff

$$\forall a, b \in Y : a \preccurlyeq b \wedge b \not\preccurlyeq a \Rightarrow a \leqslant_{ref} b \wedge b \not\leqslant_{ref} a \quad (16)$$

and iff

$$\forall a, b \in Y : a \preccurlyeq b \wedge b \not\preccurlyeq a \Rightarrow a \leqslant_{ref} b \quad (17)$$

then \leqslant_{ref} is a *weak refinement* of \preccurlyeq .

For instance, the preference relations induced by the weighted sum approach, the Tchebycheff method, and lexicographic ordering are refinements of the Pareto dominance relation \preccurlyeq_{par} . That means if a $\mathbf{a} \in X$ dominates $\mathbf{b} \in X$, then \mathbf{a} is also better than \mathbf{b} with respect to the scalarizing function – in other words: dominance is preserved after the problem transformation. The ϵ -constraint method only represents a weak refinement of \preccurlyeq_{par} ; this guarantees that dominance is not violated, that is, whenever \mathbf{a} dominates \mathbf{b} , then \mathbf{a} is not worse than \mathbf{b} regarding the chosen objective. Other terms like (weak) *compatibility* (Hansen and Jaszkiewicz 1998), (weak) *Pareto compliance* (Zitzler et al. 2008a), or (strict) *monotonicity* (Miettinen 1999) are used as well in the literature to denote a (weak) refinement.

3.2 Set-Oriented Problem Transformations

Often, it is difficult to provide the preference information necessary for a solution-oriented problem transformation. Instead of specifying in advance which Pareto optimum is sought, an alternative is to generate the Pareto-optimal set and make a decision afterward on the basis of this knowledge. Since with most applications the identification of all Pareto-optimal solutions is infeasible, one is rather interested in approximating the Pareto-optimal set, that is, finding a suitable *set* of compromise solutions. In this case, one deals with a set problem where the decision space consists of all possible sets of solutions.

- ▶ For a multiobjective optimization problem $(X, Z, \mathbf{f}, \mathbf{g}, \leq)$, the associated *set problem* is given by $(\Psi, \Omega, F, \mathbf{G}, \leq)$ where
 - $\Psi = 2^X$ is the space of decision vector sets, that is, the powerset of X ,
 - $\Omega = 2^Z$ is the space of objective vector sets, that is, the powerset of Z ,
 - F is the extension of \mathbf{f} to sets, that is, $F(A) := \{\mathbf{f}(\mathbf{a}) : \mathbf{a} \in A\}$ for $A \in \Psi$,
 - $\mathbf{G} = (G_1, \dots, G_m)$ is the extension of \mathbf{g} to sets, that is, $G_i(A) := \max \{g_i(\mathbf{a}) : \mathbf{a} \in A\}$ for $1 \leq i \leq m$ and $A \in \Psi$, and
 - \leq extends \leq to sets where $A \leq B : \Leftrightarrow \forall \mathbf{b} \in B \exists \mathbf{a} \in A : \mathbf{a} \leq \mathbf{b}$.

The dominance relation \leq on sets represents a natural extension of a dominance relation \leq on solutions. It induces a corresponding *set preference relation* \preccurlyeq on the space Ψ of solution sets and therefore the terms comparability, incomparability, etc., also apply here. In the case of \preccurlyeq_{par} and the corresponding set preference relation \preccurlyeq_{par} , a solution set is also denoted as *Pareto set approximation*, and an optimal Pareto set approximation is one that contains for each Pareto-optimal objective vector a corresponding decision vector.

Example 4 Consider the knapsack problem and the four solution sets depicted in [Fig. 5](#): both A and B dominate C , D is incomparable to A , B , and C , and A and B are *indifferent*, that is, $A \preccurlyeq B \wedge B \preccurlyeq A$, with respect to Pareto dominance. Although A contains only Pareto-optimal solutions, it does not represent an optimal Pareto set approximation as the entire Pareto-optimal front is not covered.

It is important to note that the set problem resulting from the transformation is still underspecified – similarly to the original multiobjective problem – as usually incomparable Pareto set approximations exist. Again, preference information is required to impose a total preorder on Ψ ; however, the preferences are weaker and easier to specify than the ones needed for solution-oriented problem transformations. This is also the motivation for making the problem more complex by considering sets instead of single solutions (the size of Ψ is exponential in the size of X).

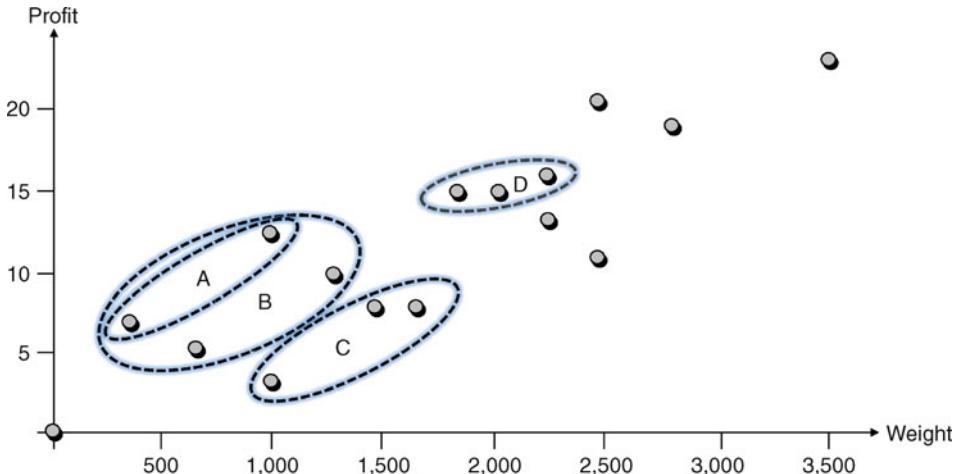
A common way to define a total preorder on Pareto set approximation relies on *quality indicators* – also denoted as performance measures or performance metrics – which represent functions that assign a tuple of solution sets a real value. Most popular are unary quality indicators:

- ▶ A (unary) *quality indicator* I is a function $I : \Psi \mapsto \mathbb{R}$ that assigns a Pareto set approximation a real value.

According to this definition, a unary quality indicator I can be regarded as a set quality measure or an objective function on solution sets. Therefore, it can be used to transform the

Fig. 5

Four Pareto set approximations for the knapsack problem instance from [Fig. 1](#).



set problem $(\Psi, \Omega, F, G, \leq)$ into a single-objective problem $(\Psi, \mathbb{R}, I, G, \leq)$ assuming that the indicator values are to be minimized.

Various quality indicators have been proposed in the literature, see Hansen and Jaszkiewicz (1998), Knowles and Corne (2002), and Zitzler et al. (2003); three selected unary indicators are briefly discussed here:

Hypervolume indicator: The *hypervolume indicator* I_H (Zitzler and Thiele 1998) gives the volume of the portion of the objective space that is weakly dominated by a specific Pareto set approximation. It can be formally defined as

$$I_H(A) := \lambda(H(A, R)) \quad (18)$$

where $R \subseteq Z$ is a given reference set of objective vectors,

$$H(A, R) := \{z \in Z : \exists \mathbf{a} \in A \exists \mathbf{r} \in R : \mathbf{f}(\mathbf{a}) \leq \mathbf{z} \leq \mathbf{r}\} \quad (19)$$

denotes the set of objective vectors that are enclosed by the front $F(A)$ given by A and the reference set R , and λ is the Lebesgue measure with $\lambda(H(A, R)) = \int_{\mathbb{R}^n} \mathbf{1}_{H(A,R)}(z) dz$ and $\mathbf{1}_{H(A,R)}$ being the characteristic function of $H(A, R)$. The hypervolume indicator is to be maximized. In [Fig. 6](#) on the left, the gray shaded area represents the set $H(A, R)$.

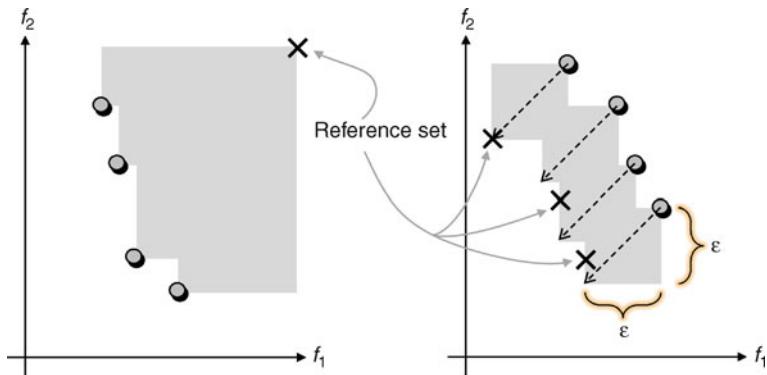
Epsilon indicator: The *epsilon indicator* I_ε (Zitzler et al. 2003) measures the objective-wise distance to a given reference set in the objective space; it is based on the concept of epsilon dominance

$$I_\varepsilon(A) := \inf_{\varepsilon} F(A) \leq_\varepsilon R \quad (20)$$

where \leq_ε is the natural extension of epsilon dominance from [Eq. 7](#) to solution sets with $A \leq_\varepsilon B : \Leftrightarrow \forall \mathbf{b} \in B \exists \mathbf{a} \in A : \mathbf{a} \leq_\varepsilon \mathbf{b}$. In other words, it provides the minimum ε such that the reference set is weakly Pareto dominated as illustrated in [Fig. 6](#) on the right; the smaller the value, the better the solution set.

Fig. 6

Illustration of the hypervolume indicator (left) and the (additive) epsilon indicator (right) for a minimization problems with two objectives each.



R indicators: The *R indicators* proposed in Hansen and Jaszkiewicz (1998) can be used to assess Pareto set approximations on the basis of scalarizing functions. The following quality indicator I_{RT} relies on the Tchebycheff method and requires multiple scalarizing functions $s_T^{(1)}, \dots, s_T^{(l)}$, each using a different combination of weights and the same reference point (z_1, \dots, z_n)

$$I_{RT}(A) := \frac{1}{l} \sum_{1 \leq i \leq l} \min_{\mathbf{a} \in A} s_T^{(i)}(\mathbf{a}) \quad (21)$$

It assumes the scalarizing functions to be minimized and simply takes the average over the best values per scalarizing function, which is to be minimized as well.

Often, it is beneficial to use a combination of quality indicators instead of a single indicator. One possibility is to define a sequence of indicators (I_1, \dots, I_l) and to apply a lexicographic order \leq_{lex} on \mathbb{R}^l , see Zitzler et al. (2008b). Thereby, the set problem $(\Psi, \Omega, F, G, \leq)$ is further transformed into the optimization problem $(\Psi, \mathbb{R}^l, (I_1, \dots, I_l), G, \leq_{lex})$. Such a hierarchy of indicators is especially useful for search as will be discussed in the next section.

Finally, note that the same considerations as with scalarizing functions apply to quality indicators: the induced set preference relation should be in accordance with the original set preference relation \leq . The relation \leq_H associated with the hypervolume indicator refines the set Pareto dominance relation \leq_{par} , while the relations \leq_ε and \leq_{RT} associated with I_ε and I_{RT} , respectively, only weakly refine \leq_{par} .

4 Search Algorithm Design

When designing a search algorithm for a multiobjective optimization problem, one either has to deal with a solution-oriented scenario or a set-oriented scenario – depending on the type of problem transformation chosen. In the first case, usually a single-objective problem emerges and the standard components of a black-box search method can be used. The latter case

requires specialized methods and by far most of the studies in the area of evolutionary multiobjective optimization focus on approximating the Pareto-optimal set. Therefore, in the following, a set-oriented problem transformation will be assumed and the corresponding algorithmic concepts that have been developed for this purpose will be discussed.

Note that the presentation focuses on evolutionary algorithms, although also other natural computing methods such as ant colony optimization and particle swarm optimization have been successfully applied to multiobjective scenarios. The general concepts are more or less the same, but the latter techniques may employ specialized procedures tailored to the algorithmic framework used.

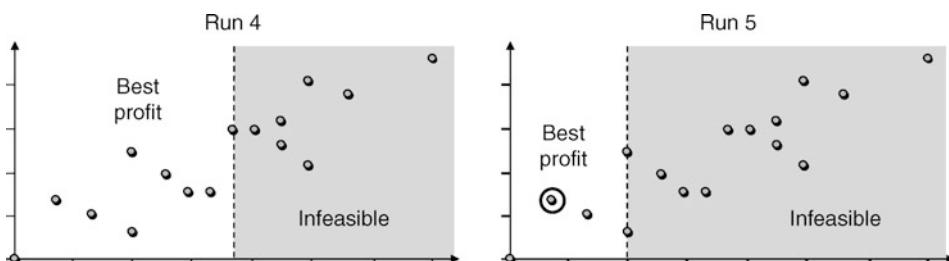
4.1 Types of Algorithms

One can distinguish between two types of search strategies to identify a suitable set of compromise solutions: the algorithms of the first category perform multiple single-objective runs, while the methods of the second category require only a single run. As to the multiple runs approach, consider, for example, the I_{RT} quality indicator from [Eq. 21](#) in combination with l Tchebycheff scalarizing functions $s_T^{(1)}, \dots, s_T^{(l)}$. For each scalarizing function $s_T^{(i)}$, a separate optimization run is carried out where the objective function f is $f = s_T^{(i)}$, that is, overall l runs are performed. Afterward, the best solutions found per run are combined to form a Pareto set approximation. A similar strategy can be devised on the basis of the ε -constraint method. First, the thresholds ε_i for the constraints that represent converted objective functions are set to ∞ , which means only the chosen objective is optimized. The best solution found is used to adapt the constraints, so that it falls into the infeasible region, and the next optimization run is carried out. This process is iterated until the feasible set equals the empty set. As before, the union of the best solutions per run form the approximation of the Pareto-optimal set. While for $n = 2$ this scheme is simple to implement as depicted in [Fig. 7](#), it becomes much more complex, if the number of optimization criteria is arbitrary, see Laumanns et al. ([2006](#)).

Most multiobjective evolutionary algorithms proposed in the literature are of the second type. Here, the population as a whole represents a Pareto set approximation and therefore a single run is sufficient. Taking this line of thought further, the successive application of mating selection, variation, and environmental selection aims at generating a new, better Pareto set

Fig. 7

Visualization of the multiple run strategy to approximate the Pareto-optimal set using the ε -constraint method. The figure shows for the knapsack problem and for two successive runs the current weight constraint and the best solution found.



approximation and therefore can be regarded as a multi-stage mutation operator on solution sets. From this perspective, a common multiobjective evolutionary algorithm implements a single-objective (1, 1)-strategy on solution sets with a clever mutation operator; this applies to almost all search strategies currently available. The idea of operating on a population of solution sets, that is, employing an evolutionary algorithm where each individual is a Pareto set approximation on its own and where possibly solution sets are recombined, has not gained much attention yet (end of 2008): there only exists a preliminary study (Bader et al. 2009) that fully addresses this issue, while some papers in the context of parallelism partially exploit this idea (Poloni 1995; Baita et al. 1995; Lee and Hajela 1996; Aherne et al. 1997; Sawai and Adachi 2000; Branke et al. 2004b; Mezmaz et al. 2006; Coello Coello et al. 2007). Thus, the potential of this multi-solution set strategy still needs to be explored.

4.2 Basic Algorithmic Concepts

The main difference between a single-objective evolutionary optimizer and a multiobjective one approximating the Pareto-optimal set lies in fitness assignment and selection, while variation has been treated similarly (and therefore is not discussed in the following). (Note that most recently (March 2009), specific set variation operators have been proposed (Bader et al. 2009; Voß et al. 2009).) The key question is how a better Pareto set approximation can be generated from the current population in a randomized fashion – independently of whether the search algorithm works with a single or multiple solution sets.

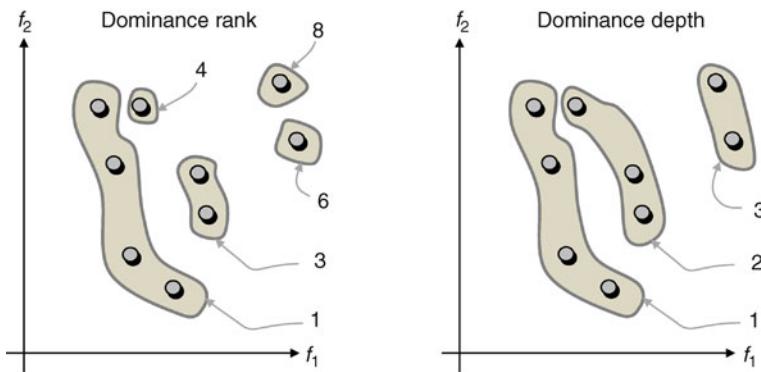
4.2.1 Fitness Assignment

One issue is to quantify the usefulness of a particular individual with regard to the entire population. The general idea behind most prevalent concepts is to assign fitness values respectively to rank the individuals according to the loss in quality of the Pareto set approximation that can be accounted to the removal of a particular individual. More specifically, the fitness of an individual $\mathbf{a} \in X$ is determined by taking the difference between the quality of the entire population P and the quality of the population without \mathbf{a} , for example, $fit_{\mathbf{a}} = I(P) - I(P \setminus \{\mathbf{a}\})$ where P is a multiset of decision vectors and I a unary quality indicator used for optimization. (Note that for reasons of simplicity, here individuals and decision vectors are not distinguished, although in practice usually a representation is required to encode decision vectors appropriately.)

A difficulty that arises is the fact that the set preference relation or the quality indicator used are usually insensitive to dominated population members. That means the quality of the population solely depends on the nondominated individuals. The consequence is that all dominated individuals are assigned a fitness of 0 when employing the above principle. For instance, the hypervolume indicator, which has become a popular measure for fitness assignment, does not change its value when dominated solutions are added to or removed from the solution set. The standard solution to this problem is to partition the population into nonoverlapping dominance classes that are organized in a hierarchy. Each class contains only mutually nondominating solutions, and the fitness values are then computed per dominance class. Furthermore, the individuals in the first class are considered superior to the second class, which in turn is better than the third class and so forth.

Fig. 8

Two different partitionings into dominance classes for a biobjective minimization problem. The numbers refer to the hierarchy of the dominance classes.



There exist different ways to create such a partitioning, Fig. 8 illustrates two popular schemes. The method of *dominance rank* (Fonseca and Fleming 1993) counts for each individual the number of population members that dominate it; this number gives the so-called dominance rank. All individuals with dominance rank 0 form the first dominance class, the second class consists of individuals with dominance rank 1, and so forth. Another method originally proposed in Goldberg (1989) and for the first time integrated in a multiobjective evolutionary algorithm by Deb and colleagues (Srinivas and Deb 1994; Deb et al. 2002a) uses the *dominance depth* – a strategy also known as nondominated sorting. Here, the minimal elements of the population form the first dominance class, that is, $\text{Min}(P, \leq_{\text{par}})$; the next class is represented by the minimal elements of the remaining population members, that is, $\text{Min}(P \setminus \text{Min}(P, \leq_{\text{par}}), \leq_{\text{par}})$, and the other classes are defined accordingly. Partitioning in terms of dominance depth has in comparison to dominance rank the additional property that classes with lower numbers dominate classes with higher numbers in the hierarchy.

As mentioned above, the fitness values, respectively, the ranking within each dominance class are determined using the underlying set measure or set preference relation. Density-based measures that resemble niching techniques like fitness sharing in single-objective evolutionary computation have been the most popular, particularly in the first decade of EMO. This approach usually relies on Euclidean distance in the objective space: it first computes the distances between all population members and on this basis then estimates the density around each individual. One example is the k th nearest neighbor method where k is a prespecified constant which is often set to 2. Here, the distance to the k th closest individual is taken as the density estimate for each population member and the individuals in each dominance class are then ranked in descending order of the density estimates (Zitzler et al. 2002). Many other techniques exist (Fonseca and Fleming 1993; Knowles and Corne 2000b; Deb et al. 2002a). All these techniques can be regarded as implementations of a specific quality indicator, I_D , which is based on Euclidean distance in the objective space. However, the problem with this type of indicator is that, in general, it does not represent a proper preorder as transitivity is violated (Knowles and Corne 2002; Zitzler et al. 2003, 2008b). This theoretical consideration can also be observed in practice: Euclidean distance-based multiobjective evolutionary algorithms tend to exhibit cyclic behavior, which means they do not progress after a certain point in time

(Laumanns et al. 2002). This can cause problems especially with applications involving many objective functions (Wagner et al. 2007).

The alternative to density-based fitness assignment is to directly use set preference relations or quality indicators that are preorders and (weak) refinements. In particular, the hypervolume indicator has been employed in this context since it provides a refinement of weak Pareto dominance and hence allows a fine-grained ranking of the individuals. Several hypervolume-based algorithms have been proposed (Emmerich et al. 2005; Igel et al. 2007; Zitzler et al. 2007) that all rely on the same principle: the population is first partitioned into dominance classes based on dominance depth and then for each dominance class the individuals are ranked according to their contribution to the overall hypervolume of the class. Precisely, the hypervolume contribution h_a of an individual a with respect to its dominance class A is given by

$$h_a = I_H(A) - I_H(A \setminus \{a\}) \quad (22)$$

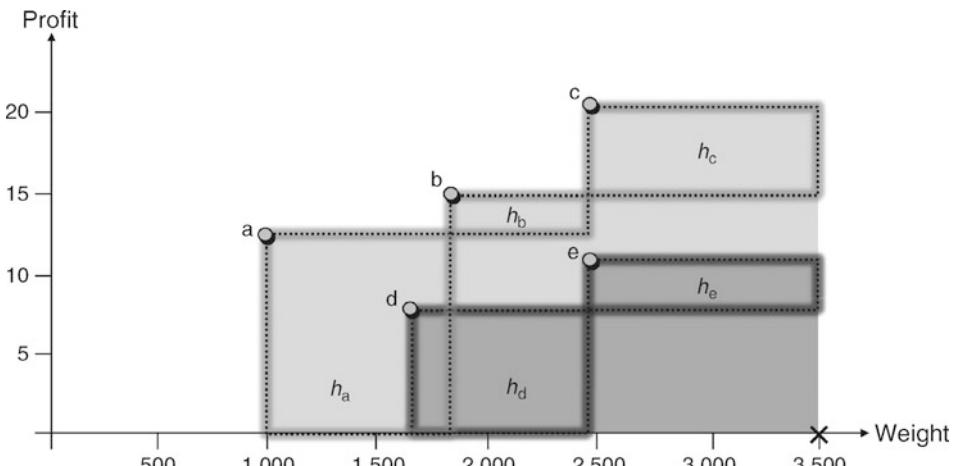
which equals the loss in hypervolume resulting from the removal of a .

Example 5 Consider the population in Fig. 9 consisting of five individuals. When using the hypervolume-based fitness assignment described above, the ranking of the individuals is a, c, b, d, e . The individuals a, b, c fall in the first dominance class A_1 and it holds $h_a > h_c > h_b$. The second dominance class A_2 contains the remaining population members where d contributes more to the overall hypervolume of A_2 than e , that is, $h_d > h_e$.

Unfortunately, the exact computation of the hypervolume values is computationally expensive (Bringmann and Friedrich 2008) and the available algorithms have a worst-case runtime complexity that is exponential in the number of objectives (Fonseca et al. 2006; Beume and Rudolph 2006). Therefore, different methods to speed up this computation have

Fig. 9

Hypervolume-based fitness assignment for a hypothetical population for the biobjective knapsack problem. The boxed areas represent the hypervolume contributions attributed to the specific individuals.



been proposed (Bradstreet et al. 2008; Bader et al. 2010) that enable hypervolume-based search also in high-dimensional objective spaces.

The majority of the proposed fitness assignment methods follows the principle sketched above, a combination of dominance-based partitioning and a quality indicator. It is interesting to note in this context that this scheme can be expressed in terms of a sequence of indicators as outlined in [Sect. 3.2](#). Regarding the example with the hypervolume indicator, let $I_H^{(i)}$ denote the indicator value of the i th dominance class A_i of a population P , that is, $I_H^{(i)}(P) := I_H(A_i)$. Then the fitness assignment procedure corresponds to a set preference relation induced by the sequence $(I_H^{(1)}, I_H^{(2)}, \dots, I_H^{(N)})$ of quality indicators where N is the population size. This idea can be extended to arbitrary set preference relations as proposed in Zitzler et al. (2008b).

Finally, also other fitness assignment variants exist that do not rely on a dominance-based partitioning (Schaffer 1985; Horn et al. 1994; Czyzak and Jaskiewicz 1998; Zitzler and Künzli 2004; Zitzler et al. 2008b). When, for instance, the quality indicator used is composed of scalarizing functions, the fitness values may be directly inferred from the scalarizing functions without further dominance checks.

4.2.2 Selection

The fitness values and the ranking of the individuals, respectively, provide the basis for mating selection and environmental selection.

Mating selection is often implemented in the same way as in single-objective evolutionary algorithms, although some concepts have been proposed that are different. For instance, Schaffer's vector evaluated genetic algorithm (Schaffer 1985) switches between the objectives during the mating selection phase, the niched pareto genetic algorithm by Horn et al. (1994) carries out dominance-based tournaments, and other approaches based on the weighted-sum method randomly choose weight combinations to determine the members of the mating pool (Ishibuchi and Murata 1996). These techniques can be regarded as randomized fitness assignment schemes; however, they are rather seldom used.

Environmental selection plays a much more important role in multiobjective search than mating selection. The first evolutionary methods for multiobjective optimization (Schaffer 1985; Fonseca and Fleming 1993; Srinivas and Deb 1994; Horn et al. 1994) used the standard genetic algorithm scheme where the offspring population entirely replaces the parent population (a comma strategy). However, when empirical studies demonstrated that elitism is of high importance when approximating the Pareto-optimal set (Zitzler and Thiele 1999; Knowles and Corne 2000b), the picture changed. Nowadays, most multiobjective search algorithms implement a plus strategy, that is, the best solutions from the union P^+ of parent population and offspring population are chosen for survival. This leads to a subset selection problem: from the N parents and M children a subset of N solutions needs to be determined that constitutes the next population. In the case of the hypervolume indicator for instance, one searches for a subset A with $|A| = N$ that has the largest hypervolume value among all subsets of size N , that is,

$$\forall B \subseteq P^+ : |B| = N \Rightarrow I_H(B) \leq I_H(A) \quad (23)$$

Determining the best subset can be computationally expensive, and often heuristic procedures are used instead. One strategy iteratively removes the worst solution from P^+ and updates the fitness values until the desired population size is reached (see Zitzler et al. 2002). Alternatively, one may remove the M worst solutions in a single step without intermediate fitness updates as,

for example, in Deb et al. (2002a); this strategy is faster, but also less accurate than the iterative approach.

The choice of the environmental selection scheme is also important for the convergence properties of a multiobjective optimizer as has been shown in Rudolph (1998, 2001), Rudolph and Agapie (2000), Laumanns et al. (2002), and Zitzler et al. (2008b). Therefore, sometimes a separate archive is maintained that stores the nondominated solutions (Knowles and Corne 2003a; Fieldsend et al. 2003). The archive may use a different selection scheme than the population; for instance, one can employ a strategy that ensures certain quality bounds (Laumanns et al. 2002).

4.3 Advanced Algorithmic Concepts

In the following, advanced concepts to integrate constraints, user preferences, and other types of optimizers are dealt with. The presentation is not exhaustive; for instance, other topics like uncertainty and robustness, which are of high practical relevance are not treated here. On the one hand, these issues can be addressed in the same manner as in single-objective optimization where various techniques have been developed (Jin and Branke 2005). On the other hand, the methodologies that have been specifically proposed in a multiobjective context (Hughes 2001; Teich 2001; Deb and Gupta 2006; Basseur and Zitzler 2006; Gaspar-Cunha and Covas 2008) form a relatively new area of research that is still maturing and does not provide a clear picture yet of potential standard concepts. Furthermore, specialized data structures and algorithms as well as parallelization schemes to speed up fitness assignment and selection are important when implementing a multiobjective optimizer (see Habenicht 1983; Mostaghim et al. 2002; Jensen 2003; Talbi et al. 2008). Such implementation issues are not covered in the following.

4.3.1 Constraint Handling

In principle, constraints can be handled in the same manner as in single-objective optimization, for example, by using repair methods, by not accepting infeasible solutions, or by employing penalty functions. However, multiobjective optimization offers new opportunities where the constraint functions are treated as objective functions. The idea is to modify the underlying preference relation such that the constraints are included and

1. Among feasible solutions, the original dominance relation holds,
2. Feasible solutions dominate infeasible solutions, and
3. Among infeasible solutions, the one with the lower constraint violation is preferred.

There are different ways how this principle can be implemented in practice, see Coello et al. (2007) for an extensive list of references. The following scheme reflects the notion proposed in Fonseca and Fleming (1998):

- ▶ Let $(X, Z, \mathbf{f}, \mathbf{g}, \leq)$ be a multiobjective optimization problem. The *constraint-integrating preference relation* \preceq_{con} is defined as

$$\begin{aligned} \mathbf{a} \preceq_{con} \mathbf{b} : \Leftrightarrow & (\mathbf{g}(\mathbf{a}) \leq \mathbf{0} \wedge \mathbf{g}(\mathbf{b}) \leq \mathbf{0} \wedge \mathbf{f}(\mathbf{a}) \leq \mathbf{f}(\mathbf{b})) \vee \\ & (\mathbf{g}(\mathbf{a}) \leq \mathbf{0} \wedge \mathbf{g}(\mathbf{b}) \not\leq \mathbf{0}) \vee \\ & (\mathbf{g}(\mathbf{a}) \not\leq \mathbf{0} \wedge \mathbf{g}(\mathbf{b}) \not\leq \mathbf{0} \wedge \mathbf{g}(\mathbf{a}) \leq_{par} \mathbf{g}(\mathbf{b})) \end{aligned} \quad (24)$$

for all $\mathbf{a}, \mathbf{b} \in X$.

This scheme compares infeasible solutions using weak Pareto dominance; alternatively, one may consider the overall constraint violation, for example, the sum of the individual violations, and only compare this scalar value.

The constraint-integrating preference relation can be used in the same way as described in [Sect. 4.2.1](#) to partition the population in dominance classes. The construction ensures that (1) a class contains either only feasible or only infeasible solutions, and (2) infeasible classes follow the feasible classes in the hierarchy of dominance classes.

Example 6 Consider the single-objective variant of the knapsack problem defined in [Eq. 3](#). To handle the weight constraint, a preference relation \preccurlyeq_{con} can be formulated as follows:

$$S \preccurlyeq_{con} T : \Leftrightarrow (f_p(S) \leq f_p(T) \wedge f_w(S) \leq w_{\max}) \vee f_w(S) \leq f_w(T) \quad (25)$$

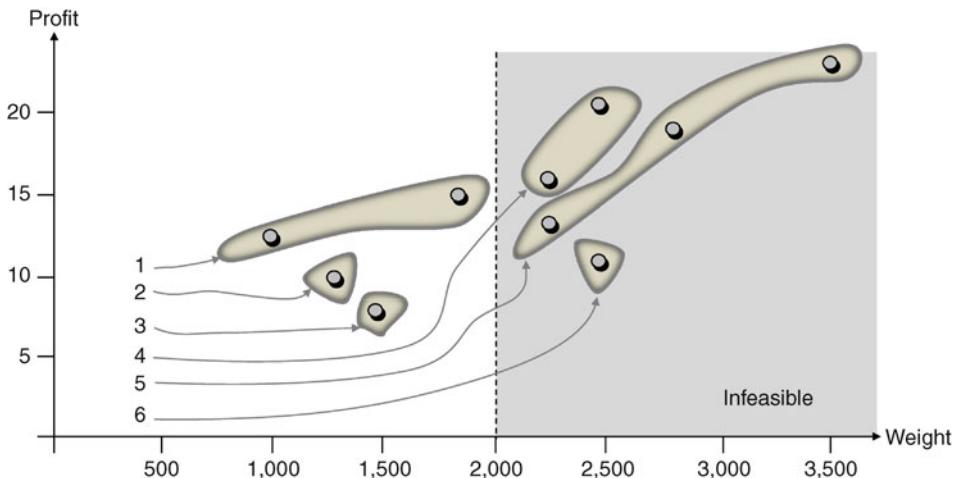
where S, T are arbitrary set of items. To illustrate how this preference relation can be used for partitioning the population, recall the knapsack problem instance from [Fig. 1](#). For a hypothetical population of ten individuals, the resulting dominance classes are depicted in [Fig. 10](#) assuming that dominance depth is used for partitioning.

4.3.2 Preference Articulation

Often, the decision maker is not interested in approximating the entire Pareto-optimal set, but has further preferences that specify certain regions of interest. Such preferences can be in the form of constraints, priorities, reference points, etc. To integrate such information in the search process is especially helpful when using the evolutionary algorithm in an interactive fashion (Jaskiewicz and Branke 2008): the optimizer first presents a general approximation set, for example, according to the hypervolume indicator, and afterward, the decision maker

Fig. 10

Partitioning of a population into dominance classes using dominance depth and a constraint-integrating preference relation. The problem is taken from [Fig. 1](#) with a weight threshold of $w_{\max} = 2,000$.



restricts respectively refines the search goal by adding further preferences; these two steps are iterated until the decision maker arrives at a final solution.

There basically exist two methodologies to guide the search on the basis of additional preference information: one can modify either the preference relation on solutions or the set preference relation. The first possibility corresponds to the aforementioned strategy to include constraints. For instance, the approach proposed in Fonseca and Fleming (1998) allows us to combine goals, which correspond to desired objective functions values, and priorities, which allow us to define a hierarchy of objectives and constraints. A further option is to change the dominance relation (Branke et al. 2001; Molina et al. 2009), for example, to reflect trade-off information or to incorporate reference points. Several other solution-oriented preference handling techniques exist (Cvetković and Parmee 2002; Rachmawati and Srinivasan 2006).

The second possibility consists in adapting the set preference relation resp. the underlying quality indicator according to the user preferences. Often, a reference point is used that specifies a region in the objective space that is of particular interest. In density-based fitness assignment procedures, this information can be incorporated by introducing a bias in the distance metric that is directly affected by the reference point (Branke and Deb 2004; Deb and Sundar 2006). Alternatively, one may consider an extension of the hypervolume indicator where a weight function puts emphasis on certain regions of the objective space (Zitzler et al. 2007); thereby, it is possible to concentrate the Pareto set approximation around one or several reference points. A similar approach can be implemented based on the epsilon indicator or the I_{RT} indicator (Zitzler et al. 2008b). Besides reference points, it is also possible to include directions: with the I_{RT} indicator, for instance, the chosen weight combinations determine which parts of the front are most important.

4.3.3 Hybridization

A promising approach to exploit knowledge about the problem structure in single-objective optimization is to combine an evolutionary algorithm with other techniques such as local search strategies or exact optimization methods. The same principle can be highly beneficial in multiobjective optimization (Knowles and Corne 2000a; Jaszkiewicz 2002), but some difficulties arise here: when for instance using a local search algorithm, this strategy focuses on the improvement of a single solution, while all population members together constitute a Pareto set approximation. Therefore, a search direction needs to be specified that corresponds to the population as a whole. In Jaszkiewicz (2002), for instance, hybrid multiobjective evolutionary algorithms based on scalarizing functions with weights have been investigated. Each time the local search procedure is invoked, a weight combination is chosen at random and the corresponding scalarizing function is taken as objective function for the local search strategy. Another approach does not use scalarizing functions, but relies on dominance to decide whether a newly generated solution replaces the current solution of the local search algorithm (Knowles and Corne 2000a).

4.4 Performance Assessment

When designing a multiobjective evolutionary algorithm or specific algorithmic concepts, it is crucial to compare the effectiveness of the new method to existing methods. To this end, both benchmark problems and comparison methodologies are needed.

4.4.1 Test Problems

Various suites of test functions have been proposed in the literature and many of them are widely used in the community, for example, the ZDT test functions (Zitzler et al. 2000), the DTLZ test functions (Deb et al. 2002b, 2005), the framework presented in Okabe et al. (2004), and the WFG suite (Huband et al. 2006). To ensure a consistent nomenclature, the proposed functions are often named according to the authors of the corresponding papers: the first letters of the authors' last names form the acronym, which may be extended by a number distinguishing specific functions of the same suite.

Artificial test functions can be designed and used to evaluate algorithms with respect to selected problem characteristics such as the shape of the front or the neighborhood structure in the decision space. Combinatorial optimization problems, in particular multiobjective versions of well-known NP hard problems, usually do not offer this flexibility, but they are often closer to reality since they can be found in various application areas. Although the problem formulation may be simple, a combinatorial problem as such can be hard to solve. Prominent examples are the multiobjective knapsack problem (Zitzler and Thiele 1999) and the multiobjective quadratic assignment problem (Knowles and Corne 2003b), which have been widely used in the EMO community.

Nevertheless, the ultimate benchmarks are real-world applications themselves (Künzli et al. 2004; Hughes 2007; Siegfried et al. 2009). The difficulty with these is that implementations need to be available in order to make them accessible for the research community. For instance, the PISA framework offers a methodology to facilitate the exchange of program code for performance assessment (Bleuler et al. 2003; Bader et al. 2008). PISA stands for "A Platform and Programming Language Independent Interface for Search Algorithms" and basically defines a text-based interface to separate general optimization principles (fitness assignment and selection) from application-specific aspects (representation, variation, and objective function evaluation). The PISA Web site (Bader et al. 2008) contains implementations of various multiobjective optimizers and benchmark problems that can be arbitrarily combined, independently of the computing platform.

4.4.2 Comparison Methodologies

Comparing stochastic algorithms for approximating the Pareto-optimal set involves comparing samples of solution sets. One can distinguish between three possibilities, how this can be accomplished: on the basis of (1) quality indicators, (2) set preference relations, and (3) attainment functions, which will be introduced later.

If the optimization goal is given in terms of a quality indicator, then the outcomes that have been obtained by running the different optimizers multiple times can be converted into samples of indicator values. Afterward, standard statistical testing procedures such as the Mann–Whitney rank sum test or Fisher's permutation test can be applied to make statistical statements about whether some algorithms generate higher quality solution sets than others, see Conover (1999). Note that multiple testing issues need to be taken into account when comparing more than two search methods (Zitzler et al. 2008a).

In the case that not a single indicator, but a more complex set preference relation is considered – possibly composed of multiple indicators – the solution sets under consideration can be compared pairwisely using this relation. On the basis of these comparisons, a standard

Mann–Whitney rank sum test can be applied, provided that the set preference relation is a total preorder. If totality is not given, for example, when directly using the weak Pareto dominance set relation, then a slightly modified testing procedure is necessary as described in Knowles et al. (2006).

The use of quality indicators or set preference relations in general implies that the outcome of the comparison is always specific to the encoded preference information. This means that Algorithm 1 can be better than Algorithm 2 when quality indicator I_1 is used, while the opposite can hold when another indicator I_2 is considered. This preference dependency can be circumvented by performing a comparison based on the weak Pareto dominance relation only. One possibility is the aforementioned methodology for set preference relations. Another possibility is the attainment function approach, which allows a more detailed analysis, but at the same time is only applicable in scenarios with few objectives.

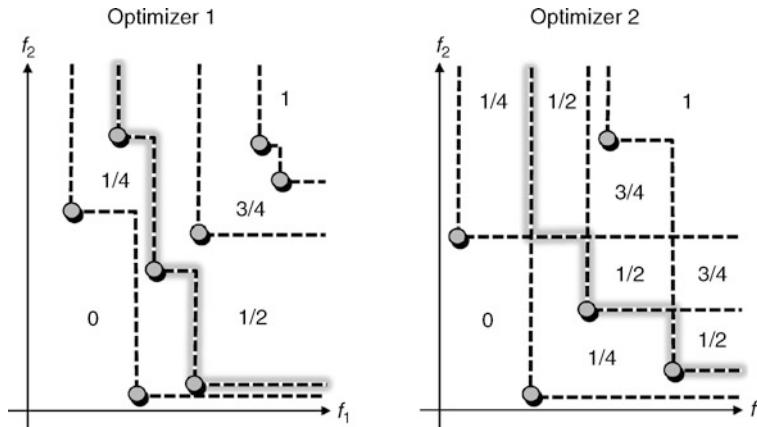
The *attainment function method* proposed in Fonseca and Fleming (1996) and Grunert da Fonseca et al. (2001) summarizes a sample of Pareto set approximations in terms of a so-called empirical attainment function. To explain the underlying idea, suppose that a certain stochastic multiobjective optimizer is run once on a specific problem. For each objective vector $\mathbf{u} \in Z$, there is a certain probability p that the resulting Pareto set approximation contains a solution \mathbf{a} such that $f(\mathbf{a})$ weakly dominates \mathbf{u} . We say p is the probability that \mathbf{u} is *attained* by the optimizer. The *attainment function* gives for each objective vector $\mathbf{u} \in Z$ the probability that \mathbf{u} is attained in one optimization run of the considered algorithm. The true attainment function is usually unknown, but it can be estimated on the basis of a sample of solution sets: one simply counts the number of Pareto set approximations by which each objective vector is attained and normalizes the resulting number with the overall sample size. The attainment function is a first-order moment measure, meaning that it estimates the probability that \mathbf{u} is attained in one optimization run of the considered algorithm *independently* of attaining any other \mathbf{u} ; however, also higher-order attainment functions may be considered (Grunert da Fonseca et al. 2001).

Example 7 Consider Fig. 11. For the scenario on the left, the four Pareto front approximations cut the objective space into five regions: the upper right region is attained in all of the runs and therefore is assigned a relative frequency of 1, the lower left region is attained in none of the runs, and the remaining three regions are assigned relative frequencies of $1/4$, $2/4$, and $3/4$ because they are attained, respectively, in one, two, and three of the four runs. In the scenario on the right, the objective space is partitioned into nine regions; the relative frequencies are determined analogously as shown in the figure.

The estimated attainment functions can be used to compare two optimizers by performing a corresponding statistical test as proposed in (Grunert da Fonseca et al. 2001). The test reveals in which regions in the objective space there are significant differences between the attainment functions. In addition, the estimated attainment functions can also be used for visualizing the outcomes of multiple runs of an optimizer. For instance, one may be interested in plotting all the goals that have been attained (independently) in 50% of the runs. The 50%-attainment surface represents the border of this subspace; roughly speaking, the 50%-attainment surface divides the objective space in two parts: the goals that have been attained and the goals that have not been attained with a frequency of at least 50%, see Fig. 11.

Fig. 11

Hypothetical outcomes of four runs for two different stochastic optimizers (left and right). The numbers in the figures give the relative frequencies according to which the distinct regions in the objective space are attained. The thick gray lines represent the 50%-attainment surfaces.



5 Applications

As outlined in the previous sections, the main use of evolutionary algorithms in multiobjective optimization scenarios lies in identifying a suitable set of compromise solutions, that is, in approximating the Pareto-optimal set. In the following, it will be briefly discussed why this type of optimization is beneficial.

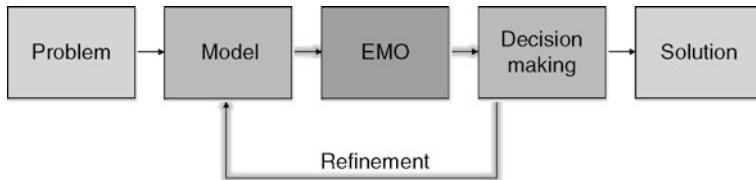
5.1 Learning and Decision Making

The main reason for formulating an optimization problem in terms of a multiple criteria model is the lack of knowledge: too little is known about the problem structure and the influence of the different optimization criteria; the same holds for user preferences, and it may even be that sufficient preferences are not available or cannot be quantified in a mathematical manner. Therefore, multiobjective optimization focuses on “modeling in progress” where the model is iteratively refined to best suit (the relevant aspects of) reality. Since evolutionary algorithms as black-box optimizers also provide high flexibility with regard to the model, evolutionary multiobjective optimization offers an excellent framework for successively learning about a problem. The workflow of this model-centered problem-solving strategy is depicted in **Fig. 12**.

Learning can have different meanings. First of all, an approximation of the Pareto-optimal set allows us to perform a trade-off analysis. Starting with a particular compromise solution, one may ask how much deterioration is necessary to obtain a certain improvement in another objective. Such information is basically given by the shape of the front. Non-convex regions of the front stand for large trade-offs, while convex parts represent low trade-offs.

Fig. 12

Model-centered problem solving using evolutionary multiobjective optimization.



Example 8 In Siegfried et al. (2009), a multiobjective groundwater management problem is tackled using an evolutionary algorithm. The task is to optimize the placement and the operation of pumping facilities over time, while considering multiple neighboring regions that are economically independent. For each region a separate cost objective function is considered, which reflects how much money needs to be spent for that region to ensure sufficient water supply for a prespecified time horizon.

⌚ [Figure 13](#) shows the approximated trade-off front for a three-region scenario. It is remarkable that there exists one solution which represents a so-called knee point (Branke et al. 2004a): it can be regarded as a decision vector that epsilon dominates the other decision vectors in the approximation set with a relatively small value for ε , cf. ⌚ [Eq. 7](#). Such information is highly useful for decision making, as it shows that – given the current estimate for the front – only small trade-offs need to be taken into account.

Another form of learning is to better understand the relationships between the considered objectives. Interesting questions in this context can be: how strongly are the objectives conflicting, can certain objectives be omitted without losing too much information, and which are the most important objectives? These are important aspects in particular from the perspective of decision making. It can become quickly infeasible to analyze a Pareto set approximation when the number of objectives increases. Therefore, an automatic dimensionality reduction of the objective space can be a valuable tool for a human user. Some studies addressed this problem and proposed methods based on (1) classical dimensionality reduction techniques such as principal component analysis (Deb and Saxena 2006; Saxena and Deb 2007), and (2) dedicated dominance-oriented reduction techniques (Brockhoff and Zitzler 2007, 2006). A remarkable result is that the notion of objective conflict always applies to groups of objectives and not to objective pairs only. Objective conflict can be defined in different ways (Deb 2001; Purshouse and Fleming 2003; Tan et al. 2005; Brockhoff and Zitzler 2007); the following notion is the most general (Brockhoff and Zitzler 2009):

- ▶ Let $(X, Z, \mathbf{f} = (f_1, \dots, f_n), \mathbf{g}, \leq_{par})$ be a multiobjective optimization problem. A set $\mathcal{F}_1 \subseteq \{f_1, \dots, f_n\}$ of objectives is denoted as *conflicting* with another set $\mathcal{F}_2 \subseteq \{f_1, \dots, f_n\}$ iff

$$\leq_{par}^{\mathcal{F}_1} \neq \leq_{par}^{\mathcal{F}_2} \quad (26)$$

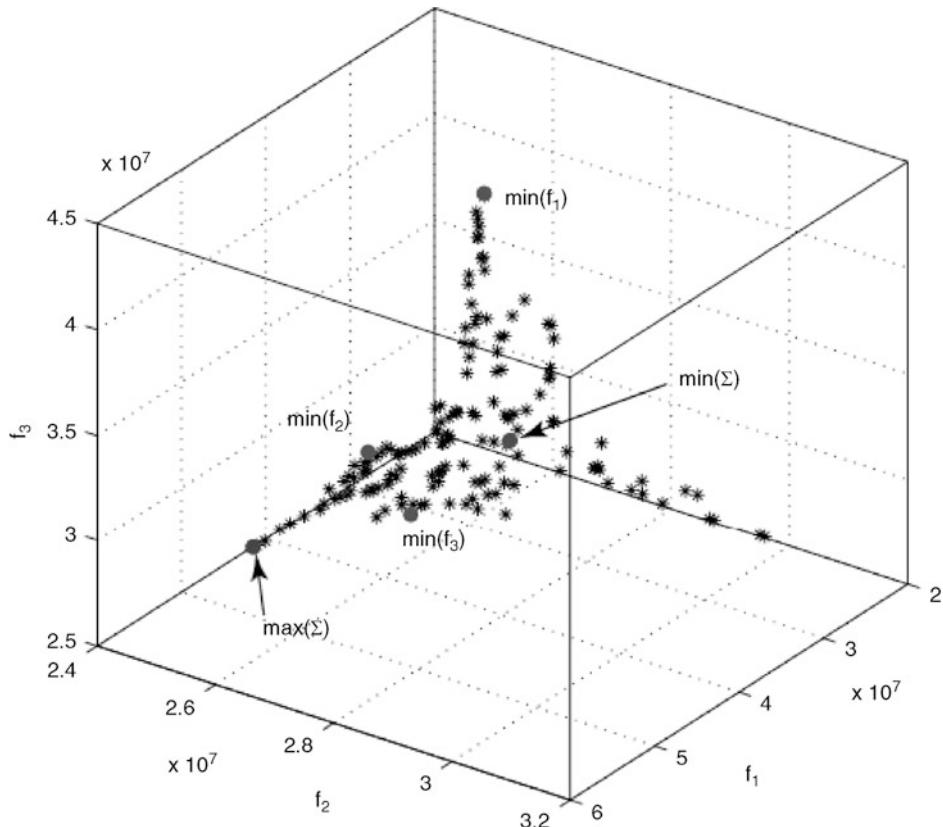
where the relation $\leq_{par}^{\mathcal{F}}$ restricts the weak Pareto dominance relation to the objectives in \mathcal{F} , that is,

$$\mathbf{u} \leq_{par}^{\mathcal{F}} \mathbf{v} : \Leftrightarrow \forall f \in \mathcal{F} : f(\mathbf{u}) \leq f(\mathbf{v}) \quad (27)$$

for all objective vectors $\mathbf{u}, \mathbf{v} \in Z$.

Fig. 13

Pareto set approximation obtained for the groundwater management application (Siegfried et al. 2009). Each axis corresponds to one cost function, all three are to be minimized. The annotated points represent the minima per objectives as well as the objective vectors providing the minimum resp. maximum unweighted sum over the objective components.



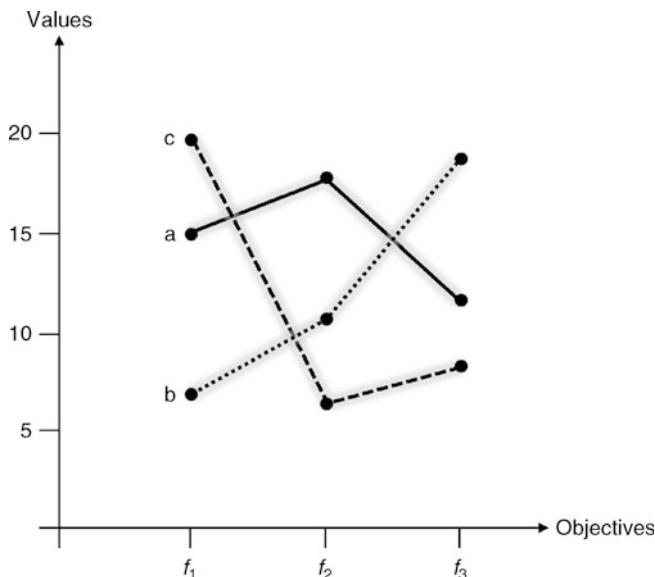
That means two groups of objectives are nonconflicting if they induce the same preference relation, which in turn implies that the Pareto-optimal sets are identical.

Example 9 [Figure 14](#) shows the parallel coordinates plot, cf. Purshouse and Fleming (2003), of three solutions **a** (solid line), **b** (dotted), and **c** (dashed) that are pairwisely incomparable. Assuming that X consists of only these three elements, the objective sets $\{f_1\}$, $\{f_2\}$, and $\{f_3\}$ are mutually conflicting, that is, all objectives are pairwisely conflicting, and also conflicting with the set $\{f_1, f_2, f_3\}$, that is, no objective alone can preserve the dominance relationships. However, the objective subset $\{f_1, f_3\}$ is not conflicting with $\{f_1, f_2, f_3\}$, which implies that the second objective could be omitted while preserving the dominance structure.

The above example shows that objectives may be removed without changing the dominance structure. Experimental results indicate that up to 50% of the objectives may be omitted

Fig. 14

Parallel coordinates plot for three solutions and three objectives. Each solution is represented by a line that connects the function values per objective.



when focusing on a subset $A \subseteq X$ of the decision space, for example, a Pareto set approximation (Brockhoff and Zitzler 2009). The same techniques can also be used to quantify the degree of conflict and to determine on this basis how strongly certain objectives are conflicting (Brockhoff and Zitzler 2006). Thereby, one can reveal which objectives are most influential and can explain the shape of the front best.

Thirdly, learning may include a combined decision space/objective space analysis of a Pareto set approximation in order to identify structural patterns. It is valuable information to reveal that certain regions of the Pareto-optimal front are correlated to and characterized by specific decision variable settings. However, to identify interesting structural patterns is itself a challenging optimization problem closely related to issues of data mining. Researchers have developed and applied dedicated clustering techniques for this purpose (Ulrich et al. 2008; Aittokoski et al. 2009). Such a clustering aims at finding groups of solutions (within a Pareto set approximation) that shows similarities in both decision space and objective space. The clustering results can later be used for a dimensionality reduction of the decision space, possibly in combination with the objective space.

5.2 Multiobjectivization

The term *multiobjectivization* has been coined by Knowles et al. (2001) and originally refers to reformulating a single-objective problem by means of a multiobjective model, which is then tackled using a set-based problem transformation. The idea is that the resulting problem is easier to solve or can be better handled than the original problem formulation – a principle that applies not only to single-objective problems, but also to multiobjective ones.

One can distinguish between three types of multiobjectivization, or more precisely ways to increase the number of objectives, to make a problem easier: (1) disaggregation, (2) decomposition, and (3) addition. The first type means that an originally multiobjective model, which was tackled using a solution-oriented problem transformation is now approached on the basis of a set-oriented problem transformation. The motivation for disaggregation is given by empirical and theoretical results. Different studies (Zitzler and Thiele 1999; Zitzler et al. 2000) have shown that a set-based multiobjective optimizer can be more effective than, for example, a weighted-sum single-objective optimizer for certain ranges of weight combinations. These empirical findings are supported by theoretical studies where a similar picture emerges (Laumanns et al. 2004a, b).

Decomposition, the second type of multiobjectivization, refers to reformulating an objective function in a different way in terms of several objectives. This reformulation can have the effect that the optimizer has more information available and therefore can search more efficiently. Scharnow et al. (2002) showed for the single-source shortest-paths problem and Neumann and Wegener (2006) demonstrated for the minimum spanning tree problem that a multiobjective approach is indeed faster in finding the (single-objective) optimum than a single-objective approach; these results are based on running time analyses. Practical evidence for these findings are provided in Knowles et al. (2001) and Handl et al. (2008).

As to the third type of multiobjectivization, adding objectives, there is a prominent application area, namely, the phenomenon of bloat in genetic programming, which was addressed first by different research groups simultaneously (Bleuler et al. 2001; Ekárt and Németh 2001; de Jong et al. 2001) and later further investigated in several follow-up studies (de Jong and Pollack 2003; Panait and Luke 2004; Bernstein et al. 2004; Kotanchek et al. 2006; Bleuler et al. 2007). The idea is to avoid useless growth of trees and the resulting decrease of search efficiency by considering the tree size as an additional objective. For different test problems, it was shown that a set-based multiobjective optimizer can find better and more compact solutions than a standard genetic programming approach and other solution-oriented approaches that combine quality and size in a single objective. In addition, there are theoretical results available that prove that a problem can become easier when adding an objective function (Brockhoff et al. 2007). Brockhoff et al. (2007) showed for a particular problem that optimizing each objective separately requires more steps in terms of running time complexity than treating the combined biobjective problem – although in the latter case a set of solutions has to be found. Clearly, the addition of objectives often makes a problem harder, as outlined in **• Sect. 2.3**, but if additional knowledge is integrated into the model using a further objective this can be beneficial for the search process.

References

- Aherne FJ, Thacker NA, Rockett PI (1997) Optimising object recognition parameters using a parallel multiobjective genetic algorithm. In: Proceedings of the 2nd IEE/IEEE international conference on genetic algorithms in engineering systems: innovations and applications (GALESIA'97). IEEE, Prague, pp 1–6
- Aittokoski T, Ayramo S, Miettinen K (2009) Clustering aided approach for decision making in computationally expensive multiobjective optimization. Optimization Meth Softw 24(2):157–174. doi: <http://dx.doi.org/10.1080/1055678080252531>
- Bader J, Bleuler S, Kunzli S, Laumanns M, Thiele L, Zitzler E (2008) PISA Website. <http://www.tik.ee.ethz.ch/sop/pisa/>
- Bader J, Brockhoff D, Welten S, Zitzler E (2009) On using populations of sets in multiobjective optimization. In: Ehrgott M et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2009), LNCS, vol 5467. Springer, pp 140–154

- Bader J, Deb K, Zitzler E (2010) Faster hypervolume-based search using Monte Carlo sampling. In: Ehrgott M et al. (eds) Conference on multiple criteria decision making (MCDM 2010), LNEMS, vol 634. Springer, Heidelberg, Germany, pp 313–326
- Baita F, Mason F, Poloni C, Ukovich W (1995) Genetic algorithm with redundancies for the vehicle scheduling problem. In: Biethahn J, Nissen V (eds) Evolutionary algorithms in management applications. Springer, Berlin, Germany, pp 341–353
- Basurra M, Zitzler E (2006) Handling uncertainty in indicator-based multiobjective optimization. *Int J Comput Intell Res* 2(3):255–272
- Bernstein Y, Li X, Ciesielski V, Song A (2004) Multi-objective parsimony enforcement for superior generalisation performance. In: Congress on evolutionary computation (CEC 2004). IEEE Press, Piscataway, NJ, pp 83–89
- Beume N, Rudolph G (2006) Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. *Tech. Rep. CI-216/06*, Sonderforschungsbereich 531 Computational Intelligence, Universität Dortmund, shorter version published at IASTED International Conference on Computational Intelligence (CI 2006)
- Bleuler S, Bader J, Zitzler E (2007) Reducing bloat in GP with multiple objectives. In: Knowles J, Corne D, Deb K (eds) Multi-objective problem solving from nature: from concepts to applications. Springer, Heidelberg, pp 177–200
- Bleuler S, Brack M, Thiele L, Zitzler E (2001) Multi-objective genetic programming: reducing bloat by using SPEA2. In: Congress on evolutionary computation (CEC 2001). IEEE, Piscataway, NJ, pp 536–543
- Bleuler S, Laumanns M, Thiele L, Zitzler E (2003) PISA—A platform and programming language independent interface for search algorithms. In: Fonseca CM, Fleming PJ, Zitzler E, Deb K, Thiele L (eds) Conference on evolutionary multi-criterion optimization (EMO 2003), LNCS, vol 2632. Springer, Berlin, pp 494–508
- Bradstreet L, While L, Barone L (2008) A fast incremental hypervolume algorithm. *IEEE Trans Evolut Comput* 12(6):714–723
- Branke J, Deb K (2004) Integrating user preferences into evolutionary multi-objective optimization. *Tech. Rep. 2004004*, Indian Institute of Technology, Kanpur, India, also published as book chapter in Jin Y (ed): Knowledge incorporation in evolutionary computation. Springer, Berlin, pp 461–477
- Branke J, Deb K, Dierolf H, Osswald M (2004a) Finding knees in multi-objective optimization. In: Runarsson TP et al. (eds) Conference on parallel problem solving from nature (PPSN VIII), LNCS, vol 3242. Springer, Berlin, pp 722–731
- Branke J, Schmeck H, Deb K, Reddy M (2004b) Parallelizing multi-objective evolutionary algorithms: cone separation. In: Congress on evolutionary computation (CEC 2004), vol 2. IEEE Service Center, Portland, OR, pp 1952–1957
- Branke J, Kausler T, Schmeck H (2001) Guidance in evolutionary multi-objective optimization. *Adv Eng Softw* 32:499–507
- Bringmann K, Friedrich T (2008) Approximating the volume of unions and intersections of high-dimensional geometric objects. In: Proceedings of the 19th international symposium on algorithms and computation (ISAAC 2008). Springer, Berlin, Germany
- Brockhoff D, Zitzler E (2006) Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: Runarsson T et al. (eds) Conference on parallel problem solving from nature (PPSN IX), vol 4193. Springer, Berlin, Germany, LNCS, pp 533–542
- Brockhoff D, Zitzler E (2007) Dimensionality reduction in multiobjective optimization: the minimum objective subset problem. In: Waldmann KH, Stocker UM (eds) Operations Research Proceedings 2006. Springer, Karlsruhe, pp 423–429
- Brockhoff D, Zitzler E (2009) Objective reduction in evolutionary multiobjective optimization: theory and applications. *Evolut Comput* 17(2):135–166
- Brockhoff D, Friedrich T, Hebbinghaus N, Klein C, Neumann F, Zitzler E (2007) Do additional objectives make a problem harder? In: Thierens D et al. (eds) Genetic and evolutionary computation conference (GECCO 2007). ACM Press, New York, NY, pp 765–772
- Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, 2nd edn. Springer, Berlin, Germany
- Coello Coello CA, Van Veldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, Norwell
- Conover WJ (1999) Practical nonparametric statistics, 3rd edn. Wiley, New York
- Cvetković D, Parmee IC (2002) Preferences and their application in evolutionary multiobjective optimisation. *IEEE Trans Evolut Comput* 6(1):42–57
- Czyzak P, Jaskiewicz A (1998) Pareto simulated annealing—a metaheuristic for multiobjective combinatorial optimization. *Multi-criteria Decis Anal* 7:34–47
- de Jong ED, Pollack JB (2003) Multi-objective methods for tree size control. *Genet Programming Evol Mach* 4:211–233
- de Jong ED, Watson RA, Pollack JB (2001) Reducing bloat and promoting diversity using multi-objective methods. In: Spector L et al. (eds) Genetic and evolutionary computation conference (GECCO

- 2001). Morgan Kaufmann, San Francisco, CA, pp 11–18
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK
- Deb K, Gupta H (2006) Introducing robustness in multi-objective optimization. *Evolut Comput* 14 (4):463–494
- Deb K, Saxena D (2006) Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Congress on evolutionary computation (CEC 2006). IEEE Press, Seattle, WJ, pp 3352–3360
- Deb K, Sundar J (2006) Reference point based multi-objective optimization using evolutionary algorithms. In: Keijzer M et al. (eds) Conference on genetic and evolutionary computation (GECCO 2006). ACM, New York, pp 635–642
- Deb K, Chaudhuri S, Miettinen K (2006) Towards estimating nadir objective vector using evolutionary approaches. In: Keijzer M et al. (eds) Conference on genetic and evolutionary computation (GECCO 2006). ACM, New York, pp 643–650
- Deb K, Pratap A, Agarwal S, Meyarivan T (2002a) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evolut Comput* 6(2):182–197
- Deb K, Thiele L, Laumanns M, Zitzler E (2002b) Scalable multi-objective optimization test problems. In: Congress on evolutionary computation (CEC 2002). IEEE Press, Honolulu, pp 825–830
- Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multi-objective optimization. In: Abraham A, Jain R, Goldberg R (eds) Evolutionary multiobjective optimization: theoretical advances and applications. Springer, chap 6, Berlin, pp 105–145
- Ehrhart M (2005) Multicriteria optimization, 2nd edn. Springer, Berlin, Germany
- Ekárt A, Németh SZ (2001) Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. *Genet Programming Evol Mach* 2:61–73
- Emmerich M, Beume N, Naujoks B (2005) An EMO algorithm using the hypervolume measure as selection criterion. In: Conference on evolutionary multi-criterion optimization (EMO 2005), LNCS, vol 3410. Springer, Berlin, pp 62–76
- Fieldsend JE, Everson RE, Singh S (2003) Using unconstrained elite archives for multiobjective optimization. *IEEE Trans Evolut Comput* 7(3):305–323
- Fleming PJ, Purshouse RC, Lygoe RJ (2005) Many-objective optimization: an engineering design perspective. In: Coello Coello CA et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2005), LNCS, vol 3410. Springer, Berlin, Germany, pp 14–32
- Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: Forrest S (ed) Conference on genetic algorithms. Morgan Kaufmann, San Mateo, CA, pp 416–423
- Fonseca CM, Fleming PJ (1995) An overview of evolutionary algorithms in multiobjective optimization. *Evolut Comput* 3(1):1–16
- Fonseca CM, Fleming PJ (1996) On the performance assessment and comparison of stochastic multiobjective optimizers. In: Parallel problem solving from nature (PPSN IV). Springer, Berlin, Germany, pp 584–593
- Fonseca CM, Fleming PJ (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: a unified formulation. *IEEE Trans Syst Man Cybern* 28(1):26–37
- Fonseca CM, Paquete L, López-Ibáñez M (2006) An improved dimension-sweep algorithm for the hypervolume indicator. In: Congress on evolutionary computation (CEC 2006), Vancouver. IEEE Press, pp 1157–1163
- Gaspar-Cunha A, Covas JA (2008) Robustness in multi-objective optimization using evolutionary algorithms. *Comput Optimization Appl* 39(1):75–96
- Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
- Grunert da Fonseca V, Fonseca CM, Hall AO (2001) Inferential performance assessment of stochastic optimizers and the attainment function. In: Zitzler E et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2001), LNCS, vol 1993. Springer, Zurich, pp 213–225
- Habenicht W (1983) Quad trees: a datastructure for discrete vector optimization problems. In: Hansen P (ed) Essays and surveys on multiple criteria decision making, LNEMS, vol 209. Springer, Berlin, pp 136–145
- Handl J, Lovell SC, Knowles J (2008) Multiobjectivization by decomposition of scalar cost functions. In: Conference on parallel problem solving from nature (PPSN X). Springer, Berlin, pp 31–40
- Hansen MP, Jaszkiewicz A (1998) Evaluating the quality of approximations of the non-dominated set. Tech. rep., Institute of Mathematical Modeling, Technical University of Denmark, iMM Technical Report IMM-REP-1998-7
- Helbig S, Pateva D (1994) On several concepts for ε -efficiency. *OR Spektrum* 16(3):179–186
- Horn J (1997) Multicriterion decision making. In: Bäck T, Fogel DB, Michalewicz Z (eds) Handbook of evolutionary computation, IOP Publishing and Oxford University Press, Bristol, UK
- Horn J, Nafpliotis N, Goldberg DE (1994) A niched pareto genetic algorithm for multiobjective

- optimization. In: Congress on evolutionary computation (CEC 1994). IEEE Press, Piscataway, pp 82–87
- Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans Evolut Comput* 10(5):477–506
- Hughes E (2001) Evolutionary multi-objective ranking with uncertainty and noise. In: Evolutionary multi-criterion optimization, Lecture notes in computer science. Springer, Berlin, pp 329–343
- Hughes EJ (2007) Radar waveform optimization as a many-objective application benchmark. In: Conference on evolutionary multi-criterion optimization (EMO 2007), LNCS, vol 4403. Springer, Heidelberg, pp 700–714
- Igel C, Hansen N, Roth S (2007) Covariance matrix adaptation for multi-objective optimization. *Evolut Comput* 15(1):1–28
- Ishibuchi H, Murata T (1996) Multi-objective genetic local search algorithm. In: IEEE (ed) Proceedings of the 1996 International conference on evolutionary computation. Nagoya, Japan, pp 119–124
- Jaszkiewicz A (2002) On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—a comparative experiment. *IEEE Trans Evolut Comput* 6(4):402–412
- Jaszkiewicz A, Branke J (2008) Interactive multiobjective evolutionary algorithms. In: Branke J, Deb K, Miettinen K, Slowinski R (eds) Multiobjective optimization: interactive and evolutionary approaches. Springer, Heidelberg, pp 179–193
- Jensen MT (2003) Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. *IEEE Trans Evolut Comput* 7(5):503–515
- Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments—a survey. *IEEE Trans Evolut Comput* 9(3):303–317
- Knowles J, Corne D (2000a) M-PAES: a memetic algorithm for multiobjective optimization. In: Congress on evolutionary computation (CEC 2000). IEEE Press, Piscataway, NJ, pp 325–332
- Knowles JD, Corne DW (2000b) Approximating the non-dominated front using the pareto archived evolution strategy. *Evolut Comput* 8(2):149–172
- Knowles J, Corne D (2002) On metrics for comparing non-dominated sets. In: Congress on evolutionary computation (CEC 2002). IEEE Press, Piscataway, NJ, pp 711–716
- Knowles J, Corne D (2003a) Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Trans Evolut Comput* 7(2):100–116
- Knowles JD, Corne DW (2003b) Instance generators and test suites for the multiobjective quadratic assignment problem. In: Fonseca CM, Fleming PJ, Zitzler E, Deb K, Thiele L (eds) Evolutionary multi-criterion optimization (EMO 2003), LNCS, vol 2632. Springer, Berlin, pp 295–310
- Knowles J, Thiele L, Zitzler E (2006) A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich
- Knowles JD, Watson RA, Corne DW (2001) Reducing local optima in single-objective problems by multi-objectivization. In: Zitzler E et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2001), LNCS, vol 1993. Springer, Berlin, pp 269–283
- Kotanek M, Smits G, Vladislavleva E (2006) Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In: Riolo RL, Soule T, Worzel B (eds) Genetic programming theory and practice IV, genetic and evolutionary computation, vol 5. Springer, chap 3
- Künzli S, Bleuler S, Thiele L, Zitzler E (2004) A computer engineering benchmark application for multiobjective optimizers. In: Coello CAC, Lamont G (eds) Applications of multi-objective evolutionary algorithms. World Scientific, Singapore, pp 269–294
- Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multiobjective optimization. *Evolut Comput* 10(3):263–282
- Laumanns M, Thiele L, Zitzler E (2004a) Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Nat Comput* 3(1):37–51
- Laumanns M, Thiele L, Zitzler E (2004b) Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Trans Evolut Comput* 8(2):170–182
- Laumanns M, Thiele L, Zitzler E (2006) An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *Eur J Oper Res* 169(3):932–942
- Lee J, Hajela P (1996) Parallel genetic algorithm implementation in multidisciplinary rotor blade design. *J Aircraft* 33(5):962–969
- Mezzam M, Melab N, Talbi E-G (2006) Using the multi-start and island models for parallel multi-objective optimization on the computational grid. In: eScience. IEEE Computer Society, Washington, DC, p 112
- Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer, Boston, MA
- Molina J, Santana LV, Hernández-Díaz A, Coello Coello CA, Caballero R (2009) G-dominance: reference point based dominance for multiobjective metaheuristics. *Eur J Oper Res* 197(2):685–692. doi:10.1016/j.ejor.2008.07.015, <http://www.sciencedirect.com/science/article/B6VCT-4T2M5WY-1/2/498e5b5a39d874c7aee53e01a5557910>
- Mostaghim S, Teich J, Tyagi A (2002) Comparison of data structures for storing pareto-sets in MOEAs.

- In: Congress on evolutionary computation (CEC 2002). IEEE Press, Piscataway, NJ, pp 843–848
- Neumann F, Wegener I (2006) Minimum spanning trees made easier via multi-objective optimization. *Nat Comput* 5(3):305–319, conference version in Beyer H-G et al. (eds.) Genetic and evolutionary computation conference, GECCO 2005, Volume 1. ACM Press, New York, pp 763–770
- Okabe T, Jin Y, Olhofer M, Sendhoff B (2004) On test functions for evolutionary multi-objective optimization. In: Parallel problem solving from nature (PPSN VIII). Springer, Berlin, pp 792–802
- Panait L, Luke S (2004) Alternative bloat control methods. In: Genetic and evolutionary computation conference (GECCO 2004), LNCS. Springer, pp 630–641
- Poloni C (1995) Hybrid GA for multi-objective aerodynamic shape optimization. In: Winter G, Periaux J, Galan M, Cuesta P (eds) Genetic algorithms in engineering and computer science. Wiley, Chichester, UK, pp 397–416
- Purshouse RC, Fleming PJ (2003) Conflict, harmony, and independence: relationships in evolutionary multi-criterion optimisation. In: Conference on evolutionary multi-criterion optimization (EMO 2003), LNCS, vol 2632. Springer, Berlin, pp 16–30
- Rachmawati L, Srinivasan D (2006) Preference incorporation in multi-objective evolutionary algorithms: a survey. In: Congress on evolutionary computation (CEC 2006). IEEE Press, pp 962–968
- Rudolph G (1998) On a multi-objective evolutionary algorithm and its convergence to the pareto set. In: Proceedings of the IEEE International conference on evolutionary computation. IEEE Press, Piscataway, pp 511–516
- Rudolph G (2001) Some theoretical properties of evolutionary algorithms under partially ordered fitness values. In: Evolutionary algorithms workshop (EAW-2001). Bucharest, pp 9–22
- Rudolph G, Agapie A (2000) Convergence properties of some multi-objective evolutionary algorithms. In: Zalzala A, Eberhart R (eds) Congress on evolutionary computation (CEC 2000), vol 2. IEEE Press, New York, pp 1010–1016
- Sawai H, Adachi S (2000) Effects of hierarchical migration in a parallel distributed parameter-free GA. In: Congress on evolutionary computation (CEC 2000). IEEE Press, Piscataway, NJ, pp 1117–1124
- Saxena DK, Deb K (2007) Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: employing correntropy and a novel maximum variance unfolding. In: Conference on evolutionary multi-criterion optimization (EMO 2007), LNCS, vol 4403. Springer, Berlin, pp 772–787
- Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette JJ (ed) Conference on genetic algorithms and their applications. Pittsburgh, PA, pp 93–100
- Scharnow J, Tinnefeld K, Wegener I (2002) Fitness landscapes based on sorting and shortest paths problems. In: Conference on parallel problem solving from nature (PPSN VII), LNCS, vol 2439. Springer, Berlin, pp 54–63
- Siegfried T, Bleuler S, Laumanns M, Zitzler E, Kinzelbach W (2009) Multi-objective groundwater management using evolutionary algorithms. *IEEE Trans Evolut Comput* 13(2):229–242
- Srinivas N, Deb K (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolut Comput* 2(3):221–248
- Steuer R, Miettinen K (2008) International Society on Multiple Criteria Decision Making. <http://www.terry.uga.edu/mcdm/>
- Talbi EG, Mostaghim S, Okabe T, Ishibuchi H, Rudolph G, Coello Coello CA (2008) Parallel approaches for multiobjective optimization. In: Branke J et al. (eds) Multiobjective optimization: interactive and evolutionary approaches. Springer, Heidelberg, pp 349–372
- Tan KC, Khor EF, Lee TH (2005) Multiobjective evolutionary algorithms and applications. Springer, London
- Teich J (2001) Pareto-front exploration with uncertain objectives. In: Conference on evolutionary multi-criterion optimization (EMO 2001). In: Zitzler E, Deb K, Thiele L, Coello Coello CA, Corne D, LNCS, vol 1993. Springer, pp 314–328
- Ulrich T, Brockhoff D, Zitzler E (2008) Pattern identification in pareto-set approximations. In: Keijzer M et al. (eds) Genetic and evolutionary computation conference (GECCO 2008). ACM, pp 737–744
- Voß T, Hansen N, Igel C (2009) Recombination for learning strategy parameters in the MO-CMA-ES. In: Ehrhart M et al. (eds) Evolutionary multi-criterion optimization. Lecture notes in computer science, vol 5467. Springer, pp 155–168
- Wagner T, Beume N, Naujoks B (2007) Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Obayashi S et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2007), LNCS, vol 4403. Springer, pp 742–756
- Winkler P (1985) Random orders. *Order* 1(1985):317–331
- Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Conference on parallel problem solving from nature (PPSN VIII), LNCS, vol 3242. Springer, Heidelberg, pp 832–842
- Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Conference on parallel problem solving from Nature (PPSN V). Amsterdam, pp 292–301

- Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans Evolut Comput* 3(4):257–271
- Zitzler E, Brockhoff D, Thiele L (2007) The hyper-volume indicator revisited: on the design of pareto-compliant indicators via weighted integration. In: Obayashi S et al. (eds) Conference on evolutionary multi-criterion optimization (EMO 2007), LNCS, vol 4403. Springer, Berlin, pp 862–876
- Zitzler E, Deb K, Thiele L (2000) Comparison of multi-objective evolutionary algorithms: empirical results. *Evolut Comput* 8(2):173–195
- Zitzler E, Knowles J, Thiele L (2008a) Quality assessment of pareto set approximations. In: Branke J, Deb K, Miettinen K, Slowinski R (eds) Multiobjective optimization: interactive and evolutionary approaches. Springer, Berlin, pp 373–404
- Zitzler E, Laumanns M, Thiele L (2002) SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou K et al. (eds) Evolutionary methods for design, optimisation and control with application to industrial problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE). Athens, Greece, pp 95–100
- Zitzler E, Thiele L, Laumanns M, Fonseca CM, Grunert da Fonseca V (2003) Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans Evolut Comput* 7(2):117–132
- Zitzler E, Thiele L, Bader J (2008b) On set-based multi-objective optimization (Revised Version). TIK Report 300, Computer Engineering and Networks Laboratory (TIK), ETH Zurich

29 Memetic Algorithms

Natalio Krasnogor

Interdisciplinary Optimisation Laboratory, The Automated Scheduling,
Optimisation and Planning Research Group, School of Computer
Science, University of Nottingham, UK
natalio.krasnogor@nottingham.ac.uk

1	<i>Introduction</i>	906
2	<i>A Pattern Language for Memetic Algorithms</i>	907
3	<i>A Pragmatic Guide to Fitting It All Together</i>	926
4	<i>Brief Notes on Theoretical Aspects</i>	928
5	<i>The Future: Toward Nature-Inspired Living Software Systems</i>	928
6	<i>Conclusions</i>	930

Abstract

Memetic algorithms (MA) has become one of the key methodologies behind solvers that are capable of tackling very large, real-world, optimization problems. They are being actively investigated in research institutions as well as broadly applied in industry. This chapter provides a pragmatic guide on the key design issues underpinning memetic algorithms (MA) engineering. It begins with a brief contextual introduction to memetic algorithms and then moves on to define a pattern language for MAs. For each pattern, an associated design issue is tackled and illustrated with examples from the literature. The last section of this chapter “fast forwards” to the future and mentions what, in our mind, are the key challenges that scientists and practitioners will need to face if memetic algorithms are to remain a relevant technology in the next 20 years.

1 Introduction

Memetic algorithms (MAs) was the name given by Moscato (1989) to a class of stochastic global search techniques that, broadly speaking, combine within the framework of evolutionary algorithms (EAs) the benefits of problem-specific local search heuristics and multi-agent systems. MAs have been successfully applied to a wide range of domains that cover problems in combinatorial optimization (Reeves 1996; Burke et al. 1996; Fleurent and Ferland 1997; He and Mort 2000; Cheng and Gen 1997; Carr et al. 2002; Tang and Yao 2007; Gutin et al. 2007), continuous optimization (Hart 1994; Niesse and Mayne 1996; Morris et al. 1998), dynamic optimization (Vavak and Fogarty 1996; Caponio et al. 2007; Wang et al. 2008), multi-objective optimization (Liu et al. 2007; Ishibuchi and Kaige 2004; Jaskiewicz 2002), etc. It could be argued that, unlike other nature-inspired algorithms such as ant colony optimization (ACO) (Dorigo et al. 1997), simulated annealing (Kirkpatrick et al. 1983), neural networks (McCulloch and Pitts 1943), evolutionary algorithms (Holland 1976), etc., memetic algorithms lack, at their core, a clear natural metaphor. Whether this is a strength or weakness of the paradigm is a question for another time and this chapter focuses first on a pragmatic software engineering presentation of this remarkably malleable search technology and then, toward the end of the chapter, argues that there is indeed a potentially powerful nature-inspired metaphor that could lead to important new breakthroughs in the field.

Early in the history of the application of EAs to real-world problems, it became apparent that a *canonical GA*, namely, one using a simple binary representation, n-point crossover, bitwise mutation, and fitness proportionate selection, could not possibly compete with tailor-made algorithms. This empirical observation resonated well with theoretical (and experimental) studies on the so called “Baldwin effect” and on “Lamarckian evolution” (Hinton and Nowlan 1987; Bull et al. 2000; Houck et al. 1997; Mayley 1996; Turney 1996; Whitley et al. 1994; Whitley and Gruau 1993) that focused on how learning could affect the process of evolution. Thus, it became apparent that the global search dynamics of EAs ought to be complemented with local search refinement provided by a suitable hybridization using problem-specific solvers including heuristics, approximate, and exact algorithms. Moreover, further theoretical results (Wolpert and Macready 1997) (and similar subsequent work) debunked the idea that effective and efficient “black box” general problem solvers were attainable, and hence gave further impetus to the school of thought that supported, as an essential methodological component, the incorporation of problem (or domain)-specific information

in EAs. (Please note that everything said so far about evolutionary algorithms can also be applied to other search frameworks such as tabu search, simulated annealing or ant colony optimization, etc.) Domain-specific knowledge was thus added to the EA framework by means of specialized crossover and mutation operators, sophisticated problem-specific representations, smart population initialization, complex fitness functions (closer to the spirit of MAs), local search heuristics and, when available, approximate and exact methods. More recently, R. Dawkins' concept of "memes" (Dawkins 1976) has been gathering pace within the memetic algorithms literature as they can be thought of as representing "evolvable" strategies for problem solving, thus breaking the mould of a fixed and static domain knowledge captured once during the design of the MAs and left untouched afterward. Thus Dawkins' memes, and their extensions (Cavalli-Sforza and Feldman 1981; Durham 1991; Gabora 1993; Blackmore 1999) as evolvable search strategies (Krasnogor 1999, 2002, 2004b; Krasnogor and Gustafson 2004; Smith 2003; Burke et al. 2007a) provide a critical link to the possibility of open-ended combinatorial and/or continuous problem solving.

Software development is a process of knowledge acquisition (Armour 2007) and the development of successful memetic algorithms is no different. The popularity behind MAs is more closely related to the relative ease by which a reasonably good solver can be implemented than to any fundamental advantage over other optimization techniques such as tabu search or simulated annealing (to name but two). Indeed, any successful nature-inspired search method owes its popularity not to an intrinsic problem-solving feature, which might be absent from a competing method, but rather to the fact that, in spite of obvious design flaws (e.g., large number of parameters, lack of operational theory for their use, etc.), they help to structure around them a healthy research and practice milieu. That is, nature-inspired search methods are computational "research programs," or "research paradigms," in their own right (Kuhn 1962). Thus the question of what are the key components of the memetic algorithms research paradigm takes center stage. The literature has a large number of papers in which a variety of methods are classified as memetic algorithms. Thus, although the large majority of memetic algorithms are instances of evolutionary algorithms-local search hybrids, numerous MAs are derived from other metaheuristics, for example, ant colony optimization (ACO) (Lee and Lee 2005), particle swarm (Liu et al. 2005), artificial immune systems (AIS) (Yanga et al. 2008), etc. What all of these implementations have in common is a *carefully choreographed interplay between (stochastic) global search and local search strategies*. The remaining parts of this chapter consider some of the key implementation strategies that, over the course of the years, have (re)appeared in the form of tried and tested algorithmic design solutions to the ubiquitous problem of how to successfully orchestrate global and local search methods in complex search spaces.

2 A Pattern Language for Memetic Algorithms

Alexander et al. (1977) introduced, within the context of architecture and urban planning, the concept of "Patterns" and "Pattern Languages":

- ▶ In this book, we present one possible pattern language,... The elements of this language are entities called patterns. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

A pattern language is then defined as a collection of interrelated patterns, with each and every one of them expressed in a concise, clear, and uniform format. The content of a pattern includes at least the following elements (Gamma et al. 1995):

- *The pattern name*, which is a concise handler to refer to both a specific problem and a tried and tested solution. By having a carefully selected set of pattern names (e.g., selection mechanism, crossover strategy, exploration strategy, diversification plan, etc.) the pattern language, that is, the vocabulary of, in our case, nature-inspired paradigms – for example, memetic algorithms – is enriched. Critically, a small increase in the size of a vocabulary creates a rich and expressive combinatorial explosion of patterns' compositions, thus opening the road for substantial research and practical experimentation. The importance of this observation will become apparent once we describe self-generating memetic algorithms.
- *The problem statement* depicting the situation in which the pattern is best applied, that is, the problem that the pattern attempts to provide a solution to, for example, maintaining pareto front diversity, etc. The problem statement might also contain a set of constraints describing situations where the pattern should not be applied or, symmetrically, conditions that must be fulfilled before the pattern can be used.
- *The solution*, in turn, provides a template on how to approach the solution of the problem to which the pattern is applied. The description in this section is not prescriptive but rather qualitative. As emphasized by Alexander et al. (1977), one might reuse a solution under myriads of different shapes, yet the essential core of all those implementations should be easily distinguishable and invariant.
- *The consequences* of applying the pattern. There are no free lunches, hence, even when a pattern might be the best (or perhaps only) solution to a given problem, its application might lead to a series of trade-offs. The more explicit and clearly stated these are, the more clear and precise the pattern language as a whole will be. For example, a pattern that calls for the reinitialization of a population due to a diversity crisis might carry with it, as obvious collateral damage, certain loss of information. Thus, by employing a reinitialization pattern, one might be forced to utilize a pattern that safeguards partial solutions.
- *Representative examples* briefly mentioning cases where the pattern has been used.

Thus, a collection of well-defined patterns, that is, a rich pattern language, substantially enhances one's ability to communicate solutions to recurring problems without the need to discuss specific implementation details. The pattern language thus serves the dual purpose of being both a taxonomy of problems and a catalog of solutions. This chapter provides a series of patterns that will be defined as per the tableaux that appears above. A reader interested in, for example, finding out about diversity handling strategies for MAs, irrespective of which underlying framework (e.g., evolutionary, ant colony, artificial immune system) the MA is

Fig. 1

In (a) the pseudocode (reproduced from Bacardit and Krasnogor (2009)) for a Memetic Learning Classifier System, (b) the pseudocode (reproduced from Merz (2003)) for an Estimation of Distribution-Like Memetic Algorithm, (c) a Memetic Particle Swarm Optimization pseudocode (reproduced from Petalas et al. (2007)), (d) and (e) pseudocode of a representative Ant Colony Optimization metaheuristic (reproduced from Cordon et al. (2002)) with a solution refining strategy, through local search, for Ant Colony Optimization (ACO) and (f) an Artificial Immune System (AIS) inspired memetic algorithm flowchart (reproduced from Yanga et al. 2008).

```

Procedure GA Cycle
Population = Initialize population
Evaluate (Population)
For it = 1 to NumIterations
Selection (Population)
Offspring = CrossOver (Population)
Mutation(Offspring)
Localsearch (Offspring)
Population = Replacement (Population, Offspring)
Evaluate (Population)
EndFor
Output : Best individual from Population

```

```

Procedure LocalSearch
Input : Population
ForEach Individual in Population
    If rand(0,1) < probLocalSearch
        Apply Rule-wise local search operators to individual/
    Endif
EndForEach

```

Output : Population

a

```

Input : N, X, c1, c2, xmin, xmax (lower & upper bounds), F (objective function),
Set t = 0.
Initialize xi(t), vi(t) in [xmin, xmax], pi(t) ← xi(t), i = 1,..., N.
Evaluate F(xi(t))
Determine the indices gi, i = 1,..., N.
While (stopping criterion is not satisfied) Do
    Update the velocities vi(t+1), i = 1,..., N, according to (1).
    Set xi(t+1) = vi(t+1), i = 1,...,N.
    Constrain each particle xi in [xmin, xmax].
    Evaluate F(xi(t+1)), i = 1,...,N.
    If F(xi(t+1)) < F(pi(t)) Then pi(t+1) ← xi(t+1)
    Else pi(t+1) ← pi(t).
    Update the indices gi.
    When (local search is applied) Do
        Choose (according to one of the Schemata 1-3) pg(t+1), q = (1,...,N).
        Apply local search on pq(t+1) and obtain a new solution, y.
        If F(y) < F(pq(t+1)) Then pq(t+1) ← y.
    End When
    Set t = t+1.
End while

```

C

```

1 Procedure ACO_Metaheuristic
2     parameter_initialization()
3     while (termination_criterion not_satisfied)
4         schedule_activities()
5             ants_generation_and_activity()
6             pheromone_evaporation()
7             daemon_actions() [optional]
8         end schedule_activities
9     end while
10 end Procedure

1 Procedure ants_generation_and_activity()
2     repeat in parallel for k=1 to m (number_of_ants)
3         new_ant(k)
4     end repeat in parallel
5 end Procedure

1 Procedure new_ant (ant.id)
2     initialize_ant (ant.id)
3     L = update_ant_memory()
4     while (current_state ≠ target_state)
5         P = compute_transition_probabilities (A,L,Ω)
6         next_state = apply_ant_decision_policy(P,Ω)
7         move_to_next_state(next_state)
8         if (on_line_step_by_step_pheromone_update)
9             deposit_pheromone_on_the_visited_edge()
10        end if
11        L = update_internal_state()
12        if (online_delayed_pheromone_update)
13            for each visited edge
14                deposit_pheromone_on_the_visited_edge()
15            end for
16        end if
17        release_ant_resources (ant.id)
18    end Procedure

```

e

```

function cMA(psize, gens, rec, ub) : X
begin
    for i = 1 to psize do p[i] := 0.5;
    X := generate (p);
    X := localSearch(X);
    X* := X;
    i := 1;
repeat
    X := generate(p);
    X := localSearch(X);
    if rec then X* = recombine (X*, X);
    else X* := generate (p);
    X* := localSearch (X* );
    if f(X) < f(X*) then Swap (X, X* );
    if f(X) > f(X*) then X* = X;
    if up then update (p, X, x, psize);
    else update (p, X, x, psize);
    i := i + 1;
until (i > gens);
return X*;
end

```

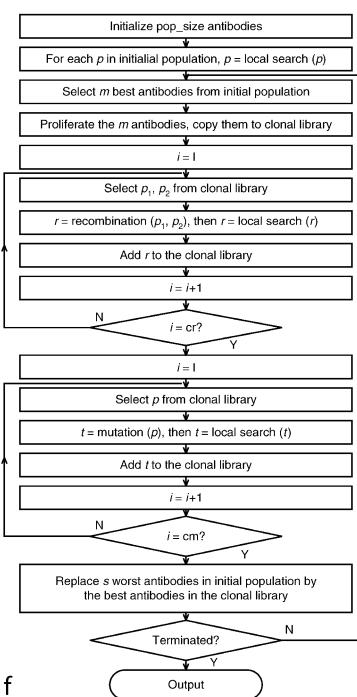
b

```

1 Procedure daemon_actions
2     for each Ss do local_search(Ss) [optional]
3         rank (S1,...,Sm) in decreasing order of solution
4             quality into (S'1,...,S'm)
5             if (best (S',Sglobal-best))
6                 Sglobal-best = S'1
7             end if
8             for μ = 1 to (σ - 1) do
9                 for each edge ars ∈ S'μ do
10                    τrs = τrs + (σ - μ) . IC(S'μ))
11                end for
12                for each edge ars ∈ Sglobal-best do
13                    τrs = τrs + σ . IC(Sglobal-best))
14                end for
15 end procedure

```

d



f

being implemented in, can quickly scan the various patterns in the pattern language catalog, identify the one related to diversity strategies and rapidly gain an idea of the tried and tested approaches, the pattern's motivation and consequences. Furthermore, the reader could then refer to the mentioned literature for concrete, detailed codes and methods.

It has been argued (Cooper 2000) that design patterns are seldom created but rather they are discovered through a process of datamining the source code base and literature base for reusable solutions to recurring problems. In this spirit, and before describing a specific pattern language for memetic algorithms, one sees the pseudocodes and flowcharts of some representative MA instances. These algorithms are reported in exactly the same form as found in the original publication so as to emphasize both the *invariants* in their architecture as well as the variety of “decorations” found in the many implementations of memetic algorithms. Other examples of “in the wild” MAs can be found in Krasnogor and Smith (2005).

Memetic improvements have been used in, for example, learning classifier systems (Bacardit and Krasnogor 2009) with the top level pseudocode shown in [Fig. 1a](#). An estimation of distribution (EDA) like MA, a compact memetic algorithm (Merz 2003) is shown in [Fig. 1b](#) (for a more recent EDA-MA see Duque et al. (2008)) while a memetic particle swarm optimization (PSO) (Petalas et al. 2007) pseudocode is depicted in [Fig. 1c](#). [Figure 1d](#) and [e](#) show a generic pseudocode of an ant colony optimization-based MA (Cordon et al. 2002) and its explicit use of solution-refining strategies (in the form of local search methods), respectively. An example of an immune system-inspired memetic algorithm's (Yanga et al. 2008) flowchart is shown in [Fig. 1f](#). The key invariant property that is present in the architecture of all these memetic algorithms is the combination of a search mechanism operating over (in principle) the entire search space with other search operators focusing on local regions of these search spaces. This key invariant holds true regardless of the nature-inspired paradigm the memetic algorithm is derived from or whether it is meant to solve an NP-hard combinatorial problem or a highly complex (e.g., multimodal, nonlinear, and multidimensional) continuous one.

We argue that the key problem that is addressed by memetic algorithms is the balance between global and local search, in other words, the strategy that different nature-inspired paradigms (e.g., ACO, AIS, etc.) might need to implement so as to benefit from a successful tradeoff between exploration and exploitation. Thus, the first top-level pattern can be defined.

Name: Memetic Algorithm Pattern (MAP) [1]

- *Problem statement:* A memetic algorithm provides solution patterns for the ubiquitous problem of how to successfully orchestrate a balanced tradeoff between exploring a search space and exploiting available (partial) solutions. It provides a suitable solution to complex problems where standard and efficient (i.e., approximation algorithms, exact algorithms, etc.) methods do not exist. The memetic algorithm is said to explore the search space through a “global” search technique, while exploitation is achieved through “local” search.
- *The solution:* This pattern relies on finding, for a given problem domain, an adequate instantiation of exploration and exploitation. Exploration is performed by “global search” methods usually implemented by means of a population-based nature-inspired method such as evolutionary algorithms, ant colony optimization, artificial immune systems, etc. Exploitation is commonly done through the use of local search methods and domain-specific heuristics. The global scale exploration is achieved by, for example, keeping track of multiple solutions or by virtue of specific “jump” operators that are able to connect

distant regions of the search space. The local scale exploitation focuses the search on the vicinity of a given candidate solution.

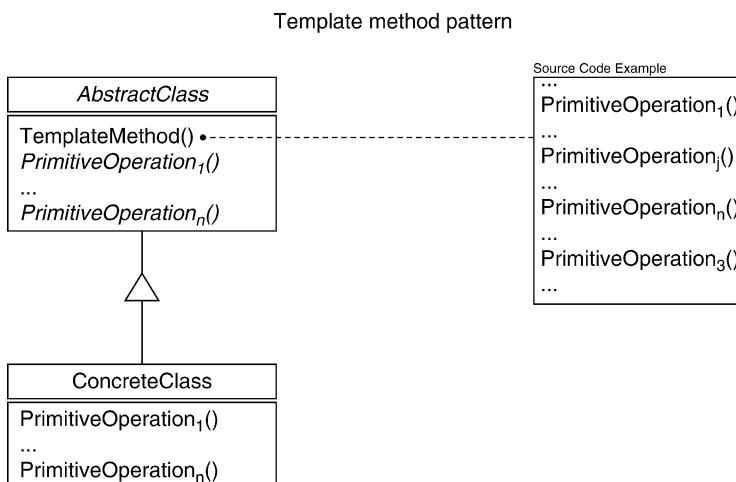
- *The consequences:* Hybridizing a global search method of any kind with local search and/or domain-specific heuristics usually results in better end-results, but this comes at the expense of increased computational time. The correct tradeoff between exploration and exploitation must be such that, were the global searcher given the same total CPU “budget” as the memetic algorithm, then, its solutions would still be worse than those derived from the MA. Needless to say, if the local searcher by itself, or through a naive multi-start shell could achieve the same quality of results as the memetic algorithm then the global searcher becomes irrelevant. Another likely consequence is that local search and domain-specific heuristics usually result in premature convergence (also called diversity crisis).
- *Examples:* Papers (Cordon et al. 2002; Yanga et al. 2008; Petalas et al. 2007; Duque et al. 2008; Bacardit and Krasnogor 2009) report on the use of memetic algorithms under a variety of nature-inspired incarnations.

The memetic algorithm patterns (MAP) can be refined through a variety of template patterns (TP) (Gamma et al. 1995), which allow for the definition of the *skeleton* of an algorithm, method, or protocol, through deferring problem-specific details to subclasses. In this way, through a judicious use of the template pattern, one can have a very generic and reusable recipe for implementing solutions to a range of, perhaps very different, problems.

⦿ *Figure 2* shows a class diagram capturing a template method pattern. The abstract class defines a template method that provides the algorithmic skeleton for a specific functionality. To achieve its functionality the template method calls one or more primitive operations (defined abstractly in the abstract class) that can be redefined in more specialized subclasses (concreteClass in the figure). ⦿ *Figures 1a-f* are examples of template method patterns for ACO, AIS, particle swarm optimization (PSO), EDA and LCS based memetic algorithms

■ Fig. 2

A UML class diagram sketching the structure for implementing Template Method patterns.
(Adapted from Gamma et al. (1995).)



respectively. Each TP captures the invariant properties of memetic algorithms when it is implemented from the perspective of a specific nature-inspired algorithm. It also captures the invariant features of MAs regardless of which nature-inspired route is used to implement it, for example, the entwining of global and local search procedures. The following sections will discuss other design patterns that are critical to the implementation of competent memetic algorithms thus extending the pattern language for MAs.

2.1 A Template Pattern for an Evolutionary Memetic Algorithm

As the design issues and critical considerations behind the implementation of MAs are almost identical across the various possible nature-inspired algorithms that could be used as templates, and as evolutionary memetic algorithms are the best known MAs, the focus of the following discussions will be on an evolutionary algorithm template pattern for MAs. The pattern language that will emerge, however, will be valid for other nature-inspired paradigms as well.

Name: Evolutionary Memetic Algorithm Template Pattern (EMATP) [2]

- *Problem statement:* the EMATP provides a viable route for solving the problem of how best to coordinate global and local search methods from within an evolutionary algorithms paradigm. Although in principle the simultaneous exploration of the search space by the evolutionary process and the exploitation (by refinement) of candidate solutions should result on an improved algorithm, this is not always the case. The expectation is that a hybridization of an EA would result in a synergistic net effect that would productively balance local and global search.
- *The solution:* the (almost) standard evolutionary cycle composed of *Initiate* → *Evaluate* → *Mate* → *Mutate* → *Select* → *RepeatUntilDone* is expanded with domain-specific operators that can refine this cycle. The refinement processes could vary for the different elements of the core EA's pipeline and could be implemented through exact, approximated or heuristic (e.g., local search) methods. Domain-specific operations are often implemented in the form of smart initializations, local search procedures, etc., that refine the input solutions to the mate and/or mutate processes as well as (more often) their outputs. Refinements could also be applied to the selection, initialization, and population management processes through, for example, fitness sharing, crowding, population structuring (e.g., cellular/lattice structures, demes, islands), and diversity management strategies.
- *The consequences:* paradoxically, although the key motivation for using memetic algorithms has been to refine solutions and converge toward good local optima (or ideally global optima) fast, sometimes the balance of local and global search is poorly implemented resulting in an untimely diversity crisis because the algorithm converges too fast. Another direct consequence of using memetic algorithms is that often, refining solutions by problem-specific strategies incurs in a substantial additional CPU commitment. Thus, it becomes essential to validate whether the exploration mechanisms of the core EA's pipeline when augmented with refining strategies do not end up producing worse solutions than having run the pure EAs with the same total CPU commitment (or the refinement strategies alone with an equivalent total CPU budget).

- Examples: Figure 3 shows a concrete example of an evolutionary memetic algorithm. A detailed description and analysis of several successful implementations of the EMATP pattern are described in Krasnogor and Smith (2005). The paper also provides a taxonomy for EMAs as well as a discussion of design issues.

Fig. 3

An evolutionary algorithm-based memetic algorithm template. The figure highlights the EA's core operators as well as the hotspots where the algorithm can be refined, that is, where "memetic" operators might come into play.

```
/* Variable and constants definitions:  
/* t is generation number  
/* IPi stands for intermediate population ith  
/* P(t) stands for the population at time t  
/* H stands for historical population archive  
*/  
Begin  
t ← 0;  
  
Initialize P(t);  
  
Evaluate P(t);  
Repeat Until (termination conditions = True) do  
  /* Mating FineGrainedScheduler, fSR, coordinates the application of */  
  /* Crossover and Refinement Operators to population P(t) */  
  
  IP1 ← fSR(Crossover, Refinement, P(t));  
  
  /* Mutation FineGrainedScheduler, fSM, coordinates the application of */  
  /* Mutation and Refinement Operators to population IP1 */  
  
  IP2 ← fSM(Mutation, Refinement, IP1);  
  
  /* Selection CoarseGrainedScheduler, cS, coordinates the application of */  
  /* Selection and Refinement Operators to population IP2 */  
  
  IP3 ← cS(Selection, Refinement, IP2);  
  
  t ← t+1;  
  P(t) ← IP3;  
  
  /* MetaScheduler, mS, coordinates the application of */  
  /* Refinement Operators to populations H ⊇ P(i) with i ∈ [0..t] */  
  
  H ← mS(Refinement, IP3);  
  
End;  
End;
```

Evolutionary algorithm backbone

Refinement hooks

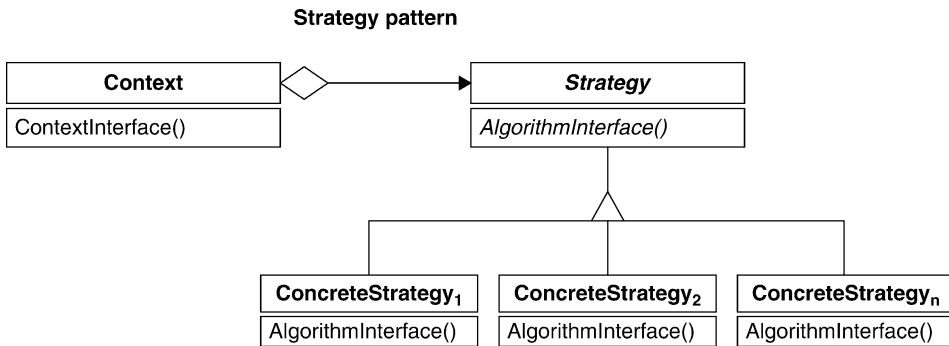
➊ *Figure 3* shows a concrete example of the EMATP. The figure identifies the key EA's components as the “backbone” and the potential places where refinement might take place as refinement hooks. Each hook is represented by a “scheduler” operator that manages the flow of information (e.g., partial/candidate solutions, time-dependent parameters, etc.) between each of the core EA's component and the refinement strategies associated with them. A formal notation for these schedulers can be found in Krasnogor and Smith (2005). A detailed study of the literature reveals that three types of schedulers can be abstracted from the various templates that are implemented. These schedulers are: a *fine-grain scheduler* for coordinating the operation of the genetic operators mutation and crossover with refinement methods, a *coarse-grain scheduler* for overseeing the interplay between population management strategies (e.g., selection) and refinement strategies and *meta scheduler* that orchestrate refinement procedures over longer timescales and larger spatial scales (e.g., islands, population structures, pareto archives). These schedulers receive their names because they operate at different level of granularities and timeframes. The fine-grain schedulers (represented by fS_R , fS_M in ➋ *Fig. 3*) have access to a very limited number of solutions and hence they have a very localized view of the current state of the search and thus the decision they can take in terms of parameter and operators adaptation is confined to small regions of the search space. The course-grain scheduler (represented by cS in ➋ *Fig. 3*), on the other hand, has access to a complete population and, perhaps, even to intermediate populations that could have been created in the main EA's pipeline. Thus it has a more global view of the search state that might be used to strategically guide the application of further refining methods to (parts of) the population it has access to. The meta scheduler (represented by mS in ➋ *Fig. 3*), on the other hand, has access to potentially the power set of all previously visited solutions and hence it has an even broader information base to decide the appropriate balance between exploration and exploitation. These schedulers also operate at different time-scales. Clearly, fS_* can operate very frequently indeed, potentially each time that a crossover or mutation event takes place. The coarse-grain scheduler operates once per generation and the mS , potentially, even less often. These differing spatial (i.e., access to solutions) and temporal (frequency of operation) features result in constraints on the complexity of the algorithmic strategies that these schedulers can implement and have a direct impact in several design issues such as whether or not a surrogate fitness function (Zhou 2004; Zhou et al. 2007) should be used, etc. Needless to say, EMATP can be implemented through a series of (object-oriented) class hierarchies. The EMATP pattern in ➋ *Fig. 3* can be encapsulated into an abstract class from which subclasses implement specific versions of the pattern. One could thus imagine a family of evolutionary memetic algorithms where one or more features are either removed from the pattern or added to it simply by overriding the behavior of the scheduler methods.

2.2 Strategy Patterns for Memetic Algorithms Design Issues

The memetic algorithms pattern language is extended by describing a series of strategy patterns associated with design issues arising from attempting to implement effective instances of the EMATP. The evolutionary algorithm's processes such as crossover and mutation, as well as the refinement strategies and the schedulers that coordinate their operations, are best thought of as strategy patterns (Gamma et al. 1995). These patterns are useful for defining a family of interchangeable algorithms.

Fig. 4

A UML class diagram sketching the structure for implementing strategy patterns. (Adapted from Gamma et al. (1995).)



The idea is that the client that employs a member of this family of algorithms, for example, the fine-grained scheduler coordinating a local search strategy with a crossover operator, should be able to use any of the available local search and/or crossover strategies without need for recoding. A most frequent example is provided when testing different, for example, crossover operators such as one-point crossover, two-point crossover and uniform crossover without the need to rewrite the EA's backbone. Similarly, one should be able to change the local search used without affecting the pattern in which it is used. That is, a strategy pattern should be used each time that a program, in this case a memetic algorithm, needs a specific service or function and when there are, in fact, several ways of executing these functions. To promote reusability, strategy patterns follow a scheme similar to the one depicted in **Fig. 4**. The choice of which strategy to implement as to perform the required function is problem- and instance-specific. Thus by using strategy patterns, one can change algorithms without causing a chain reaction of changes in the code that uses those algorithms. As argued below, there is a direct link between critical memetic algorithms design issues and strategy patterns.

2.2.1 Refinement Strategies

It has been argued that an EMATP provides conceptual solutions to the problem of balancing local and global search. The global search is performed through the implementation of some form of evolutionary algorithm while the local search is meant to be implemented by some form of refinement operator. The choice of operator refinement is a critical design decision in memetic algorithms engineering as it tremendously impacts the synergy between the EA's backbone and the refinement methods of choice.

Name: Refinement Strategy Pattern (RSP) [3]

- Problem statement:* this pattern attempts to address the following key questions: (1) what refinement operator should be used and, ultimately, (2) what fitness landscape will the MA be exploring? While the EMATP helps define the algorithmic skeleton of an

evolutionary-based memetic algorithm, it leaves to the strategy patterns the problem of how to decide what strategy would be implemented at the various refinement stages. Indeed, both theoretical and experimental work has analyzed the effectiveness of different local search strategies within memetic algorithms.

- *The solution:* the key design issue related to refinement strategies is how they prevent getting trapped in (poor) local optima, that is, search stagnation, while performing an efficient and fast local search. The answer to questions (1) and (2) above is problem, instance, and time dependent, hence a solution can only be provided at an abstract engineering level that allows for the seamless adaptation of the EMATP to different situations. More precisely, for a given set of low-level move operators, for example, n-exchange, n-swaps, bit-flip, etc., a variety of navigation rules have been implemented. Navigation rules specify how the search landscape induced by a given move operator is traversed. Some of the alternatives are breadth first search, depth first search, variable depth, etc. Besides the navigation method, the refinement strategy also needs to specify an acceptance criteria, for example, greedy, Monte Carlo, Great-Deluge (Dueck 1993; Burke et al. 2004), etc. Finally, the most sophisticated refinement strategies employ multiple move operators and acceptance criteria, for example, multimeme algorithms (Krasnogor 2004a; Krasnogor and Smith 2001), variable-neighborhood search (Hansen and Mladenovic 1998, 2001), etc. Indeed, a refinement strategy pattern might use at its core a full memetic algorithm pattern, see Romero-Campero et al. (2008) for an example. Thus, through the recursive nesting of MAP and RSP, it is possible to obtain extremely complex and versatile search methodologies.
- *The consequences:* the refinement strategy pattern comes with some “strings attached” to it. Its very flexibility means that a given memetic algorithm might be exploring not one but several search landscapes simultaneously. The precise nature of this combined search space should, ideally, be adequately analyzed either experimentally or theoretically (ideally in both ways). Fitness landscapes statistics were utilized in, for example, Kallel et al. (2001) and Merz and Freisleben (1999). Krasnogor and Smith (2008) resorted to polynomial local search theory to show the impact of various combinations of local search and genetic operators. The chapter provides a worst case analyzing for the complexity of heuristics applied to a well-known problem, the two-dimensional and Euclidean TSP. By way of simple arguments, the paper shows the PLS-completeness of a family of memetic algorithms for the TSP as well as graph partitioning and maximum network flow (both with important practical applications). It was shown that potentially very long paths to local optima exists even when the neighborhood used by some MAs are of polynomial size. Interestingly, the chapter’s arguments are also valid for a variable-neighborhood search, rather than evolutionary based, template pattern for global search. A similar result is also provided in Sudholt (2007) for MinCut, MaxSAT, and Knapsack problems.
- *Examples:* intelligent refinement strategy patterns can be seen in Jakob (2006) and Neri et al. (2007a).

2.2.2 Exact and Approximate Hybridization

The previous section presented a refinement strategy pattern that dealt with the general question of how to design a navigable fitness landscape for memetic algorithms through a

judicious selection of refinement strategies. One complementary and related design issue is that of the hybridization of evolutionary algorithms with exact and approximate methods for which a new strategy pattern is presented.

Name: Exact and Approximate Hybridization Strategy Pattern (EAHSP) [4]

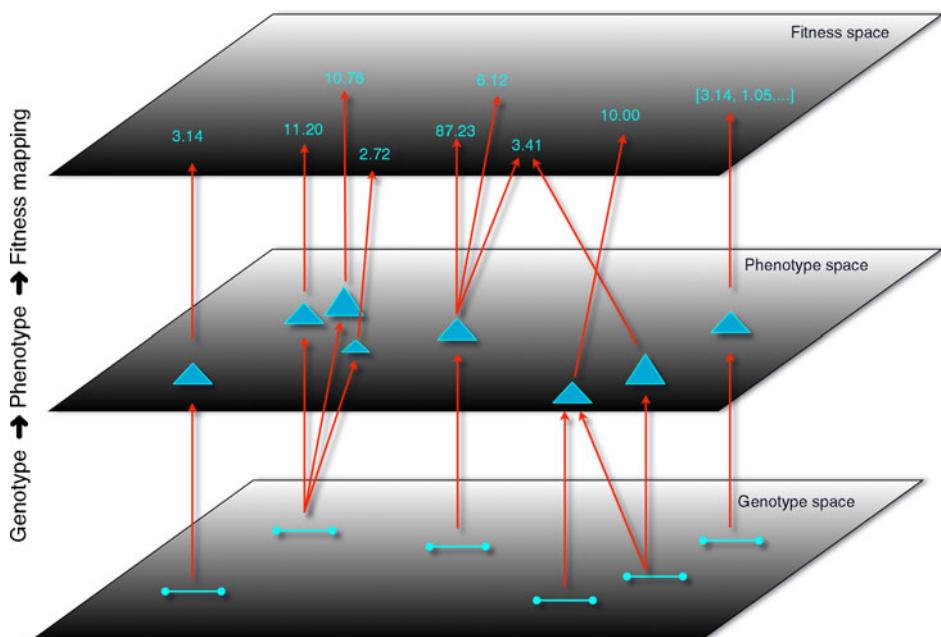
- *Problem statement:* The integration of exact and approximation method into an evolutionary memetic algorithm (EMA) pattern requires the consideration of specific design issues that are distinct than those of other (e.g., heuristic) refinement strategies. More specifically, the development of exact methods requires considerably more effort than that of a set of relatively simple local searchers. The majority of the effort goes into obtaining a tight formulation of the problem for the exact method or a suitable relaxation. Moreover, exact and approximation methods might take a substantial amount of CPU effort to run, sometimes dwarfing that is required by the EMA. Thus, how to synergistically exploit both approaches for obtaining a better solver remains a critical problem.
- *The solution:* It has been argued that there are essentially two main ways of integrating exact and EMA methods. The first approach is based on running the EMA and the exact method in tandem, that is, the EMA provides solutions to the exact method (for example, a branch and bound method) that could, in turn, use the evolved solutions as good bounds, thus helping reduce the branch and bound runtime. Symmetrically, the exact method might be used to seed EMA populations with good candidate solutions. In this loose integration, each algorithm runs essentially independent from each other but exchanging information from time to time. On the second implementation strategy the branch and bound calls the EMA from time to time and thus uses the EMA as a heuristic to produce bounds. It might call it only once as to provide initial solutions or many times over its execution, thus forming a more tightly coupled pipeline. Gallardo et al. (2007) calls these two strategies coercive and cooperative, respectively. Sometimes, when the problem is excessively large, for example, it does not fit in memory and there is no possibility of a distributed computation. Branch and bound is heuristically modified and converted into a beam search. Although this approximation loses performance, it does allow for much larger problems to be tackled (Gallardo et al. 2007). In Pirkwieser et al. (2008), the authors interleave the execution of an EMA with a Lagrangian relaxation method based on a loose coupling strategy.
- *The consequences:* the application of this strategy pattern requires a detailed modeling of the problem at hand and a realistic evaluation of the exact method and memetic algorithm relative runtimes. It should be clear that this strategy should be used if the added benefit of guaranteed results outweigh the additional efforts both in engineering the system and in running it.
- *Examples:* Mezmaz et al. (2007) explores the application of both tandem and pipeline strategies for the bi-objective flowshop scheduling problem. Raidl and Puchinger (2008) provide a detailed analysis of how to combine integer and linear programming methods with metaheuristics. Many of the strategies reported are directly applicable to EMAP.

2.2.3 Population Diversity Handling Strategies

A key characteristic of memetic algorithms is that, they combine search through a population of solutions with search focused around specific promising ones. The introduction of

Fig. 5

The mapping from genotype (i.e., solution encoding) to phenotype (i.e., encoding interpretation) to fitness (i.e., a concrete measure of a solution's worth) can be very complex.



refinement strategies to improve over a subset of solutions often produces additional selection that, in turn, brings a premature population diversity crisis. A population's lack of diversity causes the MA to allocate repeated reproductive, mutation, and refinement trials to the same individuals, thus wasting precious CPU resources. Balancing population diversity is a critical issue, that in itself necessitates a detailed analysis of what is meant by "diversity." The complexity of this task is represented in [Fig. 5](#). Only in the most idealized problem, the mapping from genotypes to phenotypes and then to fitness values is a one-to-one direct mapping. In real-world applications these mappings are highly complex. In some cases, the interpretation of a given solution representation (i.e., genotype) into a phenotype (i.e., actual solution) is nonlinear and/or stochastic and results in potentially different phenotypes for the same genotype. Similarly, fitness assignment to phenotypes might also be nonlinear and/or stochastic thus producing, for a given phenotype, potentially different fitness values. The existence of nonlinearities and stochasticity in the genotype → phenotype → fitness mappings gets compounded when multi-objective problems are tackled. Hence, the definition of population diversity used by diversity strategies must take these complex mappings into account.

Name: Population Diversity Handling Strategy Pattern (PDHSP) [5]

- Problem statement:* the goal of this pattern is to provide interchangeable strategies for the robust management of population diversity within memetic algorithms.
- The solution:* sustaining population diversity in memetic algorithms is a difficult task. It is usually tackled by a combination of strategies operating at various levels. First,

the population is usually initialized in an intelligent manner as to avoid resampling and nonrepresentative coverage of the search space at the beginning of the search. Examples of initialization mechanisms can be found in Burke et al. (1998) and Kretwski (2008). Later, during the search, population diversity maintenance can be implemented at various levels. For example Krasnogor et al. (2002) uses a memory of solution features during the mating process as to avoid generating phenotypes with over-represented features. In this work population diversity is controlled at the phenotype level. An example of a diversity preservation strategy at the fitness level is adopted by Kononova et al. (2007), where the selection of which local search to use is guided by the range of fitness in the population. Similarly, Krasnogor and Smith (2000) uses an adaptation in the local searcher to monitor and affect fitness distributions. In the context of AIS-based templates for memetic algorithms, the aging operator eliminates the oldest candidate solutions from a population so as to contribute to maintaining diversity and thus avoiding (poor) local optima. AIS and differential evolution (DE)-based solvers for multidimensional problems, where diversity handling and refinement strategies are compared, are reported in Cutello et al. (2007). Keeping track of the “age” of the solutions is quite an ubiquitous strategy used on several multi-solution templates. Sorensen and Sevaux (2006) propose several population management strategies based on measuring diversity in the solution space for multidimensional knapsack problem and weighted tardiness single-machine scheduling problems. For continuous problems, diversity is sometimes preserved through structuring the population using a cellular memetic algorithm (Nguyen et al. 2007).

- *The consequences:* Care should be taken in designing the diversity measure to use. In Burke et al. (2002, 2003) a variety of diversity measures for problems have been analyzed where mappings such as those depicted in Fig. 5 take place. Although the examples used in this chapter are based on genetic programming, the lessons learnt are universal for memetic algorithms, namely, (1) increased diversity does not always positively correlate with improved performance and (2) whether it leads to improved performance depends on the complex mapping between genotypes, phenotypes, and fitness. Thus population diversity handling strategies must be implemented in such a way that they should be easy to change and benchmark as to continuously assess their performance in specific problems.
- *Examples:* Neri et al. present a comparative study of several fitness diversity-based adaptation schemes for multimeme algorithms in Neri et al. (2007b). Landa Silva and Burke (2004) address the issue of the interplay between diversity and multi-objective optimization.

2.2.4 Surrogate Strategies

Evolutionary memetic algorithms are usually used to solve very complex and hard problems. Oftentimes, these problems also give rise to uncertainties in the assignment of the fitness values of individuals upon which selection can be applied. On the other hand, refinement strategies used within memetic algorithms are usually computationally expensive, there is no explicit knowledge of a fitness function (e.g., in interactive evolutionary design problems) and hence fitness must be “reverse-engineered.” Moreover, in some cases the objective function is extremely multi-modal and a smoothing criteria is required. The above considerations lead directly to an important design issue in memetic algorithms engineering, namely, the use of surrogate objective functions.

Name: Surrogate Objective Function Strategy Pattern (SOFSP) [6]

- *Problem statement:* Memetic algorithms are computationally intensive methods and one of the most important computational bottlenecks that must be considered when engineering an MA in the calculation of the objective (i.e., fitness) function. As shown in [Fig. 5](#), many problems are inherently noisy, that is, the same candidate solution might be assigned a different fitness value due to stochasticity or nonlinearities in their evaluation. In other cases, several promising solutions might be repeatedly improved by a refinement strategy pattern and this refinement, almost invariably, leads to a large number of additional objective function's evaluations. In optimal design problems, one often finds that no explicit knowledge of fitness is available and the quality of a solution must be inferred from samples given by experts. The aim of the surrogate objective function strategy pattern is to provide effective ways of replacing an expensive, noisy, or unknown fitness function with a suitably defined approximation. The goal is to ameliorate the cost of highly expensive fitness function evaluations while preserving fitness assignment quality by providing a “proxy” to the original objective function that is faster than the former, but of sufficient quality so that it can provide good quality estimations of the true fitness value of a candidate solution.
- *The solution:* Surrogate fitness functions have also been called metamodels, local models, and partial objective functions in the literature. Although the use of effective strategies for solving the surrogacy problem are also important to other branches of natural computation, it is particularly poignant in memetic algorithm's design. This is so because MAs should be able to cope with uncertainty in the genotype → phenotype → fitness mappings, which could lead to a noisy fitness assignment and ambiguity in the very definition of fitness but also MAs must contend with and balance the additional computational effort introduced by the refinement strategies. Explicit averaging has been used to ascertain the fitness of candidate solutions under uncertainty. This is exemplified by the reduction of variance techniques (e.g., latin hypercube sampling), reevaluation of distinct individuals (e.g., those with high fitness or variance), weighted histories (Branke [1998, 2001, 2002](#)), etc. Fitness inheritance (Llora et al. [2007](#)), artificial neural networks and related approaches (Ong et al. [2008](#)), models (Bull [1999](#)), and metamodels (Bhattacharya [2007](#)) as well as design of experiments (Sacks et al. [1989](#)) have also been used for surrogate fitness implementation. In Jin ([2005](#)) and Paenke and Jin ([2006](#)) the authors present a detailed analysis of surrogate methods for evolutionary computation.
- *The consequences:* The use of a surrogate objective function strategy is, in many real-world scenarios, unavoidable. However, its use brings forth a series of other design decisions the memetic algorithms engineer will need to take into account, for example, at what level will the approximation be more productive? the fitness level or the problem itself? Is it possible to use global models or should a collection of local approximations be preferred, etc.
- *Examples:* Zhou et al. ([2007](#)) provide specific recommendations for integrating approximate fitness models within memetic algorithms.

2.2.5 Continuous Problems

The utilization of EMATP for continuous problems presents, in addition to the previous design considerations, new opportunities and challenges. This section only focuses on the issue of integrating refinement strategies for continuous optimization within a memetic algorithm

framework. The reader must note that there is a large body of literature on the use of evolutionary computation for continuous domains, which is not dealt with here. Unlike combinatorial optimization, where it is always possible to detect when a given point is a local optimum, in the continuous case it is, in general, not possible and hence one must settle for an a priori decision on what solution precisions are acceptable, or provide a decision-making mechanism to the MA itself. This becomes even more critical when facing continuous design optimization problems with multiple optimality criteria (e.g., Siepmann et al. 2007). The following section describes a refinement strategy pattern for continuous problems that addresses some of the design issues reported in Hart et al. (2004).

Name: Continuous Problems Refinement Strategy Pattern (CPRSP) [7]

- *Problem statement:* Effective search requires an appropriate determination of search scales, which in continuous optimization is not always possible, for example, it is often impossible to determine, given a candidate solution, whether it represents a local optimum or not. The lack of explicit local scale might also result on very long searches, that when combined with lack of gradient information pose a very difficult problem for optimization techniques. Thus, whether one uses as a refinement strategy, a derivative-free method such as, for example, Nelder–Mead simplex (Nelder and Mead 1965) or Lagrangian interpolation (Berrut and Trefethen 2004), or methods that use first or second order information, for example, Gram–Schmidt orthogonalization or Broyden–Fletcher–Goldfarb–Shanno (Jongen et al. 2004; Hoshino 1971); in general, it is not advisable to rely on detecting a local optimum to stop a local search within a continuous problem.
- *The solution:* Schwefel (1993) surveys several continuous optimization methods, some of which use derivatives information and some which do not. Any of these methods can be integrated within an EMATP through a judicious balance of the amount of effort allocated to the local searchers. As in general it is not possible to run these standard methods all the way to stationary points or local optima, one relies on either truncated searches or a selective application of the local continuous optimization technique to some of the potential solutions in the population. Local search frequency and intensity parameters defining the proportion of individuals in the population that will undergo refinement and how much effort each one will be allocated are introduced. As mentioned in Nguyen et al. (2007), the values for these parameters and the decision on which of the above-mentioned refinement methods are to be used are very much problem dependent. As it is the case for discrete optimization problems, MAs applied to continuous domains also, if improperly engineered, might suffer from premature convergence through a rapid loss of diversity in the population. The use of infrequent or incomplete searches contributes to maintaining diversity but it has been shown, for example, Quang et al. (2009), that a lattice-type population structure promotes a more diverse search.
- *The consequences:* The difficulty of a priori deciding step sizes, local search probability, refinement strategy, etc., for each new problem suggests that, unless one knows beforehand the problem's characteristic features, an adaptive mechanism must be used to decide on the fly on the various choices available. Similarly to the combinatorial case (Krasnogor et al. 2002), Ong and Keane propose to use multimeme algorithms for continuous optimization problems (Ong and Keane 2004).
- *Examples:* Other examples on intelligent hybridization strategies for continuous problems could be seen in Lozano et al. (2004) and Molina et al. (2008). In Hart (2003, 2005) it is argued that many of the difficulties in guaranteeing an efficient combination of

local-global search resulting in adequate convergence properties might be tackled by discretizing the decision space. These papers provide both theoretical as well as practical guidance on how to achieve this.

2.2.6 Multimeme Strategy Pattern

In previous sections, it was argued that, for the evolutionary memetic algorithm Template Pattern to be of practical use, it should be further decorated with a series of strategy patterns. Most of these strategy patterns are conceptually linked to considerations about MA's design issues. Chief among these issues is the selection for a given problem instance, at a given point in time during the search process and for a given potential solution (out of a potentially large population), which refinement strategy to use. Deciding which refinement strategy to use can be further extended to decide not only among heuristic methods but also among approximate and exact methods in both the discrete and continuous case. The following strategy pattern addresses this design issue.

Name: Multimeme Strategy Pattern (MSP) [8]

- *Problem statement:* Deciding which type of refinement or complementary search strategy to use at any given time for a given problem instance and a set of candidate solutions is a far from trivial matter. This is particularly difficult in the absence of abundant statistical information on an algorithm's behavior in relation to problem instances' features.
- *The solution:* Rather than deciding *a priori* on a particular search strategy (meme) by which to complement an EMATP, one can incorporate adaptive or self-adaptive techniques that would allow the MA to dynamically change the refinement strategy accordingly to how search is progressing. These methods have been called multimeme algorithms. Several relatively simple schemes have been shown to be extremely successful both for discrete and continuous optimization. For example, Krasnogor and Smith (2001) Krasnogor et al. (2002), Krasnogor and Pelta (2002), Carr et al. (2002), and Krasnogor (2004a) assign to each available local search method-meme a label which, during crossover, is inherited by an offspring from the fittest parent. Thus, the most effective local search methods are adaptively used during the search process and the fittest problem-solving memes survive over generations. As to maintain local search diversity, the mutation operator can also override an inherited local search label and assign a new one, thus ensuring that search is also done in the heuristic (memetic) space. Similarly, for continuous optimization, Ong and Keane (2004) present multimeme algorithms that employ on-line reinforcement performance information to adaptively change the memes used. Also, a roulette wheel decision mechanism can be exploited for sampling memetic space, that is, stochastically selecting (with bias) from the set of available local searchers the one to use next. An extensive discussion on how to "schedule" various local searchers-memes within an EMATP, and the levels at which search performance information can be incorporated into the decision making process behind the selection of each refinement strategy, is available in Krasnogor and Smith (2005, 2008).
- *The consequences:* The on-line adaptation of the local search choice requires some kind of bookkeeping mechanism as to ascertain the usefulness of the various refinement memes during the search. Besides the simple mechanisms for scheduling multimeme algorithms

appearing in the references mentioned above, Smith (2007) discusses other ways of assigning credit to a potentially varying set of local searchers. In principle, any reinforcement-learning (Kaelbling et al. 1996) type of mechanism could be used. A related family of algorithms, called hyperheuristics (Burke et al. 2007c; Dowsland et al. 2007), has been proposed as an attempt to raise the level of generality in problem solving by intelligently managing the application of collections of “low level” heuristics. The learning mechanisms behind hyper-heuristics could also be used within multimeme algorithms (and viceversa).

- *Examples:* Other examples of multimeme strategies appear in Jakob (2006) and Neri et al. (2007a). Multimeme strategies are closely related to adaptive and self-adaptive methods, hence the reader should also refer to the vast literature on the subject. Some pointers are Smith and Smuda (1995), Smith (2001, 2007), and Landa-Silva and Le (2008).

2.2.7 Self-Generating Strategies

Before describing a strategy pattern for self-generating strategies, the etymology of “memetic” in memetic algorithms is revisited and it is argued that there indeed is a promising nature-inspired research direction that could lead to important new algorithmic variants. Memetic theory started as such with the introduction by R. Dawkins of the concept of a meme in Dawkins (1976) and later refined in Dawkins (1982):

- ▶ I think that a new kind of replicator has recently emerged on this very planet. It is staring us in the face. It is still in its infancy, still drifting clumsily about in its primeval soup, but already it is achieving evolutionary change at a rate that leaves the old gene panting far behind. The new soup is the soup of human culture. We need a name for the new replicator, a noun that conveys the idea of a unit of cultural transmission, or a unit of imitation. “Mimeme” comes from a suitable Greek root, but I want a monosyllable that sounds a bit like “gene”. I hope my classicist friends will forgive me if I abbreviate mimeme to meme. If it is any consolation, it could alternatively be thought of as being related to “memory”, or to the French word “même”. It should be pronounced to rhyme with “cream”. Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation.

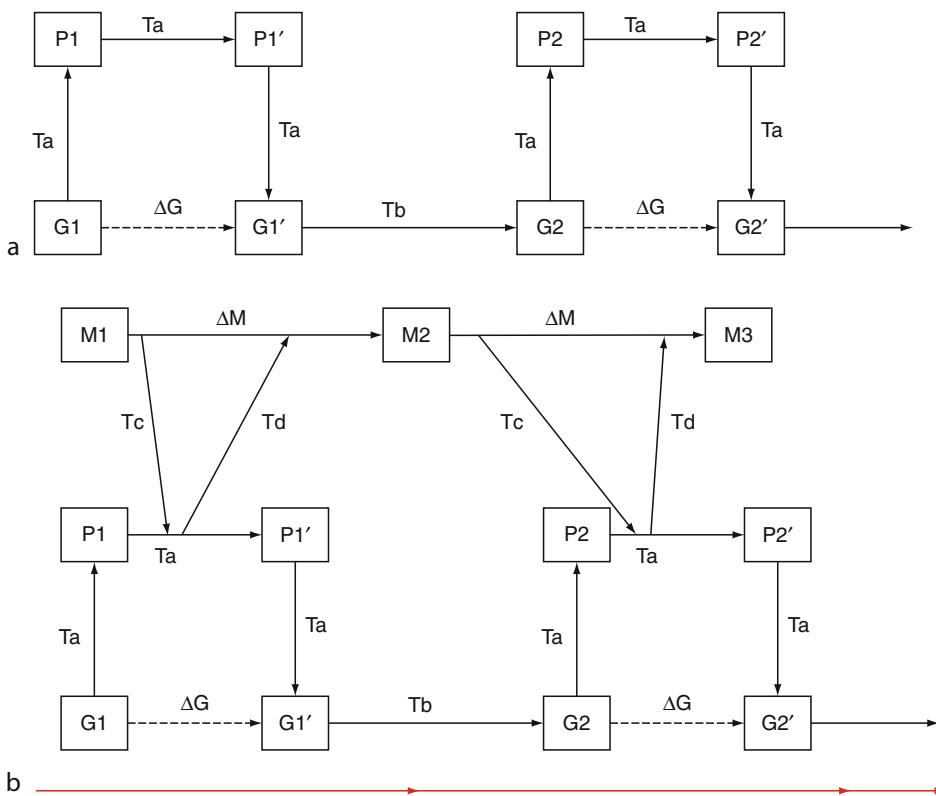
The fact that cultural phenomena could, in certain cases, be modeled or understood as some sort of evolutionary process was not a new idea, for example, the elementary unit of cultural change and/or transmission was sometimes called *m-culture* and *i-culture* (Cloak 1975), *culture-type* (Richerson and Boyd 1978), etc. Durham (1991) provides quite a compelling case for the coevolution of genes and culture in *H. Sapiens*. The fundamental innovation of memetic theory is the recognition that a dual system of inheritance, by means of the existence of two distinct replicators, moulds human culture. Moreover, these two replicators interact and co-evolve shaping each other’s environment. As a consequence, evolutionary changes at the gene level are expected to influence the second replicator, the memes. Symmetrically, evolutionary changes in the meme pool can have consequences for the genes. From a computer science perspective, this is a very appealing nature-inspired computation paradigm as it defines a possible two-scale learning and adaptation mechanism, one at the level of solutions to problems in exactly the same way in which EAs (or any other population-based

metaheuristic) attempts to solve problems and a second level in which the solving methods themselves evolve, or more generally, change. The distinction between standard EAs and a meme-gene evolutionary process can be seen in [Fig. 6a](#) and [Fig. 6b](#), respectively.

In [Fig. 6a](#) a hypothetical population of individuals is represented at two different points in time, generation 1 (G_1) and at a later generation (G_2). In the lower line, G_i for $i = 1, 2$ represents the distribution of genotypes in the population. In the upper line, P_i represents the distribution of phenotypes at the given time. Transformations, T_A , account for epigenetic phenomena, for example, interactions with the environment, in-migration and out-migrations, individual development, etc., all of them affecting the distribution of phenotypes and producing a change in the distribution of genotypes during this generation. On the other hand, transformations, T_B , account for the Mendelian principles that govern genetic inheritance and transform a distribution of genotypes G'_1 into another one G'_2 . Evolutionary computation endeavors to concentrate on the study and assessment of many different ways the cycle depicted in [Fig. 6a](#) can be implemented. This evolutionary cycle implicitly assumes the existence of only one replicator, namely genes, representing solutions to hard and complex problems. On the other hand, [Fig. 6b](#) reflects a coevolutionary system where two replicators

Fig. 6

In (a) evolutionary genetic cycle (adapted from Durham (1991) p 114), (b) Coevolutionary memetic-genetic cycle (adapted from Durham (1991) p 186).



of a different nature interact. In the context of memetic algorithms, memes represent instructions to self-improve. That is, memes specify sets of rules, programs, heuristics, strategies, behaviors, etc., defining the very essence of the underlying search algorithm. Cavalli-Sforza and Feldman (1981) suggest that memetic-genetic search processes might be driven by:

- ▶ the balance of several evolutionary forces: (1) *mutation*, which is both purposive (innovation) and random (copy error); (2) *transmission*, which is not as inert as in biology [i.e., conveyance may also be horizontal and oblique]; (3) *cultural drift* (sampling fluctuations); (4) *cultural selection* (decisions by individuals); and (5) *natural selection* (the consequences at the level of Darwinian fitness)

⦿ *Figure 6b* shows the type of transformations that are possible under a coevolutionary setting, namely, the usual transformations in terms of ‘genes’ and phenotypes distributions but also meme-phenotypes and memes-memes processes. There are mainly two transformations for memes that are depicted, T_C and T_D . Transformations T_C represents the various ways in which “cultural” instructions can reshape phenotypes distributions, for example, individuals learn, adopt, or imitate certain memes or modify other memes. T_D , on the other hand, reflects the changes in memetic distribution that can be expected from changes in phenotypic distributions, for example, those attributed to teaching, preaching, etc. Thus, Dawkin’s memes, and their extensions (Cavalli-Sforza and Feldman 1981; Durham 1991; Gabora 1993; Blackmore 1999) as evolvable search strategies (Krasnogor 1999, 2002, 2004b; Krasnogor and Gustafson 2004; Smith 2003; Burke et al. 2007a) provide a critical link to the possibility of open-ended combinatorial and/or continuous problem solving. Next, a self-generating strategy pattern is illustrated that captures some of the solutions for implementing the above concepts.

Name: Self-Generating Strategy Pattern (SGSP) [9]

- *Problem statement:* The essential problem these strategies tackle can be succinctly put as “how to implement a search mechanism that learns how to search?” or “how to optimize through learning in a reusable manner?” (Please note that this does not refer to the well-known fact that many learning processes can be recast as optimization ones.) This problem statement is not about adaptation or self-adaptation mechanisms that mainly deal with parameter learning within fixed algorithmic templates, but rather how to infer, on the fly, new algorithmic templates that could be useful at a later time.
- *The solution:* Needless to say, this is one of the more recent cutting-edge developments in MA and the proposed solutions are still in their infancy. Self-generating MAs (sometimes called coevolutionary MAs) were first proposed as coevolutionary MAs in Krasnogor (1999) and these first ideas were explained in some more detail in Krasnogor (2002). The first experimental confirmation of the potential behind these strategies came with Krasnogor and Gustafson (2002), Smith (2002a, b), Krasnogor and Gustafson (2003, 2004), and Krasnogor (2004b). The key concept behind these new algorithms was the capturing of algorithmic building blocks through a formal grammar. Through a process not unlike grammatical evolution (O’Neill and Ryan 2003), sentences in the language induced by the grammar were evolved. These evolved sentences represented entire search methods that were self-generated on the fly while attempting to solve a given problem. The above algorithms were used for a range of difficult problems such as polynomial time solvable with low epistasis NK landscapes, polynomial time solvable with high epistasis NK landscapes all the way up to NP hard NK landscapes with both low and high epistasis as well as in randomly generated benchmarks, simplified models of proteins structure prediction and on a real-world bioinformatics application, namely, protein structure comparison.

- *The consequences:* It is evident that a self-generating strategy can only be used in a setting where the problem is either sufficiently hard, that it is worth paying the additional cost of searching an algorithmic design space or for which no good heuristics are available and one is interested in evolving them. Interestingly, after one has evolved a set of refinement strategies, these can be readily used (without need to rediscover them) in new instances or problems, under a multimeme strategy setting. Thus, the self-generating mechanisms described in the papers mentioned above provide a concrete implementation for the nature-inspired gene-meme coevolutionary process depicted in [Fig. 6b](#).
- *Examples:* Recent papers exploiting GP-type learning processes for evolving problem solvers are Geiger et al. (2006), Burke et al. (2006, 2007a, b), Pappa and Freitas (2007), Bader-El-Din and Poli (2007, 2008), Fukunaga (2008), Tabacman et al. (2008), and Tay and Ho (2008). It is important to note that from this set of examples, only Fukunaga (2008) evolve algorithms, the others evolve rules or evaluation functions that guide other heuristics. Thus Krasnogor and Gustafson (2002, 2003, 2004), Smith (2002a, b), Krasnogor (2004b), and Fukunaga (2008) are the closest examples available of a coupled optimization-learning process such as the one depicted in [Fig. 6b](#).

3 A Pragmatic Guide to Fitting It All Together

In previous sections, the core nine terms defining a pattern language for memetic algorithms were described. From an analysis of the literature and software available, the following patterns define our language.

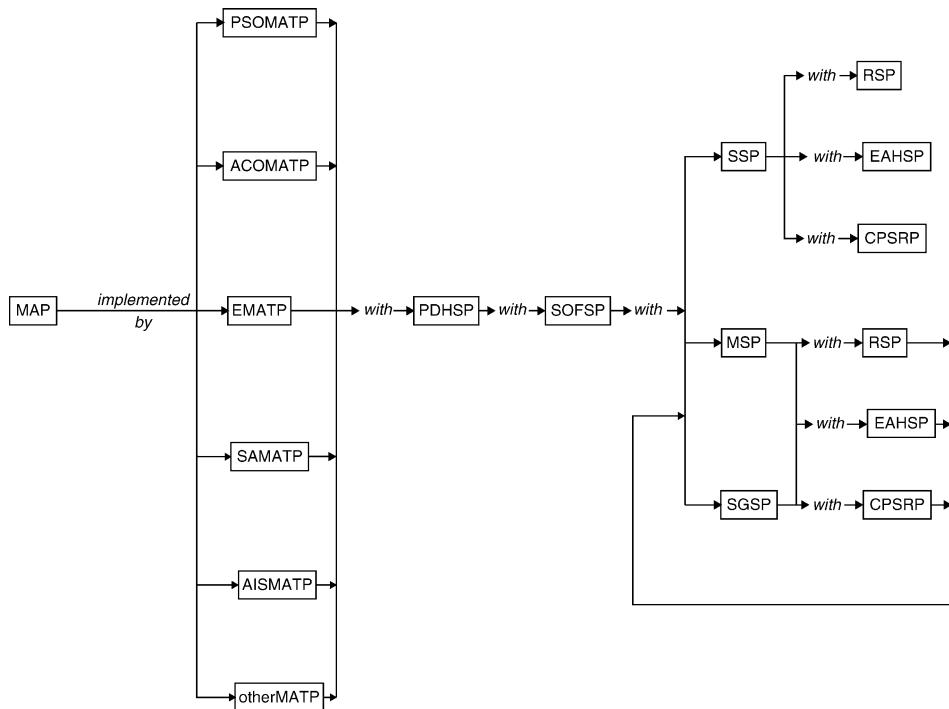
At the top of the conceptual hierarchy, the memetic algorithm pattern (MAP) appears. Its aim is to provide organizational principles for effective global-local search on hard problems. MAP can be implemented through a number of available (successful) templates some of which are based on particle swarm optimization, ant colony optimization, evolutionary algorithms, etc. Each of these gives rise to new terms in the language, namely, the EMATP (for the evolutionary memetic algorithm algorithmic template), the ACOMATP (for the ant colony memetic algorithm template), the SAMATP (for the simulated annealing memetic algorithm template), etc. Each of these nature-inspired paradigm templates have their own “idiosyncrasies.” However, at the time of implementing them as memetic algorithms, the following common features are captured in new design patterns, namely, the refinement strategy pattern (RSP), with focus on heuristic and local search methods, the exact and approximate hybridization strategy pattern (EAHSP) that defines ways in which expensive exact (or approximate) algorithms are integrated with a memetic algorithms template pattern under any nature-inspired realization. Two other terms in our pattern language are the population diversity handling strategy pattern (PDHSP), dealing with ways to preserve – in the face of aggressive refinement strategies – global diversity and the surrogate objective function strategy pattern (SOFSP) whose role is to define methods for dealing within a memetic algorithm with very expensive/noisy/undetermined objective functions. Finally, the pattern language presented also includes terms for defining the “generation” of memetic algorithm one is trying to implement. A memetic algorithm of the first generation in which only one refinement strategy is used, is called a simple strategy pattern (SSP) and that is the canonical MAs (e.g., see [Fig. 3](#)). The multimeme strategy pattern (MSP) and the self-generating strategy pattern (SGSP), which provide methods for utilizing several refinement strategies simultaneously and

for synthesizing new refinement strategies on the fly respectively, are second and third generation MAs. Thus the resulting pattern language has at least the following terms: {MAP, EMATP, PSOMATP, ACOMATP, SAMATP, . . . , AISMATP, RSP, EASHP, CPRSP, PDHSP, SOFSP, SSP, MSP, SGSP}.

These patterns are functionally related as depicted in [Fig. 7](#) that represents the series of design decisions involved in the implementation of memetic algorithms. One could choose to implement an MA by following, for example, an ant colony optimization template or – more often – an evolutionary algorithm template or any of the other template patterns. Each one of these nature-inspired template patterns will have their own algorithmic peculiarities, for example, crossovers and mutations for EAs, pheromones updating rules for ACOs, etc., but all of them when taken as a memetic algorithm will need to address the issues of population diversity, refinement strategies, exact algorithms, surrogate objective functions, single meme versus multimeme versus self-generation, etc. In order to instantiate code for any of these design patterns, one can simply look at the description of the design pattern provided in this chapter and refer to any of the literature references given within the pattern description. Thus, [Fig. 7](#) provides a reference handbook for MAs engineering.

Fig. 7

Integrative view of the design patterns for the memetic algorithms patterns language. A path through the graph represents a series of design decision that an algorithms engineer would need to take while implementing an MA. Each design pattern provides solutions to specific problems the pattern addresses. By indexing through the patterns' acronyms into the main text in this chapter, the reader can have access to examples from the literature where the issues have been solved to satisfaction.



4 Brief Notes on Theoretical Aspects

Memetic algorithms' notorious successes in practical applications notwithstanding, no systematic effort has been made to try to understand them from a theoretical viewpoint. Memetic algorithms do not adhere to pure EA theory, not even those applied to continuous problems for which convergence analysis are somewhat easier. Moreover, MAs are also much more complex than other heuristic methods such as greedy-like algorithms for which precise theoretical knowledge is possible. Thus, while the body of empirical knowledge is steadily growing, their theoretical underpinnings remain poorly understood. A collection of disjoint, but only partial, analyses were published in, e.g., Krasnogor and Smith (2005), Sudholt (2007) for MAs applied to combinatorial optimization and Hart (2003, 2005) for continuous optimization. Two emerging methodologies to assess heuristic performance that might in the future lead to important insights for MAs are PLS-theory (Johnson et al. 1988; Yannakakis 1997) and Domination analysis (Glover and Punnen 1997; Gutin and Yeo 2006). Recent results based on both of these theoretical approaches to heuristic analysis are compiled in Michiels et al. (2007). For an organizational perspective of the field, the reader is also referred to the papers by Krasnogor and Smith (2005) and Ong et al. (2006) in which several best practices are described and research lines proposed.

5 The Future: Toward Nature-Inspired Living Software Systems

Memetic algorithms have gone through three major transitions in their evolution. The first transition was coincidental with their introduction and the bulk of the research and applications produced was based on the use of, mainly, an evolutionary algorithm with one specialized local searcher. The second transition brought the concept of multimeme algorithms that expanded the repertoire of both the key skeleton, that is, EAs were no longer the only global search template used but rather any global-(population) based search could be employed, and gave way to the constraint on having a single local searcher. The third major transition is taking place right now and involves the incorporation of explicit learning mechanisms that can, while optimizing, "distill" new refinement operators. These new operators, in turn, can be reused in a later application of the MA, thus amortizing the cost invested in discovering them. This third transition is also producing more frequent and more confident use of exact methods in tandem or hybridized with MAs. This third major transition is likely to produce very exciting results over the next few years and indeed much research remains to be done in the theory and practice of memetic algorithms.

The question that needs to be briefly addressed is "what's next?" "where do memetic algorithms go from here?"

Memetic algorithms in particular and optimization algorithms in general are but one class of the millions of different software systems humans create to solve problems. The vast majority – if not all – of software systems development follows, to various degrees of formalism and details, a process of requirement collection, specification, design, implementation, testing, deployment, and long term maintenance. There are, indeed, a variety of software engineering techniques that advocate the concurrent realization or elimination altogether of (some of) these steps as a way to achieve a more agile software development. The unrelenting scaling up of available computational, storage, and communication power is

bringing closer the day when software will cease to be created following such a process and will, instead, be “planted/seeded” within a production environment (e.g., a factory, university, bank, etc.) and then “grown” from the bottom-up into a fully developed functional software system. Although within the remit of optimization problems, self-generating techniques such as Krasnogor and Gustafson (2002), Smith (2002a, b), Krasnogor and Gustafson (2003, 2004), Krasnogor (2004b), Burke et al. (2006, 2007a, b), Pappa and Freitas (2007), Bader-El-Den and Poli (2008), Fukunaga (2008), and Tabacman et al. (2008) are moving incrementally in this direction, perhaps an even more radical scenario can be imagined. Consider, for example, a given production environment, a factory, where a series of optimization problems must be solved routinely, such as personnel rostering (Π_{pr}), job-shop schedules (Π_{jss}), path planning (Π_{pp}), inventory management (Π_{in}), etc. Each of these problems, during long periods of time (weeks, months, years, and even decades) give rise to large collections of problem instances (I_{Π_j} with $j \in \{\Pi_{pr}, \Pi_{jss}, \Pi_{pp}, \Pi_{in}\}$). Current practice dictates that for each Π , a tailor-made solver is used that in practice works well for the associated $I(\Pi)$. A key observation worth making is that the distribution of instances for each of the problems arising from a given factory (or bank, university, supermarket, airport, etc.) is not random but rather strongly dependent on physical, legal, commercial, and societal constraints operating over the factory. That is, the instances derived from each one of these problems, for a given factory, will have strong common features and similarities. This, in turn, provides a tremendous opportunity for a more organic and autonomic problem solving. What we have in mind can best be described as a new discipline of “pluripotential problem solvers (PPS).” The idea behind pluripotential problem solvers is that a minimum set of self-generating features would be hardcoded into a given pluripotential solver cell (PSC), then this software cell would be immersed (in multiple copies) into the factory environment and bombarded with problem instances over a period of time. After a while, some of the PSC would differentiate along problem instance lines (very much as a stem cell differentiates along cell type lines). Once differentiated, a mature PSC (or solver cell for simplicity) would only be sensitive to new instances of the same problem. Furthermore, as time progresses, the solver cell would be able to incorporate new problem-specific and instance-distribution-specific problem solving capabilities (e.g., perhaps using some refinement of the technologies described in the self-generating strategy pattern) thus further specializing. In the end, the set of initially planted PSC would have been developing along very specialized lines within a specific production environment. Needless to say, if one were to transplant a specialized solver cell operating on, let’s say, Π_{jss} in factory 1 as a solver for Π_{jss} in factory 2, the performance of the solver will be limited because, unless the distributions of instances generated in both factories are very similar, the specialized solver cell would not be able to cope with the new ones as it would have lost its capacity to specialize. What one would rather do is seed again in factory 2 a new set of PSC and let them mature within their production environment. The scenario described above is predicated under two key assumptions. The first one is that one does not expect to have a cutting-edge solver “out of the tin” (for which arguably one would be willing to pay considerable amounts of money) but rather than one would pay less (or nothing) for a PSC and be willing to wait until the software grows into a very specialized, niche-specific, cutting-edge solver. The second assumption is that computational power, communication bandwidth, and storage capacity will keep increasing while the price per floating point operation and gigabytes transmitted/stored will go on decreasing. The advent of pluripotential problem solvers will require a concerted, creative, and unorthodox research on the amalgamation of optimization algorithms, data mining, and machine learning as well as more in-depth studies of nature-inspired

principles that – complementing evolutionary approaches – so far have not been tapped for optimization, namely, the creative power of self-assembly and self-organization (Krasnogor et al. 2008).

6 Conclusions

This chapter provided an unorthodox introduction to memetic algorithms. It analyzed memetic algorithms as they appear in the literature and are used in practice, rather than conforming to a top down definition of what a memetic algorithm is and then, based on a definition, prescribe how to implement them. The approach taken here allows for a pragmatic view of the field and provides a recipe for creating memetic algorithms by resorting to a shared pattern language. This emerging pattern language for memetic algorithms serves as a catalog of parts (or concerns) that an algorithms engineer might want to resort to for solving specific design issues arising from the need to interlink global search and local search for hard complex problems. Due to space and time constraints, design patterns for multiobjective (Burke and Landa-Silva 2004; Li and Landa-Silva 2008) problems, parallelization strategies (Alba and Tomassini 2002; Nebro et al. 2007), etc., have not been considered, and how these important new components of the pattern language would interact with the patterns described here is also not discussed. It is hoped that the pattern language for MAs presented in this chapter will, in the future, be expanded and refined by researchers and practitioners alike.

Acknowledgments

The author would like to acknowledge the many friends and colleagues with whom he has collaborated over the years. Their ideas, scientific rigor and enthusiasm for memetic algorithms has been a continuous source of inspiration and challenges. The author would also like to thank Jonathan Blakes and James Smaldon for their valuable comments during the preparation of this paper. The author wishes to acknowledge funding from the EPSRC for projects EP/D061571/1 and EP/C523385/1. Finally, the editors of this book are thanked for giving the author an opportunity to contribute.

References

- Alba E, Tomassini M (2002) Parallelism and evolutionary algorithms. *IEEE Trans Evolut Comput* 6:443–462
- Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl-King I, Angel S (1977) *A pattern language - towns, buildings, construction*. Oxford University Press, New York
- Armour P (2007) The conservation of uncertainty, exploring different units for measuring software. *Commun ACM* 50:25–28
- Bacardit J, Krasnogor N (2009) Performance and efficiency of memetic Pittsburgh learning classifier systems. *Evolut Comput* 17(3)
- Bader-El-Den M, Poli R (2008) Evolving heuristics with genetic programming. In: GECCO '08: Proceedings of the 10th annual conference on genetic and evolutionary computation, ACM, New York, pp 601–602, doi: <http://doi.acm.org/10.1145/1389095.1389212>
- Bader-El-Din MB, Poli R (2007) Generating SAT local-search heuristics using a GP hyper-heuristic framework. In: LNCS 4926. Proceedings of the 8th international conference on artificial evolution, Honolulu, pp 37–49
- Berrut JP, Trefethen L (2004) Barycentric Lagrange interpolation. *SIAM Rev* 46(3):501–517

- Bhattacharya M (2007) Surrogate based EA for expensive optimization problems. In: Proceedings for the IEEE congress on evolutionary computation (CEC), Singapore, pp 3847–3854
- Blackmore S (1999) The meme machine. Oxford University Press, Oxford
- Branke J (1998) Creating robust solutions by means of an evolutionary algorithm. In: Parallel problem solving from nature PPSN V, Amsterdam, pp 119–128
- Branke J (2001) Reducing the sampling variance when searching for robust solutions. In: Spector L, et al. (eds) Proceedings of the genetic and evolutionary computation conference, Kluwer, San Francisco, CA, pp 235–242
- Branke J (2002) Evolutionary Optimization in Dynamic Environments. Kluwer, Boston, MA
- Bull L (1999) On model-based evolutionary computation. *J Soft Comput Fus Found Methodol Appl* 3: 76–82
- Bull L, Holland O, Blackmore S (2000) On meme–gene coevolution. *Artif Life* 6:227–235
- Burke E, Landa-Silva J (2004) The design of memetic algorithms for scheduling and timetabling problems. In: Hart W, Krasnogor N, Smith J (eds) Recent advances in memetic algorithms. Springer, pp 289–312
- Burke E, Newall J, Weare R (1996) A memetic algorithm for university exam timetabling. In: Burke E, Ross P (eds) The practice and theory of automated timetabling, Lecture notes in computer science, vol 1153. Springer, Berlin, pp 241–250
- Burke E, Newall J, Weare R (1998) Initialization strategies and diversity in evolutionary timetabling. *Evolut Comput* 6:81–103
- Burke E, Bykov Y, Newall J, Petrovic S (2004) A time-predefined local search approach to exam timetabling problems. *IIE Trans* 36:509–528
- Burke E, Gustafson S, Kendall G, Krasnogor N (2002) Advanced population diversity measures in genetic programming. In: Guervos JM, Adamidis P, Beyer H, Fernandez-Villacañas J, Schwefel H (eds) 7th International conference parallel problem solving from nature, PPSN, Springer, Granada, Spain, Lecture notes in computer science, vol 2439. Springer, New York, pp 341–350
- Burke E, Gustafson S, Kendall G, Krasnogor N (2003) Is increased diversity beneficial in genetic programming: an analysis of the effects on fitness. In: IEEE congress on evolutionary computation, CEC, IEEE, Canberra, pp 1398–1405
- Burke E, Hyde M, Kendall G (2006) Evolving bin packing heuristics with genetic programming. In: Runarsson T, Beyer HG, Burke E, Merelo-Guervos J, Whitley D, Yao X (eds) Proceedings of the 9th International conference on parallel problem solving from nature (PPSN 2006), LNCS 4193. Springer, pp 860–869
- Burke E, Hyde M, Kendall G, Woodward J (2007a) Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In: Proceedings of the genetic and evolutionary computation conference (GECCO 2007), ACM, London, pp 1559–1565
- Burke E, Hyde M, Kendall G, Woodward J (2007b) Scalability of evolved on line bin packing heuristics. In: Proceedings of the congress on evolutionary computation (CEC 2007). Singapore, pp 2530–2537
- Burke E, McCollum B, Meisels A, Petrovic S, Qu R (2007c) A graph-based hyper-heuristic for timetabling problems. *Eur J Oper Res* 176:177–192
- Caponio A, Cascella G, Neri F, Salvatore N, Sumner M (2007) A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. *IEEE Trans Syst Man Cybern Part B* 37:28–41
- Carr R, Hart W, Krasnogor N, Burke E, Hirst J, Smith J (2002) Alignment of protein structures with a memetic evolutionary algorithm. In: Langdon W, Cantu-Paz E, Mathias K, Roy R, Davis D, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter M, Schultz A, Miller J, Burke E, Jonoska N (eds) GECCO-2002: Proceedings of the genetic and evolutionary computation conference, Morgan Kaufmann, San Mateo, CA
- Cavalli-Sforza L, Feldman M (1981) Cultural transmission and evolution: a quantitative approach. Princeton University Press, Princeton, NJ
- Cheng R, Gen M (1997) Parallel machine scheduling problems using memetic algorithms. *Comput Ind Eng* 33(3–4):761–764
- Cloak F (1975) Is a cultural ethology possible. *Hum Ecol* 3:161–182
- Cooper J (2000) Java design patterns: a tutorial. Addison-Wesley, Boston, MA
- Cordon O, Herrera F, Stutzle T (2002) A review on the ant colony optimization metaheuristic: basis, models and new trends. *Mathware Soft Comput* 9:141–175
- Cutello V, Krasnogor N, Nicosia G, Pavone M (2007) Immune algorithm versus differential evolution: a comparative case study using high dimensional function optimization. In: International conference on adaptive and natural computing algorithms, ICANNGA 2007. LNCS, Springer, Berlin, pp 93–101
- Dawkins R (1976) The selfish gene. Oxford University Press, New York
- Dawkins R (1982) The extended phenotype. Freeman, Oxford
- Dorigo M, Gambardella L (1997) Ant colony system: a cooperative learning approach to the travelling salesman problem. *IEEE Trans Evolut Comput* 1(1): 53–66
- Dowsland K, Soubeiga E, Burke EK (2007) A simulated annealing hyper-heuristic for determining shipper sizes. *Eur J Oper Res* 179:759–774

- Dueck G (1993) New optimisation heuristics. the Great Deluge algorithm and record-to-record travel. *J Comput Phys* 104:86–92
- Duque T, Goldberg D, Sastry K (2008) Improving the efficiency of the extended compact genetic algorithm. In: GECCO '08: Proceedings of the 10th annual conference on genetic and evolutionary computation, ACM, New York, pp 467–468. doi: <http://doi.acm.org/10.1145/1389095.1389181>
- Durham W (1991) Coevolution: genes, culture and human diversity. Stanford University Press, Stanford, CA
- Fleurent C, Ferland J (1997) Genetic and hybrid algorithms for graph coloring. *Ann Oper Res* 63:437–461
- Fukunaga A (2008) Automated discovery of local search heuristics for satisfiability testing. *Evolut Comput* 16(1):31–61, doi: 10.1162/evco.2008.16.1.31, URL <http://www.mitpressjournals.org/doi/abs/10.1162/evco.2008.16.%1.31>, pMID: 18386995, <http://www.mitpressjournals.org/doi/pdf/10.1162/evco.2008.16.1.31>
- Gabora L (1993) Meme and variations: a computational model of cultural evolution. In: L Nadel, Stein D (eds) 1993 Lectures in complex systems. Addison-Wesley, Boston, MA, pp 471–494
- Gallardo J, Cotta C, Fernandez A (2007) On the hybridization of memetic algorithms with branch-and-bound techniques. *Syst Man Cybern Part B IEEE Trans* 37(1):77–83. doi: 10.1109/TSMCB.2006.883266
- Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns, elements of reusable object-oriented software. Addison-Wesley, Reading, MA
- Geiger CD, Uzsoy R, Aytug H (2006) Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach. *J Scheduling* 9(1):7–34
- Glover F, Punnen A (1997) The traveling salesman problem: new solvable cases and linkages with the development of approximation algorithms. *J Oper Res Soc* 48:502–510
- Gutin G, Yeo A (2006) Domination analysis of combinatorial optimization algorithms and problems. In: Graph theory, combinatorics and algorithms, operations research/computer science interfaces, vol 34. Springer, New York, pp 145–171
- Gutin G, Karapetyan D, Krasnogor N (2007) Memetic algorithm for the generalized asymmetric traveling salesman problem. In: Pavone M, Nicosia G, Pelta D, Krasnogor N (eds) Proceedings of the 2007 workshop on nature inspired cooperative strategies for optimisation. Studies in computational intelligence. Springer, Berlin
- Hansen P, Mladenovic N (1998) Variable neighborhood search for the p -median. *Location Sci* 5(4):207–226
- Hansen P, Mladenovic N (2001) Variable neighborhood search: principles and applications. *Eur J Oper Res* 130:449–467
- Hart W (2003) Locally-adaptive and memetic evolutionary pattern search algorithms. *Evolut Comput* 11:29–52
- Hart W (2005) Rethinking the design of real-coded evolutionary algorithms: making discrete choices in continuous search domains. *J Soft Comput Fus Found Methodol Appl* 9:225–235
- Hart W, Krasnogor N, Smith J (2004) Recent advances in memetic algorithms, studies in fuzziness and soft computing, vol 166, Springer, Berlin/Heidelberg/New York, chap Memetic Evolutionary Algorithms, pp 3–27
- Hart WE (1994) Adaptive global optimization with local search. Ph.D. thesis, University of California, San Diego, CA
- He L, Mort N (2000) Hybrid genetic algorithms for telecommunications network back-up routing. *BT Technol J* 18(4):42–50
- Hinton G, Nowlan S (1987) How learning can guide evolution. *Complex Syst* 1:495–502
- Holland JH (1976) Adaptation in natural and artificial systems. The University of Michigan Press, New York
- Hoshino S (1971) On Davies, Swann and Campey minimisation process. *Comput J* 14:426
- Houck C, Joines J, Kay M, Wilson J (1997) Empirical investigation of the benefits of partial lamarckianism. *Evolut Comput* 5(1):31–60
- Ishibuchi H, Kaige S (2004) Implementation of simple multiobjective memetic algorithms and its application to knapsack problems. *Int J Hybrid Intell Syst* 1(1–2):22–35
- Jakob W (2006) Towards an adaptive multimeme algorithm for parameter optimisation suiting the engineers needs. In: Runarsson TP, et al. (eds) Proceedings of the IX parallel problem solving from nature conference (PPSN IX). Lecture notes in computer science 4193. Springer, Berlin, pp 132–141
- Jaszkiewicz A (2002) Genetic local search for multi-objective combinatorial optimization. *Eur J Oper Res* 137
- Jin Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput Fus Found Methodol Appl* 9:3–12
- Johnson D, Papadimitriou C, Yannakakis M (1988) How easy is local search. *J Comput Syst Sci* 37:79–100
- Jongen H, Meer K, Triesch E (2004) Optimization theory. Springer, New York
- Kaelbling L, Littman M, Moore A (1996) Reinforcement learning: a survey. *J Artif Intell Res* 4:237–285
- Kallel L, Naudts B, Reeves C (2001) Properties of fitness functions and search landscapes. In: Kallel L, Naudts B, Rogers A (eds) Theoretical aspects of evolutionary computing. Springer, Berlin, pp 175–206

- Kirkpatrick S, Gelatt C, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220:671–680
- Kononova A, Hughes K, Pourkashian M, Ingham D (2007) Fitness diversity based adaptive memetic algorithm for solving inverse problems of chemical kinetics. In: IEEE Congress on Evolutionary Computation (CEC), IEEE, Singapore, pp 2366–2373
- Krasnogor N (1999) Coevolution of genes and memes in memetic algorithms. In: Wu A (ed) Proceedings of the 1999 genetic and evolutionary computation conference, Graduate students workshop program, San Francisco, CA, <http://www.cs.nott.ac.uk/nxk/PAPERS/memetic.pdf>, (poster)
- Krasnogor N (2002) Studies on the theory and design space of memetic algorithms. Ph.D. thesis, University of the West of England, Bristol, <http://www.cs.nott.ac.uk/nxk/PAPERS/thesis.pdf>
- Krasnogor N (2004a) Recent advances in memetic algorithms, Studies in fuzziness and soft computing, vol 166, Springer, Berlin, Heidelberg New York, chap Towards robust memetic algorithms, pp 185–207
- Krasnogor N (2004b) Self-generating metaheuristics in bioinformatics: the protein structure comparison case. *Genet Programming Evol Mach* 5(2):181–201
- Krasnogor N, Gustafson S (2002) Toward truly “memetic” memetic algorithms: discussion and proof of concepts. In: Corne D, Fogel G, Hart W, Knowles J, Krasnogor N, Roy R, Smith JE, Tiwari A (eds) Advances in nature-inspired computation: the PPSN VII workshops, PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading, UK ISBN 0-9543481-0-9
- Krasnogor N, Gustafson S (2003) The local searcher as a supplier of building blocks in self-generating memetic algorithms. In: Hart JS WE, Krasnogor N (eds) Fourth international workshop on memetic algorithms (WOMA4), In GECCO 2003 workshop proceedings. Chicago, IL
- Krasnogor N, Gustafson S (2004) A study on the use of “self-generation” in memetic algorithms. *Nat Comput* 3(1):53–76
- Krasnogor N, Pelta D (2002) Fuzzy memes in multi-meme algorithms: a fuzzy-evolutionary hybrid. In: Verdegay J (ed) Fuzzy sets based heuristics for optimization. Springer, Berlin
- Krasnogor N, Smith J (2000) A memetic algorithm with self-adaptive local search: TSP as a case study. In: Whitley D, Goldberg D, Cantu-Paz E, Spector L, Parmee I, Beyer HG (eds) GECCO 2000: Proceedings of the 2000 genetic and evolutionary computation conference, Morgan Kaufmann, San Francisco, CA
- Krasnogor N, Smith J (2001) Emergence of profitable search strategies based on a simple inheritance mechanism. In: Spector L, Goodman E, Wu A, Langdon W, Voigt H, Gen M, Sen S, Dorigo M, Pezeshj S, Garzon M, Burke E (eds) GECCO 2001: Proceedings of the 2001 genetic and evolutionary computation conference, Morgan Kaufmann, San Francisco, CA
- Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: Model, taxonomy and design issues. *IEEE Trans Evolut Algorithms* 9(5):474–488
- Krasnogor N, Smith J (2008) Memetic algorithms: the polynomial local search complexity theory perspective. *J Math Model Algorithms* 7:3–24
- Krasnogor N, Blackburne B, Hirst J, Burke E (2002) Multimeme algorithms for protein structure prediction. In: Guervos JM, Adamidis P, Beyer H, Fernandez-Villacanas J, Schwefel H (eds) 7th International conference parallel problem solving from nature, PPSN, Springer, Berlin/Heidelberg, Granada, Spain, Lecture notes in computer science, vol 2439. Springer, pp 769–778
- Krasnogor N, Gustafson S, Pelta D, Verdegay J (eds) (2008) Systems self-assembly: multidisciplinary snapshots, Studies in multidisciplinarity, vol 5. Elsevier, Spain
- Kretwski M (2008) A memetic algorithm for global induction of decision trees. In: Proceedings of SOFSEM: theory and practice of computer science. Lecture notes in computer science, Springer, New York, pp 531–540
- Kuhn T (1962) The structure of scientific revolution. University of Chicago Press, Chicago, IL
- Landa-Silva D, Le KN (2008) A simple evolutionary algorithm with self-adaptation for multi-objective optimisation. Springer, Berlin, pp 133–155
- Landa-Silva J, Burke EK (2004) Using diversity to guide the search in multi-objective optimization. World Scientific, Singapore, pp 727–751
- Lee Z, Lee C (2005) A hybrid search algorithm with heuristics for resource allocation problem. *Inf Sci* 173:155–167
- Li H, Landa-Silva D (2008) Evolutionary multi-objective simulated annealing with adaptive and competitive search direction. In: Proceedings of the 2008 IEEE congress on evolutionary computation (CEC 2008). IEEE Press, Piscataway, NJ, pp 3310–3317
- Liu BF, Chen HM, Chen JH, Hwang SF, Ho SY (2005) Meswarm: memetic particle swarm optimization. In: GECCO ’05: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM, New York, pp 267–268. doi: <http://doi.acm.org/10.1145/1068009.1068049>
- Liu D, Tan KC, Goh CK, Ho WK (2007) A multiobjective memetic algorithm based on particle swarm optimization. *Syst Man Cybern Part B IEEE Trans* 37(1):42–50. doi: 10.1109/TSMCB.2006.883270
- Llora X, Sastry K, Yu T, Goldberg D (2007) Do not match, inherit: fitness surrogates for genetics-based

- machine learning techniques. In: Proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, San Mateo, CA, pp 1798–1805
- Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. *Evolut Comput* 12(3):273–302
- Mayley G (1996) Landscapes, learning costs and genetic assimilation. *Evolut Comput* 4(3):213–234
- McCulloch W, Pitts W (1943) A logical calculus of the ideas immanent in nervous system. *Bull Math Biophys* 5:115–133
- Merz P (2003) The compact memetic algorithm. In: Proceedings of the IV International workshop on memetic algorithms (WOMA IV). GECCO 2003, Chicago, IL. <http://w210.ub.uni-tuebingen.de/portal/woma4/>
- Merz P, Freisleben B (1999) Fitness landscapes and memetic algorithm design. In: New ideas in optimization, McGraw-Hill, Maidenhead, pp 245–260
- Mezmaz M, Melab N, Talbi EG (2007) Combining meta-heuristics and exact methods for solving exactly multi-objective problems on the grid. *J Math Model Algorithms* 6:393–409
- Michiels W, Aarts E, Korst J (2007) Theoretical aspects of local search. Monographs in theoretical computer science. Springer, New York
- Molina D, Lozano M, Garcia-Martinez C, Herrera F (2008) Memetic algorithm for intense local search methods using local search chains. In: Hybrid meta-heuristics: 5th international workshop. Lecture notes in computer science. Springer, Berlin/Heidelberg/New York, pp 58–71
- Morris GM, Goodsell DS, Halliday RS, Huey R, Hart WE, Belew RK, Olson AJ (1998) Automated docking using a lamarkian genetic algorithm and an empirical binding free energy function. *J Comp Chem* 14:1639–1662
- Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, CA
- Nebro A, Alba E, Luna F (2007) Multi-objective optimization using grid computing. *Soft Comput* 11:531–540
- Nelder J, Mead R (1965) A simplex method for function minimization. *Comput J* 7(4):308–313. doi: 10.1093/comjnl/7.4.308
- Neri F, Jari T, Cascella G, Ong Y (2007a) An adaptive multimeme algorithm for designing HIV multidrug therapies. *IEEE/ACM Trans Comput Biol Bioinformatics* 4(2):264–278
- Neri F, Tirronen V, Karkkainen T, Rossi T (2007b) Fitness diversity based adaptation in multimeme algorithms: a comparative study. In: Proceedings of the IEEE congress on evolutionary computation. IEEE, Singapore, pp 2374–2381
- Nguyen QH, Ong YS, Lim MH, Krasnogor N (2007) A comprehensive study on the design issues of memetic algorithm. In: Proceedings of the 2007 IEEE congress on evolutionary computation. IEEE, Singapore, pp 2390–2397
- Niesse J, Mayne H (Sep. 15, 1996) Global geometry optimization of atomic clusters using a modified genetic algorithm in space-fixed coordinates. *J Chem Phys* 105(11):4700–4706
- O'Neill M, Ryan C (2003) Grammatical evolution: evolutionary automatic programming in an arbitrary language. Genetic Programming, vol 4. Springer, Essex
- Ong Y, Keane A (2004) Meta-lamarckian learning in memetic algorithms. *IEEE Trans Evolut Comput* 8:99–110
- Ong Y, Lim M, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. *IEEE Trans Syst Man Cybern Part B* 36:141–152
- Ong Y, Lum K, Nair P (2008) Hybrid evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers. *Comput Opt Appl* 39:97–119
- Paenke I, Jin J (2006) Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *IEEE Trans Evolut Comput* 10: 405–420
- Pappa G, Freitas A (2007) Discovering new rule induction algorithms with grammar-based genetic programming. In: Maimon O, Rokach L (eds) Soft computing for knowledge discovery and data mining. Springer, New York, pp 133–152
- Petalas Y, Parsopoulos K, Vrahatis M (2007) Memetic particle swarm optimisation. *Ann Oper Res* 156:99–127
- Pirkwieser S, Raidl GR, Puchinger J (2008) A Lagrangian decomposition/evolutionary algorithm hybrid for the knapsack constrained maximum spanning tree problem. In: Cotta C, van Hemert J (eds) Recent advances in evolutionary computation for combinatorial optimization. Springer, Valencia, pp 69–85
- Quang Q, Ong Y, Lim M, Krasnogor N (2009) Adaptive cellular memetic algorithm. *Evolut Comput* 17(2): 231–256
- Raidl GR, Puchinger J (2008) Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In: Blum C, et al. (eds) Hybrid Metaheuristics - an emergent approach for combinatorial optimization. Springer, Berlin/Heidelberg/New York, pp 31–62
- Reeves C (1996) Hybrid genetic algorithms for bin-packing and related problems. *Ann Oper Res* 63:371–396

- Richerson P, Boyd R (1978) A dual inheritance model of the human evolutionary process: I. Basic postulates and a simple model. *J Soc Biol Struct* 1:127–154
- Romero-Campero F, Cao H, Camara M, Krasnogor N (2008) Structure and parameter estimation for cell systems biology models. In: Keijzer, M et al. (eds) Proceedings of the genetic and evolutionary computation conference (GECCO-2008), ACM, Seattle, WA, pp 331–338
- Sacks J, Welch W, Mitchell T, Wynn H (1989) Design and analysis of computer experiments. *Stat Sci* 4:409–435
- Schwefel H (1993) Evolution and optimum seeking: the sixth generation. Wiley, New York, NY
- Siepmann P, Martin C, Vancea I, Moriarty P, Krasnogor N (2007) A genetic algorithm approach to probing the evolution of self-organised nanostructured systems. *Nano Lett* 7(7):1985–1990
- Smith J (2001) Modelling GAs with self adaptive mutation rates. In: GECCO-2001: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Francisco, CA
- Smith J (2002a) Co-evolution of memetic algorithms: Initial results. In: Merelo, Adamitis, Beyer, Fernandez-Villacans, Schwefel (eds) Parallel problem solving from nature – PPSN VII, LNCS 2439. Springer, Spain, pp 537–548
- Smith J (2002b) Co-evolution of memetic algorithms for protein structure prediction. In: Hart K, Smith J (eds) Proceedings of the third international workshop on memetic algorithms, New York
- Smith J (2003) Co-evolving memetic algorithms: A learning approach to robust scalable optimisation. In: Proceedings of the 2003 congress on evolutionary computation. Canberra, pp 498–505
- Smith JE (2007) Credit assignment in adaptive memetic algorithms. In: GECCO '07: Proceedings of the 9th annual conference on genetic and evolutionary computation, ACM, New York, pp 1412–1419. doi: <http://doi.acm.org/10.1145/1276958.1277219>
- Smith R, Smuda E (1995) Adaptively resizing populations: algorithms, analysis and first results. *Complex Syst* 1(9):47–72
- Sorensen K, Sevaux M (2006) MA:PM: memetic algorithms with population management. *Comput Oper Res* 33:1214–1225
- Sudholt D (2007) Memetic algorithms with variable-depth search to overcome local optima. In: Proceedings of the 2007 conference on genetic and evolutionary computation (GECCO), ACM, New York, pp 787–794
- Tabacman M, Bacardit J, Loiseau I, Krasnogor N (2008) Learning classifier systems in optimisation problems: a case study on fractal travelling salesman problems. In: Proceedings of the international workshop on learning classifier systems, Lecture notes in computer science, Springer, New York, URL <http://www.cs.nott.ac.uk/nxk/PAPERS/maxi.pdf>
- Tang M, Yao X (2007) A memetic algorithm for VLSI floorplanning. *Syst Man Cybern Part B IEEE Trans* 37(1):62–69. doi: 10.1109/TSMCB.2006.883268
- Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Comput Ind Eng* 54(3):453–473
- Turney P (1996) How to shift bias: lessons from the Baldwin effect. *Evolut Comput* 4(3):271–295
- Vavak F, Fogarty T (1996) Comparison of steady state and generational genetic algorithms for use in non-stationary environments. In: Proceedings of the 1996 IEEE conference on evolutionary computation, Japan, pp 192–195
- Wang H, Wang D, Yang S (2009) A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Comput* 13(8–9)
- Whitley L, Gruau F (1993) Adding learning to the cellular development of neural networks: evolution and the Baldwin effect. *Evolut Comput* 1:213–233
- Whitley L, Gordon S, Mathias K (1994) Lamarckian evolution, the Baldwin effect, and function optimisation. In: Davidor Y, Schwefel HP, Männer R (eds) PPSN, Lecture notes in computer science, vol 866. Springer, Berlin, pp 6–15
- Wolpert D, Macready W (1997) No free lunch theorems for optimisation. *IEEE Trans Evolut Comput* 1(1): 67–82
- Yanga J, Suna L, Leeb H, Qiand Y, Liang Y (2008) Clonal selection based memetic algorithm for job shop scheduling problems. *J Bionic Eng* 5:111–119
- Yannakakis M (1997) Computational complexity. In: Aarts E, Lenstra J (eds) Local search in combinatorial optimization. Wiley, New York, pp 19–55
- Zhou Z (2004) Hierarchical surrogate-assisted evolutionary optimization framework. In: Congress on evolutionary computation, 2004. CEC 2004. Portland, pp 1586–1593
- Zhou Z, Ong Y, Lim M, Lee B (2007) Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Comput Fus Found Methodol Appl* 11:957–971

30 Genetics-Based Machine Learning

Tim Kovacs

Department of Computer Science, University of Bristol, UK
kovacs@cs.bris.ac.uk

1	<i>Introduction</i>	938
2	<i>A Framework for GBML</i>	941
3	<i>GBML Areas</i>	947
4	<i>Conclusions</i>	972

Abstract

This is a survey of the field of genetics-based machine learning (GBML): the application of evolutionary algorithms (ES) to machine learning. We assume readers are familiar with evolutionary algorithms and their application to optimization problems, but not necessarily with machine learning. We briefly outline the scope of machine learning, introduce the more specific area of supervised learning, contrast it with optimization and present arguments for and against GBML. Next we introduce a framework for GBML, which includes ways of classifying GBML algorithms and a discussion of the interaction between learning and evolution. We then review the following areas with emphasis on their evolutionary aspects: GBML for subproblems of learning, genetic programming, evolving ensembles, evolving neural networks, learning classifier systems, and genetic fuzzy systems.

1 Introduction

Genetics-based machine learning (GBML) is the application of evolutionary algorithms (EAs) to machine learning. We assume readers are familiar with EAs, which are well documented elsewhere, and their application to optimization problems. In this introductory section we outline the scope of machine learning, introduce the more specific area of supervised learning, and contrast it with optimization. However, the treatment is necessarily brief and readers who desire to work in GBML are strongly advised to first gain a solid foundation in non-evolutionary approaches to machine learning. ➤ Sect. 2 describes a framework for GBML, which includes ways of classifying GBML algorithms and a discussion of the interaction between learning and evolution. ➤ Sect. 3 reviews the work of a number of GBML communities with emphasis on their evolutionary aspects. Finally, ➤ Sect. 4 concludes the chapter.

What is Missing

Given the breadth of the field and the volume of the literature, the coverage herein is necessarily somewhat arbitrary and misses a number of significant subjects. These include a general introduction to machine learning including the structure of learning problems and their fitness landscapes (which we must exploit in order to learn efficiently), non-evolutionary algorithms (which constitute the majority of machine learning methods, and include both simple and effective methods), and theoretical limitations of learning (such as the *no free lunch* theorem for supervised learning (Wolpert 1996) and the *conservation law of generalization* (Schaffer 1994)). Also missing is coverage of GBML for clustering, reinforcement learning, Bayesian networks, artificial immune systems, artificial life, and application areas. Finally, some areas which have been touched on have been given an undeservedly cursory treatment, including EAs for data preparation (e.g., feature selection), coevolution, and comparisons between GBML and non-evolutionary alternatives. However, Freitas (2002a) contains good treatments of GBML for, among others, clustering and data preparation.

1.1 Machine Learning

Machine learning is concerned with machines that improve with experience and reason inductively or abductively in order to optimize, approximate, summarize, generalize from specific examples to general rules, classify, make predictions, find associations, propose

explanations, and propose ways of grouping things. For simplicity, we will restrict ourselves to classification and optimization problems.

Inductive Generalization

Inductive generalization refers to the inference of unknown values from known values. Induction differs from deduction in that the unknown values are in fact *unknowable*, which gives rise to fundamental limitations in what can be learned. (If at a later time new data makes all such values known the problem ceases to be inductive.) Given that the unknown values are unknowable, we *assume* they are correlated with the known values and we seek to learn the correlations. We formulate our objective as maximizing a function of the unknown values. In evolutionary computation this objective is called the fitness function, whereas in other areas the analogous feedback signal may be known as the error function, or by other names. There is *no need for induction* if: (i) all values are known and (ii) there is enough time to process them. We consider two inductive problems: function optimization and learning. We will not deal with abduction.

1-Max: A Typical Optimization Problem

The 1-max problem is to maximize the number of 1s in a binary string of length n . The optimal solution is trivial for humans although it is less so for EAs. The *representation* of this problem follows. Input: none. Output: bit strings of length n . *Data generation*: we can generate as many output strings as time allows, up to the point where we have enumerated the search space (in which case the problem ceases to be inductive). *Training*: the fitness of a string is the number of 1s it contains. We can evaluate a learning method on this task by determining how close it gets to the known optimal solution. In more realistic problems, the optimum is not known and we may not even know the maximum possible fitness. Nonetheless, for both toy and realistic problems, we can evaluate how much training was needed to reach a certain fitness and how a learning method compares to others.

Classification of Mushrooms: A Typical Learning Problem

Suppose we want to classify mushroom species as poisonous or edible given some training data consisting of features of each species (color, size and so on) including edibility. Our task is to learn a hypothesis, which will classify new species whose edibility is unknown. *Representation*: the input is a set of nominal attributes and the output is a binary label indicating edibility. *Data generation*: a fixed dataset of input/output examples derived from a book. Typically the dataset is far, far smaller than the set of possible inputs, and we partition it into train and test sets. *Training*: induce a hypothesis which maximizes classification accuracy on the train set. *Evaluation*: evaluate the accuracy of the induced hypothesis on the test set, which we take as an indication of how well a newly encountered species might be classified.

Terminology in Supervised Learning

Although many others exist, we focus on the primary machine learning paradigm: standard supervised learning (SL), of which the preceding mushroom classification task is a good example. In SL we have a dataset of labeled input/output pairs. Inputs are typically called instances or exemplars and are factored into attributes (also called features), while outputs are called classes (for classification tasks) or the output is called the dependent variable (for regression tasks).

Comparison of Supervised Learning and Optimization

In SL we typically have limited training data and it is crucial to find a good inductive bias for later use on new data. Consequently, we *must* evaluate the generalization of the induced hypothesis from the train set to the previously unused test set. In contrast, in optimization we can typically generate as much data as time allows and we can typically evaluate any output. We are concerned with finding the optimum output in minimum time, and, specifically, inducing which output to evaluate next. As a result no test set is needed.

Issues in Supervised Learning

A great many issues arise in SL including overfitting, underfitting, producing human readable results, dealing with class imbalances in the training data, asymmetric cost functions, noisy and nonstationary data, online learning, stream mining, learning from particularly small datasets, learning when there are very many attributes, learning from positive instances only, incorporating bias and prior knowledge, handling structured data, and using additional unlabeled data for training. None of these will be dealt with here.

1.2 Arguments For and Against GBML

GBML methods are a niche approach to machine learning and much less well-known than the main non-evolutionary methods, but there are many good reasons to consider them.

Accuracy

Importantly, the classification accuracy of the best evolutionary and non-evolutionary methods are comparable (Freitas 2002a, Sect. 12.1.1).

Synergy of Learning and Evolution

GBML methods exploit the synergy of learning and evolution, combining global and local search and benefitting from the Baldwin effect's smoothing of the fitness landscape (Sect. 2.3).

Epistasis

There is some evidence that the accuracy of GBML methods may not suffer from epistasis as much as typical non-evolutionary greedy search (Freitas 2002a, Sect. 12.1.1).

Integrated Feature Selection and Learning

GBML methods can combine feature selection and learning in one process. For instance, feature selection is intrinsic in LCS methods (Sect. 3.5).

Adapting Bias

GBML methods are well-suited to adapting inductive bias. We can adapt representational bias by, for example, selecting rule condition shapes (Sect. 3.5.3), and algorithmic bias by, for example, evolving learning rules (Sect. 3.4).

Exploiting Diversity

We can exploit the diversity of a population of solutions to combine and improve predictions (the ensemble approach, (Sect. 3.3) and to generate Pareto sets for multi-objective problems.

Dynamic Adaptation

All the above can be done dynamically to improve accuracy, to deal with nonstationarity, and to minimize population size. This last is of interest in order to reduce overfitting, improve run-time, and improve readability.

Universality

Evolution can be used as a wrapper for *any* learner.

Parallelization

Population-based search is easily parallelized.

Suitable Problem Characteristics

From an optimization perspective, learning problems are typically large, non-differentiable, noisy, epistatic, deceptive, and multimodal (Miller et al. 1989). To this list we could add high-dimensional and highly constrained. EAs are a good choice for such problems. See Cantú-Paz and Kamath (2003) and [Sect. 3.4](#) for more arguments in favor of and against GBML.

Algorithmic Complexity

GBML algorithms are typically more complex than their non-evolutionary alternatives. This makes them harder to implement and harder to analyze, which means there is less theory to guide parameterization and development of new algorithms.

Increased Run-time

GBML methods are generally much slower than the non-evolutionary alternatives.

Suitability for a Given Problem

No single learning method is a good choice for all problems. For one thing the bias of a given GBML method may be inappropriate for a given problem. Problems to which GBML methods are particularly prone include prohibitive run-time (or set-up time) and that simpler and/or faster methods may suffice. Furthermore, even where GBML methods perform better, the improvements may be marginal. See the strengths, weaknesses, opportunities, threats (SWOT) analysis of GBML in Orriols-Puig et al. ([2008b](#)) for more.

2 A Framework for GBML

The aim of the framework presented in this section is to structure the range of GBML systems into more specific categories about which we can make more specific observations than we could about GBML systems as a whole. We present two categorizations. In the first ([Sect. 2.1](#)), GBML systems are classified by their role in learning; specifically their applications to i) subproblems of machine learning, ii) learning itself, or iii) meta-learning. In the second categorization ([Sect. 2.2](#)), GBML systems are classified by their high-level algorithmic approach as either Pittsburgh or Michigan systems. Following this, in [Sect. 2.3](#) we briefly review ways in which learning and evolution interact and in [Sect. 2.4](#) we consider various models of GBML not covered earlier.

Before proceeding, we note that evolution can output a huge range of phenotypes, from scalar values to complex learning agents, and that agents can be more or less plastic (independent of evolution). For example, if evolution outputs a fixed hypothesis, that hypothesis has no plasticity. In contrast, evolution can output a neural net which, when trained with backpropagation, can learn much. (In the latter approach, evolution may specify the network structure while backpropagation adapts the network weights.)

Structure of GBML Systems

We can divide any evolutionary (meta)-learning system into the following parts: (i) *Representation*, which consists of the genotype (the learner's genes) and phenotype (the learner itself, built according to its genes). In simple cases, the genotype and phenotype may be identical, for example with the simple ternary LCS rules of [Sect. 3.5.2](#). In other cases, the two are very different and the phenotype may be derived through a complex developmental process (as in nature); see [Sect. 3.4](#) on developmental encodings for neural networks. (ii) *Feedback*, which consists of the learner's objective function (e.g., the error function in supervised learning) and the fitness function which guides evolution. (iii) *The production system*, which applies the phenotypes to the learning problem. (iv) *The evolutionary system*, which adapts genes.

2.1 Classifying GBML Systems by Role

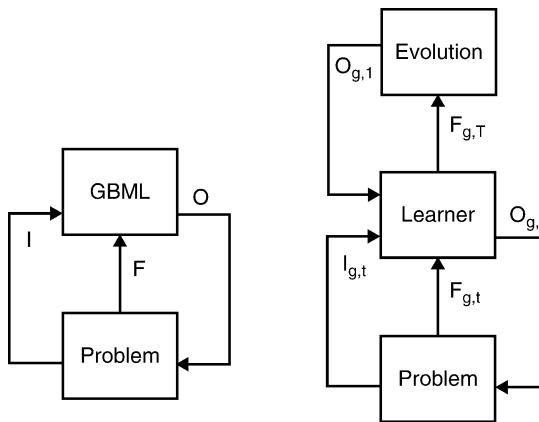
In order to contrast learning and meta-learning, we define learning as a process which outputs a fixed hypothesis. Accordingly, when evolution adapts hypotheses it is a learner and when it adapts learners it is a meta-learner. However, this distinction between learning and meta-learning should not be overemphasized; if evolution outputs a learner with little plasticity then evolution may be largely responsible for the final hypothesis, and in this case plays both a learning and a meta-learning role. Furthermore, both contribute to the ultimate goal of adaptation, and in [Sect. 2.3](#) we will see ways in which they interact.

Evolution as learning is illustrated in the left of [Fig. 1](#), which shows a GBML agent interacting directly with the learning problem. In contrast, the right of the figure shows GBML as meta-learning: the learner (or a set of learners) is the output of evolution, and the learner interacts directly with the learning problem while evolution interacts with it only through learners. At time step 1 of each generation, evolution outputs a learning agent and at the generation's final step, T, it receives an evaluation of the learner's fitness. During the intervening time steps the learner interacts with the problem. This approach to meta-learning is *universal* as any learner can be augmented by GBML, and is related to the wrapper approach to feature selection in [Sect. 3.1](#).

Meta-learning is a broad term with different interpretations but the essential idea is *learning about learning*. A meta-learner may optimize parameters of a learner, learn which learner to apply to a given input or a given problem, learn which representation(s) to use, optimize the update rules used to train learners, learn an algorithm which solves the problem, evolve an ecosystem of learners, and potentially be open ended. See Vilalta and Drissi ([2002](#)) and Giraud-Carrier and Keller ([2002](#)) on non-evolutionary meta-learning and Burke et al. ([2003](#)), Krasnogor ([2004](#)), Krasnogor and Gustafson ([2004](#)), and Burke and Kendall ([2005](#)) on the hyperheuristics (*heuristics to learn heuristics*) approach, of which a subset is evolutionary.

Fig. 1

(Left) GBML as learner. Input, Output, and Fitness shown. (Right) GBML as meta-learner. Subscripts denote generation and time step ($1 \dots T$).



A third role for evolution is application to various subproblems of learning including feature selection, feature construction, and other optimization roles within learning agents. In these cases evolution neither outputs the final hypothesis nor outputs a learning agent which does so. **Section 3.1** deals with such applications.

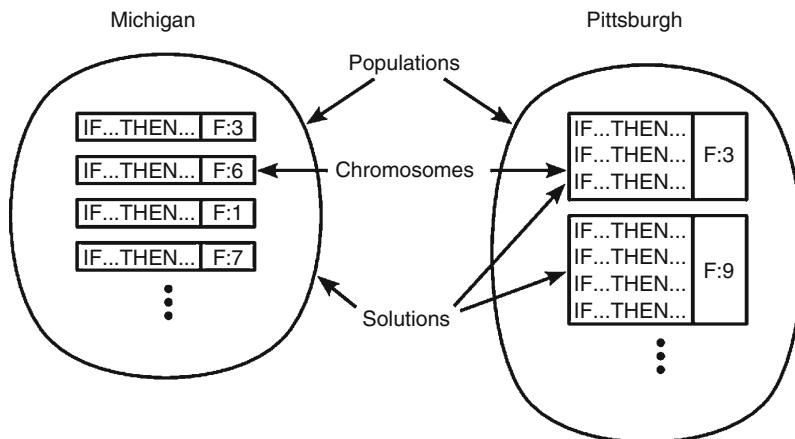
2.2 Classifying GBML Systems Algorithmically

In the Pittsburgh (Pitt) approach, one chromosome encodes one solution. We assume that fitness is assigned to chromosomes, so in Pitt systems it is assigned to solutions. This leaves a credit assignment problem: how did the chromosome's component genes contribute to the observed fitness of the chromosome? This is left to evolution as this is what EAs are designed to deal with. In the Michigan approach, one solution is (typically) represented by many chromosomes and so fitness is assigned to partial solutions. Credit assignment differs from the Pitt case as chromosomes not only compete for reproduction but may also complement and cooperate with each other. This gives rise to the issues of how to encourage cooperation, complementarity, and coverage of the inputs, all of which makes designing an effective fitness function more complex than in Pitt systems. In Michigan systems, the credit assignment problem is how to measure a chromosome's contributions to the overall solution, as reflected in the various aspects of fitness just mentioned. To sum up the difficulty in Michigan systems: the best set of chromosomes may not be the set of best (i.e., fittest) chromosomes (Freitas 2002a). To illustrate, **Fig. 2** depicts the representation used by Pitt and Michigan versions of the rule-based systems called learning classifier systems (LCS) (see **Sect. 3.5**). In a Pittsburgh LCS, a chromosome is a variable-length *set* of rules, while in a Michigan LCS, a chromosome is a single fixed-length rule.

Although the Pittsburgh and Michigan approaches are generally presented as two discrete cases, some hybrids exist (e.g., Wilcox (1995)).

Fig. 2

Michigan and Pittsburgh rule-based systems compared. The F:x associated with each chromosome indicates its fitness.



Pittsburgh and Michigan Compared

Pittsburgh systems (especially naive implementations) are slower, since they evolve more complex structures and they assign credit at a less specific (and hence less informative) level. (See, however, [Sect. 3.5.5](#) on the windowing approach to improving run-time and Bacardit et al. (2009a) for an approach which avoids performing matching operations between rule conditions and irrelevant features.) Additionally, their chromosomes and their genetic operators are more complex. On the other hand they face less complex credit assignment problems and hence are more robust, that is, more likely to adapt successfully. Michigan systems use a finer grain of credit assignment than the Pittsburgh approach, which means bad partial solutions can be deleted without restarting from scratch. This makes them more efficient and also more suitable for incremental learning. However, credit assignment is more complex in Michigan systems. Since the solution is a *set* of chromosomes: (i) the population must not converge fully, and (ii) as noted, the best set of chromosomes may not be the set of best chromosomes.

The two approaches also tend to be applied in different ways. Pitt systems are typically used offline and are algorithm-driven; the main loop processes each chromosome in turn and seeks out data to evaluate them (which is how a standard genetic algorithm (GA) works, although fitness evaluation is typically simpler in a GA). In contrast, Michigan systems are typically used online and are data-driven; the main loop processes each data input in turn and seeks out applicable chromosomes (see [Fig. 3](#)). As a result, Michigan systems are more often used as learners (though not necessarily more often as meta-learners) for reinforcement learning, which is almost always online. The Michigan approach has mainly been used with LCS. See Greene and Smith (1993), Janikow (1993), Wilcox (1995), Freitas (2002b) and Kovacs (2004) for comparison of the approaches.

Iterative Rule Learning

IRL is a variation on the Michigan approach in which, as usual, one solution is represented by many chromosomes, but only the single best chromosome is selected after each run, which

Fig. 3**A basic Michigan algorithm.**

On each time step:

1. Identify match set: subset of population which match current input
2. Compute support in match set for each class
3. Select class
4. Identify action set: subset of match set which advocate selected class
5. Update action set based on feedback
6. Optionally alter population

alters the coevolutionary dynamics of the system. The output of multiple runs is combined to produce the solution. The approach originated with SIA (Supervised Inductive Algorithm) (Venturini 1993; Juan Liu and Tin-Yau Kwok 2000), a supervised genetic rule learner.

Genetic Cooperative-Competitive Learning

GCCL is another Michigan approach in which each generation is ranked by fitness and a *coverage-based filter* then allocates inputs to the first rule which correctly covers them. Inputs are only allocated to one rule per generation and rules which have no inputs allocated die at the end of a generation. The collective accuracy of the remaining rules is compared to the previous best generation, which is stored offline. If the new generation is more accurate (or the same but has fewer rules) it replaces the previous best. Examples include COGIN (Greene and Smith 1993, 1994), REGAL (Giordana and Neri 1995), and LOGENPRO (Wong and Leung 2000).

2.3 The Interaction of Learning and Evolution

This section briefly touches on the rich interactions between evolution and learning.

Memetic Learning

We can characterize evolution as a form of global search, which is good at finding good basins of attraction, but poor at finding the optimum of those basins. In contrast, many learning methods are forms of local search and have the opposite characteristics. We can get the best of both by combining them, which generally outperforms either alone (Yao 1999). For example, evolving the initial weights of a neural network and then training them with gradient descent can be two orders of magnitude faster than using random initial weights (Floreano et al. 2008). Methods which combine global and local search are called *memetic* algorithms (Hart et al. 2004, 2005; Ong et al. 2006, 2007; Smith 2007; Ong et al. 2009; Rozenberg et al. 2012). See Krasnogor and Smith (2005) for a self-contained tutorial.

Darwinian and Lamarckian Evolution

In Lamarckian evolution/inheritance, learning during an individual's lifetime directly alters the genes passed to offspring, so offspring inherit the result of their parents' learning. This does not occur in nature but can in computers and has the potential to be more efficient than Darwinian evolution since the results of learning are not thrown away. Indeed, Ackley and

Littman (1992) showed Lamarckian evolution was much faster on stationary learning tasks but Sasaki and Tokoro (1997) showed Darwinian evolution was generally better on nonstationary tasks. See also Whitley et al. (1994), Yamasaki and Sekiguchi (2000), Pereira and Costa (2001), and Whiteson and Stone (2006).

The Baldwin Effect

The Baldwin effect is a two-part dynamic between learning and evolution which depends on *Phenotypic Plasticity* (PP): the ability to adapt (e.g., learn) during an individual's lifetime. The first aspect is this. Suppose a mutation would have no benefit except for PP. Without PP, the mutation does not increase fitness, but with PP it does. Thus PP helps evolution to adopt beneficial mutations; it effectively smooths the fitness landscape. A possible example from nature is lactose tolerance in human adults. At a recent point in human evolution a mutation occurred, which allowed adult humans to digest milk. Subsequently, humans learned to keep animals for milk, which in turn made the mutation more likely to spread. The smoothing effect on the fitness landscape depends on PP; the greater the PP the more potential there is for smoothing. All GBML methods exploit the Baldwin effect to the extent that they have PP. See Whiteson and Stone (2006, Sect. 7.2) for a short review of the Baldwin effect in reinforcement learning.

The second aspect of the Baldwin effect is genetic assimilation. Suppose PP has a cost (e.g., learning involves making mistakes). If PP can be replaced by new genes, it will be; for instance a learned behavior can become instinctive. This allows learned behaviors to become inherited without Lamarckian inheritance.

Turney (1996) has connected the Baldwin effect to inductive bias. All inductive algorithms have a bias and the Baldwin effect can be seen as a shift from weak to strong bias. When bias is weak, agents rely on learning; when bias is strong, agents rely on instinctive behavior.

Evaluating Evolutionary Search by Evaluating Accuracy

Kovacs and Kerber (2004) point out that high classification accuracy does not imply effective genetic search. To illustrate, they initialized XCS (Wilson 1995) with random condition/action rules and disabled evolutionary search. Updates to estimates of rule utility, however, were made as usual. They found the system was still able to achieve very high training set accuracy on the widely used 6 and 11 multiplexer tasks since ineffective rules were simply given low weight in decision making, though neither removed nor replaced. Care is therefore warranted when attributing good accuracy to genetic search. A limitation of this work is that test set accuracy was not evaluated.

2.4 Other GBML Models

This section covers some models which are orthogonal to those discussed earlier.

Online Evolutionary Computation

In many problems, especially sequential ones, feedback is very noisy and needs averaging. Whiteson and Stone (2006) allocated trials to chromosomes in proportion to their fitness with the following procedure. At each new generation, each chromosome is evaluated once only. Subsequent evaluations are allocated using a softmax distribution based on the initial fitnesses and the average fitness of a chromosome is recalculated after each evaluation. In nonstationary

problems a recency-weighted average of fitness samples is used. This approach is called *online evolutionary computation*. Its advantages are that less time is wasted evaluating weaker chromosomes, and in cases where mistakes matter, fewer mistakes are made by agents during fitness evaluations. However, the improvement is only on average; worst case performance is not improved. This is related to other work on optimizing noisy fitness functions (Stagge 1998; Beielstein and Markon 2002), except that they do not reduce online mistakes.

Steady-State EAs

Whereas standard generational EAs replace the entire population each generation, steady state EAs replace a subset (e.g., only two in XCS). This approach is standard in Michigan LCS because they minimize disruption to the population, which is useful for online learning. Steady state EAs introduce selection for deletion as well as reproduction and this is typically biased toward lower fitness chromosomes or to reduce crowding.

Co-evolving Learners and Problems

Another possibility not mentioned in our earlier classifications is to coevolve both learners and problems. When successful, this allows learners to gradually solve harder problems rather than tackling the most difficult problems from the start. It also allows us to search the space of problems to find those which are harder for a given learner, and to explore the dynamics between learners and problems.

3 GBML Areas

This section covers the main GBML research communities. These communities are more disjoint than the methods they use and the lines between them are increasingly blurring. For example, LCS often evolve neural networks and fuzzy rules, and some are powered by genetic programming. Nonetheless, the differences between the communities and their approaches is such that it seemed most useful to structure this section by community and not, for example, by phenotype or learning paradigm; such integrated surveys of GBML are left to the future. Many communities have reinvented the same ideas, yet each has its own focus and strengths and so each has much to learn from the others.

3.1 GBML for Subproblems of Learning

This section briefly reviews ways in which evolution has been used not for the primary task of learning – generating hypotheses – but for subproblems including data preparation and optimization within other learning methods.

Evolutionary Feature Selection

Some attributes (features) of the input are of little or no use in classification. We can simplify and speed learning by selecting only useful attributes to work with, especially when there are very many attributes and many contribute little. EAs are widely used in the wrapper approach to feature selection (John et al. 1994) in which the base learner (the one which generates hypotheses) is treated as a black box to be optimized by a search algorithm. In this, EAs usually give good results compared to non-evolutionary methods (Jain and Zongker 1997;

Sharpe and Glover (1999); Kudo and Skalansky (2000) but there are exceptions (Jain and Zongker 1997). In Cantú-Paz (2002) Estimation of Distribution Algorithms were found to give similar accuracy but run more slowly than a GA. More generally we can weight features (instead of making an all-or-nothing selection) and some learners can use weights directly, for example, weighted k-nearest neighbors (Raymer et al. 2000). The main drawback of EAs for feature selection is their slowness compared to non-evolutionary methods. See Martin-Bautista and Vila (1999) and Freitas (2002a, b) for overviews, and Stout et al. (2008), Bacardit et al. (2009b) for some recent real-world applications.

Evolutionary Feature Construction

Some features are not very useful by themselves but can be when combined with others. We can leave the base learner to discover this or we can preprocess data to construct informative new features by combining existing ones, for example new feature $f_{\text{new}} = f_1 \text{ AND } f_3 \text{ AND } f_8$. This is also called constructive induction and there are different approaches. GP has been used to construct features out of the original attributes, for example, Hu (1998), Krawiec (2002), and Smith and Bull (2005). The original features have also been linearly transformed by evolving a vector of coefficients (Kelly and Davis 1991; Punch et al. 1993). Simultaneous feature transformation and selection has had good results (Raymer et al. 2000).

Other Subproblems of Learning

EAs have been used in a variety of other ways. One is training set optimization in which we can partition the data into training sets (Romaniuk 1994), select the most useful training inputs (Ishibuchi and Nakashima 2000), and even generate synthetic inputs (Zhang and Veenker 1991; Cho and Cha 1996). EAs have also been used for optimization within a learner, for example, Kelly and Davis (1991) optimized weighted k-nearest neighbors with a GA, Cantú-Paz and Kamath (2003) optimized decision tree tests using a GA and an evolution strategy (ES) and Thompson (1998, 1999) optimized voting weights in an ensemble. Janikow (1993) replaced beam search in AQ with a genetic algorithm and similarly Tameddoni-Nezhad and Muggleton (2000, 2003), Divina and Marchiori (2002), and Divina et al. (2002, 2003) have investigated inductive logic programming driven by a GA.

3.2 Genetic Programming

Genetic Programming (GP) is a major evolutionary paradigm which evolves programs (Vanneschi and Poli 2012). The differences between GP and GAs are not precise but typically GP evolves variable-length structures, typically trees, in which genes can be functions. See Woodward (2003) for discussion. Freitas (2002a) discusses differences between GAs and GP which arise because GP representations are more complex. Among the pros of GP: (i) it is easier to represent complex languages, such as first-order logic in GP, (ii) it is easier to represent complex concepts compactly, and (iii) GP is good at finding novel and complex patterns overlooked by other methods. Among the cons of GP: (i) expressive representations have large search spaces, (ii) GP tends to overfit / does not generalize well, and (iii) variable-length representations suffer from bloat (see, e.g., Poli et al. 2008)).

While GAs are typically applied to function optimization, GP is widely applied to learning. To illustrate, there are many learning problems among the set of “typical GP problems” defined by Koza (1992), which have become more-or-less agreed benchmarks for the

GP community (Vanneschi and Poli 2012), there are many learning problems. These include the multiplexer and parity Boolean functions, symbolic regression of mathematical functions and the intertwined spirals problem, which involves the classification of two-dimensional points as belonging to one of the two spirals. GP usually follows the Pittsburgh approach. We cover the two representations most widely used for learning with GP: GP trees and decision trees.

3.2.1 GP Trees

GP Trees for Classification

Figure 4 shows the 3 multiplexer Boolean function as a truth table on the left and as a GP tree on the right. To classify an input with the GP tree: (i) instantiate the leaf variables with the input values, (ii) propagate values upward from leaves through the functions in the non-leaf nodes and (iii) output the value of the root (top) node as the classification.

GP Trees for Regression

In regression problems leaves may be constants or variables and non-leaves are mathematical functions. Figure 5 shows a real-valued function as an algebraic expression on the left and as

Fig. 4

Two representations of the 3 multiplexer function: truth table (left) and GP tree (right).

A	B	C	Class
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

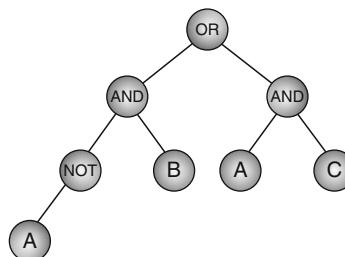
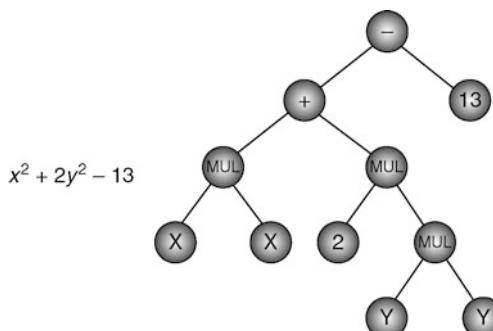


Fig. 5

Two representations of a real-valued function.



a GP tree on the right. (Note that $x^2 + 2y^2 - 13 = ((x*x) + (2*(y*y))) - 13$.) The output of the tree is computed in the same way as in the preceding classification example.

3.2.2 Decision Trees

► *Figure 6* shows the 3 multiplexer as a truth table and as a decision tree. To classify an input in such a tree: (i) start at the root (top) of tree, (ii) follow the branch corresponding to the value of the attribute in the input, (iii) repeat until a leaf is reached, and (iv) output the value of the leaf as the classification of the input.

Evolving First-Order Trees

First-order trees use both propositional and first-order internal nodes. Rouwhorst and Engelbrecht (2000) found first-order logic made trees more expressive and allowed much smaller solutions than found by the rule learner CN2 or the tree learner C4.5, with similar accuracy.

Oblique (Linear) Trees

Whereas conventional tree algorithms learn axis-parallel decision boundaries, oblique trees make tests on a linear combination of attributes. The resulting trees are more expressive but have a larger search space. See Bot and Langdon (2000).

Evolving Individual Nodes in Decision Trees

In most GP-based tree evolvers, an individual is a complete tree but in Marmelstein and Lamont (1998) each individual is a tree node. The tree is built incrementally: one GP run is made for each node. This is similar to IRL in ► Sect. 2.2 but results are added to a tree structure rather than a list.

3.2.3 Extensions to GP

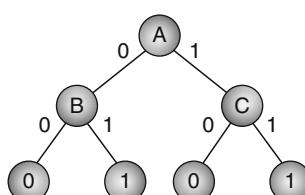
Ensemble Methods and GP

Ensemble ideas have been used in two ways. First, to reduce fitness computation time and memory requirements by training on subsamples of the data. The bagging approach has been used in Folino et al. (2003) and Iba (1999) and the boosting approach in Song et al. (2005).

■ Fig. 6

Two representations of the 3 multiplexer function: truth table (left) and decision tree (right).

A	B	C	Class
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Although not an ensemble technique, the limited error fitness (LEF) method introduced in Gathercole and Ross (1997) as a way of reducing GP run-time works in a similar manner: in LEF, the proportion of the training set used to evaluate fitness depends on the individual's performance. The second ensemble approach improves accuracy by building an ensemble of GP trees. In Keijzer and Babovic (2000) and Paris et al. (2001) each run adds one tree to the ensemble and weights are computed with standard boosting.

GP Hyperheuristics

Schmidhuber (1987) proposed a meta-GP system evolving evolutionary operators as a way of expanding the power of GP's evolutionary search. Instead of evolving decision rules Krasnogor proposes applying GP to the much harder task of evolving classification algorithms, represented using grammars (Krasnogor 2002, 2004; Krasnogor and Gustafson 2004). Freitas (2002a, Sect. 12.2.3) sketches a similar approach, which he calls *algorithm induction*, while (Pappa and Freitas 2010) goes into the subject in much more detail. Burke et al. (2009) also deal with GP hyperheuristics.

3.2.4 Conclusions

Lack of Test Sets in GP

GP terminology follows a convention in the GA field since at least (Holland 1986) in which *brittleness* refers to overfitting or poor generalization to unseen cases, and *robustness* refers to good generalization. A feature of the GP literature is that GP is usually evaluated only on the training set (Kushchu 2002; Vanneschi and Poli 2012). Kushchu has also criticized the way in which test sets have been used (Kushchu 2002). Nonetheless GP has the same need for test sets to evaluate generalization as other methods (Kuschu 2002) and as a result, the ability of GP to perform inductive generalization is one of the open issues for GP identified in Vanneschi and Poli (2012). See Kushchu (2002) and Vanneschi and Poli (2012) for methods which have been used to encourage generalization in GP, many of which can be applied to other methods.

Reading

See Koza's 1994 book (Koza 1994) for the basics of evolving decision trees with GP, Wong and Leung's 2000 book on data mining with grammar-based GP (Wong and Leung 2000), Freitas' 2002 book (Freitas 2002a) for a good introduction to GP, decision trees and both evolutionary and non-evolutionary learning, Poli, Langdon, and McPhee's free 2008 GP book (Poli et al. 2008), and Vanneschi and Poli's chapter on GP in this volume (Vanneschi and Poli 2012). The GP bibliography has over 5,000 entries (Langdon et al. 2009).

3.3 Evolving Ensembles

Ensembles, also called *multiple classifier systems* and *committee machines*, is the field which studies how to combine predictions from multiple sources. Ensemble methods are widely applicable to evolutionary systems where a population intrinsically provides multiple predictors. Ensemble techniques can be used with any learning method, although they are most useful for unstable learners, whose hypotheses are sensitive to small changes in their training. Ensembles can be heterogeneous (composed of different types of predictors) in which case they are called *hybrid* ensembles. Relatively few studies of hybrid systems exist (Brown et al. 2005)

but see for example Woods et al. (1997), Cho and Park (2001), and Chandra and Yao (2006). Ensembles enjoy good theoretical foundations (Brown et al. 1996; Tumer and Ghosh 1996), perform very well in practice (Caruana and Niculescu-Mizil 2006) and were identified by Dietterich as one of the four current directions for machine learning in 1998 (Dietterich 1998). While the key advantage of using ensembles is better test-set generalization, there are others: ensembles can perform more complex tasks than individual members, the overall system can be easier to understand and modify and ensembles are more robust/degrade more gracefully than individual predictors (Sharkey 1996).

Working with an ensemble raises a number of issues. How to create or select ensemble members? How many members are needed? When to remove ensemble members? How to combine their predictions? How to encourage diversity in members? There are many approaches to these issues, among the best known of which are bagging (Breiman 1996, 1998) and boosting (Chandra and Yao 2006a; Meir and Rätsch 2003).

Creating a good ensemble is an inherently multi-objective problem (Valentini and Masulli 2002). In addition to maximizing accuracy we also want to maximize diversity in errors; after all, having multiple identical predictors provides no advantage. On the other hand, an ensemble of predictors which make different errors is very useful since we can combine their predictions so that the ensemble output is at least as good on the training set as the average predictor (Krogh and Vedelsby 1995). Hence we want to create accurate predictors with diverse errors (Dietterich 1998; Hansen and Salamon 1990; Krogh and Vedelsby 1995; Opitz and Maclin 1999; Opitz and Slavlik 1996). In addition we may want to minimize ensemble size in order to reduce run-time and to make the ensemble easier to understand. Finally, evolving variable-length chromosomes without pressure toward parsimony results in bloat (Poli et al. 2008), in which case we have a reason to minimize the size of individual members.

3.3.1 Evolutionary Ensembles

Although most ensembles are non-evolutionary, evolution has many applications within ensembles. (i) *Classifier creation and adaptation*: providing the ensemble with a set of candidate members. (ii) *Voting*: Lam and Suen (1995), Thompson (1998, 1999), and Cho (1999) evolve weights for the votes of ensemble members. (iii) *Classifier selection*: the winners of evolutionary competition are added to the ensemble. (iv) *Feature selection*: generating diverse classifiers by training them on different features (see [Sect. 3.1](#) and Kuncheva (2004, Sect. 8.1.4)). (v) *Data selection*: generating diverse classifiers by training on different data (see [Sect. 3.1](#)). All these approaches have non-evolutionary alternatives. We now go into more detail on two of the above applications.

Classifier Creation and Adaptation

Single-objective evolution is common in evolving ensembles. For example, Liu and Yao (1999) combines accuracy and diversity into a single objective. In comparison, multiobjective evolutionary ensembles are rare (Chandra and Yao 2006) but they are starting to appear for example Abbass (2003) and Chandra and Yao (2006a, b). In addition to upgrading GBML to multi-objective GBML, other measures can be taken to evolve diversity, for example, fitness sharing (Liu et al. 2000) and the coevolutionary fitness method we describe next. Gagné et al. (2007) compare boosting and coevolution of learners and problems – both gradually focus on

cases which are harder to learn – and argue that coevolution is less likely to overfit noise. Their coevolution-inspired fitness works as follows: Let Q be a set of reference classifiers. The hardness of a training input, x_i , is based on how many members of Q misclassify it. The fitness of a classifier is the sum of hardnesses of the inputs, x_i , it classifies correctly. This method results in accurate, yet error-diverse classifiers, since both are required to obtain high fitness. Gagné et al. exploit the population of classifiers to provide Q . They also introduce a greedy margin-based scheme for selection of ensemble members. They find that a simpler off-line version of their evolving ensemble learning (EEL) approach dominates their online version as the latter lacks a way to remove bad classifiers. Good results were obtained compared to Adaboost on six UCI (Asuncion and Newman 2009) datasets.

Evolutionary Selection of Members

There are two extremes. Usually each run produces one member of the ensemble and many runs are needed. Sometimes, however, the entire population is eligible to join the ensemble, in which case only one run is needed. The latter does not resolve the *ensemble selection problem*: which candidates to use? There are many combinations possible from just a small pool of candidates, and, as with selecting a solution set from a Michigan population, the set of best individuals may not be the best set of individuals (that is, the best ensemble). The selection problem is formally equivalent to the feature selection problem (Gagné et al. 2007) (☞ Sect. 3.2). See, for example, Sirlantzis et al. (2001) and Ruta and Gabrys (2001) for evolutionary approaches.

3.3.2 Conclusions

Research directions for evolutionary ensembles include multiobjective evolution (Chandra and Yao 2006a), hybrid ensembles (Chandra and Yao 2006b), and minimizing ensemble complexity (Liu et al. 2000).

Reading

Key works include Opitz and Shavlik's classic 1996 paper on evolving NN ensembles (Opitz and Shavlik 1996), Kuncheva's 2004 book on ensembles (Kuncheva 2004), Chandra and Yao's 2006 discussion of multiobjective evolution of ensembles (Chandra and Yao 2006a), Yao and Islam's 2008 review of evolving NN ensembles (Yao and Islam 2008) and Brown's 2005 and 2010 surveys of ensembles (Brown et al. 2005; Brown 2010). We cover evolving NN ensembles in ☞ Sect. 3.4.

3.4 Evolving Neural Networks

The study of neural networks (NNs) is a large and interdisciplinary area. The term artificial neural network (ANN) is often used to distinguish simulations from biological NNs, but having noted this we shall refer simply to NNs. When evolution is involved such systems may be called evolving artificial neural networks (EANNs) Yao (1999) or evolving connectionist systems (ECosSs) (Kasabov 2007).

A neural network consists of a set of nodes, a set of directed connections between a subset of nodes, and a set of weights on the connections. The connections specify inputs and outputs to and from nodes and there are three forms of nodes: input nodes (for input to the network from the outside world), output nodes, and hidden nodes, which only connect to other nodes.

Nodes are typically arranged in layers: the input layer, hidden layer(s), and output layer. Nodes compute by integrating their inputs using an activation function and passing on their activation as output. Connection weights modulate the activation they pass on and in the simplest form of learning weights are modified while all else remains fixed. The most common approach to learning weights is to use a gradient descent-based learning rule such as back-propagation. The *architecture* of a NN refers to the set of nodes, connections, activation functions, and the plasticity of nodes (that is, whether they can be updated or not). Most often all nodes use the same activation function and in virtually all cases all nodes can be updated. Evolution has been applied at three levels: weights, architecture, and learning rules. In terms of architecture, evolution has been used to determine connectivity, select activation functions, and determine plasticity.

Representations

Three forms of representations have been used: (i) direct encoding (Yao 1999; Floreano et al. 2008) in which all details (connections and nodes) are specified, (ii) indirect encoding (Yao 1999; Floreano et al. 2008) in which general features are specified (e.g., number of hidden layers and nodes) and a learning process determines the details, and (iii) developmental encoding (Floreano et al. 2008) in which a developmental process is genetically encoded (Kitano 1990; Gruau 1995; Nolfi et al. 1994; Husbands et al. 1994; Pal and Bhandari 1994; Sziranyi 1996). Implicit and developmental representations are more flexible and tend to be used for evolving architectures, while direct representations tend to be used for evolving weights alone.

Credit Assignment

Evolving NNs virtually always use the Pittsburgh approach although there are a few Michigan systems (Andersen and Tsoi 1993; Smith and Cribbs 1994; Smith and Cribbs 1997). In Michigan systems, each chromosome specifies only one hidden node, which raises issues. How should the architecture be defined? A simple method is to fix it in advance. How can we make nodes specialize? Two options are to encourage diversity during evolution, for example with fitness sharing, or, after evolution, by pruning redundant nodes (Andersen and Tsoi 1993).

Adapting Weights

Most NN learning rules are based on gradient descent, including the best known: back-propagation (BP). BP has many successful applications, but gradient descent-based methods require a continuous and differentiable error function and often get trapped in local minima (Sutton 1986; Whitley et al. 1990).

An alternative is to evolve the weights which has the advantages that EAs do not rely on gradients and can work on discrete fitness functions. Another advantage of evolving weights is that the same evolutionary method can be used for different types of network (feedforward, recurrent, and higher order), which is a great convenience for the engineer (Yao 1999). Consequently, much research has been done on evolution of weights. Unsurprisingly fitness functions penalize NN error but they also typically penalize network complexity (number of hidden nodes) in order to control overfitting. The expressive power of a NN depends on the number of hidden nodes: fewer nodes = less expressive = fits training data less, while more nodes = more expressive = fits data better. As a result, if a NN has too few nodes it underfits, while with too many nodes it overfits. In terms of training rate, there is no clear winner between evolution and gradient descent; which is better depending on the problem

(Yao 1999). However, Yao (1999) states that evolving weights and architecture is better than evolving weights alone and that evolution seems better for reinforcement learning and recurrent networks. Floreano (2008) suggests evolution is better for dynamic networks. Happily we do not have to choose between the two approaches.

Evolving and Learning Weights

Evolution is good at finding a good basin of attraction but poor at finding the optimum of the basin. In contrast, gradient descent has the opposite characteristics. To get the best of both (Yao 1999) we should evolve initial weights and then train them with gradient descent. Floreano (2008) claims that this can be two orders of magnitude faster than beginning with random initial weights.

Evolving Architectures

Architecture has an important impact on performance and can determine whether a NN under- or over-fits. Designing architectures by hand is a tedious, expert, trial-and-error process. Alternatives include constructive NNs, which grow from a minimal network and destructive NNs, which shrink from a maximal network. Unfortunately, both can become stuck in local optima and can only generate certain architectures (Angeline et al. 1994). Another alternative is to evolve architectures. Miller et al. (1989) make the following suggestions (quoted from Yao (1999)) as to why EAs should be suitable for searching the space of architectures.

- ▶ The surface is infinitely large since the number of possible nodes and connections is unbounded.
- ▶ The surface is nondifferentiable since changes in the number of nodes or connections are discrete and can have a discontinuous effect on EANN's performance.
- ▶ The surface is complex and noisy, since the mapping from an architecture to its performance is indirect, strongly epistatic, and dependent on the evaluation method used.
- ▶ The surface is deceptive since similar architectures may have quite different performance.
- ▶ The surface is multimodal since different architectures may have similar performance.

There are good reasons to evolve architectures and weights simultaneously. If we learn with gradient descent there is a many-to-one mapping from NN genotypes to phenotypes (Yao and Liu 1997). Random initial weights and stochastic learning lead to different outcomes, which makes fitness evaluation noisy, and necessitates averaging over multiple runs, which means the process is slow. On the other hand, if we evolve architectures and weights simultaneously we have a one-to-one genotype to phenotype mapping, which avoids the problem above and results in faster learning. Furthermore, we can co-optimize other parameters of the network (Floreano 2008) at the same time. For example, Belew et al. (1992) found the best networks had a very high learning rate which may have been optimal due to many factors such as initial weights, training order, and amount of training. Without co-optimizing, architecture and weights evolution would not have been able to take all factors into account at the same time.

Evolving Learning Rules (Yao 1999)

There is no one best learning rule for all architectures or problems. Selecting rules by hand is difficult and if we evolve the architecture then we do not know *a priori* what it will be.

A way to deal with this is to evolve the learning rule, but we must be careful: the architectures and problems used in learning the rules must be representative of those to which it will eventually be applied. To get general rules, we should train on general problems and architectures, not just one kind. On the other hand, to obtain a training rule specialized for a specific architecture or problem type, we should train just on that architecture or problem.

One approach is to evolve only learning rule parameters (Yao 1999) such as the learning rate and momentum in backpropagation. This has the effect of adapting a standard learning rule to the architecture or problem at hand. Non-evolutionary methods of adapting training rules also exist. Castillo et al. (2007), working with multi-layer perceptrons, found evolving the architecture, initial weights, and rule parameters together as good or better than evolving only the first two or the third.

We can also evolve new learning rules (Yao 1999; Radi and Poli 2003). Open-ended evolution of rules was initially considered impractical and instead Chalmers (1990) specified a generic, abstract form of update and evolved its parameters to produce different concrete rules. The generic update was a linear function of ten terms, each of which had an associated evolved real-valued weight. Four of the terms represented local information for the node being updated while the other six terms were the pairwise products of the first four. Using this method Chalmers was able to rediscover the delta rule and some of its variants. This approach has been used by a number of others and has been reported to outperform human-designed rules (Dasdan and Oflazer 1993). More recently, GP was used to evolve novel types of rules from a set of mathematical functions and the best new rules consistently outperformed standard backpropagation (Radi and Poli 2003). Whereas architectures are fixed, rules could potentially change over their lifetime (e.g., their learning rate could change) but evolving dynamic rules would naturally be much more complex than evolving static ones.

3.4.1 Ensembles of NNs

Most methods output a single NN (Yao and Islam 2008) but a population of evolving NNs is naturally treated as an ensemble and recent work has begun to do so. Evolving NNs is inherently multiobjective: we want accurate yet simple and diverse networks. Some works combine these objectives into one fitness function while others are explicitly multiobjective.

Single-Objective Ensembles

Yao and Liu (1998) used EPNet's (Yao and Liu 1997) population as an ensemble without modifying the evolutionary process. By treating the population as an ensemble, the result outperformed the population's best individual. Liu and Yao (1990) pursued accuracy and diversity in two ways. The first was to modify backpropagation to minimize error and maximize diversity using an approach they call negative correlation learning (NCL) in which the errors of members become negatively correlated and hence diverse. The second method was to combine accuracy and diversity in a single objective. Evolutionary ensembles for NCL (EENCL) (Liu et al. 2000) automatically determines the size of an ensemble. It encourages diversity with fitness sharing and NCL, and it deals with the ensemble member selection problem (● Sect. 3.3.1) with a cluster-and-select method (see Jin and Sendhoff (2004)). First we cluster candidates, based on their errors, on the training set so that clusters of

candidates make similar errors. Then we select the most accurate in each cluster to join the ensemble; the result is the ensemble can be much smaller than the population. Cooperative neural net ensembles (CNNE) (Islam et al. 2003) used a constructive approach to determine the number of individuals and how many hidden nodes each has. Both contribute to the expressive power of the ensemble and CNNE was able to balance the two to obtain suitable ensembles. Unsurprisingly, it was found that more complex problems needed larger ensembles.

Multiobjective Ensembles

Memetic pareto artificial NN (MPANN) (Abbass 2003) was the first ensemble of NNs to use multiobjective evolution. It also uses gradient-based local search to optimize network complexity and error. Diverse and accurate ensembles (DIVACE) (Chandra and Yao 2006a) uses multi-objective evolution to maximize accuracy and diversity. Evolutionary selection is based on non-dominated sorting (Srinivas and Deb 1994), a cluster-and-select approach is used to form the ensemble, and search is provided by simulated annealing and a variant of differential evolution (Storn and Price 1996). DIVACE-II (Chandra and Yao 2006) is a heterogeneous multi-objective Michigan approach using NNs, support vector machines, and radial basis function nets. The role of crossover and mutation is played by bagging (Breiman 1996) and boosting (Freund and Schapire 1996), which produce accurate and diverse candidates. Each generation bagging and boosting make candidate ensemble members and only dominated members are replaced. The accuracy of DIVACE-II was very good compared to 25 other learners on the Australian credit card and diabetes datasets and it outperformed the original DIVACE.

3.4.2 Yao's Framework for Evolving NNs

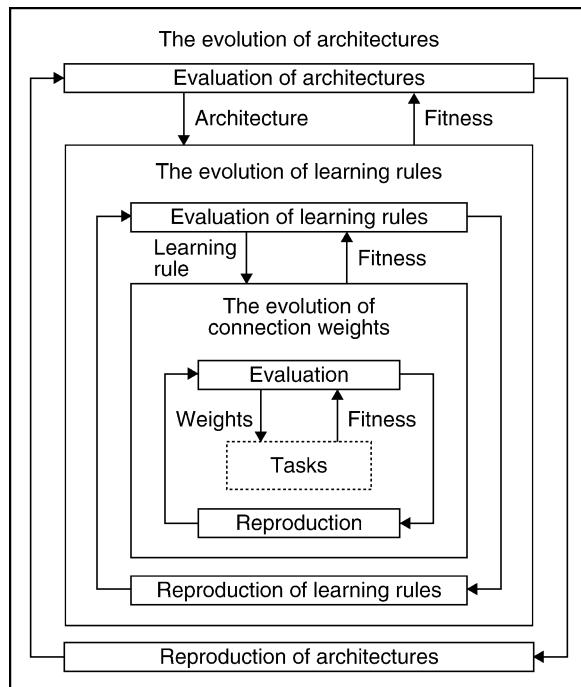
❶ *Figure 7* shows Yao's framework for evolving architectures, training rules, and weights as nested processes (Yao 1999). Weight evolution is the innermost as it occurs at the fastest time scale, while either rule or architecture evolution is outermost. If we have prior knowledge, or are interested in a specific class of either rule or architecture, this constrains the search space and Yao suggests the outermost should be the one which constrains it most. The framework can be thought of as a three-dimensional space of evolutionary NNs where 0 on each axis represents one-shot search and infinity represents exhaustive search. If we remove references to EAs and NNs, it becomes a general framework for adaptive systems.

3.4.3 Conclusions

Evolution is widely used with NNs, indeed according to Floreano et al. (2008) most studies of neural robots in real environments use some form of evolution. Floreano et al. go on to claim that evolving NNs can be used to study “brain development and dynamics because it can encompass multiple temporal and spatial scales along which an organism evolves, such as genetic, developmental, learning, and behavioral phenomena. The possibility to coevolve both the neural system and the morphological properties of agents . . . adds an additional valuable perspective to the evolutionary approach that cannot be matched by any other approach.” (Floreano et al. 2008, p. 59).

Fig. 7

Yao's framework for evolving architectures, training rules, and weights.



Reading

Key reading on evolving NNs includes Yao's classic 1999 survey (Yao 1999), Kasabov 2007 book (Kasabov 2007), Floreano, Dürr, and Mattiussi's 2008 survey (Floreano et al. 2008), which includes reviews of evolving dynamic and neuromodulatory NNs, and Yao and Islam's 2008 survey of evolving NN ensembles (Yao and Islam 2008).

3.5 Learning Classifier Systems (LCS)

Learning classifier systems (LCS) originated in the GA community as a way of applying GAs to learning problems. The LCS field is one of the oldest, largest, and most active areas of GBML. The majority of LCS research is currently carried out on XCS (Wilson 1995; Butz and Wilson 2001) and its derivatives XCSF (Wilson 2001b, 2002a) for regression/function approximation, and UCS (Bernadó-Mansilla and Garrell-Guiu 2003; Orriols-Puig and Bernadó-Mansilla 2008) for supervised learning.

The Game of the Name

Terminology has been contentious in this area (Heitkötter and Beasley 2001). LCS are also widely simply called classifier systems (abbreviated CS or CFS) and sometimes evolutionary (learning) classifier systems. At one time GBML referred exclusively to LCS. None of these names is very satisfactory but the field appears to have settled on LCS.

The difficulty in naming the field relates in part to the difficulty in defining what an LCS is (Smith 1992; Holland et al. 2000). In practice, what is accepted as an LCS has become more inclusive over the years. A reasonable definition of an LCS would be an evolutionary rule-based system – except that a significant minority of LCS are not evolutionary! On the other hand, most non-evolutionary rule-based systems are not considered LCS, so the boundaries of the field are defined more by convention than principle. Even EA practitioners are far from unanimous; work continues to be published which some would definitely consider forms of LCS, but which make no reference to the term and which contain few or no LCS references.

(L)CS has at times been taken to refer to Michigan systems only (see, e.g., Greene and Smith (1993)) but it now generally includes Pitt systems as well, as implied by the name and content of the international workshop on learning classifier systems (IWLCS) which includes both Pitt and Michigan, evolutionary and non-evolutionary systems. As a final terminological note, rules in LCS are often referred to as “classifiers”.

3.5.1 Production Systems and Rule(Set) Parameters

LCS evolve condition-action (IF–THEN) rules. Recall from [Sect. 2.2](#) and [Fig. 2](#) that in Michigan rule-based systems a chromosome is a single rule, while in Pittsburgh systems a chromosome is a variable-length *set* of rules. Pittsburgh, Michigan, IRL, and GCCL are all used. Michigan systems are rare elsewhere but are the most common form of LCS. Within LCS, IRL is most common with fuzzy systems, but see Aguilar-Ruiz et al. (2003) for a non-fuzzy version. In LCS, we typically evolve rule conditions and actions although non-evolutionary operators may act upon them. In addition, each phenotype has parameters associated with it and these parameters are typically learned rather than evolved using the Widrow–Hoff update or similar (see Lanzi et al. (2006b) for examples). In Michigan LCS parameters are associated with each rule but in Pittsburgh systems they are associated with each ruleset. For example, in UCS the parameters are: fitness, mean action set size (to bias a deletion process, which seeks to balance action set sizes), and experience (a count of the number of times a rule has been applied, in order to estimate confidence in its fitness). In GAssist (a supervised Pittsburgh system) the only parameter is fitness. Variations of the above exist; in some cases, rules predict the next input or read and write to memory.

3.5.2 Representing Conditions and Actions

The most common representation in LCS uses fixed-length strings with binary inputs and outputs and ternary conditions. In a simple Michigan version (see, e.g., Wilson (1995)), each rule has one action and one condition from $\{0, 1, \#\}$ where # is a wildcard, matching both 0 and 1 in inputs. For example, the condition 01# matches two inputs: 010 and 011. Similar representations were used almost exclusively prior to approximately 2000 and are inherited from GAs and their preference for minimal alphabets. (Indeed, ternary conditions have an interesting parallel with ternary schemata (Reeves and Rowe 2002) for binary GAs.) Such rules individually have limited expressive power (Schuurmans and Schaeffer 1989) (but see also Booker (1991)), which necessitates that solutions are sets of rules. More insidiously, the lack of individual expressiveness can be a factor in pathological credit assignment (strong/fit overgeneralis

(Kovacs 2004)). Various extensions to the simple scheme described above have been studied (see (Kovacs 2004, Sect. 2.2.2)).

Real-Valued Intervals

Following Bacardit (2004, p. 84) we distinguish two approaches to real-valued interval representation in conditions. The first is representations based on discretization: HIDER* uses *natural coding* (Giraldez et al. 2003), ECL clusters attribute values and evolves constraints on them (Divina et al. 2003) while GAssist uses *adaptive discretization intervals* (Bacardit 2004). The second approach is to handle real values directly. In HIDER genes specify a lower and upper bound (where lower is always less than upper) (Aguilar-Ruiz et al. 2003). In Corcoran and Sen (1994) a variation of HIDER's scheme is used where the attribute is ignored when the upper bound is less than the lower. Interval representations are also used in Wilson (2001a) and Stone and Bull (2003). Finally, Wilson (2000) specifies bounds using centre and spread genes.

Default/Exception Structures

Various forms of default/exception rule structures have been used with LCS. It has been argued that they should increase the number of solutions possible without increasing the search space and should allow gradual refinement of knowledge by adding exceptions (Holland 1986). However, the space of *combinations* of rules is much larger than the set of rules and the evolutionary dynamics of default/exception rule combinations have proved difficult to manage in Michigan systems. Nonetheless, default rules can significantly reduce the number of rules needed for a solution (Valenzuela-Rendón 1989) and there have been some successes.

➤ *Figure 8* illustrates three representations for a Boolean function. The leftmost is a truth table, which lists all possible inputs and their outputs. The middle representation is the ternary language commonly used by LCS, which requires only four rules to represent the eight input/output pairs, thanks to the generalization provided by the # symbol. Finally, on the right, a default rule ($\#\# \rightarrow 1$) has been added to the ternary representation. This rule matches all inputs and states that the output is always 1. This rule is incorrect by itself, but the two rules above it provide exceptions and, taken together, the three accurately represent the function using one less rule than the middle representation. One difficulty in evolving such

■ Fig. 8

Three representations for the 3 multiplexer function.

Truth table			
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ternary rules
0 0 # → 0
0 1 # → 1
1 # 0 → 0
1 # 1 → 1

Default rule
0 0 # → 0
1 # 0 → 0
→ 1

default/exception structures lies in identifying which rules are the defaults and which the exceptions; a simple solution is to maintain the population in some order and make earlier rules exceptions to later ones (as in a decision list (Rivest 1987)). This is straightforward in Pitt systems in which individual rulesets are static but is more complex in Michigan populations in which individual rules are created and deleted dynamically. The other issue is how to assign credit to the overall multi-rule structure. In Pittsburgh systems this is again straightforward since fitness is assigned only at the level of rulesets, but in Michigan systems each rule has a fitness, and it is not obvious how to credit the three rules in the default/exception structure in a way which recognizes their cooperation.

The Pittsburgh GABIL (De Jong and Spears 1991) and GAssist (Bacardit 2004) use decision lists and often evolve default rules spontaneously (e.g., a fully general last rule). Bacardit found that enforcing a fully general last rule in each ruleset in GAssist (and allowing evolution to select the most useful class for such rules) was effective (Bacardit 2004).

In Michigan systems, default/exception structures are called default hierarchies. Rule specificity has been used as the criterion for determining which rules are exceptions and accordingly conflict resolution methods have been biased according to specificity. There are, however, many problems with this approach (Smith and Goldberg 1991). It is difficult for evolution to produce these structures since they depend on cooperation between otherwise competing rules. The structures are unstable since they are interdependent; unfortunate deletion of one member alters the utility of the entire structure. As noted, they complicate credit assignment and conflict resolution since exception rules must override defaults (Wilson 1989; Smith and Goldberg 1991). There are also problems with the use of specificity to prioritize rules. For one, having fewer #s does not mean a rule actually matches fewer inputs; counting #s is a purely syntactic measure of generality. For another, there is no reason why exception rules should be more specific. The consequence of these difficulties is that there has not been much interest in Michigan default hierarchies since the early 1990s (but see Vallim et al. (2003)) and indeed not all Michigan LCS support them (e.g., ZCS (Wilson 1995), XCS/XCSF and UCS do not). Nonetheless, the idea should perhaps be revisited and an ensembles perspective might prove useful.

Other Representations for Conditions and Actions

A great range of other representations has been used, particularly in recent years. These include VL₁ logic (Michalski et al. 1986) as used in GIL (Janikow 1993), first-order logic (Mellor 2005a, b, 2008), decision lists as used in GABIL (De Jong and Spears 1991) and GAssist (Bacardit 2004), messy encoding (Lanzi 1999a), ellipses (Butz 2005) and hyperellipses (Butz et al. 2006), hyperspheres (Marshall and Kovacs 2006), convex hulls (Lanzi and Wilson 2006a), tile coding (Lanzi et al. 2006) and a closely related hyperplane coding (Booker 2005a, b), GP trees (Ahluwalia and Bull 1999; Lanzi 1999b, 2001), Boolean networks defined by GP (Bull 2009), support vectors (Loiacono et al. 2007), edges of an augmented transition network (Landau et al. 2005), gene expression programming (Wilson 2008), fuzzy rules (see **• Sect. 3.6**), and neural networks (Smith and Cribbs 1994; Cribbs and Smith 1996; Smith and Cribbs 1997; Bull and O’Hara 2002; O’Hara and Bull 2005; Dam et al. 2008; Howard et al. 2008; Howard and Bull 2008). GALE (Llorá and Garrell 2001; Llorá 2002; Llorá and Wilson 2004) has used particularly complex representations, including the use of GP to evolve trees defining axis-parallel and oblique hyper-rectangles (Llorá and Wilson 2004), and evolved prototypes, which are used with a k-nearest-neighbor classifier. The prototypes need not be fully specified; some attributes can be left undefined. This representation has also been used in

GAssist (Bacardit 2004). There has been limited work with alternative action representations including computed actions (Tran et al. 2007; Lanzi and Loiacono 2007) and continuous actions (Wilson 2007).

3.5.3 Evolutionary Selection of Representations

As we have seen, there are many representations to choose from. Unfortunately, it is generally not clear which might be best for a given problem or part of a problem. One approach is to let evolution make these choices. This can be seen as a form of meta-learning in which evolution adapts the inductive bias of the learner. In Bacardit (2004) and Bacardit et al. (2007), evolution was used to select default actions in decision lists in GAssist. GAssist's initial population was seeded with last rules, which together advocated all possible classes and over time evolution selected the most suitable of these rules. To obtain good results, it was necessary to encourage diversity in default actions. In GALE (Llorá 2002), evolution selects both classification algorithms and representations. GALE has elements of cellular automata and artificial life: individuals are distributed on a two-dimensional grid. Only neighbors within r cells interact: two neighbors may perform crossover, an individual may be cloned and copied to a neighboring cell, and an individual may die if its neighbors are fitter. A number of representations have been used: rule sets, prototypes, and decision trees (orthogonal, oblique, and multivariate based on nearest neighbor). Decision trees are evolved using GP while prototypes are used by a k-nearest-neighbor algorithm to select outputs. An individual uses a particular representation and classification algorithm and hence evolutionary selection operates on both. Populations may be homogeneous or heterogeneous and in Llorá and Wilson (2004) GALE was modified to interbreed orthogonal and oblique trees.

In the representational ecology approach (Marshall and Kovacs 2006) condition shapes were selected by evolution. Two Boolean classification tasks were used: a plane function, which is easy to describe with hyperplanes, but hard with hyperspheres, and a sphere function, which has the opposite characteristics. Three versions of XCS were used: with hyperplane conditions (XCS-planes), with hyperspheres (XCS-spheres), and both (XCS-both). XCS was otherwise unchanged, but in XCS-both the representations compete due to XCS's pressure against overlapping rules (Kovacs 2004). In XCS-both planes and spheres do not interbreed; they constitute genetically independent populations, that is, species. In terms of classification accuracy XCS-planes did well on the plane function and poorly on the sphere function while XCS-sphere showed the opposite results. XCS-both performed well on both; there was no significant difference in accuracy compared to the better single-representation version on each problem. Furthermore, XCS-both selected the more appropriate representation for each function. In terms of the amount of training needed, XCS-both was similar to XCS-sphere on the sphere function but was significantly slower than XCS-plane on the plane function.

Selecting Discretization Methods and Cut Points

GAssist's Adaptive Discretization Intervals (ADI) approach has two parts (Bacardit 2004). The first consists of adapting interval sizes. To begin, a discretization algorithm proposes cut points for each attribute and this defines the finest discretization possible, said to be composed of micro-intervals. Evolution can merge and split macro-intervals, which are composed of micro-intervals, and each individual can have different macro-intervals. The second part consists of selecting discretization algorithms. Evolution is allowed to select discretization

algorithms for each attribute or rule from a pool including uniform width, uniform frequency, ID3, Fayyad and Irani, M  ntaras, USD, ChiMerge, and random. Unfortunately, evolving the best discretizers was found to be difficult and the use of ADI resulted in only small improvements in accuracy. However, further work was suggested.

3.5.4 Optimization of Population Representation: Macroclassifiers

Wilson (1995) introduced an optimization for Michigan populations called macroclassifiers. He noted that as an XCS population converges on the solution set, many identical copies of this set accumulate. A macroclassifier is simply a rule with an additional *numerosity* parameter, which indicates how many identical virtual rules it represents. Using macroclassifiers saves a great deal of run-time compared to processing a large number of identical rules. Furthermore, macroclassifiers provide interesting statistics on evolutionary dynamics. Empirically, macroclassifiers perform essentially as the equivalent “micro” classifiers (Kovacs 1996).  Figure 9 illustrates how the rules m and m' in the top can be represented by m alone in the bottom by adding a numerosity parameter.

3.5.5 Rule Discovery

LCS are interesting from an evolutionary perspective, particularly Michigan systems in which evolutionary dynamics are more complex than in Pittsburgh systems. Where Pittsburgh systems face two objectives (evolving accurate and parsimonious rulesets) Michigan systems face a third: coverage of the input (or input/output) space. Furthermore, Michigan systems have coevolutionary dynamics as rules both cooperate and compete. Since the level of selection (rules) is lower than the level of solutions (rulesets) researchers have attempted to coax better results by modifying rule fitness calculations with various methods. Fitness sharing and crowding have been used to encourage diversity and, hence, coverage. Fitness sharing is naturally based on inputs (see ZCS) while crowding has been implemented by making deletion probability proportional to the degree of overlap with other rules (as in XCS). Finally, restricted mating as implemented by a niche GA plays an important role in XCS and UCS (see  Sect. 3.5.5).

 Fig. 9

A population of microclassifiers (top) and the equivalent macroclassifiers (bottom).

Rule	Cond.	Action	Strength
m	# # 0 0 1 1	1	200.0
m'	# # 0 0 1 1	1	220.0
n	# # 0 0 1 1	0	100.0
o	0 0 1 1 1 0	1	100.0

Rule	Cond.	Action	Strength	Numerosity
m	# # 0 0 1 1	1	200.0	2
n	# # 0 0 1 1	0	100.0	1
o	0 0 1 1 1 0	1	100.0	1

Windowing in Pittsburgh LSS

As noted in [Sect. 2.2](#) naive implementations of the Pittsburgh approach are slower than Michigan systems, which are themselves slow compared to non-evolutionary methods. The naive Pitt approach is to evaluate each individual on the entire data set, but much can be done to improve this. Instead, windowing methods (Fürnkranz 1998) learn on subsets of the data to improve runtime. Windowing has been used in Pitt LCS since, at least, ADAM (Greene and Smith 1987). More recently GAssist used incremental learning by alternating strata (ILAS) (Bacardit 2004) which partitions the data into n strata, each with the same class distribution as the entire set. A different stratum is used for fitness evaluation each generation. On larger data sets speed-up can be an order of magnitude. Windowing has become a standard part of recent Pittsburgh systems applied to real-world problems (e.g., Bacardit and Krasnogor (2008) and Bacardit et al. (2008, 2009)).

Many ensemble methods improve classification accuracy by sampling data in similar ways to windowing techniques, which suggests the potential for both improved accuracy and runtime, but this has not been investigated in LCS.

Michigan Rule Discovery

Most rule discovery work focuses on Michigan LCS as they are more common and their evolutionary dynamics are more complex. The rest of this section deals with Michigan systems although many ideas, such as self-adaptive mutation, could be applied to Pitt systems. Michigan LCS use the steady state GAs introduced in [Sect. 2.4](#) as they minimize disruption to the rule population during on-line learning. An unusual feature of Michigan LCS is the emphasis placed on minimizing population size, for which various techniques are used: niche GAs, the addition of a generalization term in fitness, subsumption deletion, condensation and various compaction methods.

Niche GAs

Whereas in a standard panmictic GA all rules are eligible for reproduction, in a niche GA mating is restricted to rules in the same action set (which is considered a niche). (See [Fig. 3](#) on action sets.) The input spaces of rules in an action set overlap and their actions agree, which suggests their predictions will tend to be related. Consequently, mating these related rules is more effective, on average, than mating rules drawn from the entire population. This is a form of speciation since it creates non-interbreeding sub-populations. However, the niche GA has many other effects (Wilson 2001). First, a strong bias toward general rules, since they match more inputs and hence appear in more action sets. Second, pressure against overlapping rules, since they compete for reproduction (Kovacs 2004). Third, complete coverage of the input space, since competition occurs for each input. The niche GA was introduced in Booker (1989) and originally operated in the match set but was later further restricted to the action set (Wilson 1998). It is used in XCS and UCS and is related to *universal suffrage* (Giordana and Saitta 1994).

EDAs Instead of GAs

Recently Butz et al. (2005, 2006) and Butz and Pelikan (2006) replaced XCS's usual crossover with an estimation of distribution algorithm (EDA)-based method to improve solving of difficult hierarchical problems, while (Llorà et al. 2005a, b) introduced CCS: a Pitt LCS based on compact GAs (a simple form of EDA).

Subsumption Deletion

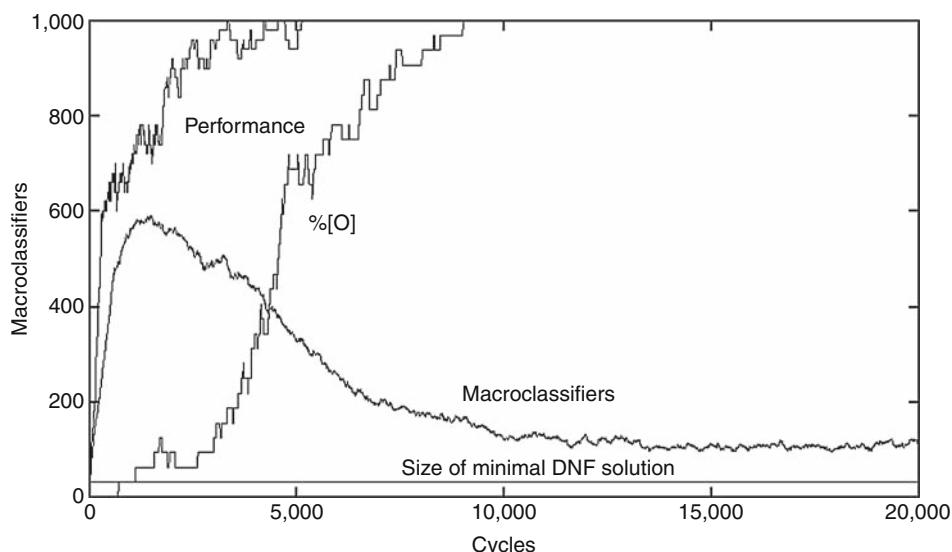
A rule x logically subsumes a rule y when x matches a superset of the inputs y matches and they have the same action. For example, $00\# \rightarrow 0$ subsumes $000 \rightarrow 0$ and $001 \rightarrow 0$. In XCS x is allowed to subsume y if: (i) x logically subsumes y , (ii) x is sufficiently accurate and (iii) x is sufficiently experienced (has been evaluated sufficiently) so we can have confidence in its accuracy. Subsumption deletion was introduced in XCS (see Butz and Wilson (2001)) and takes two forms. In *GA subsumption*, when a child is created, we check to see if its parents subsume it, which constrains accurate parents to only produce more general children. In *action set subsumption*, the most general of the sufficiently accurate and experienced rules in the action set is given the opportunity to subsume the others. This removes redundant, specific rules from the population but is too aggressive for some problems.

Michigan Evolutionary Dynamics

Michigan LCS have interesting evolutionary dynamics and plotting macroclassifiers is a useful way to monitor population convergence and parsimony. [Figure 10](#) illustrates by showing XCS learning the 11 multiplexer function. The performance curve is a moving average of the proportion of the last 50 inputs which were classified correctly, $\%[O]$ shows the proportion of the minimal set of 16 ternary rules XCS needs to represent this function (indicated by the straight line labeled “Size of minimal DNF solution” in the figure) and macroclassifiers were explained in [Sect. 3.5.4](#). In this experiment, the population was initially empty and was seeded by covering [Sect. 3.5.5](#). “Cycles” refers to the number of inputs presented, inputs were drawn uniform randomly from the input space, the population size limit was 800, all input/output pairs were in both the train and test sets, GA subsumption was used but action set subsumption was not and curves are the average of 10 runs. Other settings are as in Wilson (1995).

Fig. 10

Evolutionary dynamics of XCS on the 11 multiplexer.



Note that XCS continues to refine its solution (population) after 100% performance is reached and that it finds the minimal representation (at the point where $\%[O]$ reaches the top of the figure) but that continued crossover and mutation generate extra transient rules, which make the population much larger.

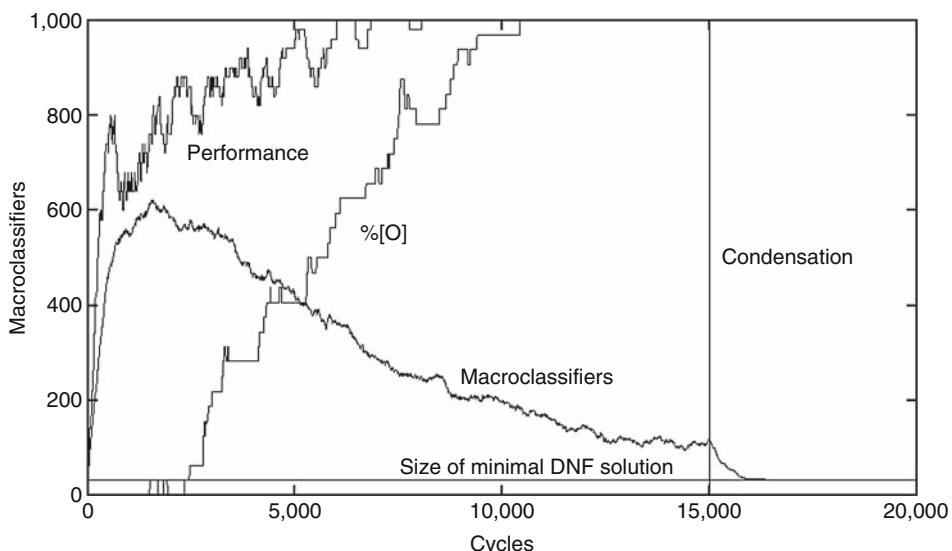
Condensation

As illustrated by [Fig. 10](#), an evolved population normally contains many redundant and low-fitness rules. These rules are typically transient, but more are generated while the GA continues to run. Condensation (Wilson 1998; Kovacs 1996) is a very simple technique to remove such rules, which consists of running the system with crossover and mutation turned off; we only clone and delete existing rules. [Figure 11](#) repeats the experiment from [Fig. 10](#) but switches after 15,000 cycles to condensation after which the population quickly converges to the minimal solution. Other methods of compacting the population have been investigated (Kovacs 1997; Wilson 2002b; Dixon et al. 2002).

Tuning Evolutionary Search

XCS is robust to class imbalances (Orriols-Puig and Bernadó-Mansilla 2006) but for very high imbalances tuning the GA based on a facetwise model improved performance (Orriols-Puig and Bernadó-Mansilla 2006; Orriols-Puig et al. 2007b). Self-tuning evolutionary search has also been studied. The mutation rate can be adapted during evolution for example Hurst and Bull (2003, 2004), Howard et al. (2008), and Butz et al. (2008), while De Jong et al. (1993) dynamically controls use of two generalization operators: each has a control bit specifying whether it can be used and control bits evolve with the rest of the genotypes.

Fig. 11
XCS with condensation on the 11 multiplexer.



Non-evolutionary Rule Discovery

Evolution has been supplemented by heuristics in various ways. Covering, first suggested in Holland (1976), creates a rule to match an unmatched input. It can be used to create (“seed”) the initial population (Venturini 1993; Wilson 1995; Hekanaho 1995) or to supplement the GA throughout evolution (Wilson 1995). Kovacs (2004, p. 42) found covering each action set was preferable to covering the match set when applying XCS to sequential tasks. Most covering/seeding is done as needed but instead (Juan Liu and Tin-Yau Kwok 2000) selects inputs at the center of same-class clusters. For other non-evolutionary operators see (Booker 1989; Riolo 1987), the work on corporations of rules (Wilson and Goldberg 1989; Smith 1994; Tomlinson and Bull 1998, 2002; Tomlinson 1999) and the work on non-evolutionary LCS.

Non-evolutionary LCS

Although LCS were originally conceived as a way of applying GAs to learning problems (Holland and Reitman 1978), not all LCS include a GA. Various heuristics have been used to create and refine rules in for example YACS (Gerard et al. 2002) and MACS (Gérard and Sigaud 2003). A number of systems have been inspired by psychological models of learning. ACS (Stolzmann 1996; Butz 2002) and ACS2 (Butz 2002a) are examples, although ACS was also later supplemented by a GA (Butz et al. 2000a, b). Another is AgentP, a specialized LCS for maze tasks (Zatuchna 2004, 2005).

3.5.6 LCS Credit Assignment

While credit assignment in Pittsburgh LCS is a straightforward matter of multiobjective fitness evaluation, as noted in [Sect. 3.5.5](#) it is far more complex in Michigan systems with their more complex evolutionary dynamics. Credit assignment is also more complex in some learning paradigms, particularly reinforcement learning, which we will not cover here. Within supervised learning, credit assignment is more complex in regression tasks than in classification. These difficulties have been the major issue for Michigan LCS and have occupied a considerable part of the literature, particularly prior to the development of XCS, which provided a reasonable solution for both supervised and reinforcement learning.

Strength and Accuracy in Michigan LCS

Although we are not covering reinforcement learning, Michigan LCS have traditionally been designed for such problems. XCS/XCSF are reinforcement learning systems, but since supervised learning can be formulated as simplified reinforcement learning, they have been applied to SL tasks. Consequently, we now very briefly outline the difference between the two major forms of Michigan reinforcement learning LCS.

In older (pre-1995) reinforcement learning, LCS fitness is proportional to the magnitude of reward and is called *strength*. Strength is used both for conflict resolution and as fitness in the GA (see e.g., ZCS (Wilson 1994)). Such LCS are referred to as strength-based and they suffer from many difficulties with credit assignment (Kovacs 2004), the analysis of which is quite complex. Although some strength-based systems incorporate accuracy as a component of fitness, their fitness is still proportional to reward. In contrast, the main feature of XCS is that it adds a prediction parameter, which estimates the reward to be obtained if the action advocated by a rule is taken. Rule fitness is proportional to the accuracy of reward prediction and not to its magnitude, which avoids many problems strength-based systems have with

credit assignment. In XCS, accuracy is estimated from the variance in reward and since overgeneral rules have high variance they have low fitness. Although XCS has proved robust in a range of applications, a major limitation is that the accuracy estimate conflates several things: (i) overgenerality in rules, (ii) noise in the training data, and (iii) stochasticity in the transition function in sequential problems. In contrast, strength-based systems may be less affected by noise and stochasticity since they are little affected by reward variance. See Kovacs (2004) for analysis of the two approaches.

Prediction Updates

To update rule predictions while training, the basic XCS system (Wilson 1995; Butz and Wilson 2001) uses the Widrow–Hoff update for nonsequential problems and the Q-learning update for sequential ones. Various alternatives have been used: average rewards (Tharakunnel and Goldberg 2002; Lanzi and Loiacono 2006), gradient descent (Butz et al. 2005b; Lanzi et al. 2007), and eligibility traces (Drugowitsch and Barry 2005). The basic XCSF uses NLMS (linear piecewise) prediction (Wilson 2000b, 2002a) but Lanzi et al. (2006b) has compared various alternative classical parameter estimation (RLS and Kalman filter) and gain adaptation algorithms (K1, K2, IDBD, and IDD). He found that Kalman filter and RLS have significantly better accuracy than the others and that Kalman filter produces more compact solutions than RLS. There has also been recent work on other systems; UCS is essentially a supervised version of XCS and the main difference is its prediction update. Bull has also studied simplified LCS (Bull 2005).

Evolutionary Selection of Prediction Functions

In Lanzi et al. (2008), Lanzi selects prediction functions in XCSFHP (XCSF with Heterogeneous Predictors) in a way similar to the selection of condition types in the representational ecology approach in [Sect. 3.5.3](#). Polynomial functions (linear, quadratic, and cubic) and constant, linear and NN predictors were available. XCSFHP selected the most suitable predictor for regression and sequential tasks and performed almost as well as XCSF using the best single predictor.

Theoretical Results

Among the notable theoretical works on LCS, Lanzi (2002a) demonstrates that XCS without generalization implements tabular Q-learning, Butz et al. (2005a) investigate the computational complexity of XCS in a probably approximately correct (PAC) setting, and Wada et al. (2005a, b, c, 2007) analyze credit assignment and relate LCS to mainstream reinforcement learning methods. Kovacs (2004) identifies pathological rule types: strong overgeneral and fit overgeneral rules, which are overgeneral yet stronger/fitter than not-overgeneral competitors. Fortunately, such rules are only possible under specific circumstances. A number of papers seek to characterize problems which are hard for LCS (Goldberg et al. 1992; Kovacs 2000, 2001, 2004; Bernadó-Mansilla and Ho 2005; Bagnall and Zatuchna 2005) while others model evolutionary dynamics (Butz et al. 2004a, b, 2007; Butz 2006; Orriols-Puig 2007c, d) and yet others attempt to reconstruct LCS from first principles using probabilistic models (Drugowitsch 2007, 2008; Edakunni et al. 2009).

Hierarchies and Ensembles of LCS

Hierarchical LCS have been studied for some time and Barry (1996) reviews early work. Dorigo and Colombetti (1998), Donnart and Meyer (1996a, b), and Donnart (1998) apply hierarchical LCS to robot control while Barry (2000) uses hierarchical XCSs to learn long

sequences of actions. The ensembles field (● Sect. 3.3) studies how to combine predictions (Kuncheva 2004) and all the above could be reformulated as ensembles of LCS. Some recent work has taken this approach (Dam et al. 2005; Bull et al. 2007).

3.5.7 Conclusions

LCS face certain inherent difficulties; Michigan systems face complex credit assignment problems while in Pittsburgh systems, run-time can be a major issue. The same is true for all GBML systems, but the Michigan approach has been explored far more extensively within LCS than elsewhere. Recently, there has been much integration with mainstream machine learning and much research on representations and credit assignment algorithms. Most recent applications have been to data mining and function approximation, although some work continues on reinforcement learning. Future directions are likely to include exposing more of the LCS to evolution and further integration with machine learning, ensembles, memetic algorithms, and multiobjective optimization.

Reading

No general up-to-date introduction to LCS exists. For the basics see Goldberg (1989) and the introductory parts of Kovacs (2004) or Butz (2006). For a good introduction to representations and operators see chapter 6 of Freitas (2002a). For a review of early LCS see Barry (2000). For reviews of LCS research see Wilson and Goldberg (1989), Lanzi and Riolo (2000), and Lanzi (2008). For a review of state-of-the-art GBML and empirical comparison to non-evolutionary pattern recognition methods see Orriols-Puig et al. (2008b). For other comparisons with non-evolutionary methods see Bonelli and Alexandre (1991), Greenyer (2000), Saxon and Barry (2000), Wilson (2000), Bernadó et al. (2002), and Bernadó-Mansilla and Garrell-Guiu (2003). Finally, the LCS bibliography (Kovacs 2009) has over 900 references.

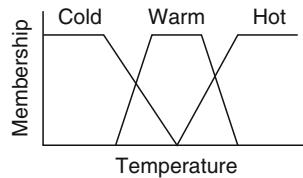
3.6 Genetic Fuzzy Systems

Following the section on LCS, this section covers a second actively developing approach to evolving rule-based systems. We will see that the two areas overlap considerably and that the distinction between them is somewhat arbitrary. Nonetheless, the two communities and their literatures are somewhat disjoint.

Fuzzy logic is a major paradigm in soft computing which provides a means of approximate reasoning not found in traditional crisp logic. Genetic fuzzy systems (GFS) apply evolution to fuzzy learning systems in various ways: GAs, GP, and evolution strategies have all been used. We will cover a particular form of GFS called genetic fuzzy rule-based systems (FRBS), which are also known as learning fuzzy classifier systems (LFCS) (Bonarini 2000) or referred to as, for example, “genetic learning of fuzzy rules” and (for reinforcement learning tasks) “fuzzy Q-learning”. Like other LCS, FRBS evolve if-then rules but in FRBS the rules are fuzzy. Most systems are Pittsburgh, but there are many Michigan examples (Valenzuela-Rendón 1991, 1998; Geyer-Schulz 1997; Bonarini 2000; Orriols-Puig et al. 2007a, 2008a; Casillas et al. 2007). In addition to FRBS, we briefly cover genetic fuzzy NNs, but we do not cover genetic fuzzy clustering (see Oscar-Cordón et al. (2001)).

In the terminology of fuzzy logic, ordinary scalar values are called *crisp* values. A *membership function* defines the degree of match between crisp values and a set of fuzzy linguistic

terms. The set of terms is a *fuzzy set*. The following figure shows a membership function for the set {cold, warm, and hot}.



Each crisp value matches *each* term to some degree in the interval [0,1], so, for example, a membership function might define 5° as 0.8 cold, 0.3 warm and 0.0 hot. The process of computing the membership of each term is called fuzzification and can be considered a form of discretization. Conversely, defuzzification refers to computing a crisp value from fuzzy values.

Fuzzy rules are condition/action (IF–THEN) rules composed of a set of linguistic variables (e.g., temperature, humidity), which can each take on linguistic terms (e.g., cold, warm, and hot). For example:

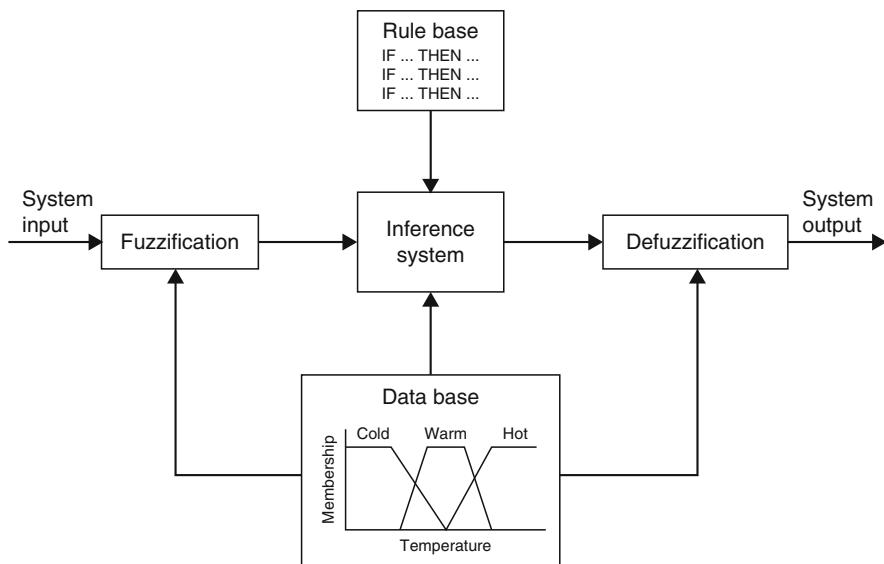
```
IF temperature IS cold AND humidity IS high THEN heater IS high
IF temperature IS warm AND humidity IS low THEN heater IS medium
```

As illustrated in Fig. 12, a fuzzy rule-based system consists of:

- A rule base (RB) of fuzzy rules
- A data base (DB) of linguistic terms and their membership functions
- Together the RB and DB are the *knowledge base* (KB)
- A fuzzy inference system which maps from fuzzy inputs to a fuzzy output
- Fuzzification and defuzzification processes

Fig. 12

Components and information flow in a fuzzy rule-based system. (Adapted from Hekanaho 1995.)



3.6.1 Evolution of FRBSs

We distinguish (i) genetic tuning and (ii) genetic learning of DB, RB, or inference engine parameters.

Genetic Tuning

The concept behind genetic tuning is to first train a hand-crafted FRBS and then to evolve the DB (linguistic terms and membership functions) to improve performance. In other words, we do not alter the hand-crafted rule base but only tune its parameters. Specifically, we can adjust the shape of the membership functions and the parameterized expressions in the (adaptive) inference system and adapt defuzzification methods.

Genetic Learning

The concept of genetic learning is to evolve the DB, RB, or inference engine parameters. There are a number of approaches. In *genetic rule learning*, we usually predefine the DB by hand and evolve the RB. In *genetic rule selection* we use the GA to remove irrelevant, redundant, incorrect, or conflicting rules. This is a similar role to condensation in LCS (see [Sect. 3.5.5](#)). In *genetic KB learning* we learn both the DB and RB. We either learn the DB first and then learn the RB or we iteratively learn a series of DBs and evaluate each one by learning an RB using it.

It is also possible to learn components simultaneously, which may produce better results though the larger search space makes it slower and more difficult than adapting components independently. As examples, Morimoto et al. (1997) learn the DB and RB simultaneously while Homaitar and McCormick (1995) learn KB components and inference engine parameters simultaneously.

Recently, Sánchez and Couso (2007) claimed that all existing GFS have been applied to crisp data and that with such data the benefits of GFS compared to other learning methods are limited to linguistic interpretability. However, GFS has the potential to outperform other methods on fuzzy data and they identify three cases (Sánchez and Couso 2007, p. 558):

1. Crisp data with hand-added fuzziness
2. Transformations of data based on semantic interpretations of fuzzy sets
3. Inherently fuzzy data

They argue that GFS should use fuzzy fitness functions in such cases to deal directly with the uncertainty in the data and propose such systems as a new class of GFS to add to the taxonomy of Herrera (2008).

3.6.2 Genetic Neuro-fuzzy Systems

A neuro-fuzzy system (NFS) or fuzzy neural network (FNN) is any combination of fuzzy logic and neural networks. Among the many examples of such systems, Liangjie and Yanda (1996) use a GA to minimize the error of the NN, Hanebeck and Schmidt (1996) use both a GA and backpropagation to minimize error, Perneel and Thelen (1995) optimize a fuzzy expert system using a GA and NN, and Morimoto et al. (1997) use a NN to approximate the fitness function for a GA, which adapts membership functions and controls rules. See Cordón et al.

(2001) for an introduction to NFS, Linkens and Nyongesa (1996) for a review of EAs, NNs, and fuzzy logic from the perspective of intelligent control, and He et al. (1999) for a discussion of combining the three. Kolman and Margaliot (2009) introduce fuzzy all-permutations rule-bases (FARBs), which are mathematically equivalent to NNs.

3.6.3 Conclusions

Herrera (2008, p. 38) lists the following active areas within GFS:

1. Multi-objective genetic learning of FRBSs: interpretability–precision trade-off
2. GA-based techniques for mining fuzzy association rules and novel data mining approaches
3. Learning genetic models based on low quality data (e.g., noisy data)
4. Genetic learning of fuzzy partitions and context adaptation
5. Genetic adaptation of inference engine components
6. Revisiting the Michigan-style GFSs

Herrera also lists (p. 42) current issues for GFS:

1. Human readability
2. New data mining tasks: frequent and interesting pattern mining, mining data streams
3. Dealing with high dimensional data

Reading

There is a substantial GFS literature. Notable works include the four seminal 1991 papers on genetic tuning of the DB (Karr 1991), the Michigan approach (Valenzuela-Rendón 1989), the Pittsburgh approach (Thrift 1991), and relational matrix-based FRBS (Pham and Karaboga 1991). Subsequent work includes Geyer-Schulz's 1997 book on Michigan fuzzy LCS learning RBs with GP (Geyer-Schulz 1997), Bonarini's 2000 introductory chapter from an LCS perspective (Bonarini 2000), Mitra and Hayashi's 2000 survey of neuro-fuzzy rule generation methods (Mitra and Hayashi 2000), Cordon et al.'s 2001 book on genetic fuzzy systems in general (Cordón et al. 2001), Angelov's 2002 book on evolving FRBS (Angelov 2002), chapter 10 of Freitas' 2002 book on evolutionary data mining (Freitas 2002a), Herrera's 2008 survey article on GFS (Herrera 2008) (which lists further key reading), and finally Kolman and Margaliot's 2009 book on the neuro-fuzzy FARB approach (Kolman and Margaliot 2009).

4 Conclusions

The reader should need no convincing that GBML is a very diverse and active area. Although much integration with mainstream machine learning has taken place in the last 10 years, more is needed. The use of multiobjective EAs in GBML is spreading. Integration with ensembles is natural given the population-based nature of EAs but it is only just beginning. Other areas which need attention are memetics, meta-learning, hyperheuristics, and estimation of distribution algorithms. In addition to further integration with other areas, the constituent areas of GBML need more interaction with each other.

Two persistent difficulties for GBML are worth highlighting. First, run-time speed remains an issue as EAs are much slower than most other methods. While this sometimes matters little

(e.g., in off-line learning with modest datasets), it is sometimes equally critical (e.g., in stream mining). Various methods to speed up GBML exist (see e.g., Freitas (2002a, Sect. 12.1.3) and more research is warranted, but this may simply remain a weakness. The second difficulty is theory. EA theory is notoriously difficult and when coupled with other processes becomes even less tractable. Nonetheless, substantial progress has been made in the past 10 years, most notably with LCS.

Other active research directions will no doubt include meta-learning such as the evolution of bias (e.g., selection of representation), evolving heuristics and learning rules for specific problem classes, and other forms of self-adaptation. In the area of data preparation, Freitas (2002a, Sect. 12.2.1) argues that attribute construction is a promising area for GBML and that filter methods for feature selection are faster than wrappers and deserve more GBML research. Finally, many specialized learning problems (not to mention specific applications) remain little or unexplored with GBML, including ranking, semi-supervised learning, transductive learning, inductive transfer, learning to learn, stream mining, and no doubt others that have not yet been formulated.

Acknowledgments

Thanks to my editor Thomas Bäck for his patience and encouragement, and to Larry Bull, John R. Woodward, Natalio Krasnogor, Gavin Brown and Arjun Chandra for comments.

Glossary

EA	Evolutionary Algorithm
FRBS	Fuzzy Rule-Based System
GA	Genetic Algorithm
GBML	Genetics-Based Machine Learning
GFS	Genetic Fuzzy System
GP	Genetic Programming
LCS	Learning Classifier System
NN	Neural Network
SL	Supervised Learning

References

- Abbass HA (2003) Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Comput*, 15(11):2705–2726
- Ackley DH, Littman ML (1992) Interactions between learning and evolution. In: Langton C, Taylor C, Rasmussen S, Farmer J (eds) *Artificial life II: Santa Fe institute studies in the sciences of Complexity*, vol 10. Addison-Wesley, New York, pp 487–509
- Aguilar-Ruiz J, Riquelme J, Toro M (2003) Evolutionary learning of hierarchical decision rules. *IEEE Trans Syst Man Cybern B* 33(2):324–331
- Ahluwalia M, Bull L (1999) A genetic programming-based classifier system. In: Banzhaf et al. (eds) *GECCO-99: Proceedings of the genetic and evolutionary computation conference*. Morgan Kaufmann, San Francisco, pp 11–18

- Andersen HC, Tsoi AC (1993) A constructive algorithm for the training of a multi-layer perceptron based on the genetic algorithm. *Complex Syst*, 7(4):249–268
- Angeline PJ, Sauders GM, Pollack JB (1994) An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans Neural Netw* 5:54–65
- Angelov P (2002) Evolving rule-based models: a tool for design of flexible adaptive systems. *Studies in fuzziness and soft computing*, vol 92. Springer, Heidelberg
- Asuncion A, Newman DJ (2009) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/ML_Repository.html
- Bacardit J, Stout M, Hirst JD and Krasnogor N (2008) Data mining in proteomics with learning classifier systems. In: Bull L, Bernadó Mansilla E, Holmes J (eds) *Learning classifier systems in data mining*. Springer, Berlin, pp 17–46
- Bacardit J, Burke EK, Krasnogor N (2009a) Improving the scalability of rule-based evolutionary learning. *Memetic Comput* 1(1):55–57
- Bacardit J, Stout M, Hirst JD, Valencia A, Smith RE, Krasnogor N (2009b) Automated alphabet reduction for protein datasets. *BMC Bioinformatics*, vol 10, 6
- Bacardit J (2004) Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time. PhD thesis, Universitat Ramon Llull, Barcelona, Spain
- Bacardit J, Goldberg DE, Butz MV (2007) Improving the performance of a Pittsburgh learning classifier system using a default rule. In: Kovacs T, Llòra X, Takadama K, Lanzi PL, Stolzmann W, Wilson SW (eds) *Learning Classifier Systems. International workshops, IWLCS 2003–2005, Revised selected papers, Lecture notes in computer science*, vol 4399. Springer, Berlin, pp 291–307
- Bacardit J, Krasnogor N (2008) Empirical evaluation of ensemble techniques for a Pittsburgh learning classifier system. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llòra X, Takadama K (eds) *Learning Classifier Systems. 10th and 11th International Workshops (2006–2007), Lecture notes in computer science*, vol 4998. Springer, Berlin, pp 255–268
- Bagnall AJ, Zatuchna ZV (2005) On the classification of maze problems. In: Bull L, Kovacs T (eds) *Applications of learning classifier systems*. Springer, Berlin, pp 307–316
- Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakielo M, Smith RE (eds) (1999) *GECCO-99: Proceedings of the genetic and evolutionary computation conference*. Morgan Kaufmann, San Francisco, CA
- Barry A (1996) Hierarchy formulation within classifiers system: a review. In: Goodman EG, Uskov VL, Punch WF (eds) *Proceedings of the first international conference on evolutionary algorithms and their application EVCA'96*. The Presidium of the Russian Academy of Sciences, Moscow, pp 195–211
- Barry A (2000) XCS performance and population structure within multiple-step environments. PhD thesis, Queen's University Belfast, Belfast
- Beielstein T, Markon S (2002) Threshold selection, hypothesis tests and DOE methods. In: 2002 congress on evolutionary computation. IEEE Press, Washington, DC, pp 777–782
- Belew RK, McInerney J, Schraudolph NN (1992) Evolving networks: using the genetic algorithm with connectionistic learning. In: Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) *Proceedings of the 2nd conference on artificial life*. Addison-Wesley, New York, pp 51–548
- Bernadó E, Llòra X, Garrell JM (2002) XCS and GALE: a comparative study of two learning classifier systems on data mining. In: Lanzi PL, Stolzmann W, Wilson SW (eds) *Advances in learning classifier systems, Lecture notes in artificial intelligence*, vol 2321. Springer, Berlin, pp 115–132
- Bernadó-Mansilla E, Garrell-Guiu JM (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolut Comput* 11(3):209–238
- Bernadó-Mansilla E, Ho TK (2005) Domain of competence of XCS classifier system in complexity measurement space. *IEEE Trans Evolut Comput* 9(1):82–104
- Bonarini A (2000) An introduction to learning fuzzy classifier systems. In: Lanzi PL, Stolzmann W, Wilson SW (eds) *Learning classifier systems: from foundations to applications, Lecture note in artificial intelligence*, vol 1813. Springer, Berlin, pp 83–104
- Bonelli P, Alexandre P (1991) An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains. In: Booker LB, Belew RK (eds) *Proceedings of the 14th international conference on genetic algorithms (ICGA'91)*. Morgan Kaufmann, San Francisco, CA, pp 288–295
- Booker LB (1989) Triggered rule discovery in classifier systems. In: Schaffer JD (ed) *Proceedings of the 3rd international conference on genetic algorithms (ICGA-89)*. Morgan Kaufmann, San Francisco, CA, pp 265–274
- Booker LB (1991) Representing attribute-based concepts in a classifier system. In: Rawlins GJE (ed) *Proceedings of the first workshop on foundations of genetic algorithms (FOGA91)*. Morgan Kaufmann, San Mateo, CA, pp 115–127
- Booker LB (2005a) Adaptive value function approximations in classifier systems. In: *GECCO '05: proceedings of the 2005 workshops on genetic and evolutionary computation*. ACM, New York, pp 90–91

- Booker LB (2005b) Approximating value functions in classifier systems. In: Bull L, Kovacs T (eds) Foundations of learning classifier systems (Studies in fuzziness and soft computing), Lecture notes in artificial intelligence, vol 183, Springer, Berlin, pp 45–61
- Booker LB, Belew RK (eds) (1991) Proceedings of the 4th international conference on genetic algorithms (ICGA91). Morgan Kaufmann, San Francisco, CA
- Bot MCJ, Langdon WB (2000) Application of genetic programming to induction of linear classification trees. In: Genetic programming: proceedings of the 3rd European conference (EuroGP 2000), Lecture notes in computer science, vol 1802, Springer, Berlin, pp 247–258
- Breiman L (1996) Bagging predictors. *Mach Learn* 24 (2):123–140
- Breiman L (1998) Arcing classifiers. *Ann Stat* 26(3): 801–845
- Brown G (2010) Ensemble learning. In: Sammut C, Webb G (eds) Encyclopedia of machine learning. Springer, Berlin
- Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: A survey and categorisation. *J Inform Fusion* 6(1):5–20
- Bull L (2009) On dynamical genetic programming: simple Boolean networks in learning classifier systems. *IJPEDS* 24(5):421–442
- Bull L, Studley M, Bagnall T, Whittle I (2007) On the use of rule-sharing in learning classifier system ensembles. *IEEE Trans Evolut Comput* 11:496–502
- Bull L (2005) Two simple learning classifier systems. In: Bull L, Kovacs T (eds) Foundations of learning classifier systems (Studies in fuzziness and soft computing), Lecture notes in artificial intelligence, vol 183, Springer, Berlin, pp 63–90
- Bull L, O'Hara T (2002) Accuracy-based neuro and neuro-fuzzy classifier systems. In: Langdon WB, Cantú-Paz E, Mathias K, Roy R, Davis R, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter MA, Schultz AC, Miller JF, Burke E, Jonoska N (eds) GECCO 2002: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Francisco, CA, pp 905–911
- Burke EK, Hyde MR, Kendall G, Ochoa G, Ozcan E, Woodward JR (2009) Exploring hyper-heuristic methodologies with genetic programming. In: Mumford C, Jain L (eds) Collaborative computational intelligence. Springer, Berlin
- Burke EK, Kendall G (2005) Introduction. In: Burke EK, Kendall G (eds) Search methodologies: introductory tutorials in optimization and decision support techniques. Springer, Berlin, pp 5–18
- Burke EK, Kendall G, Newall J, Hart E, Russ P, Schulenburg S (2003) Hyper-heuristics: an emerging direction in modern search technology. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Norwell, MA, pp 457–474
- Butz MV, Kovacs T, Lanzi PL, Wilson SW (2004b) Toward a theory of generalization and learning in XCS. *IEEE Trans Evolut Comput* 8(1):8–46
- Butz MV (2002a) An algorithmic description of ACS2. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems: from foundations to applications, Lecture notes in artificial intelligence, vol 2321. Springer, Berlin, pp 211–229
- Butz MV (2002b) Anticipatory learning classifier systems. Kluwer, Norwell, MA
- Butz MV, Goldberg DV, Stolzmann W (2000a) Introducing a genetic generalization pressure to the anticipatory classifier system – part 1: theoretical approach. In: Whitley D, Goldberg D, Cantú-Paz E, Spector L, Parmee I, Beyer H-G (eds) Proceedings of genetic and evolutionary computation conference (GECCO 2000). Morgan Kaufmann, San Francisco, CA, pp 34–41
- Butz MV, Goldberg DE, Stolzmann W (2000b) Introducing a genetic generalization pressure to the anticipatory classifier system – part 2: performance analysis. In: Whitley D, Goldberg D, Cantú-Paz E, Spector L, Parmee I, Beyer H-G (eds) Proceedings of genetic and evolutionary computation conference (GECCO 2000). Morgan Kaufmann, San Francisco, CA, pp 42–49
- Butz MV, Wilson SW (2001) An algorithmic description of XCS. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Advances in learning classifier systems, Lecture notes in artificial intelligence, vol 1996. Springer, Berlin, pp 253–272
- Butz MV (2005) Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. In: Beyer HG et al. (eds) Proceedings of the genetic and evolutionary computation conference (GECCO 2005). ACM, New York, pp 1835–1842
- Butz MV (2006) Rule-based evolutionary online learning systems: a principled approach to LCS analysis and design. Studies in fuzziness and soft computing. Springer, Berlin
- Butz MV, Goldberg DE, Lanzi PL (2004a) Bounding learning time in XCS. In: Genetic and evolutionary computation (GECCO 2004), Lecture notes in computer science, vol 3103. Springer, Berlin, pp 739–750
- Butz MV, Goldberg DE, Lanzi PL (2005a) Computational complexity of the XCS classifier system. In: Bull L, Kovacs T (eds) Foundations of learning classifier systems, Studies in fuzziness and soft computing, Lecture notes in artificial intelligence, vol 183. Springer, Berlin, pp 91–126
- Butz MV, Goldberg DE, Lanzi PL (2005b) Gradient descent methods in learning classifier systems: improving XCS performance in multistep problems. *IEEE Trans Evolut Comput* 9(5):452–473

- Butz MV, Goldberg DE, Lanzi PL, Sastry K (2007) Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity. *GP and Evol Machines* 8(1):5–37
- Butz MV, Lanzi PL, Wilson SW (2006) Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure. In: Cattolico M (ed) *Proceedings of the genetic and evolutionary computation conference (GECCO 2006)*. ACM, New York, pp 1457–1464
- Butz MV, Pelikan M (2006) Studying XCS/BOA learning in Boolean functions: structure encoding and random Boolean functions. In: Cattolico M et al. (eds) *Genetic and evolutionary computation conference, GECCO 2006*. ACM, New York, pp 1449–1456
- Butz MV, Pelikan M, Llorà X, Goldberg DE (2005) Extracted global structure makes local building block processing effective in XCS. In: Beyer HG, O'Reilly UM (eds) *Proceedings of the genetic and evolutionary computation conference, GECCO 2005*. ACM, New York, pp 655–662
- Butz MV, Pelikan M, Llorà X, Goldberg DE (2006) Automated global structure extraction for effective local building block processing in XCS. *Evolut Comput* 14(3):345–380
- Butz MV, Stalph P, Lanzi PL (2008) Self-adaptive mutation in XCSF. In *GECCO '08: Proceedings of the 10th annual conference on genetic and evolutionary computation*. ACM, New York, pp 1365–1372
- Cantú-Paz E, Kamath C (2003) Inducing oblique decision trees with evolutionary algorithms. *IEEE Trans Evolut Comput* 7(1):54–68
- Cantú-Paz E (2002) Feature subset selection by estimation of distribution algorithms. In: *GECCO '02: Proceedings of the genetic and evolutionary computation conference*. Morgan Kaufmann, San Francisco, CA, pp 303–310
- Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: *ICML '06: Proceedings of the 23rd international conference on machine learning*. ACM, New York, pp 161–168
- Casillas J, Carse B, Bull L (2007) Fuzzy-XCS: a Michigan genetic fuzzy system. *IEEE Trans Fuzzy Syst* 15: 536–550
- Castillo PA, Merelo JJ, Arenas MG, Romero G (2007) Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. *Inform Sciences*, 177 (14):2884–2905
- Chalmers D (1990) The evolution of learning: An experiment in genetic connectionism. In: Touretsky E (ed) *Proceedings 1990 connectionist models summer school*. Morgan Kaufmann, San Francisco, CA pp 81–90
- Chandra A, Yao X (2006a) Ensemble learning using multi-objective evolutionary algorithms. *J Math Model Algorithm* 5(4):417–445
- Chandra A, Yao X (2006b) Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing* 69(7–9):686–700
- Cho S, Cha K (1996) Evolution of neural net training set through addition of virtual samples. In: *Proceedings of the 1996 IEEE international conference on evolutionary computation*. IEEE Press, Washington DC, pp 685–688
- Cho S-B (1999) Pattern recognition with neural networks combined by genetic algorithm. *Fuzzy Set Syst* 103:339–347
- Cho S-B, Park C (2004) Speciated GA for optimal ensemble classifiers in DNA microarray classification. In: *Congress on evolutionary computation (CEC 2004)*, vol 1. pp 590–597
- Corcoran AL, Sen S (1994) Using real-valued genetic algorithms to evolve rule sets for classification. In: *Proceedings of the IEEE conference on evolutionary computation*. IEEE Press, Washington DC, pp 120–124
- Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) *Genetic fuzzy systems*. World Scientific, Singapore
- Cribbs III HB, Smith RE (1996) Classifier system renaissance: new analogies, new directions. In: Koza JR, Goldberg DE, Fogel DB, Riolo RL (eds) *Genetic programming 1996: proceedings of the first annual conference*. MIT Press, Cambridge, MA, Stanford University, CA, USA, 28–31 July 1996. pp 547–552
- Dam HH, Abbass HA, Lokan C, Yao X (2008) Neural-based learning classifier systems. *IEEE Trans Knowl Data Eng* 20(1):26–39
- Dam HH, Abbass HA, Lokan C (2005) DXCS: an XCS system for distributed data mining. In: Beyer HG, O'Reilly UM (eds) *Genetic and evolutionary computation conference, GECCO 2005*. pp 1883–1890
- Dasdan A, Oflazer K (1993) Genetic synthesis of unsupervised learning algorithms. Technical Report BU-CEIS-9306, Department of Computer Engineering and Information Science, Bilkent University, Ankara
- De Jong KA, Spears WM (1991) Learning concept classification rules using genetic algorithms. In: *Proceedings of the twelfth international conference on artificial intelligence IJCAI-91*, vol 2. Morgan Kaufmann, pp 651–656
- De Jong KA, Spears WM, Gordon DF (1993) Using genetic algorithms for concept learning. *Mach Learn* 3:161–188
- Dietterich TG (1998) Machine-learning research: four current directions. *AI Mag* 18(4):97–136
- Divina F, Keijzer M, Marchiori E (2002) Non-universal suffrage selection operators favor population diversity in genetic algorithms. In: *Benelearn 2002: proceedings of the 12th Belgian-Dutch conference*

- on machine learning (Technical report UU-CS-2002-046). pp 23–30
- Divina F, Keijzer M, Marchiori E (2003) A method for handling numerical attributes in GA-based inductive concept learners. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2003). Springer, Berlin, pp 898–908
- Divina F, Marchiori E (2002) Evolutionary concept learning. In: Langdon WB, Cantú-Paz E, Mathias K, Roy R, Davis D, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter MA, Schultz AC, Miller JF, Burke E, Jonoska N (eds) GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco, CA, New York, 9–13 July 2002, pp 343–350
- Dixon PW, Corne D, Oates MJ (2002) A ruleset reduction algorithm for the XCS learning classifier system. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems, 5th international workshop (IWLCS 2002), Lecture notes in computer science, vol 2661. Springer, Berlin, pp 20–29
- Donnart J-Y (1998) Cognitive architecture and adaptive properties of an motivationally autonomous animat. PhD thesis, Université Pierre et Marie Curie, Paris, France
- Donnart J-Y, Meyer J-A (1996a) Hierarchical-map Building and Self-positioning with MonaLysa. *Adapt Behav* 5(1):29–74
- Donnart J-Y, Meyer J-A (1996b) Learning reactive and planning rules in a motivationally autonomous animat. *IEEE Trans Syst Man Cybern B Cybern* 26 (3):381–395
- Dorigo M, Colombetti M (1998) Robot shaping: an experiment in behavior engineering. MIT Press/Bradford Books, Cambridge, MA
- Drugowitsch J, Barry A (2005) XCS with eligibility traces. In: Beyer H-G, O'Reilly U-M (eds) Genetic and evolutionary computation conference, GECCO 2005. ACM, New York, pp 1851–1858
- Drugowitsch J (2008) Design and analysis of learning classifier systems: a probabilistic approach. Springer, Berlin
- Drugowitsch J, Barry A (2007) A formal framework and extensions for function approximation in learning classifier systems. *Mach Learn* 70(1):45–88
- Edakunni NE, Kovacs T, Brown G, Marshall JAR, Chandra A (2009) Modelling UCS as a mixture of experts. In: Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO'09). ACM, pp 1187–1194
- Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. *Evol Intell* 1(1): 47–62
- Folino G, Pizzuti C, Spezzano G (2003) Ensemble techniques for parallel genetic programming based classifiers. In: Proceedings of the sixth European conference on genetic programming (EuroGP'03), Lecture notes in computer science, vol 2610. Springer, Berlin, pp 59–69
- Freitas AA (2002a) Data mining and knowledge discovery with evolutionary algorithms. Springer, Berlin
- Freitas AA (2002b) A survey of evolutionary algorithms for data mining and knowledge discovery. In: Ghosh A, Tsutsui S (eds) Advances in evolutionary computation. Springer, Berlin, pp 819–845
- Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: Proceedings of the international conference on machine learning (ICML'96), Bari, Italy, pp 148–156
- Freund Y, Schapire R (1999) A short introduction to boosting. *J Jpn Soc Artif Intell* 14(5):771–780
- Fürnkranz J (1998) Integrative windowing. *J Artif Intell Res* 8:129–164
- Gagné C, Sebag M, Schoenauer M, Tomassini M (2007) Ensemble learning for free with evolutionary algorithms? In: GECCO '07: Proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, New York, pp 1782–1789
- Gathercole C, Ross P (1997) Tackling the Boolean even n parity problem with genetic programming and limited-error fitness. In: Koza JR, Deb K, Dorigo M, Fogel DB, Garzon M, Iba H, Riolo RL (eds) Genetic programming 1997: proceedings second annual conference. Morgan Kaufmann, San Francisco, CA, pp 119–127
- Gérard P, Sigaud O (2003) Designing efficient exploration with MACS: Modules and function approximation. In: Cantú-Paz E, Foster JA, Deb K, Davis D, Roy R, O'Reilly U-M, Beyer H-G, Standish R, Kendall G, Wilson S, Harman M, Wegener J, Dasgupta D, Potter MA, Schultz AC, Dowsland K, Jonoska N, Miller J (eds) Genetic and evolutionary computation – GECCO-2003, Lecture notes in computer science, vol 2724. Springer, Berlin, pp 1882–1893
- Gerard P, Stolzmann W, Sigaud O (2002) YACS, a new learning classifier system using anticipation. *J Soft Comput* 6(3–4):216–228
- Geyer-Schulz A (1997) Fuzzy rule-based expert systems and genetic machine learning. Physica, Heidelberg
- Giordana A, Neri F (1995) Search-intensive concept induction. *Evolut Comput* 3:375–416
- Giordana A, Saitta L (1994) Learning disjunctive concepts by means of genetic algorithms. In: Proceedings of the international conference on machine learning, Brunswick, NJ, pp 96–104
- Giraldez R, Aguilar-Ruiz J, Riquelme J (2003) Natural coding: a more efficient representation for evolutionary learning. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2003). Springer, Berlin, pp 979–990

- Giraud-Carrier C, Keller J (2002) Meta-learning. In: Meij J (ed) Dealing with the data flood. STT/Beweton, Hague, The Netherlands
- Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
- Goldberg DE, Horn J, Deb K (1992) What makes a problem hard for a classifier system? In: Collected abstracts for the first international workshop on learning classifier system (IWLCS-92). (Also technical report 92007 Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign). Available from ENCORE (<ftp://ftp.krl.caltech.edu/pub/EC/Welcome.html>) in the section on Classifier Systems
- Greene DP, Smith SF (1993) Competition-based induction of decision models from examples. *Mach Learn* 13:229–257
- Greene DP, Smith SF (1994) Using coverage as a model building constraint in learning classifier systems. *Evolut Comput* 2(1):67–91
- Greene DP, Smith SF (1987) A genetic system for learning models of consumer choice. In: Proceedings of the second international conference on genetic algorithms and their applications. Morgan Kaufmann, San Francisco, CA, pp 217–223
- Greenyer A (2000) The use of a learning classifier system JXCS. In: van der Putten P, van Someren M (eds) CoIL challenge 2000: the insurance company case. Leiden Institute of Advanced Computer Science, June 2000. Technical report 2000–09
- Gruau F (1995) Automatic definition of modular neural networks. *Adapt Behav* 3(2):151–183
- Hanebeck D, Schmidt K (1996) Genetic optimization of fuzzy networks. *Fuzzy set syst* 79:59–68
- Hansen LK, Salamon P (1990) Neural network ensembles. *IEEE Trans Pattern Anal Mach Intell* 12(10): 993–1001
- Hart WE, Krasnogor N, Smith JE (eds) (2004) Special issue on memetic algorithms, evolutionary computation, vol 12, 3
- Hart WE, Krasnogor N, Smith JE (eds) (2005) Recent advances in memetic algorithms, Studies in fuzziness and soft computing, vol 166. Springer, Berlin
- He L, Wang KJ, Jin HZ, Li GB, Gao XZ (1999) The combination and prospects of neural networks, fuzzy logic and genetic algorithms. In: IEEE midnight-sun workshop on soft computing methods in industrial applications. IEEE, Washington, DC, pp 52–57
- Heitkötter J, Beasley D (2001) The hitch-hiker's guide to evolutionary computation (FAQ for comp.ai.genetic). Accessed 28/2/09. <http://www.aip.de/~ast/EvolvCompFAQ/>
- Hekanaho J (1995) Symbiosis in multimodal concept learning. In: Proceedings of the 1995 international conference on machine learning (ICML'95). Morgan Kaufmann, San Francisco, pp 278–285
- Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolut Intell* 1(1):27–46
- Holland JH (1976) Adaptation. In: Rosen R, Snell FM (eds) Progress in theoretical biology. Plenum, New York
- Holland JH (1986) Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Mitchell T, Michalski R, Carbonell J (eds) Machine learning, an artificial intelligence approach. vol. II, chap. 20. Morgan Kaufmann, San Francisco, CA, pp 593–623
- Holland JH, Booker LB, Colombetti M, Dorigo M, Goldberg DE, Forrest S, Riolo RL, Smith RE, Lanzi PL, Stolzmann W, Wilson SW (2000) What is a learning classifier system? In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems: from foundations to application, Lecture notes in artificial intelligence, vol 1813. Springer, Berlin, pp 3–32
- Holland JH, Holyoak KJ, Nisbett RE, Thagard PR (1986) Induction: processes of inference, learning, and discovery. MIT Press, Cambridge, MA
- Holland JH, Reitman JS (1978) Cognitive systems based on adaptive algorithms. In: Waterman DA, Hayes-Roth F (eds) Pattern-directed Inference Systems. Academic Press, New York. Reprinted in: Evolutionary Computation. The Fossil Record. Fogel DB (ed) (1998) IEEE Press, Washington DC. ISBN: 0-7803-3481-7
- Homaifar A, McCormick E (1995) Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans Fuzzy Syst*, 3(2):129–139
- Howard D, Bull L (2008) On the effects of node duplication and connection-orientated constructivism in neural XCSF. In: Keijzer M et al. (eds) GECCO-2008: proceedings of the genetic and evolutionary computation conference. ACM, New York, pp 1977–1984
- Howard D, Bull L, Lanzi PL (2008) Self-adaptive constructivism in neural XCS and XCSF. In: Keijzer M et al. (eds) GECCO-2008: proceedings of the genetic and evolutionary computation conference. ACM, New York, pp 1389–1396
- Hu Y-J (1998) A genetic programming approach to constructive induction. In: Genetic programming 1998: proceedings of the 3rd annual conference. Morgan Kaufmann, San Francisco, CA, pp 146–151
- Hurst J, Bull L (2003) Self-adaptation in classifier system controllers. *Artifi Life Robot* 5(2):109–119
- Hurst J, Bull L (2004) A self-adaptive neural learning classifier system with constructivism for mobile robot control. In: Yao X et al. (eds) Parallel problem solving from nature (PPSN VIII), Lecture notes

- in computer science, vol 3242. Springer, Berlin, pp 942–951
- Husbands P, Harvey I, Cliff D, Miller G (1994) The use of genetic algorithms for the development of sensorimotor control systems. In: Gaussier P, Nicoud J-D (eds) From perception to action. IEEE Press, Washington DC, pp 110–121
- Iba H (1999) Bagging, boosting and bloating in genetic programming. In: Proceedings of the genetic and evolutionary computation conference (GECCO'99). Morgan Kaufmann, San Francisco, CA, pp 1053–1060
- IEEE (2000) Proceedings of the 2000 congress on evolutionary computation (CEC'00). IEEE Press, Washington DC
- Ishibuchi H, Nakashima T (2000) Multi-objective pattern and feature selection by a genetic algorithm. In: Proceedings of the 2000 genetic and evolutionary computation conference (GECCO'2000). Morgan Kaufmann, San Francisco, CA, pp 1069–1076
- Islam MM, Yao X, Murase K (2003) A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans Neural Netw* 14:820–834
- Jain A, Zongker D (1997) Feature selection: evaluation, application and small sample performance. *IEEE Trans. Pattern Anal Mach Intell* 19(2):153–158
- Janikow CZ (1991) Inductive learning of decision rules in attribute-based examples: a knowledge-intensive genetic algorithm approach. PhD thesis, University of North Carolina
- Janikow CZ (1993) A knowledge-intensive genetic algorithm for supervised learning. *Mach Learn* 13:189–228
- Jin Y, Sendhoff B (2004) Reducing fitness evaluations using clustering techniques and neural network ensembles. In: Genetic and evolutionary computation conference (GECCO–2004), Lecture notes in computer science, vol 3102. Springer, Berlin, pp 688–699
- John G, Kohavi R, Phleger K (1994) Irrelevant features and the feature subset problem. In: Proceedings of the 11th international conference on machine learning. Morgan Kaufmann, San Francisco, CA, pp 121–129
- Juan Liu J, Tin-Yau Kwok J (2000) An extended genetic rule induction algorithm. In: Proceedings of the 2000 congress on evolutionary computation (CEC'00). IEEE Press, Washington DC, pp 458–463
- Kelly JD Jr, Davis L (1991) Hybridizing the genetic algorithm and the k nearest neighbors classification algorithm. In: Booker LB, Belew RK (eds) Proceedings of the 4th international conference on genetic algorithms (ICGA91). Morgan Kaufmann, San Francisco, CA, pp 377–383
- Karr C (1991) Genetic algorithms for fuzzy controllers. *AI Expert* 6(2):26–33
- Kasabov N (2007) Evolving connectionist systems: the knowledge engineering approach. Springer, Berlin
- Keijzer M, Babovic V (2000) Genetic programming, ensemble methods, and the bias/variance/tradeoff – introductory investigation. In: Proceedings of the European conference on genetic programming (EuroGP'00), Lecture notes in computer science, vol 1802. Springer, Berlin, pp 76–90
- Kitano H (1990) Designing neural networks by genetic algorithms using graph generation system. *J Complex Syst* 4:461–476
- Kolman E, Margaliot M (2009) Knowledge-based neurocomputing: a fuzzy logic approach, Studies in fuzziness and soft computing, vol 234. Springer, Berlin
- Kovacs T (1996) Evolving optimal populations with XCS classifier systems. Master's thesis, University of Birmingham, Birmingham, UK
- Kovacs T (1997) XCS classifier system reliably evolves accurate, complete, and minimal representations for Boolean functions. In: Chawdhry PK, Roy R, Pant RK (eds) Soft computing in engineering design and manufacturing. Springer, London, pp 59–68 [ftp://ftp.cs.bham.ac.uk/pub/authors/T.Kovacs/index.html](http://ftp.cs.bham.ac.uk/pub/authors/T.Kovacs/index.html)
- Kovacs T (2000) Strength or accuracy? Fitness calculation in learning classifier systems. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems: from foundations to applications. Lecture notes in artificial intelligence, vol 1813. Springer, Berlin, pp 143–160
- Kovacs T (2004) Strength or accuracy: credit assignment in learning classifier systems. Springer, Berlin
- Kovacs T (2009) A learning classifier systems bibliography. Department of Computer Science, University of Bristol. <http://www.cs.bris.ac.uk/~kovacs/lcs/search.html>
- Kovacs T, Kerber M (2001) What makes a problem hard for XCS? In: Lanzi PL, Stolzmann W, Wilson SW (eds) Advances in learning classifier systems, Lecture notes in artificial intelligence, vol 1996. Springer, Berlin, pp 80–99
- Kovacs T, Kerber M (2004) High classification accuracy does not imply effective genetic search. In: Deb K et al. (eds) Proceedings of the 2004 genetic and evolutionary computation conference (GECCO), Lecture notes in computer science, vol 3102. Springer, Berlin, pp 785–796
- Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA
- Koza JR (1994) Genetic Programming II. MIT Press
- Krasnogor N (2002) Studies on the theory and design space of memetic algorithms. PhD thesis, University of the West of England
- Krasnogor N, Smith JE (2005) A tutorial for competent memetic algorithms: model, taxonomy and

- design issues. *IEEE Trans Evolut Comput* 9(5): 474–488
- Krasnogor N (2004) Self-generating metaheuristics in bioinformatics: the protein structure comparison case. *GP and Evol Machines* 5(2):181–201
- Krasnogor N, Gustafson S (2004) A study on the use of self-generation in memetic algorithms. *Natural Comput* 3(1):53–76
- Krawiec K (2002) Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *GP and Evol Machines* 3(4):329–343
- Krogh A, Vedelsby J (1995) Neural network ensembles, cross validation and active learning. *NIPS* 7:231–238
- Kudo M, Skalansky J (2000) Comparison of algorithms that select features for pattern classifiers. *Pattern Recogn* 33:25–41
- Kuncheva LI (2004) Combining pattern classifiers: methods and algorithms. Wiley, Hoboken
- Kushch I (2002) An evaluation of evolutionary generalization in genetic programming. *Artif Intell Rev* 18(1):3–14
- Lam L, Suen CY (1995) Optimal combination of pattern classifiers. *Pattern Recogn Lett* 16:945–954 See Kuncheva (2004a) p.167
- Landau S, Sigaud O, Schoenauer M (2005) ATNoSFERES revisited. In: Proceedings of the genetic and evolutionary computation conference GECCO-2005. ACM, New York, pp 1867–1874
- Langdon W, Gustafson S, Koza J (2009) The genetic programming bibliography. <http://www.cs.bham.ac.uk/~wbl/biblio/>
- Lanzi PL (1999a) Extending the representation of classifier conditions, part I: from binary to messy coding. In: Banzhaf W et al. (eds) GECCO-99: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Francisco, CA, pp 337–344
- Lanzi PL (1999b) Extending the representation of classifier conditions part II: from messy coding to S-expressions. In: Banzhaf W et al. (eds) GECCO-99: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Francisco, CA, pp 345–352
- Lanzi PL (2002a) Learning classifier systems from a reinforcement learning perspective. *J Soft Comput* 6(3–4):162–170
- Lanzi PL (2001) Mining interesting knowledge from data with the XCS classifier system. In: Spector L, Goodman ED, Wu A, Langdon WB, Voigt H-M, Gen M, Sen S, Dorigo M, Pezeshk S, Garzon MH, Burke E (eds) Proceedings of the genetic and evolutionary computation conference (GECCO-2001). Morgan Kaufmann, San Francisco, CA, pp 958–965
- Lanzi PL (2008) Learning classifier systems: then and now. *Evolut Intell* 1(1):63–82
- Lanzi PL, Loiacono D, Wilson SW, Goldberg DE (2006a) Classifier prediction based on tile coding. In: Genetic and evolutionary computation – GECCO-2006. ACM, New York, pp 1497–1504
- Lanzi PL, Loiacono D, Wilson SW, Goldberg DE (2006b) Prediction update algorithms for XCSF: RLS, Kalman filter and gain adaptation. In: Genetic and Evolutionary Computation – GECCO-2006. ACM, New York, pp 1505–1512
- Lanzi PL, Loiacono D, Zanini M (2008) Evolving classifiers ensembles with heterogeneous predictors. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llorà X, Takadama K (eds) Learning classifier systems. 10th and 11th international workshops (2006–2007), Lecture notes in computer science, vol 4998. Springer, Berlin, pp 218–234
- Lanzi PL, Riolo RL (2000) A roadmap to the last decade of learning classifier system research (from 1989 to 1999). In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems: from foundations to applications, Lecture notes in artificial intelligence, vol 1813. Springer, Berlin, pp 33–62
- Lanzi PL, Stolzmann W, Wilson SW (eds) (2000) Learning classifier systems: from foundations to applications, Lecture notes in artificial intelligence, vol 1813. Springer, Berlin
- Lanzi PL, Stolzmann W, Wilson SW (eds) (2001) Advances in learning classifier systems, Lecture notes in artificial intelligence, vol 1996. Springer, Berlin
- Lanzi PL, Stolzmann W, Wilson SW (eds) (2002) Advances in learning classifier systems, Lecture notes in artificial intelligence, vol 2321. Springer, Berlin
- Lanzi PL, Butz MV, Goldberg DE (2007) Empirical analysis of generalization and learning in XCS with gradient descent. In: Lipson H (ed) Proceedings of the Genetic and evolutionary computation conference, GECCO 2007, vol 2. ACM, New York, pp 1814–1821
- Lanzi PL, Loiacono D (2006) Standard and averaging reinforcement learning in XCS. In: Cattolico M (ed) Proceedings of the 8th annual conference on genetic and evolutionary computation, GECCO 2006. ACM, New York, pp 1480–1496
- Lanzi PL, Loiacono D (2007) Classifier systems that compute action mappings. In: Lipson H (ed) Proceedings of the Genetic and evolutionary computation conference, GECCO 2007. ACM, New York, pp 1822–1829
- Lanzi PL, Wilson SW (2006) Using convex hulls to represent classifier conditions. In: Cattolico M (ed) Proceedings of the genetic and evolutionary computation conference (GECCO 2006). ACM, New York, pp 1481–1488
- Liangjie Z, Yanda L (1996) A new global optimizing algorithm for fuzzy neural networks. *Int J Elect* 80(3):393–403

- Linkens DA, Nyongesa HO (1996) Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications. *IEE Proc Contr Theo Appl* 143(4):367–386
- Liu Y, Yao X (1999) Ensemble learning via negative correlation. *Neural Netwro* 12:1399–1404
- Liu Y, Yao X, Higuchi T (2000) Evolutionary ensembles with negative correlation learning. *IEEE Trans Evolut Comput* 4(4):380–387
- Llorà X (2002) Genetic based machine learning using fine-grained parallelism for data mining. PhD thesis, Enginyeria i Arquitectura La Salle. Ramon Llull University
- Llorà X, Garrell JM (2001) Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Spector L, Goodman ED, Wu A, Langdon WB, Voigt H-M, Gen M, Sen S, Dorigo M, Pezeshk S, Garzon MH, Burke E (eds) Proceedings of the genetic and evolutionary computation conference (GECCO'2001). Morgan Kaufmann, San Francisco, CA, pp 461–468
- Llorà X, Sastry K, Goldberg DE (2005a) Binary rule encoding schemes: a study using the compact classifier system. In: Rothlauf F (ed) Proceedings of the 2005 conference on genetic and evolutionary computation GECCO '05. ACM Press, New York, pp 88–99
- Llorà X, Sastry K, Goldberg DE (2005b) The compact classifier system: scalability analysis and first results. In: Rothlauf F (ed) Proceedings of the IEEE congress on evolutionary computation, CEC 2005. IEEE, Press, Washington, DC, pp 596–603
- Llorà X, Wilson SW (2004) Mixed decision trees: minimizing knowledge representation bias in LCS. In: Kalyanmoy Deb et al. (eds) Proceedings of the genetic and evolutionary computation conference (GECCO-2004), Lecture notes in computer science, Springer, Berlin, pp 797–809
- Loiacono D, Marelli A, Lanzi PL (2007) Support vector regression for classifier prediction. In: GECCO '07: Proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, Berlin, pp 1806–1813
- Marmelstein RE, Lamont GB (1998) Pattern classification using a hybrid genetic algorithm – decision tree approach. In: Genetic programming 1998: proceedings of the 3rd annual conference (GP'98). Morgan Kaufmann, San Francisco, CA, pp 223–231
- Marshall JAR, Kovacs T (2006) A representational ecology for learning classifier systems. In: Keijzer M et al. (ed) Proceedings of the 2006 genetic and evolutionary computation conference (GECCO 2006). ACM, New York, pp 1529–1536
- Martin-Bautista MJ, Vila M-A (1999) A survey of genetic feature selection in mining issues. In: Proceedings of the congress on evolutionary computation (CEC'99). IEEE Press, Washington, DC, pp 1314–1321
- Meir R, Rätsch G (2003) An introduction to boosting and leveraging. In: Advanced lectures on machine learning. Springer, Berlin, pp 118–183
- Mellor D (2005a) A first order logic classifier system. In: Rothlauf F (ed), GECCO '05: Proceedings of the 2005 conference on genetic and evolutionary computation. ACM, New York, pp 1819–1826
- Mellor D (2005b) Policy transfer with a relational learning classifier system. In: GECCO Workshops 2005. ACM, New York, pp 82–84
- Mellor D (2008) A learning classifier system approach to relational reinforcement learning. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llorà X, Takadama K (eds) Learning classifier systems. 10th and 11th international workshops (2006–2007), Lecture notes in computer science, vol 4998. Springer, New York, pp 169–188
- Michalski RS, Mozetic I, Hong J, Lavrac N (1986) The AQ15 inductive learning system: an overview and experiments. Technical Report UIUCDCS-R-86-1260, University of Illinois
- Miller GF, Todd PM, Hegde SU (1989) Designing neural networks using genetic algorithms. In: Schaffer JD (ed) Proceedings of the 3rd international conference genetic algorithms and their applications, Morgan Kaufmann, San Francisco, CA, pp 379–384
- Mitra S, Hayashi Y (2000) Neurofuzzy rule generation: survey in soft computing framework. *IEEE Trans Neural Netwro* 11(3):748–768
- Morimoto T, Suzuki J, Hashimoto Y (1997) Optimization of a fuzzy controller for fruit storage using neural networks and genetic algorithms. *Eng Appl Artif Intell* 10(5):453–461
- Nolfi S, Migliolo O, Parisi D (1994) Phenotypic plasticity in evolving neural networks. In: Gaussier P, Nicoud J-D (eds) From perception to action. IEEE Press, Washington, DC, pp 146–157
- O'Hara T, Bull L (2005) A memetic accuracy-based neural learning classifier system. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2005). IEEE Press, Washington, DC, pp 2040–2045
- Ong Y-S, Krasnogor N, Ishibuchi H (eds) (2007) Special issue on memetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics - Part B*
- Ong Y-S, Lim M-H, Neri F, Ishibuchi H (2009) Emerging trends in soft computing - memetic algorithms, *Special Issue of Soft Computing*. vol 13, 8–9
- Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: A comparative study. *IEEE Trans Syst Man Cybern B* 36(1):141–152

- Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. *J Artif Intell Res* 11:169–198
- Opitz DW, Shavlik JW (1996) Generating accurate and diverse members of a neural-network ensemble. *Advances in neural information processing systems*, vol 8. Morgan Kaufmann, pp 535–541
- Orriols-Puig A, Bernadó-Mansilla E (2006) Bounding XCS's parameters for unbalanced datasets. In: Keijzer M et al. (eds) *Proceedings of the 2006 genetic and evolutionary computation conference (GECCO 2006)*. ACM, New York, pp 1561–1568
- Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2007a) Fuzzy-UCS: preliminary results. In: Lipson H (ed) *Proceedings of the genetic and evolutionary computation conference*, GECCO 2007. ACM, New York, pp 2871–2874
- Orriols-Puig A, Goldberg DE, Sastry K, Bernadó-Mansilla E (2007b) Modeling XCS in class imbalances: population size and parameter settings. In: Lipson H et al. (eds) *Genetic and evolutionary computation conference*, GECCO 2007. ACM, New York, pp 1838–1845
- Orriols-Puig A, Goldberg DE, Sastry K, Bernadó-Mansilla E (2007c) Modeling XCS in class imbalances: population size and parameter settings. In: Lipson H (eds) *Proceedings of the genetic and evolutionary computation conference*, GECCO 2007. ACM, New York, pp 1838–1845
- Orriols-Puig A, Sastry K, Lanzi PL, Goldberg DE, Bernadó-Mansilla E (2007d) Modeling selection pressure in XCS for proportionate and tournament selection. In: Lipson H (ed) *Proceedings of the genetic and evolutionary computation conference*, GECCO 2007. ACM, New York, pp 1846–1853
- Orriols-Puig A, Bernadó-Mansilla E (2008) Revisiting UCS: description, fitness sharing, and comparison with XCS. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llorà X, Takadama K (eds) *Learning classifier systems. 10th and 11th international workshops (2006–2007)*, Lecture notes in computer science, vol 4998. Springer, Berlin, pp 96–111
- Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2008a) Evolving fuzzy rules with UCS: preliminary results. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llorà X, Takadama K (eds) *Learning classifier systems. 10th and 11th international workshops (2006–2007)*, Lecture notes in computer science, vol 4998. Springer, Berlin, pp 57–76
- Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2008b) Genetic-based machine learning systems are competitive for pattern recognition. *Evolut Intell* 1(3):209–232
- Pal S, Bhandari D (1994) Genetic algorithms with fuzzy fitness function for object extraction using cellular networks. *Fuzzy Set Syst* 65(2–3):129–139
- Pappa GL, Freitas AA (2010) Automating the design of data mining algorithms. An evolutionary computation approach. *Natural computing series*. Springer
- Paris G, Robilliard D, Fonlupt C (2001) Applying boosting techniques to genetic programming. In: *Artificial evolution 2001, Lecture notes in computer science*, vol 2310. Springer, Berlin, pp 267–278
- Pereira FB, Costa E (2001) Understanding the role of learning in the evolution of busy beaver: a comparison between the Baldwin effect and Lamarckian strategy. In: *Proceedings of the genetic and evolutionary computation conference (GECCO–2001)*. Morgan Kaufmann, San Francisco, pp 884–891
- Perneel C, Themlin J-M (1995) Optimization of fuzzy expert systems using genetic algorithms and neural networks. *IEEE Trans Fuzzy Syst* 3 (3):301–312
- Pham DT, Karaboga D (1991) Optimum design of fuzzy logic controllers using genetic algorithms. *J Syst Eng* 1:114–118
- Poli R, Langdon WB, McPhee NF (2008) A field guide to genetic programming, freely available at <http://www.gp-field-guide.org.uk/lulu.com>
- Punch WF, Goodman ED, Pei M, Chia-Shun L, Hovland P, Enbody R (1993) Further research on feature selection and classification using genetic algorithms. In: Forrest S (ed) *Proceedings of the 5th international conference on genetic algorithms (ICGA93)*. Morgan Kaufmann, San Francisco, CA, pp 557–564
- Radi A, Poli R (2003) Discovering efficient learning rules for feedforward neural networks using genetic programming. In: Abraham A, Jain L, Kacprzyk J (eds) *Recent advances in intelligent paradigms and applications*. Springer, Berlin, pp 133–159
- Raymer ML, Punch WF, Goodman ED, Kuhn LA, Jain AK (2000) Dimensionality reduction using genetic algorithms. *IEEE Trans Evolut Comput* 4(2): 164–171
- Reeves CR, Rowe JE (2002) *Genetic algorithms – principles and perspectives. A guide to GA theory*. Kluwer, Norwell
- Riolo RL (1987) Bucket brigade performance: I. long sequences of classifiers. In: Grefenstette JJ (eds) *Proceedings of the 2nd international conference on genetic algorithms (ICGA'87)*, Lawrence Erlbaum Associates, Cambridge, MA, pp 184–195
- Rivest RL (1987) Learning decision lists. *Mach Learn* 2(3):229–246
- Romanuk S (1994) Towards minimal network architectures with evolutionary growth networks. In: *Proceedings of the 1993 international joint conference on neural networks*, vol 3. IEEE Press, Washington, DC, pp 1710–1713

- Rouwhorst SE, Engelbrecht AP (2000) Searching the forest: using decision trees as building blocks for evolutionary search in classification databases. In: Proceedings of the 2000 congress on evolutionary computation (CEC00). IEEE Press, Washington, DC, pp 633–638
- Rozenberg G, Bäck T, Kok J (eds) (2012) Handbook of natural computing. Springer, Berlin
- Ruta D, Gabrys B (2001) Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In: Kittler J, Roli F (eds) Proceedings of the 2nd international workshop on multiple classifier systems, Lecture notes in computer science, vol 2096. Springer, Berlin, pp 399–408. See Kuncheva (2004a) p.321
- Sánchez L, Couso I (2007) Advocating the use of imprecisely observed data in genetic fuzzy systems. *IEEE Trans Fuzzy Syst* 15(4):551–562
- Sasaki T, Tokoro M (1997) Adaptation toward changing environments: why Darwinian in nature? In: Husbands P, Harvey I (eds) Proceedings of the 4th European conference on artificial life. MIT Press, Cambridge, MA, pp 145–153
- Saxon S, Barry A (2000) XCS and the Monk's problems. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems: from foundations to applications, Lecture notes in artificial intelligence, vol 1813. Springer, Berlin, pp 223–242
- Schaffer C (1994) A conservation law for generalization performance. In: Hirsh H, Cohen WW (eds) Machine learning: proceedings of the eleventh international conference. Morgan Kaufmann, San Francisco, CA, pp 259–265
- Schaffer JD (ed) (1989) Proceedings of the 3rd international conference on genetic algorithms (ICGA-89), George Mason University, June 1989. Morgan Kaufmann, San Francisco, CA
- Schmidhuber J (1987) Evolutionary principles in self-referential learning. (On learning how to learn: The meta-meta-... hook.). PhD thesis, Technische Universität München, Germany
- Schuurmans D, Schaeffer J (1989) Representational difficulties with classifier systems. In: Schaffer JD (ed) Proceedings of the 3rd international conference on genetic algorithms (ICGA-89). Morgan Kaufmann, San Francisco, CA, pp 328–333
- Sharkey AJC (1996) On combining artificial neural nets. *Connection Sci* 8(3–4):299–313
- Sharpe PK, Glover RP (1999) Efficient GA based techniques for classification. *Appl Intell* 11:277–284
- Sirlantzis K, Fairhurst MC, Hoque MS (2001) Genetic algorithms for multi-classifier system configuration: a case study in character recognition. In: Kittler J, Roli F (eds) Proceedings of the 2nd international workshop on multiple classifier systems, Lecture notes in computer science, vol 2096. Springer, Berlin, pp 99–108. See Kuncheva (2004a) p.321
- Smith JE (2007) Coevolving memetic algorithms: a review and progress report. *IEEE Trans Syst Man Cybern B Cybern* 37(1):6–17
- Smith MG, Bull L (2005) Genetic programming with a genetic algorithm for feature construction and selection. *GP and Evol Machines* 6(3):265–281
- Smith RE (1992) A report on the first international workshop on learning classifier systems (IWLCS-92). NASA Johnson Space Center, Houston, Texas, Oct. 6–9. <ftp://lumpi.informatik.uni-dortmund.de/pub/LCS/papers/lcs92.ps.gz> or from ENCORE, The Electronic Appendix to the Hitch-Hiker's Guide to Evolutionary Computation (<ftp://ftp.krl.caltech.edu/pub/EC/Welcome.html>) in the section on Classifier Systems
- Smith RE (1994) Memory exploitation in learning classifier systems. *Evolut Comput* 2(3):199–220
- Smith RE, Cribbs HB (1994) Is a learning classifier system a type of neural network? *Evolut Comput* 2(1): 19–36
- Smith RE, Goldberg DE (1991) Variable default hierarchy separation in a classifier system. In: Rawlins GJE (ed) Proceedings of the first workshop on foundations of genetic algorithms. Morgan Kaufmann, San Mateo, pp 148–170
- Smith RE, Cribbs III HB (1997) Combined biological paradigms. *Robot Auton Syst* 22(1):65–74
- Song D, Heywood MI, Zincir-Heywood AN (2005) Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Trans Evolut Comput* 9(3):225–239
- Srinivas N, Deb K (1994) Multi-objective function optimization using non-dominated sorting genetic algorithm. *Evolut Comput* 2(3):221–248
- Stagge P (1998) Averaging efficiently in the presence of noise. In: Parallel problem solving from nature, vol 5. pp 188–197
- Stolzmann W (1996) Learning classifier systems using the cognitive mechanism of anticipatory behavioral control, detailed version. In: Proceedings of the first European workshop on cognitive modelling. Berlin, TU, pp 82–89
- Stone C, Bull L (2003) For real! XCS with continuous-valued inputs. *Evolut Comput* 11(3):298–336
- Storn R, Price K (1996) Minimizing the real functions of the ICEC'96 contest by differential evolution. In: Proceedings of the IEEE international conference Evolutionary Computation. IEEE Press, Washington, DC, pp 842–844
- Stout M, Bacardit J, Hirst JD, Krasnogor N (2008) Prediction of recursive convex hull class assignment for protein residues. *Bioinformatics* 24(7): 916–923

- Sutton RS (1986) Two problems with backpropagation and other steepest-descent learning procedures for networks. In: Proceedings of the 8th annual conference cognitive science society. Erlbaum, pp 823–831
- Sziranyi T (1996) Robustness of cellular neural networks in image deblurring and texture segmentation. *Int J Circuit Theory App* 24(3):381–396
- Tamaddoni-Nezhad A, Muggleton SH (2000) Searching the subsumption lattice by a genetic algorithm. In: Cussens J, Frisch A (eds) Proceedings of the 10th international conference on inductive logic programming. Springer, Berlin, pp 243–252
- Tamaddoni-Nezhad A, Muggleton S (2003) A genetic algorithms approach to ILP. In: Inductive logic programming, Lecture notes in computer science, vol 2583. Springer, Berlin, pp 285–300
- Tharakunnel K, Goldberg D (2002) XCS with average reward criterion in multi-step environment. Technical report, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign
- Thompson S (1998) Pruning boosted classifiers with a real valued genetic algorithm. In: Research and development in expert systems XV – proceedings of ES'98. Springer, Berlin, pp 133–146
- Thompson S (1999) Genetic algorithms as postprocessors for data mining. In: Data mining with evolutionary algorithms: research directions – papers from the AAAI workshop, Tech report WS-99-06. AAAI Press, Menlo Park, CA, pp 18–22
- Thrift P (1991) Fuzzy logic synthesis with genetic algorithms. In: Booker LB, Belew RK (eds) Proceedings of 4th international conference on genetic algorithms (ICGA'91). Morgan Kaufmann, San Francisco, CA, pp 509–513
- Tomlinson A (1999) Corporate classifier systems. PhD thesis, University of the West of England
- Tomlinson A, Bull L (1998) A corporate classifier system. In: Eiben AE, Bäck T, Shoenauer M, Schwefel H-P (eds) Proceedings of the fifth international conference on parallel problem solving from nature – PPSN V, Lecture notes in computer science, vol 1498. Springer, Berlin, pp 550–559
- Tomlinson A, Bull L (2002) An accuracy-based corporate classifier system. *J Soft Comput* 6(3–4):200–215
- Tran TH, Sanza C, Duthen Y, Nguyen TD (2007) XCSF with computed continuous action. In: Genetic and evolutionary computation conference (GECCO 2007). ACM, New York, pp 1861–1869
- Tumer K, Ghosh J (1996) Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recogn* 29(2):341–348
- Turney P (1996) How to shift bias: lessons from the Baldwin effect. *Evolut Comput* 4(3):271–295
- Valentini G, Masulli F (2002) Ensembles of learning machines. In: WIRN VIETRI 2002: Proceedings of the 13th Italian workshop on neural nets-revised papers. Springer, Berlin, pp 3–22
- Valenzuela-Rendón M (1989) Two analysis tools to describe the operation of classifier systems. PhD thesis, University of Alabama. Also TCGA technical report 89005
- Valenzuela-Rendón M (1991) The fuzzy classifier system: a classifier system for continuously varying variables. In: Booker LB, Belew RK (eds) Proceedings of the 4th international conference on genetic algorithms (ICGA'91). Morgan Kaufmann, San Francisco, CA, pp 346–353
- Valenzuela-Rendón M (1998) Reinforcement learning in the fuzzy classifier system. *Expert Syst Appl* 14:237–247
- Vallim R, Goldberg D, Llorà X, Duque T, Carvalho A (2003) A new approach for multi-label classification based on default hierarchies and organizational learning. In: Proceedings of the genetic and evolutionary computation conference, workshop sessions: learning classifier systems. ACM, New York, pp 2017–2022
- Vanneschi L, Poli R (2012) Genetic programming: introduction, applications, theory and open issues. In: Rozenberg G, Bäck T, Kok J (eds) Handbook of natural computing. Springer, Berlin
- Venturini G (1993) SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In: Brazdil PB (ed) ECML-93 - Proceedings of the European conference on machine learning. Springer, Berlin, pp 280–296
- Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. *Artif Intell Rev* 18(2):77–95
- Wada A, Takadama K, Shimohara K, Katai O (2005c) Learning classifier systems with convergence and generalization. In: Bull L, Kovacs T (eds) Foundations of learning classifier systems. Springer, Berlin, pp 285–304
- Wada A, Takadama K, Shimohara K (2005a) Counter example for Q-bucket-brigade under prediction problem. In: GECCO Workshops 2005. ACM, New York, pp 94–99
- Wada A, Takadama K, Shimohara K (2005b) Learning classifier system equivalent with reinforcement learning with function approximation. In: GECCO Workshops 2005. ACM, New York, pp 92–93
- Wada A, Takadama K, Shimohara K (2007) Counter example for Q-bucket-brigade under prediction problem. In: Kovacs T, LLóra X, Takadama K, Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems. International workshops, IWLCS 2003-2005, revised selected papers, Lecture notes in computer science, vol 4399. Springer, Berlin, pp 128–143
- Whitley D, Goldberg D, Cantú-Paz E, Spector L, Parmee I, Beyer HG (eds) (2000) Proceedings of the genetic and

- evolutionary computation conference (GECCO-2000). Morgan Kaufmann, San Francisco, CA
- Whiteson S, Stone P (2006) Evolutionary function approximation for reinforcement learning. *J Mach Learn Res* 7:877–917
- Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Comput* 14(3):347–361
- Whitley D, Gordon VS, Mathias K (1994) Lamarckian evolution, the Baldwin effect and function optimization. In: Parallel problem solving from nature (PPSN-III). Springer, Berlin, pp 6–15
- Wilcox JR (1995) Organizational learning within a learning classifier system. Master's thesis, University of Illinois. Also Technical Report No. 95003 IlliGAL
- Wilson SW (2001a) Mining oblique data with XCS. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Advances in learning classifier systems, third international workshop, IWLCS 2000, Lecture notes in computer science, vol 1996. Springer, Berlin, pp 158–176
- Wilson SW (1989) Bid competition and specificity reconsidered. *Complex Syst* 2:705–723
- Wilson SW (1994) ZCS: a zeroth level classifier system. *Evolut Comput* 2(1):1–18. <http://prediction-dynamics.com/>
- Wilson SW (1995) Classifier fitness based on accuracy. *Evolut Comput* 3(2):149–175. <http://prediction-dynamics.com/>
- Wilson SW (1998) Generalization in the XCS classifier system. In: Koza JR, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Fogel DB, Garzon MH, Goldberg DE, Iba H, Riolo R (eds) Genetic programming 1998: proceedings of the third annual conference, Morgan Kaufmann, San Francisco, CA, pp 665–674. <http://prediction-dynamics.com/>
- Wilson SW (1999) Get real! XCS with continuous-valued inputs. In: Booker L, Forrest S, Mitchell M, Riolo RL (eds) Festschrift in honor of John H. Holland. Center for the Study of Complex Systems. pp 111–121. <http://prediction-dynamics.com/>
- Wilson SW (2000) Mining oblique data with XCS. In: Proceedings of the international workshop on learning classifier systems (IWLCS-2000), in the joint workshops of SAB 2000 and PPSN 2000. Extended abstract
- Wilson SW (2001b) Function approximation with a classifier system. In: Spector L, Goodman ED, Wu A, Langdon WB, Voigt HM, Gen M, Sen S, Dorigo M, Pezeshk S, Garzon MH, Burke E (eds) Proceedings of the genetic and evolutionary computation conference (GECCO-2001). Morgan Kaufmann, San Francisco, CA, pp 974–981
- Wilson SW (2002a) Classifiers that approximate functions. *Natural Comput* 1(2–3):211–234
- Wilson SW (2002b) Compact rulesets from XCSI. In: Lanzi PL, Stolzmann W, Wilson SW (eds) Advances in learning classifier systems, Lecture notes in artificial intelligence, vol 2321. Springer, Berlin, pp 196–208
- Wilson SW (2007) Three architectures for continuous action. In: Kovacs T, LLòra X, Takadama K, Lanzi PL, Stolzmann W, Wilson SW (eds) Learning classifier systems. International workshops, IWLCS 2003–2005, revised selected papers, Lecture notes in computer science, vol 4399. Springer, Berlin, pp 239–257
- Wilson SW (2008) Classifier conditions using gene expression programming. In: Bacardit J, Bernadó-Mansilla E, Butz M, Kovacs T, Llorà X, Takadama K (eds) Learning classifier systems. 10th and 11th international workshops (2006–2007), Lecture notes in computer science, vol 4998. Springer, Berlin, pp 206–217
- Wilson SW, Goldberg DE (1989) A critical review of classifier systems. In: Schaffer JD (ed) Proceedings of the 3rd international conference on genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 244–255. <http://prediction-dynamics.com/>
- Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. *Neural Comput* 8(7): 1341–1390
- Wong ML, Leung KS (2000) Data mining using grammar based genetic programming and applications. Kluwer, Norwell
- Woods K, Kegelmeyer W, Bowyer K (1997) Combination of multiple classifiers using local accuracy estimates. *IEEE Trans Pattern Anal Mach Intell* 19:405–410
- Woodward JR (2003) GA or GP? That is not the question. In: Proceedings of the 2003 congress on evolutionary computation, CEC2003. IEEE Press, Washington DC, pp 1056–1063
- Yamasaki K, Sekiguchi M (2000) Clear explanation of different adaptive behaviors between Darwinian population and Lamarckian population in changing environment. In: Proceedings of the fifth international symposium on artificial life and robotics. pp 120–123
- Yao X (1999) Evolving artificial neural networks. *Proc IEEE* 87(9):1423–1447
- Yao X, Islam MM (2008) Evolving artificial neural network ensembles. *IEEE Comput Intell Mag* 3(1):31–42
- Yao X, Liu Y (1997) A new evolutionary system for evolving artificial neural networks. *IEEE Trans Neural Netw* 8:694–713
- Yao X, Liu Y (1998) Making use of population information in evolutionary artificial neural networks. *IEEE Trans Syst Man Cybern B* 28(3):417–425
- Zatuchna ZV (2005) AgentP: a learning classifier system with associative perception in maze environments. PhD thesis, University of East Anglia

Zatuchna ZV (2004) AgentP model: Learning Classifier System with Associative Perception. In 8th parallel problem solving from nature international conference (PPSN VIII). pp 1172–1182

Zhang B-T, Veenker G (1991) Neural networks that teach themselves through genetic discovery of

novel examples. In: Proceedings 1991 IEEE international joint conference on neural networks (IJCNN'91) vol 1. IEEE Press, Washington DC, pp 690–695

31 Coevolutionary Principles

Elena Popovici¹ · Anthony Bucci² · R. Paul Wiegand³ · Edwin D. de Jong⁴

¹Icosystem Corporation, Cambridge, MA, USA

elena@icosystem.com

²Icosystem Corporation, Cambridge, MA, USA

anthony@icosystem.com

³Institute for Simulation and Training, University of Central Florida,
Orlando, FL, USA

wiegand@ist.ucf.edu

⁴Institute of Information and Computing Sciences, Utrecht University,
The Netherlands

dejong@cs.uu.nl

1	<i>Introduction</i>	988
2	<i>Problems and Solutions</i>	993
3	<i>Design of Coevolutionary Algorithms</i>	1004
4	<i>Analysis of Coevolution</i>	1011
5	<i>The Future of Coevolution</i>	1025
6	<i>Concluding Remarks</i>	1028

Abstract

Coevolutionary algorithms approach problems for which no function for evaluating potential solutions is present or known. Instead, algorithms rely on the aggregation of outcomes from interactions among evolving entities in order to make selection decisions. Given the lack of an explicit yardstick, understanding the dynamics of coevolutionary algorithms, judging whether a given algorithm is progressing, and designing effective new algorithms present unique challenges unlike those faced by optimization or evolutionary algorithms. The purpose of this chapter is to provide a foundational understanding of coevolutionary algorithms and to highlight critical theoretical and empirical work done over the last two decades. This chapter outlines the ends and means of coevolutionary algorithms: what they are meant to find, and how they should find it.

1 Introduction

The inspiration for coevolutionary algorithms (CoEAs) is the same as for traditional evolutionary algorithms (EAs): attempt to harness the Darwinian notions of *heredity* and *survival of the fittest* for simulation or problem-solving purposes. To put it simply, a *representation* is chosen to encode some aspects of potential solutions to a problem into individuals, those individuals are altered during search using genetic-like *variation* operators such as mutation and crossover, and search is directed by selecting better individuals as determined by a fitness *evaluation*. With any luck, the iteration of these steps will eventually lead to high-quality solutions to a problem, if problem solving is the aim, or to interesting or realistic system behavior.

Usually, EAs begin with a *fitness function*, which for the purposes of this chapter is a function of the form $f : G \rightarrow \mathbb{R}$ that assigns a real value to each possible genotype in G . (Note that in certain branches of biology, fitness refers to the number of offspring an individual receives; thus, the use of the term fitness function in this way, while common in the evolutionary computation literature, breaks somewhat with biological tradition. However, seemingly less loaded alternative terms such as *objective function* evoke multi-objective optimization, a topic that will also be discussed briefly. In order to avoid confusion with other uses of the term *objective*, the term fitness function is used throughout the chapter.) Given such a function, the fitness relationship between any two genotypes $g_1, g_2 \in G$ is clear: $f(g_1)$ is compared with $f(g_2)$ to see which is more fit. By contrast, CoEAs do not use such a direct metric of the fitness of individuals. Instead, two individuals are compared on the basis of their outcomes from interactions with other individuals. (For this reason, the fitness in a CoEA has been called *subjective fitness*, subject as it is to the changing population(s). The objectively given and unchanging fitness function used in typical EA applications is *objective*.) As should become apparent as this chapter unfolds, this seemingly small change to how evaluation is done creates a variety of fundamental differences from traditional evolutionary computation. Most importantly, the fitness ranking of two individuals can change over time, a phenomenon that cannot happen in an ordinary EA. Indeed, virtually all of the techniques and ideas presented in this chapter are affected one way or another by this fundamental question: How could one possibly build an effective algorithm when any time one believes A is better than B it is possible that later one will believe B is better than A ?

This chapter is constructed to provide a foundational understanding of coevolutionary algorithms from a problem-solving point of view. Along the way, it will survey some of the many CoEAs that have been developed and the wide variety of domains to which they have been applied. Before going into detail, however, two coevolutionary algorithm schemes and problem classes are presented to give context for what follows.

1.1 Two Simple Coevolutionary Algorithms

In a *single population* CoEA (☞ [Algorithm 1](#)), individuals are evaluated by interacting with other individuals from that population. In a *multi-population* CoEA (☞ [Algorithm 2](#)), individuals in one population interact with individuals in one or several other populations.

Algorithm 1 SINGLE POPULATION CoEA

```
Initialize population
Select evaluators from population
Evaluate individuals from population by interacting with evaluators
while not done do
    Select parents from population
    Produce children from parents via variation
    Select evaluators from (children+, parents)
    Evaluate individuals from children by interacting with evaluators
    Select survivors for next generation
end while
return solution
```

Algorithm 2 MULTI POPULATION CoEA

```
for each pop ∈ populations do
    Initialize pop
    Select evaluators from (populations – pop)
    Evaluate individuals from pop by interacting with evaluators
end for
while not done do
    for each pop ∈ populations do
        Select parents from pop
        Produce children from parents via variation
        Select evaluators from (populations – pop)
        Evaluate individuals from children by interacting with evaluators
        Select survivors for next generation
    end for
end while
return solution
```

There has been some amount of controversy over whether a single-population algorithm could be fairly called coevolutionary. After all, biologists use the term coevolution to refer to genetic influence of two species over each other. Since a population refers to a collection of members of a species, it seems that one must have two populations to have coevolution. A pragmatic approach to the term is taken here. Problems in which the ranking of two entities can change depending on the presence or absence of other entities cause a certain set of issues for population-based problem-solving algorithms, issues that do not arise with ordinary optimization or multi-objective optimization problems. While these issues manifest differently in single- and multi-population algorithms, they are nevertheless present and must be taken into account if one hopes to design a successful problem-solving algorithm. Thus, at the risk of breaking with the biological terminology, both single- and multi-population algorithms are referred to as coevolutionary.

While many CoEAs are variations of these two main frameworks, several details have been omitted, some of which will be discussed later in this chapter. For example, a common mechanism not addressed by those frameworks is for a CoEA (either single- or multi-population) to have an *archive* that serves to evaluate individuals or to store potential solutions and is updated from the main population(s). Also, multi-population CoEAs can perform simultaneous or concurrent evolutionary steps, while the example algorithm is sequential. Finally, these two algorithmic schemes do not specify which interactions occur or how the results of interaction outcomes are aggregated into an evaluation that can be used by a selection method. These points will be addressed in [Sect. 2](#).

1.2 Problem Classes

As a consequence of subjective evaluation, the dynamics of coevolutionary algorithms can be frustratingly complex. One can view the interactions oneself as the basic unit of evaluation; in that view, the fitness landscape for an individual changes as a function of the content of the population(s). Intuitions about fitness landscapes from evolutionary computation do not easily map to this population-dependent, dynamic landscape situation. Nevertheless, coevolutionary algorithms ideally will leverage this mutability of the fitness landscape, adaptively focusing on relevant areas of a search space. This can be particularly helpful when problem spaces are very large or infinite. Additionally, some coevolutionary algorithms appear natural for domains that contain certain, known structure (Potter 1997; Stanley 2004) since search on smaller components in the larger structure can be emphasized. Most usefully, though, coevolution is appropriate for domains that have no intrinsic objective measure, which will be called *interactive domains*.

While the analogy with dynamic optimization (where the landscape also changes, but independently of the search process) may ground the reader, historically it has not generated insights into how to design successful coevolutionary algorithms. Instead, a fruitful approach has been to view interactive domains (and the problems that can be defined over them) as static, but structurally different and more complex than those of traditional optimization. This chapter is structured around this view.

Historically, the terms *cooperative* and *competitive* have been used to classify the domains to which coevolution is often applied. Indeed, game theory provides some guidance for making such distinctions. It can be argued that while such distinctions are relevant and at times useful for classifying the interactive domain over which an algorithm operates, they have

not been appropriate for classifying *problems* or algorithms. The interactive domain, like the fitness function, simply gives values to interactions. The problem specifies what to find and the algorithm finds it. Chess is an interactive domain, finding a chess-playing strategy capable of grand master rating is a problem. Experience shows that details of the problem definition and algorithm design have much more impact on the overall behavior of a CoEA than details of the interactive domain.

Thus, problems are primarily divided into classes based on what constitutes a solution. Two types of problems are highlighted: *test-based* problems and *compositional* problems. A test-based problem is one in which the quality of a potential solution is determined by its performance when interacting with some set of tests. By contrast, in compositional problems the quality of a solution to the problem involves an interaction among many components that together might be thought of as a team or assembly.

1.3 Successful Applications of Coevolutionary Algorithms

In the subsequent sections interactive domains and coevolutionary algorithms will be described in detail; however, it is useful to have some context with which to begin the discussion. Below are three simple successful applications of coevolution to three different problems. They will be presented as examples for the ideas discussed above, but they were also chosen for their historical impact.

1.3.1 Single Population CoEA Applied to a Test-Based Problem

In single population CoEAs applied to test-based problems, individuals serve two roles: at times they are used as (components of) potential solutions, while at other times they are used as tests to provide evaluation information about other individuals. The problem of finding good game-playing strategies can be viewed as a test-based problem since strategies are tested by playing against other strategies. Chellapilla and Fogel's AI checkers player *Blondie24* is a popular and successful application of a CoEA to such a problem (Chellapilla and Fogel 1999). A strategy in Blondie24 employs a traditional minimax algorithm, but uses a neural network for board evaluation. Individuals in the CoEA are vectors of weights for a fixed-structure neural network, and are evaluated by playing against other individuals. Points are awarded based on the win/loss/draw record, and fairly standard evolutionary programming methods are used to select players, hence weight vectors, with better records. The CoEA employed for this problem was able to produce a checkers player that is competitive with the best existing human and AI players.

- ▶ *Interactive Domain:* Game of checkers; *Test:* Opponent playing strategy; *Potential Solution:* Playing strategy; *Problem:* Find a strategy that beats the most opponents.

1.3.2 Two-Population CoEA Applied to a Test-Based Problem

Hillis' coevolution of sorting networks and challenging data sets uses two populations (Hillis 1990). One population represents sorting networks (that is, arrangements of

compare-and-swap circuits) while another represents unsorted data sets to test a network's sorting capability. The goal of the algorithm is to produce the smallest network possible that correctly sorts any given data set, but it does this while simultaneously honing ever more challenging and representative data sets. The technique produced a 61-comparator network, which is just one comparison larger than what was, at the time, the smallest-known network for a 16-input problem. A similar, non-coevolutionary technique described in that work was unable to produce networks smaller than 63 comparators.

- ▶ *Interactive Domain:* Running sorting network on data set; *Test:* Data set; *Potential Solution:* Sorting network; *Problem:* Find smallest, correct network.

1.3.3 Multi-population CoEA Applied to a Compositional Problem

There are a number of successful applications of CoEAs to compositional problems wherein problems are decomposed in some way and separate EAs are applied in parallel to the components, even though evaluation must involve some kind of aggregation or composition of components from the whole system. Perhaps the oldest, and still among the most successful, of these is Husbands and Mills' work on job-shop scheduling (Husbands and Mill 1991). In this work, individuals encode potential floor plans for managing jobs involving the processing of a particular widget constructed by the shop, and separate populations are used to optimize these process plans for each widget. Fitness, however, includes accounting for shared resources in the shop (time, space, etc.). There is also a population of arbitrators, agents that resolve conflicts between process plans for different widgets. Resulting schedulers can deal with a great deal of uncertainty on the job-shop floor.

While perhaps not as well known as the Blondie24 or sorting network examples, this work showcases the difference between a cooperative *domain* and compositional *problem*. In this case, the solution to the problem is a collection of floor plans (one for each component) and an arbitrator, which means the problem is compositional. However, the floor plans compete with one another in the evaluation function because they must share resources, so it is not strictly cooperative.

- ▶ *Interactive Domain:* Determination of complete job-shop schedule; *Components:* floor plans for different widgets, arbitrator for conflict resolution; *Potential Solution:* set of floor plans for each widget plus an arbitrator; *Problem:* Find an efficient and robust job-shop schedule.

1.4 Chapter Organization

History has shown that the naïve application of CoEAs to ill-understood domains is as likely to produce confusing and unsatisfactory results as to succeed. Therefore, at least from the perspective of problem solving, a careful algorithm design process is critical. This chapter is organized to emphasize this process: define the problem, define the relationship between the algorithm and the problem, implement the algorithm in a principled way, and analyze its behavior based on these principles.

The next section establishes clear definitions of domains, problems, and the types of solutions one expects to find. ◉ [Section 3](#) describes how problems relate to evaluation and

representation choices within a coevolutionary algorithm. This is followed by the presentation of analytical approaches currently available for understanding coevolutionary algorithm design, performance, and behavior. The concluding section provides some broader views of coevolutionary systems and outlines possible future directions of research and application.

2 Problems and Solutions

Coevolutionary algorithms are typically applied to interactive domains. Such domains generally lack an objective function giving a value to each potential solution. Rather, interactive domains encode the outcomes of interactions between two or more entities; depending on the domain, individual entities or the interaction itself may receive value as a result of an interaction. An algorithm must then decide how to use these outcomes to make decisions about which entities to promote in the next generation and which entities to demote or discard.

In a problem-solving context, an algorithm will need more than just an interactive domain. It will also require some way of deciding which entities in the domain are better than others. Coupled with information like that, an interactive domain becomes a co-search or co-optimization problem.

This section is concerned with detailing interactive domains, co-search, and co-optimization problems in the abstract, surveying some examples from both test-based and compositional problems, and beginning the discussion of what it means to extract a solution from such problems. Here, all dynamic or algorithmic considerations are left aside and instead the focus is on formalizing static definitions of problem classes to which CoEAs have been applied. Coevolutionary algorithms and their behavior on these problem classes are discussed in subsequent sections.

2.1 Interactive Domains

The formal notion of interactive domain can be defined as follows.

Definition 1 (Interactive Domain) An *interactive domain* consists of one or more functions, called *metrics*, of the form $p : X_1 \times X_2 \times \dots \times X_n \rightarrow R$, where

- Each i with $1 \leq i \leq n$ is a *domain role*
- An element $x \in X_i$ is an *entity* (playing the domain role i)
- Each X_i is an *entity set* (for the domain role i)
- A tuple $(x_1, x_2, \dots, x_n) \in X_1 \times \dots \times X_n$ is an *interaction*
- The value $p(x_1, x_2, \dots, x_n) \in R$ is an *outcome* (of the interaction)
- The ordered set R is the *outcome set*

□

Some remarks about this definition are in order.

Interactions may be relatively simple, such as applying a sorting network to an input sequence; or highly complex, such as simulating the job-shop scheduling activities given some arbitrator and several floor plans. Further, they may be direct, such as, again, applying a sorting network to an input sequence; or indirect, via an *environment*, such as the board

and dice in a checkers game or the shared resources in the scheduling example. We abstract away from such low level details and we are merely concerned with the outcome of the interaction.

The discussion of problem classes in [Sect. 1.2](#) touched on the distinction between cooperative and competitive domains. Here, those terms will be linked directly to the interactive domain. *Cooperative domains* have n metrics p_i , one for each domain role. The metric p_i is interpreted as giving an outcome to the entities playing role i . In many examples, $p_i = p_j$ for all roles i and j , so that the metrics only differ by which entity receives the outcome (but see [\(Popovici 2006\)](#) for a more nuanced discussion of cooperative domains). *Competitive domains*, by contrast, typically have only two roles. The two corresponding metrics often obey $p_2 = -p_1$, making the domain equivalent to a zero-sum game. Naturally, there is an enormous space of interactive domains that do not fall into either the cooperative or competitive categories. Note also that the terms “cooperative coevolution” and “competitive coevolution” refer to coevolutionary algorithms operating on such domains.

While an interactive domain has n entity sets, two or more of these sets may be the same. The word *type* will be used to refer to the sets from which entities are drawn, independently of what domain roles the entities are playing. There are special cases in which all domain roles are played by entities of the same set and, furthermore, the outcomes a particular entity receives do not depend on which role it plays. These will be referred to as *role-symmetric* domains.

Example 1 (Rock–paper–scissors) The game rock–paper–scissors furnishes a particularly simple example of an interactive domain. This game is generally expressed with a payoff matrix

	<i>Rock</i>	<i>Paper</i>	<i>Scissors</i>
<i>Rock</i>	0	-1	0
<i>Paper</i>	1	0	-1
<i>Scissors</i>	-1	1	0

To view rock–paper–scissors as an interactive domain, observe

- There are two roles
- $X_1 = X_2 = \{rock, paper, scissors\}$ are the entity sets, so there is one type
- The matrix encodes a single metric p , whose value is given to the entity on the line. This domain is role symmetric; for instance, *rock* receives the same payoff versus *paper* regardless of whether it is playing the first role or the second.

Example 2 Here is another example coming from an abstract game

	t_1	t_2	t_3
s_1	1	1	0
s_2	0	1	2
s_3	2	1	1

As an interactive domain, note

- There are two roles
- $X_1 = \{s_1, s_2, s_3\}; X_2 = \{t_1, t_2, t_3\}$ are the entity sets (there are two types)
- The matrix encodes a single metric p . This domain is not role symmetric

For a large majority of interactive domains that have been studied in practice, the outcome set R is a subset of \mathbb{R} ; however, that requirement is not strictly necessary (see (Bucci and Pollack 2002) for a discussion of ordered outcome sets that are not subsets of the reals).

Finally, note that this definition of interactive domain closely resembles the notion of the game defined by Ficici (2004). What we are calling entity is there called a behavior, reflecting that work's closer focus on agents selecting behaviors. What we are here calling an interaction is there called an event, and the notion of outcome corresponds to what Ficici calls a measurement. That work should be consulted for a more detailed treatment of interactive domains.

2.2 Co-search and Co-optimization Problems

A function $f: X \rightarrow \mathbb{R}$ gives values to elements of X . However, more information is needed to define a *problem* that can be solved. For instance, one may wish to find all the elements of X that maximize the value of f . Or, one may wish to find only one such element. Or, one may wish to find the elements that minimize f . Notice that for a fixed function f , there are many, distinct ways to decide what makes a “good” element of X . To give a well-defined search or optimization problem, one must specify not just the function f but also one way for deciding which elements of X are to be found.

By analogy, once an interactive domain is defined, one has a way of giving one or more values to interactions of entities, but does not yet have a well-defined problem to solve. The purpose of this section is to detail the kinds of problem that can be defined over interactive domains. These will be called *co-search problems* and *co-optimization problems*. First:

Definition 2 (Co-search Problem) Given an interactive domain, a *co-search problem* over it consists of

- A non-empty subset $I \subset (1, \dots, n)$ of the n domain roles
- A set \mathcal{C} of *potential solutions* aggregated from entities of the domain roles in I
- A *solution concept* that specifies a partitioning of the potential solutions into (actual) solutions and non-solutions. This is represented as a subset $\mathcal{S} \subset \mathcal{C}$ of the potential solutions

□

The last two components of the definition will be examined in more detail.

Potential Solutions: There are two critical points to emphasize:

- In many problems of practical interest, no single entity makes for a sufficient solution. Rock–paper–scissors stands as an intuitive example: none of the entities *rock*, *paper*, or *scissors* is, by itself, a reasonable strategy to play in this game. In such cases, potential solutions are *aggregated* from, meaning built or constructed out of, multiple entities.
- In some problems, one would like potential solutions to either lie in or be aggregated from entities in a single domain role, while in other problems, potential solutions should be aggregates of entities from several domain roles. The subset I is used to distinguish cases like these. If $I = (1)$, solutions are aggregated only from entities playing domain role 1, while if $I = (1, 2, \dots, n)$, solutions are aggregated from entities playing all n roles. One could, of course, have an I that is not all of $(1, \dots, n)$ but contains more than one domain role.

➊ Section 2.3 will give examples of what is meant by saying potential solutions are “aggregated from” entities. While a fully general formalization of this notion is beyond the scope of this chapter, in typical examples, potential solutions are sets of entities, mixtures of (distributions over) entities, tuples of entities, or perhaps combinations of these. (For instance, Nash equilibria are usually pairs of mixtures.) The notion of potential solution here is closely related to the idea of a *configuration* in (Ficici 2004), which should be consulted for more detail; Popovici (2006) discusses the idea of aggregating solutions in more detail as well.

Solution Concepts: The phrase “solution concept” originates in game theory (Osborne and Rubinstein 1994) and is largely synonymous with solution specification or solution definition. The phrase has been widely used in coevolution since being imported into the field by Ficici (2004). While the idea of a solution concept is presented in our abstract, as a subset of \mathcal{C} , in reality, solutions are identified using the metrics of the interactive domain. (What Ficici calls an *intensional* solution concept (Ficici 2004).) That is, it is typical to seek entities or combinations of entities that in some sense optimize the metrics of the interactive domain.

➋ Section 2.4 illustrates the point by describing several examples of solution concepts that have arisen in practice.

A *co-optimization problem* is closely related to a co-search problem. However, instead of a solution concept, a co-optimization problem specifies an order on the potential solutions and implicitly requires that solutions be maximal elements of the order. Specifically:

Definition 3 (Co-optimization Problem) Given an interactive domain, a *co-optimization problem* over it consists of

- A non-empty subset $I \subset (1, \dots, n)$ of the n domain roles;
- A set \mathcal{C} of *potential solutions* built from entities of the domain roles in I ;
- An ordering (which may be a partial order or even a preorder) \leq of \mathcal{C} such that if $c_1, c_2 \in \mathcal{C}$ and $c_1 \leq c_2$, c_2 is interpreted as being no worse a solution than c_1 . (Note that \leq is an ordering on potential solutions, not on numbers. In defining it, it may be the case that lower values from the domain’s metrics are better.) □

A co-optimization problem can be converted into a co-search problem by defining the solution concept $\mathcal{S} \subset \mathcal{C}$ to be the set of maximal elements of \leq . In that sense, co-optimization is a more refined notion than co-search.

As a final bit of terminology, recall that both co-search and co-optimization problems have a subset I of the domain roles that specifies which entities are aggregated into potential solutions. Let \widehat{I} denote the complement of I in $(1, \dots, n)$, so, for instance, if $n = 3$ and $I = (1, 2)$ then $\widehat{I} = (3)$. Then two *problem roles* can be distinguished:

- If $i \in I$, X_i is a *component role* and $x \in X_i$ are *component entities* or *components*
- if $i \in \widehat{I}$, X_i is a *test role* and $x \in X_i$ are *test entities* or *tests*

Note that \widehat{I} may be empty, meaning there is no test role explicitly defined by the problem. However, I cannot be empty by definition, meaning there is always at least one set of entities playing the role of component.

To summarize some of the key concepts relating to entities that have been introduced in this section:

- An interactive domain defines two or more *domain roles* and entities that play each role.
- The notion of type is coarser than that of domain role: several domain roles might correspond to the same type of entity, but each type corresponds to at least one domain role.
- A co-search or co-optimization problem lies at a conceptually higher level, defined over an interactive domain.
- Co-search and co-optimization problems define two *problem roles*, components and tests.
- At the problem level, the component role corresponds to one or more lower-level domain roles. The test role may correspond to zero or more domain roles.
- Since types sit at the domain role level, the relationship between types and problem roles can be complicated. There may be components of different types, tests of different types, components and tests that both have the same type, etc.

☞ [Section 1.2](#), besides distinguishing cooperative and competitive domains, also mentioned the distinction between compositional and test-based problems. Before elaborating on potential solutions in ☞ [Sect. 2.3](#) and solution concepts in ☞ [Sect. 2.4](#), we will define and illustrate these two important classes of co-search and co-optimization problems. Briefly, compositional and test-based problems correspond to the extremes where $|I| = n$ and $|I| = 1$, respectively. The middle ground between these extremes is virtually unexplored.

2.2.1 Compositional Problems

A *compositional problem* is a co-search or co-optimization problem in which all domain roles are used as components to build solutions. That is, $|I| = n$ and, conversely, $|\widehat{I}| = 0$. The idea is that each entity in each domain role is a component. To build potential solutions, one must use entities from each of the domain roles; one does not have a complete potential solution until one has used at least one component from each domain role. An intuitive example is a baseball team: one does not have a baseball team until one has a pitcher, a catcher, outfielders, etc.

Compositional problems have largely been explored under the rubric of cooperative coevolutionary algorithms (CCEAs). Besides theoretical works that study features of algorithm behavior on arbitrary or abstract test problems (Jansen and Wiegand [2003a, b, 2004](#)), or empirical work of algorithm dynamics on simple test problems (Popovici and De Jong [2005a, c](#)), work applying CCEAs to multivariate function optimization or multi-agent learning have also appeared. In multivariate function optimization, one treats each input variable as a distinct domain role, so that each entity in a particular domain role is a potential setting for one of the input variables (Potter and De Jong [1994, 2000](#); Panait et al. [2004](#)). Multi-agent learning applications have treated each domain role as the space of possible behaviors or actions for agents that form a team performing a task together (Parker and Blumenthal [2003](#); Panait et al. [2006c](#)).

2.2.2 Test-Based Problems

A *test-based problem* is a co-search or co-optimization problem in which $|I| = 1$; that is, in which a single domain role contains components, and all other domain roles are tests. Intuitively speaking, the test entities are used to probe or give information about the potential solutions that can be aggregated from the components.

Test-based problems are discussed as such in (De Jong and Pollack [2004](#)) and analyzed in connection with multi-objective optimization in (De Jong and Bucci [2007](#)); however, the idea

is implicit in early work on Pareto coevolution (Noble and Watson 2001; Ficici and Pollack 2001) and is more explicit in later works (De Jong 2004a, c; Monroy et al. 2006). Bucci and Pollack (2002, 2003b) formally analyze test-based problems from the perspective of order-theory, while Bucci and Pollack (2005) and Panait and Luke (2006) treat compositional problems using ideas developed to study test-based problems. De Jong (2005), Jensen (2001), Branke and Rosenbusch (2008), Barbosa (1999), and Stuermer et al. (2009) approach two types of test-based problems not based on Pareto dominance.

In many of these works, the terms *candidate*, *candidate solution*, or *learner* are used to describe what is here described as a component. Likewise, the term *test* is used in those works to denote what is here called an entity playing the test role.

2.3 Potential Solutions in Co-search and Co-optimization Problems

❷ Section 2.2 defined a co-search problem as one that, given an interactive domain, specifies a solution concept as a subset of a set of *potential solutions* that are built from the interactive domain. This section is intended to detail, by example, several common extant ways of creating potential solutions from entities.

2.3.1 Single Entity

The simplest and most obvious set of potential solutions is a set of entities. Consider an interactive domain with n roles and a co-search problem with $I = (1)$, so that the component role is being played by the entities in domain role 1. Then, one set of potential solutions for this problem is the set X_1 of components itself. In this case, the set \mathcal{C} is X_1 , so that a solution concept \mathcal{S} gives a subset of X_1 .

Seminal work by Hillis (1990) and Sims (1994) both use single entities as potential solutions. Hillis' algorithm sought a single sorting network; there was no sense in which two or more sorting networks could be combined. Likewise, Sims' algorithm sought a morphology and behavior for a simulated creature; again, there was no mechanism for combining several creatures into a composite.

Stipulating that single entities serve as solutions is appropriate in certain domains, particularly those involving the design of a complex object (that is, when no clear way of combining entities together is present); or, those in which one expects that a single entity that solves the problem may exist. However, as noted above about rock–paper–scissors, many domains have no single entity that is adequate to solve the problem.

2.3.2 Sets

Consider, for concreteness, that we have an interactive domain with two roles and our co-search problem specifies that $I = (1)$. The entities in X_1 are components, while those in X_2 are tests. In this example, the set \mathcal{C} is the powerset of X_1 , $\mathcal{P}(X_1)$. A solution concept gives a subset of $\mathcal{P}(X_1)$, equivalently one or more subsets of X_1 , where each subset is a solution.

Set-based notions of potential solution have been studied under the umbrella of Pareto coevolution (Ficici and Pollack 2001; Noble and Watson 2001). Treating each test as

an objective to be optimized, the problem-solving goal is to find (an approximation of) the non-dominated front among the components. Therefore, a potential solution is a subset of X_1 , in other words, an element of $\mathcal{C} = \mathcal{P}(X_1)$. The non-dominated front itself is the solution.

When solution concepts are further elaborated in [Sect. 2.4](#), one will see the situation is more subtle; in fact, a set related to, but not quite, the non-dominated front is desirable. However, that discussion will be deferred until then.

2.3.3 Mixtures

Let's maintain the co-search problem of the previous example. However, let's say that a solution is not a subset of X_1 , but a probability distribution over X_1 . Following Vose ([1999](#)), denote the set of these by A^{X_1} . Then another choice for the set of potential solutions \mathcal{C} is A^{X_1} .

Mixture-based solutions are most often discussed in the context of Nash equilibria; the Nash memory ([Ficici and Pollack 2003](#)) and parallel Nash memory ([Oliehoek et al. 2006](#)) are examples of algorithms that treat mixtures as potential solutions.

2.3.4 Compositions in General

Sets and mixtures are both examples of aggregates; that is, potential solutions created by putting together components drawn from one or more of the component roles defined in the problem. Sets and mixtures are particularly simple kinds of aggregates. More complicated compositions are conceivable.

Compositional coevolution in general aims to discover good assemblies. The originally stated aim of cooperative coevolutionary algorithms, from which the more general notion of compositional coevolution sprang, was to attack the problem of evolving complicated objects by explicitly breaking them into parts, evolving the parts separately, and then assembling the parts into a working whole ([Potter and De Jong 2000](#)). The nature of the “coadapted subcomponents” discussed in that work was not strictly defined; hence, the idea is that any subcomponent of any assembly (aggregate) was fair game for the method. Since then, compositional coevolution has been applied to coevolving teams of agents performing a task, for instance, where the composition here is of several different agents collaborating as a team.

In the language developed in this section, agents or coadapted subcomponents are domain entities playing the component role of a compositional problem. A collaboration among entities is an interaction, and the means for evaluating a team or assembly would constitute the metric of the problem. The potential solutions in these cases, then, are the possible tuples over the domain roles; that is, $\mathcal{C} = X_1 \times \dots \times X_n$.

Though the translation into the notion of a co-search problem is less obvious, another important, nontrivial example of composition found in coevolutionary algorithms are in applications of neuro-evolutionary algorithms. The neuro-evolution through augmenting topologies (NEAT) algorithm ([Stanley and Miikkulainen 2002a](#)) has been applied to coevolve robot controllers in a simulated robot duel ([Stanley and Miikkulainen 2002a, b, 2004](#)) as well as players of a variant of the video game Pong ([Monroy et al. 2006](#)). A hallmark of the NEAT algorithm is the separation between components of neural networks, which consist of small assemblies of neurons and associated weighted synapses, and the topology of a

complete network, which functions as a blueprint for how assemblies are put together. A second, important feature of NEAT emphasized in these works is the possibility for complexification or elaboration: since neural network topologies are not limited by NEAT, they can grow to arbitrary complexity, elaborating on previously evolved behaviors. In contradistinction to typical CCEAs, which prespecify the size and complexity of compositions, NEAT permits open-ended complexity increase. (And, one would hope, an increase in capability as well.)

The Symbiogenic Evolutionary Adaptation Model (SEAM) described by Watson (2002) includes a form of variational operator designed to explicitly compose evolving entities. Additionally, SEAM employs a group evolutionary mechanism of composition in conjunction with notions of test-based evolution by using Pareto dominance to determine whether variation is likely to result in productive compositions.

Finally, it is worth pointing out that in general game-theoretic analysis, Nash equilibria are pairs of mixtures of pure strategies. In many cases, only the first mixture in the pair is used to solve a problem; the other is there for testing purposes. However, should one actually need both mixtures in the pair, the set of potential solutions would be $\Lambda^{X_1} \times \Lambda^{X_2}$, which involves two levels of aggregating: first mixing (over X_1 and X_2), then pairing.

2.4 Solution Concepts

This section will detail several solution concepts for both compositional and test-based problems that have been used in coevolutionary algorithms. Recall that a solution concept specifies a subset of the potential solutions, $\mathcal{S} \subset \mathcal{C}$. Each example will detail the interactive domain, the co-search problem including the space of potential solutions, and how solutions are determined from the metric(s) of the domain. Solution concepts for compositional problems are presented first.

2.4.1 Compositional Solution Concepts

Ideal Team

Consider an interactive domain with n domain roles, entity sets X_1, X_2, \dots, X_n and one metric $p : X_1 \times \dots \times X_n \rightarrow \mathbb{R}$. Consider also a co-search problem over this domain with potential solutions drawn from the set of tuples $\mathcal{C} = X_1 \times \dots \times X_n$, so in fact this is a compositional problem. Observe there is a one-to-one correspondence between a potential solution and an interaction. The ideal team solution concept defines as solutions those potential solutions that maximize the value received from the metric

$$\mathcal{S} = \{\bar{x} \in \mathcal{C} \mid \forall \bar{x}' \in \mathcal{C}. p(\bar{x}) \leq p(\bar{x}') \Rightarrow p(\bar{x}) = p(\bar{x}')\} \quad (1)$$

where the shorthand \bar{x} is used to denote an arbitrary tuple in \mathcal{C} . In multi-agent parlance, the teams for which no other teams perform better are ideal teams.

The ideal team is not the only relevant compositional solution concept, and the question has been posed to what extent cooperative coevolution algorithms converge to it (Wiegand 2004). Another solution concept of interest arising from that investigation is *maximizing robustness*. The idea here is that rather than maximizing the outcome of a single potential solution, a solution should be robust in the sense that a “small” change in one of the components results in “small” changes in the composite’s value.

2.4.2 Test-Based Solution Concepts

Unless otherwise specified, the following examples of solution concepts for test-based problems are appropriate for problems of the following form:

- The interactive domain has two roles with entity sets X_1 and X_2 .
- There is a single metric $p : X_1 \times X_2 \rightarrow \mathbb{R}$.
- The co-search problem specifies X_1 as the only component role and X_2 as the only test role.

What varies in these examples is the set of potential solutions and how the subset of solutions is defined. These will be detailed now.

Best Worst Case

Best worst case operates over single-entity potential solutions, meaning $\mathcal{C} = X_1$. This solution concept specifies as solutions those components that maximize the minimum possible outcome over interactions with all tests. (This version has also been called *maximin*. The corresponding *minimax* can similarly be defined.) That is,

$$\mathcal{S} = \left\{ x \in \mathcal{C} \mid \forall x' \in \mathcal{C}. \min_{t \in X_2} p(x, t) \leq \min_{t \in X_2} p(x', t) \Rightarrow \min_{t \in X_2} p(x, t) = \min_{t \in X_2} p(x', t) \right\} \quad (2)$$

This criterion is appropriate in real-world domains where one needs to protect against the worst scenario possible.

Simultaneous Maximization of All Outcomes

Simultaneous maximization of all outcomes requires a solution to be a component that maximizes its outcome over all possible tests simultaneously. That is, $\mathcal{C} = X_1$ is a single entity set of potential solutions, and the solution concept is

$$\mathcal{S} = \{x \in \mathcal{C} \mid \forall x' \in \mathcal{C} \forall t \in X_2. [p(x, t) \leq p(x', t) \Rightarrow p(x, t) = p(x', t)]\} \quad (3)$$

This solution concept has a limited application scope, as for many problems, there does not exist a single potential solution that simultaneously maximizes the outcome against all possible tests.

Maximization of Expected Utility

Maximization of expected utility (MEU) is also relevant when $\mathcal{C} = X_1$. It specifies as solutions those components that maximize the expected score against a randomly selected opponent

$$\mathcal{S} = \{x \in \mathcal{C} \mid \forall x' \in \mathcal{C}. \mathbf{E}(p(x, X_2)) \leq \mathbf{E}(p(x', X_2)) \Rightarrow \mathbf{E}(p(x, X_2)) = \mathbf{E}(p(x', X_2))\} \quad (4)$$

where one is abusing notation and treating X_2 as a uniformly distributed random variable ranging over the set of tests, so that $\mathbf{E}(p(x, X_2))$ is the expected value of $p(x, t)$ when t is selected uniformly randomly from X_2 .

❸ [Equation 4](#) is equivalent to

$$\mathcal{S} = \left\{ x \in \mathcal{C} \mid \forall x' \in \mathcal{C}. \sum_{t \in X_2} p(x, t) \leq \sum_{t \in X_2} p(x', t) \Rightarrow \sum_{t \in X_2} p(x, t) = \sum_{t \in X_2} p(x', t) \right\} \quad (5)$$

when all the sums are defined. That is, maximizing expected utility is equivalent to maximizing the sum of outcome values over all tests when that sum is defined for all the components. Thus, MEU essentially assumes that all tests are of equal importance, which limits its generality.

Nash Equilibrium

The Nash equilibrium solution concept is inspired by game theory (Osborne and Rubinstein 1994) and operates over mixtures of components. When the problem is as before, $\mathcal{C}_1 = \Lambda^{X_1}$. However, deviating slightly from the previous examples, problems with $I = (1, 2)$ will also be considered, so that $\mathcal{C}_2 = \Lambda^{X_1} \times \Lambda^{X_2}$. That is, there are no tests, so in fact \mathcal{C}_2 defines a compositional problem. Furthermore, in both cases it will be assumed that there are two metrics, $p_1 : X_1 \times X_2 \rightarrow \mathbb{R}$ and $p_2 : X_1 \times X_2 \rightarrow \mathbb{R}$ interpreted as giving outcomes to entities in X_1 and X_2 , respectively. In many domains such as those arising from zero sum games, $p_1(x, y) = -p_2(x, y)$ for all $x \in X_1$ and $y \in X_2$.

Consider the case \mathcal{C}_2 first. Let $\alpha \in \Lambda^{X_1}$ and $\beta \in \Lambda^{X_2}$ be two mixtures over X_1 and X_2 , respectively. These can be written as formal sums

$$\alpha = \sum_{x \in X_1} \alpha_x \cdot x \quad (6)$$

and

$$\beta = \sum_{y \in X_2} \beta_y \cdot y \quad (7)$$

where α_x is the probability assigned to the component $x \in X_1$ by the mixture α , and β_y is the probability assigned to the test $y \in X_2$ by β and, since α and β are both distributions, $\sum_{x \in X_1} \alpha_x = 1$ and $\sum_{y \in X_2} \beta_y = 1$. Using this notation, define a function $\mathbf{E}_{p1} : \Lambda^{X_1} \times \Lambda^{X_2} \rightarrow \mathbb{R}$ such that for all $(\alpha, \beta) \in \Lambda^{X_1} \times \Lambda^{X_2}$

$$\mathbf{E}_{p1}(\alpha, \beta) = \sum_{\substack{x \in X_1 \\ y \in X_2}} \alpha_x \cdot \beta_y \cdot p_1(x, y) \quad (8)$$

$\mathbf{E}_{p1}(\alpha, \beta)$ is interpreted as giving the expected outcome that the mixture α receives when interacting with β . $\mathbf{E}_{p2}(\alpha, \beta)$ is defined similarly and gives the expected outcome to β .

A Nash equilibrium is a pair $(\alpha, \beta) \in \Lambda^{X_1} \times \Lambda^{X_2}$ such that neither α nor β can unilaterally change to some other mixture and receive a higher payoff. That is, (α, β) is a Nash equilibrium if the following two conditions hold:

- For all $\alpha' \in \Lambda^{X_1}$, $\mathbf{E}_{p1}(\alpha, \beta) \leq \mathbf{E}_{p1}(\alpha', \beta) \Rightarrow \mathbf{E}_{p1}(\alpha, \beta) = \mathbf{E}_{p1}(\alpha', \beta)$
- For all $\beta' \in \Lambda^{X_2}$, $\mathbf{E}_{p2}(\alpha, \beta) \leq \mathbf{E}_{p2}(\alpha, \beta') \Rightarrow \mathbf{E}_{p2}(\alpha, \beta) = \mathbf{E}_{p2}(\alpha, \beta')$

The Nash equilibrium solution concept for problems with $\mathcal{C}_2 = \Lambda^{X_1} \times \Lambda^{X_2}$ is then

$$\mathcal{S}_2 = \{(\alpha, \beta) \in \mathcal{C}_2 \mid (\alpha, \beta) \text{ is a Nash equilibrium}\} \quad (9)$$

For $\mathcal{C}_1 = \Lambda^{X_1}$, in other words problems in which one only cares about the mixture over components in X_1 , the Nash equilibrium solution concept is

$$\mathcal{S}_1 = \pi_1(\mathcal{S}_2) \quad (10)$$

where π_1 projects a pair (α, β) onto the first coordinate, α .

An attractive feature of the Nash equilibrium as a solution concept is that Nash equilibria have certain “security” guarantees: The expected payoff to α in a Nash equilibrium (α, β) can be no lower than $E_{p1}(\alpha, \beta)$, regardless of the strategy with which it interacts. In a game like $\{\text{rock}, \text{paper}, \text{scissors}\}$, for example, any individual strategy like *rock* will receive a -1 payoff against some opponent (if *paper* plays against *rock*, for example). The mixture $\frac{1}{3} \cdot \text{rock} + \frac{1}{3} \cdot \text{paper} + \frac{1}{3} \cdot \text{scissors}$, by contrast, will never receive an expected payoff lower than 0 against any opponent because $(\frac{1}{3} \cdot \text{rock} + \frac{1}{3} \cdot \text{paper} + \frac{1}{3} \cdot \text{scissors}, \frac{1}{3} \cdot \text{rock} + \frac{1}{3} \cdot \text{paper} + \frac{1}{3} \cdot \text{scissors})$ is a Nash equilibrium for that game and $E_{p1}(\frac{1}{3} \cdot \text{rock} + \frac{1}{3} \cdot \text{paper} + \frac{1}{3} \cdot \text{scissors}, \frac{1}{3} \cdot \text{rock} + \frac{1}{3} \cdot \text{paper} + \frac{1}{3} \cdot \text{scissors}) = 0$. Note that this is a kind of worst-case guarantee for the mixture α that is similar in spirit to that sought in the best worst case solution concept; the primary difference is that the best worst case seeks single components with such a guarantee, while Nash equilibrium seeks a mixture of many components.

Pareto Optimal Set

Multi-objective optimization extends traditional optimization through the introduction of multiple *objectives*. This may be viewed as the use of a function that is vector valued rather than scalar. In Pareto-based solution concepts, every possible test is viewed as an objective to be optimized.

Potential solutions for the Pareto optimal set solution concept are subsets of X_1 ; that is, $\mathcal{C} = \mathcal{P}(X_1)$. Here, the set of solutions will consist of a single member, the non-dominated front, which is defined

$$\mathcal{F} = \{x \in \mathcal{C} \mid \forall x' \in \mathcal{C}. [\forall t \in X_2. [p(x, t) \leq p(x', t)] \Rightarrow \forall t \in X_2. [p(x, t) = p(x', t)]]\} \quad (11)$$

Let the *Pareto covering* order be an order on X_1 defined as follows:

$$x \preceq x' \quad \text{if } \forall t \in X_2. p(x, t) \leq p(x', t) \quad (12)$$

for all x and x' in X_1 . This definition essentially says that the outcome of x' against any test is at least as high as that of x . However, \preceq is not a total order on X_1 , because it is possible that $p(x, t) < p(x', t)$ while $p(x, t') > p(x', t')$ for two tests $t, t' \in X_2$. Furthermore, while \preceq is reflexive and transitive, it need not be a partial order: two distinct components x and x' might receive precisely the same outcomes on all tests, so $x \preceq x'$ and $x' \preceq x$ both hold and antisymmetry fails to hold.

The definition of \mathcal{F} can be simplified using \preceq

$$\mathcal{F} = \{x \in \mathcal{C} \mid \forall x' \in \mathcal{C}. x \prec x' \Rightarrow x' \preceq x\} \quad (13)$$

The Pareto optimal set solution concept is defined as

$$\mathcal{S} = \{\mathcal{F}\} \quad (14)$$

Some remarks:

- There is a formal similarity between \bullet Eq. 3, the simultaneous maximization of all objectives, and the non-dominated front \mathcal{F} defined in \bullet Eq. 11. They differ only in the placement of the quantifier $\forall t \in X_2$. However, while formally similar, these two solution concepts have significantly different interpretations: while \bullet Eq. 13 shows that \mathcal{F} is the set of *maximal* elements of the order \preceq , in fact \bullet Eq. 3 is the set of *maxima* of that order, the difference being that maxima must be larger than all other components across all tests simultaneously. Thus, the non-dominated front puts weaker constraints on its members

than the simultaneous maximization of all objectives, and one would expect the former to be a larger set of components.

- The Pareto optimal set solution concept has only one member, \mathcal{F} , which is itself a set of components. This definition is chosen for consistency with the other solution concepts; each solution concept is a set whose members are solutions. The non-dominated front forms a single, unique solution to a multi-objective problem. A limitation of the Pareto optimal set solution concept is that this set may be very large; while not being covered is a sensible minimum requirement for solutions, for certain problems it may be insufficiently weak, in that it can insufficiently narrow down the set of possible solutions.
- As noted, two components may receive precisely the same outcomes against all possible tests. Thus, while they may be distinct components in the set X_1 , as far as performance against the tests in X_2 goes, they are indistinguishable.

Pareto Optimal [Minimal] Equivalence Set

The indistinguishability of some elements of the non-dominated front entails this set may be too large. One can partition this set in equivalence classes by this property, like so. Say that $x \sim x'$ if $x \preceq x' \wedge x' \preceq x$; that is, x and x' are equivalent if they receive the same outcomes against all tests in X_2 . \sim is an equivalence relation on X_1 . (This is a simple, well-known consequence of \preceq being a preorder.) It is reasonable to suppose that one only needs one representative of each equivalence class under \sim , and this intuition leads to two more solution concepts.

A Pareto optimal equivalence set is a set containing *at least one* element from each equivalence class of the full Pareto optimal set. Note that while there is a single Pareto optimal set, there may be a combinatorial number of Pareto optimal equivalence sets. A Pareto optimal minimal equivalence set contains *exactly one* component from each equivalence class of the full Pareto optimal set. There may be a combinatorial number of Pareto optimal minimal equivalence sets as well.

These two solution concepts typically define a smaller set of potential solutions than the Pareto optimal set. Yet, depending on the characteristics of the problem, such a set can still be very large.

3 Design of Coevolutionary Algorithms

As the previous sections have shown, there is a wide spectrum of problems in interactive domains. While throughout this chapter we discuss generic issues that any algorithm targeted at such co-search problems will have to address, in this section, the main focus is on issues that are specific to approaching these problems via coevolutionary algorithms. Like with any method, application of coevolutionary algorithms to a problem tends to go more smoothly and have more opportunity for success when algorithm engineers *first* go through the process of formalizing the domain and the problem, and *then* design the algorithm.

For domains that are inherently interactive, the formalizing process involves more rationalization than choosing. However, sometimes one may want to reformulate a noninteractive domain into an interactive one. The domains of problems featuring the ideal team solution concept are often the result of such a reformulation. In such cases, many different choices may exist as to how to decompose elements in the original domain, and they may have different effects on problem difficulty.

The design process entails finding useful and productive ways of mapping problem particulars into the algorithmic framework, and heuristic search tends to require some means of making qualitative comparisons and decisions in order to guide the path of the algorithm. There are principles behind, and effects from these decisions. In this section, the choices available to the algorithm designer are discussed, while the biases introduced by these mapping choices are discussed throughout [Sect. 4](#).

Once the engineer identifies the types, domain roles, problem roles, and potential solution set, they must determine how a CoEA can be structured to represent them, and encode data structures within the algorithm to instantiate them in a way that is consistent with the problem's solution concept. Such decisions are considered to relate to *representation*. Additionally, one must decide how to explore the set of interactions, how outcomes from multiple interaction will be aggregated for selection purposes, and how interaction information is communicated throughout various parts of the algorithm. Such decisions are considered to relate to *evaluation*.

3.1 Representation

Successful design and application of any heuristic depends on a number of key representation decisions for how search knowledge is encoded and manipulated. In any evolutionary system, these choices involve questions about how aspects of potential solutions are represented genotypically, modified by genetic operators, and expressed phenotypically.

Coevolution brings additional subtleties to representation issues that are worth special consideration. For one, the relationship between basic terms such as *individuals*, *populations*, and *solutions* can be quite complicated in coevolution. Related to these terms are the foundational, domain-centric terms discussed above such as *test* and *component*, and a well-designed CoEA should consider how these problem-based concepts map to representations used by the search system. Additionally, many modern CoEAs make use of archival and memory mechanisms for a variety of purposes, and it is important to understand how they relate to the above notions. These notions will be discussed in turn and examples of common/possible mappings from problem to algorithm will be provided.

3.1.1 Individuals

In most traditional evolutionary systems, individuals represent potential solutions to some search problem, and the EA manipulates encoded potential solutions by modifying them using variational operators (mutation, crossover) and choosing the better from among them using selection operators (proportionate selection, rank selection, etc.). When optimizing some real-valued function, an individual might be encoded as a vector of real values to be used as arguments for the function, mutations could be accomplished by applying vectors of Gaussian noise, and selection might involve simply choosing those with the highest objective function value. Since the goal of the search is to find an optimal argument vector, one can typically equate potential solution and individual. Even when the EA is searching program spaces, such as in genetic programming, traditional algorithms still make use of individuals who are essentially procedural solutions to the problem.

In coevolution, individuals serve the same mechanistic purpose as in any evolutionary algorithm: they are the fundamental units being manipulated by the search operators

themselves. That is, mutation and crossover modify individuals, and selection chooses among them.

Definition 4 (Individual) An *individual* is a representational unit that is subject to selection and variational operators. \square

But individuals in a CoEA will not always represent potential solutions to the problem being solved. In coevolution, one must think of individuals in terms of how they relate to types, domain roles, and problem roles, *in addition to* traditional issues of how to encode problem aspects. For example, suppose an AI is being developed for controlling an airplane that generalizes over different flight conditions and this is formalized as follows: an interactive domain consists of airplane controllers interacting with different flight conditions/scenarios; the problem is test-based where the potential solutions are controllers and the flight scenarios are tests; and the solution concept is maximum expected utility. One might design a CoEA that maintains two kinds of individuals, corresponding to the two types of the domain, namely individuals representing controllers and individuals representing flight scenarios. But one still has to decide how controllers and scenarios will *actually be encoded*, which will involve additional choices. The consideration of how types, domain roles, and problem roles map to individuals is subtly different from traditional genotype/phenotype decisions.

Most often, individuals represent entities from the domain's types. Thus, for a domain with two symmetric roles, one may have a single kind of individual, and such individuals might participate in interactions in either of the domain's roles. In general, one will have at least as many kinds of individuals as types in the domain. From a problem-role perspective, individuals might represent components of a potential solution or tests helping to judge the quality of potential solutions. Hillis (1990) provides a less common example, where some of the individuals do not correspond to entities, but to sets of entities, namely, those playing the test role. The relationship between individual and solution will be revisited in [Sect. 3.1.4](#) dedicated to solutions.

3.1.2 Populations

Just as with traditional EAs, it is useful to think of populations as simply collections of individuals. Most traditional EAs (excepting island models) have only a single, explicit population, while many CoEAs have multiple populations.

But even in traditional methods, the notion of a population can be a bit murky when dealing with mechanisms such as spatial models, niching, or restrictive mating (Horn 1995; Spears 1994) since stable subsets of individuals in the overall population can arise between which very little genetic material is exchanged. These subsets are sometimes referred to as “populations,” and it is interesting to note that many of these are essentially coevolutionary in nature since fitness can be quite subjective (e.g., niching methods using fitness sharing).

Definition 5 (Population) A *population* is:

1. A set (or multiset) of individuals
2. A subset of individuals isolated from other individuals by some kind of barrier (e.g., inability to interbreed successfully, geographic, etc.) \square

Designing effective CoEAs involves decisions about how aspects of a problem map to populations. The main purpose of the populations is to provide an exploration mechanism for the entity sets of the different types in the domain. Thus, a CoEA will generally maintain at least one population per type and most often a single population per type.

For example, for an ideal team problem in a domain with a fixed number of asymmetric roles (and therefore types), the typical CoEA will maintain one population for each. For a test-based problem in a domain with two asymmetric roles (and therefore types), the typical CoEA will also maintain one population for each type.

For domains with two symmetric roles (thus a single type), the problem definition itself may not distinguish between the component role, and the test role, and the algorithm designer can choose to do the same, by maintaining a single population, or to make the distinction explicit by maintaining two populations. A unique work by Potter (1997) features an ideal team problem in a domain with a variable (unbounded) number of symmetric roles (neural networks with variable number of neurons) approached via a CoEA with a dynamic number of populations, all containing individuals of the same type.

Populations may also be used to represent potential solutions, for example when these are aggregates of entities (and therefore individuals) from one type, as is the case with Pareto optimal solution concepts. (This is analogous to traditional multi-objective optimization, where the population can be used as an approximation of the Pareto-front.)

Finally, yet importantly, individuals in a population, by taking part in interactions, serve the purpose of evaluating individuals in the same or other populations. This is because while a problem definition may specify only some of the domain roles as contributing to the potential solution set, an algorithm needs some criteria based on which to select and promote individuals for the test roles as well.

3.1.3 Archives

Coevolutionary methods often employ another kind of collection, typically referred to as an *archive*. Archives span generations, and thus can be considered a kind of search memory. The purpose of the archives is to help the populations with, or release them from, some of their multiple duties. Archives can allow algorithm designers to separate exploration from evaluation and/or solution representation.

Definition 6 (Archive) An *archive* is a collection of individuals that spans multiple generations of a coevolutionary algorithm. □

Thus, archives often contain the end solution (Rosin and Belew 1997) or actually are the solution, mainly when potential solutions are aggregations of entities (De Jong 2004a, c, 2005; Oliehoek et al. 2006; Ficici and Pollack 2003). So an analogy in traditional evolutionary computing (EC) for this purpose of an archive is the *best-so-far* individual. But they need not only represent solutions, they can and typically do influence the search – a bit like elitism would in a traditional generational EA.

Archives can also serve for evaluation purposes. A simple example of this are *hall-of-fame* methods (Rosin and Belew 1997) where successful individuals discovered during search are added to a hall-of-fame and part of the fitness of new individuals comes from the outcome of interactions with some subset of individuals from that archive.

In domains with multiple types, there might be an archive for each type (in addition to a population for each type). Most archive-based CoEAs involve an `UPDATE` step where individuals in a current population are considered for inclusion in the corresponding archive, and individuals in that archive may be reorganized or removed. There is a certain appeal to this approach because there are often straightforward ways to explicitly implement a particular solution concept by crafting the correct update procedure for an archive.

3.1.4 Solutions

A potential solution can be many different things even as part of the problem specification, the latter being a higher-level notion defined over the low-level elements of an interactive domain. It should therefore not be surprising that, as already seen, a potential solution can map to different things in the algorithm. A potential solution may be a single individual, in which case it may be extracted out of

- Any population (e.g., if approaching a maximum expected utility problem in a domain with two symmetric roles via two populations)
- A specific population (e.g., if approaching a maximum expected utility problem in a domain with two asymmetric roles via two populations)
- An archive (e.g., if using a hall-of-fame approach)

A potential solution may also be a set of individuals, in which case it may be

- The entire contents or a subset of an archive or population (e.g., for CoEAs approaching Pareto optimal solution concepts)
- A collection of individuals, one from each archive/population (e.g., for CoEAs approaching the ideal team solution concept for asymmetric domains)

Further, there may be populations/archives in a CoEA that never contribute any individuals to a potential solution (e.g., populations/archives corresponding to test roles in a test-based problem that are nevertheless used during evaluation).

3.2 Evaluation

Some issues concerning evaluation are pertinent to both single- and multi-population CoEAs. Regardless of whether interactions occur between individuals in the same population or in different populations, decisions need to be made as to what *interactions* should be assessed and how the outcomes of those interactions should be *aggregated* to give individuals fitness. When using multiple populations, the additional issue of *communication* between these populations arises. Each of these three matters will be discussed in turn.

3.2.1 Interactions

The definition of interactions was discussed in the previous section. Here, the focus is on the *selection* of interactions (also referred to as the *interaction method* or, in cooperative domains, *collaboration method*). The simplest choice is to assess all interactions possible, given the

individuals present in the system at evaluation time. This has been referred to as full mixing or complete mixing. While no additional decisions would have to be made, this choice has the disadvantage of being very expensive, as the time cost of the algorithm is counted in interactions assessed. To reduce cost, one must assess only some of all possible interactions. This immediately raises the question “which ones?”

There are two main approaches to choosing a subset of interactions: individual centric and population centric. With the individual-centric approach, one individual at a time is considered, a set of interactions is chosen for that individual to take part in, and after the interactions are assessed, the individual’s fitness is computed. These interactions may be reused (but generally are not) for computing the fitness of the other individuals that took part in them. These other individuals have historically been called *collaborators* in cooperative domains and *opponents* in competitive domains. In this context, the phrase *sample size* denotes the number of interactions used per fitness evaluation. Note that if there is no reuse, the number of interactions an individual takes part in may be greater than the number of interactions used for its evaluation. Thus, while the sample size may be (and usually is) the same for all individuals, the number of interactions that individuals are involved in may vary.

A simple and quite common approach is to use a single interaction per fitness evaluation and have the other individuals in this interaction be the best from their respective populations. When solving ideal team problems, this has been termed *single-best collaboration method* by Wiegand (2004), while for domains considered competitive, it has been referred to as *last elite opponent (LEO) evaluation* (Sims 1994). More generally, it is called the *best-of-generation* interaction scheme.

With the population-centric approach, a *topology* of interactions is picked such that any individual in the population(s) is used at least once, these interactions are assessed and then fitness is computed for all individuals. Tournaments, such as single-elimination or round-robin, are the most common examples for single-population algorithms. Both for single- and for multi-population models, shuffle and pair (also called bipartite pairing) is a population-centric method that simplifies the reuse of assessed interactions (meant to reduce computational time). With the single-elimination tournament, different individuals will participate in a different number of interactions. An approach introduced by Jaśkowski et al. (2008) uses repeated tournaments not to compute fitness for individuals, but to directly select individuals (the winner from each tournament) for breeding.

Information available from previous evaluation rounds may be used to influence how interactions are chosen for the current evaluation round. We call this *fitness bias*. Usually, individual-centric approaches use such biases, while population-centric ones do not. Clearly, to be able to perform such biasing, the algorithm must store some information about previous evaluations. This information usually consists of individuals and their fitness, but can also be more complex in nature (e.g., include other properties of individuals or relationships between individuals). The amount of information can range from remembering the last best individual to remembering all interaction assessments ever performed. With generational CoEAs, intermediary approaches include saving one or a few (usually the best) individuals from each previous generation, saving all individuals in the previous generation, or other memory or archive method (Panait et al. 2006a; De Jong 2004a, c; Ficici and Pollack 2003; Oliehoek et al. 2006).

Additional reviews and comparisons of various methods for selecting interactions can be found in the literature (Angeline and Pollack 1993; Panait and Luke 2002; Sims 1994).

With individual-centric methods, some of the additional choices (in particular the sample size) may be dynamic (i.e., they vary at run-time as a function of time) (Panait and Luke 2005; Panait et al. 2006c) or adaptive (i.e., they vary at run-time as a function of the internal state of the algorithm, e.g., population diversity and operator success) (Panait and Luke 2006; Panait et al. 2006a; Panait 2006).

Additional choices must be made when using spatially embedded CoEAs. These tend to use individual-centric approaches and interactions are localized in space, namely, neighborhood based. The size of the neighborhood corresponds to the sample size, but the shape is an additional decision to be made. Fitness-biased selection of interactions can still be used (generally within the neighborhood).

3.2.2 Aggregation

If an individual participates in a single interaction, then it must obtain a value from it, and that value becomes the individual's fitness. But when an individual participates in multiple interactions and thus obtains multiple values, then a choice must be made about how to aggregate these values.

One approach is to input values from multiple interactions into some computations and output a *single value* per individual (to be used as fitness). The computations can simply use all values obtained by the individual and determine the *best*, the *worst*, or the *average* of those values. A comparison of these three methods can be found in (Wiegand et al. 2001). Other, more complex computations can take into account values from other individuals as well, as is the case for competitive fitness sharing (Rosin and Belew 1995, 1997) or the biasing technique introduced by Panait et al. (2003, 2004). The advantage of the single-value approach is that traditional parent-selection methods can then be used by the algorithm based on single-valued fitness. Unfortunately, different ways of computing a single value have different (and strong) biases, and it is not always straightforward to tell which bias is more or less helpful for the solution concept at hand. In particular, when dealing with the ideal team solution concept, the biases of the averaging method proved harmful (Wiegand 2004) while choosing the best is helpful (Panait 2006). Even so, there has been some amount of controversy over the utility of biases (Bucci and Pollack 2005; Panait and Luke 2006); Bucci (2007) goes so far as to argue that single-valued fitness assessments are to be blamed for a number of pathological algorithm dynamics.

The alternative is to have fitness be a *tuple* of values, usually the values obtained by the individual from multiple interactions. Selecting parents based on such fitnesses requires more specialized methods, in particular, ones akin to multi-objective EAs. The tuples of different individuals of the same role must be somehow comparable, which imposes constraints on the interactions that generated the values in the tuples. The advantage of this approach is that its bias may be more appropriate for solution concepts such as the Pareto optimal set or simultaneous maximization of all outcomes, and it is mostly algorithms targeted at these solution concepts that use tuple fitness (De Jong 2004a, c; Ficici and Pollack 2001). An example using tuple fitness for the ideal team solution concept can be found in (Bucci and Pollack 2005).

3.2.3 Communication

When using a CoEA with multiple populations, in order for individuals in one population to interact with individuals from other populations, the populations must have access to one

another's contents. This is an issue of communication, and thus entails choices typical of any distributed system, such as coordination, flow, and frequency.

In terms of *coordination*, communication can be synchronous or asynchronous. In the *asynchronous* case, the populations evolve at their own pace and communicate with one another through shared memory. They decide independently when to write new information about their state to the shared memory and when to check the memory for new information about the other populations (which may or may not be available). Asynchronous CoEAs are uncommon. In the *synchronous* case, there is a centralized clock that dictates when the populations exchange information.

In terms of *flow* (in Wiegand's (2004) hierarchy of CoEA properties, flow is called *update timing*) the asynchronous model is always *parallel*, in the sense that at any point in time, there may be more than one population running. (When a parallel CoEA is run on a single processor, this translates into the fact that there is no guarantee about the order in which the populations run.) The synchronous model can be either parallel or sequential (also called *serial*). In the *parallel* case, all populations run simultaneously for a certain period of time (dictated by the central clock), after which they all pause and exchange (communicate) information and then they all continue. In the *sequential* case, at any point in time, there is a single population running and populations take turns in a round-robin fashion.

Frequency refers to the number of evaluation–selection–breeding cycles that a population goes through between two communication events. The frequency may be uniform across all populations or it may differ from one population to another.

4 Analysis of Coevolution

Particularly in the last decade, there has been a great deal of activity in analyzing coevolutionary algorithms and their use as co-optimizers. There are two major drivers of this field. One is the fact that most co-optimization problems pose difficulties not encountered in traditional optimization, namely, measuring (and achieving) algorithmic progress and performance. The other is that coevolutionary algorithms as dynamical systems exhibit behaviors not akin to traditional evolutionary algorithms.

The first half of this section clarifies the complex issues pertaining to interactive domains and co-optimization problems and places much extant research in the context of addressing such issues. The second half surveys the wide variety of analytical methods applied to understanding the search and optimization biases of coevolutionary algorithms, as well as the insights that were gained.

4.1 Challenges of Co-optimization: Progress and Performance

For traditional optimization, judging the success of any algorithm attempting to solve a given problem instance is a fairly straightforward matter. There may be different criteria of success, such as the highest quality potential solution(s) found with a given budget of function evaluations or, conversely, what evaluation budget is needed to find potential solutions whose quality is above a certain threshold. Such success metrics also allow for easy comparisons of algorithm performance, even when aggregated over classes of problems.

Additionally, even when concerned with the behavior of a single algorithm on a single problem instance, a straightforward notion of progress through time exists. With each new function evaluation, the algorithm can easily keep track of the best quality potential solution found so far. Even if the quality of the overall best is not known, the algorithm can detect when it gets closer to it. Moreover, should an algorithm actually discover a global optimum, whether or not it is recognized, there is no danger of “losing” it. The best-so-far quality metric is a monotonic function of time. (The potential solution with the optimal quality may change, should the algorithm be deliberately implemented to replace the current best-so-far potential solution with a new one of equal quality.)

All of this is largely possible because the quality of a potential solution can be obtained with a single function evaluation and thus immediately perceived by the algorithm, which in turn allows reliable comparison of any two potential solutions. As trivial as the above statements may appear, they are no longer a given in many co-optimization problems.

Consider first the one area that is most similar to traditional optimization: compositional problems using the ideal team solution concept. The unique feature of these problems is that the interaction set is actually the same as the potential solution set *and* the interaction function serves as the potential-solution quality function. Thus, the quality of a potential solution can be obtained with a single function evaluation and any two potential solutions can reliably be compared. This, of course, is not very surprising since many of these problems are actually obtained via decomposition from traditional optimization problems. Therefore, measuring performance and progress is not an issue.

Nevertheless, approaching these problems via coevolution raises new issues. For example, how does one assign credit to the components of potential solutions that are evolving in different populations? More generally, how do different algorithm design choices affect algorithm performance? Much research around this topic has shown that, even for these most traditional-optimization-similar problems, design heuristics for traditional evolutionary algorithms do not transfer as-is to coevolution (Popovici and De Jong 2004). Still, the ability to easily measure performance and progress greatly facilitates the study of coevolutionary algorithms for ideal team compositional problems.

Note that the equality of the potential solution set and the interaction set is not enough to guarantee ease of performance/progress assessment. Consider compositional problems using the pure Nash equilibria solution concept. This problem is not expressed as a co-optimization problem, but as a generic co-search problem. It is fairly obvious that the single evaluation of the interaction function for this potential solution is not enough to determine whether it constitutes a Nash equilibrium. To determine that, one would need to, in turn, fix all but one component of the potential solution and evaluate interactions corresponding to, in the worst case, the entire set of entities for that component. While this does not mean exploring the entire interaction set, it does mean fully exploring all the component sets, which is to be assumed intractable. A negative decision might be reached sooner than that, but to reach a positive decision exhaustive search must be performed.

For test-based problems, the potential solution set is always different from the interaction set and determining the quality of a potential solution requires evaluating interactions corresponding to entire entity sets (the test sets). Should this be tractable, the problem may be expressed as a traditional optimization problem. We assume that is not the case.

Consider a generic search algorithm that, at every step, evaluates a new interaction and outputs a potential solution. (Algorithms that may not be ready to output a potential solution after every evaluation can be interpreted as outputting their latest output again until ready.)

How can one judge whether the algorithm makes progress toward the solution to the entire problem? How can its output be compared with the output of a different algorithm that has evaluated the same number of interactions? Since determining the true quality of a potential solution is generally computationally intractable for real problems, one is left with attempting to compare different potential solutions based on incomplete information.

The above observations paint a rather bleak view with respect to determining whether an algorithm could tackle one of these problems. However, the matter is not hopeless, though more can be said and done for some problems than for others.

For example, for Pareto dominance solution concepts, it has been shown that for any given set of n component entities (components), an *ideal evaluation set* of at most $n^2 - n$ test entities (tests) exists that results in exactly the same dominance relations among the components as would result from using the set of all possible tests (De Jong and Pollack 2004). Of course, the guaranteed existence of this limited set of tests leaves open the question of how to identify them. However, an operational criterion is available to determine whether any given test should be included in such a limited test set. Specifically, the criterion of whether a test induces a *distinction* between two components, introduced by Ficici and Pollack (2001), can be used for this purpose; see (De Jong and Pollack 2004).

The vast majority of research on these issues has concentrated on progress of (co-optimization) algorithms in individual runs and assessing performance on test problems. Only recently have aggregate performance comparisons between algorithms started receiving attention.

With respect to progress in individual runs, one fact facilitates the analysis: the incomplete information that an algorithm sees, and based on which one should compare different potential solutions outputted by the algorithm, is increasing over time. The remarkable discovery of Ficici (2004) is that some problems (solution concepts) permit the construction of algorithms that can properly leverage this incomplete but increasing information to guarantee that outputted potential solutions have monotonically increasing *global/true* quality over time. Any such algorithm is said to guarantee *global monotonicity* and the said solution concepts are called *globally monotonic*.

Some problems of practical interest have solution concepts that are not globally monotonic. This makes it difficult or even impossible to devise algorithms that guarantee global monotonic progress as described above. Still, for some of these problems, algorithms can be built that guarantee a different form of monotonicity called *local monotonicity*, which entails that the quality of outputted potential solutions with respect to all information seen so far will increase with time.

Both globally and locally monotonic algorithms guarantee convergence to a globally optimal solution, if the interaction space is finite and any point in this space can be generated with nonzero probability at any time.

When neither global nor local monotonicity guarantees can be achieved, some surrogate techniques can be used to instrument what the algorithm is doing, though they do not inform about progress toward the problem-solving goal. Finally, while all of the above assume the intractability of computing global quality, one research approach is to study various co-optimization algorithms on test problems, for which global quality can actually be computed. This metric is not made available to the algorithm, but used externally for instrumentation, and the hope is that the observed results will transfer to real problems.

The remainder of this section reviews global and local monotonicity, surrogate measures of progress, test-bed analytical techniques and aggregate performance comparisons.

4.1.1 Monotonicity

Central to the work on monotonicity is the idea that the solution concept can be interpreted as more general than just a means of partitioning the entire set of potential solutions based on the entire set of interactions. Namely, it can be extended to apply to any context consisting of a subset of interactions and a corresponding subset of potential solutions (i.e., potential solutions that can be built with entities that participated in the given interactions). Thus, it can be viewed as a means for an algorithm to output a potential solution when queried, based on all or some of the entities it has generated and interactions it has assessed. (Conversely, any mechanism for selecting a potential solution to be outputted from the algorithm can be viewed as a solution concept.)

In general, it may be difficult for an algorithm to determine the output of the solution concept when applied to the entire history of entities discovered and interactions assessed if it has discarded some of the entities or some of the interaction outcomes. However, for some solution concepts this is possible. For example, for the best worst case solution concept, an algorithm needs to remember only the worst value seen for any potential solution and not all the interactions it took part in, the other entities involved in those interactions or their outcomes.

One question that arises is whether an algorithm able to report the application of the solution concept to its entire history (whether by means of never discarding any information or by cleverly storing parts or aggregates of this information) can guarantee that the *global* quality of the potential solutions it outputs will never decrease over time, regardless of the means of generating new interactions to be assessed. Ficici (2004) showed that this is possible for some solution concepts, which we call *globally monotonic*, but not for others. Out of the solution concepts presented in [Sect. 2.4](#), the ones proven to be globally monotonic are the Nash equilibrium, the Pareto optimal minimal equivalence set (Ficici 2004), maximization of all outcomes, and ideal team (Service 2009). Solution concepts proven not to be globally monotonic are maximization of expected utility (Ficici 2004), best worst case (Service 2009; Popovici and De Jong 2009), and all Pareto dominance ones besides the minimal equivalence set (Ficici 2004).

For the globally monotonic solution concepts, the challenge from an algorithmic perspective is being able to report the application of the solution concept to the entire history while limiting the amount of history that must be stored. The algorithms mentioned below are in many cases able to discard information without jeopardizing the guarantee of global monotonicity. Yet, all of the globally monotonic algorithms discussed below require unlimited memory.

For the Pareto optimal minimal equivalence set solution concept, De Jong (2004a) introduced an algorithm called *IPCA* (Incremental Pareto-Coevolution Archive) that guarantees global monotonicity (Popovici and De Jong 2009) while discarding information along the way. The key to achieving that lies in a carefully designed archiving mechanism. A variant named the LAyered Pareto Coevolution Archive (LAPCA) (De Jong 2004c) addresses the unbounded memory issue by maintaining a limited number of non-dominated layers. While it no longer provides the monotonicity guarantee, this algorithm achieves good progress on test problems. Similarly, the *Nash memory* algorithm (Ficici and Pollack 2003; Ficici 2004), guarantees global monotonic progress for the solution concept of the Nash equilibrium in symmetric zero-sum domains while discarding information. As before, bounding the memory loses the theoretical monotonicity guarantee but still achieves good progress on test problems. Based on the *Nash memory*, the *parallel Nash memory* algorithm (Oliehoek et al. 2006)

guarantees global monotonic progress for the solution concept of the Nash equilibrium in asymmetric games. For the maximization of all outcomes solution concept, the *covering competitive algorithm* of Rosin (1997) guarantees global monotonic progress while discarding information.

While the guarantee of increasing global quality may not be possible for solution concepts that are not globally monotonic, this does not mean that all hope is lost for problems with such solution concepts. For such problems it may still be possible to devise algorithms that guarantee that the quality of outputted potential solutions *with respect to the interactions seen so far* increases monotonically over time. Solution concepts with this property are called *locally monotonic*.

For example, while the maximization of expected utility (MEU) solution concept introduced in [Sect. 2.4.2](#) is not globally monotonic, a variant of it, namely, maximum utility sum applied to domains with a positive-valued metric, allows algorithms guaranteeing locally monotonic progress (Popovici and De Jong 2009). As the name suggests, the MEU solution concept is an optimization criterion, where the function to be optimized is utility sum. When extended to any context consisting of a subset of interactions and a corresponding subset of potential solutions, the special property this function has (if the metric values are positive) is that the “local” value for a potential solution in some context can only increase when the context is expanded.

Based on this property, for maximum utility sum applied to domains with a binary-valued metric, De Jong (2005) introduced an algorithm called *MaxSolve* that is guaranteed to output the application of the solution concept to the entire history (and therefore an increasing value with respect to an increasing context), while discarding information along the way. This property, coupled with unbounded memory, allows *MaxSolve* to achieve local monotonicity.

As Popovici and De Jong (2009) showed, for the best worst case solution concept, even the local monotonicity guarantee cannot be fully achieved via algorithms that use the solution concept as an output mechanism (and probably not in general).

While monotonicity is an interesting property that has served in clarifying the behavior and goals of coevolutionary solution concepts and algorithms, it should be made clear that guarantees of monotonicity can have limited value for a practitioner. Three important things to keep in mind are that (a) the solution concept is most often a given, not a choice, and for some solution concepts there are no known algorithms with monotonicity guarantees; (b) all algorithms known to have monotonicity guarantees require unbounded memory; and (c) even if the quality of potential solutions can only improve over time, this provides no information about the *rate* at which progress will be achieved. For these reasons, an approach that may turn out to have more practical relevance is to consider the *performance* of coevolutionary algorithms. This topic is treated in [Sect. 4.1.4](#).

4.1.2 Surrogate Progress Metrics

For problems where the above monotonicity guarantees are unobtainable, two questions remain. One is how to design algorithms for tackling these problems. Another is how to instrument any such algorithm. The former is still an open research question. Here, the latter is addressed from the perspective of coevolutionary algorithms. However, while these techniques have been introduced in the context of coevolution, some of them may be adapted to be applicable to generic co-optimization algorithms.

Many techniques have been introduced over the years, and while they cannot tell how the algorithm is doing with respect to the goal, they may still provide useful information about algorithm behavior. The techniques involve computing metrics that are external to the algorithm (i.e., they do not influence its behavior) but rely on its history. They may measure all potential solutions considered by the algorithm or only select those that the algorithm outputs. Measuring all can provide more information, but it may make it difficult to extract the important patterns within and may also pose a challenge for visualization. One must be aware that compressing information (e.g., by means of averaging) may actually eliminate important trends in the data. The trends to look for usually include increase, decrease, stagnation, noise, and repeated values.

The first such technique was introduced by Cliff and Miller (1995) in the form of CIAO plots. The acronym stands for “current individual ancestral opponent.” For a two-population CoEA, a CIAO plot is a matrix in which rows represent generations of one population and columns represent generations of the other population. Every cell represents an interaction between the best individual in each population (as reported by the algorithm) at the corresponding generations. Thus, individuals from later generations of one population interact with individuals from early generations of the other population (ancestors). The cells are color-coded on a gray scale, and can be constructed to reflect success from the perspective of either population.

The “master tournament” metric of Floreano and Nolfi (1997) is basically a compression of the information in CIAO plots. Averages are taken along lines for one population and across columns for the other population.

Both these methods are computationally expensive, as they require the evaluation of n^2 interactions, where n is the number of generations. The master tournament metric makes it easier to identify “broadly successful” individuals, but the averaging it performs may obscure some circularities that can be observed using the CIAO plots. An in-depth critique of CIAO plots is provided in (Cartlidge and Bullock 2004b).

While these metrics were introduced in the context of domains with two asymmetric roles, they are easily applicable to domains with two symmetric roles, whether approached by single-population or two-population CoEAs.

In the context of a domain with two symmetric roles and a two-population CoEA, Stanley and Miikkulainen (2002b) introduced a less costly technique called “dominance tournament.” At each generation, the best individual in each population is determined and the two of them are paired; out of their interaction, the more successful one is designated the generation champion. The dominance property is then defined as follows. The champion from the first generation is automatically considered dominant. In every subsequent generation, the champion is paired only with previous dominant champions and it is itself labeled dominant only if it is more successful than all of them. The method is easily applicable to domains with two symmetric roles and a single-population CoEA. The paper also suggests a (less straightforward) extension to some domains with two asymmetric roles.

CIAO, master tournament, and dominance tournament are all techniques that track only the best of generation individual. This was contrasted by Bader-Natal and Pollack (2004) who introduced a “population-differential” technique monitoring all individuals in each generation. Plots similar to the CIAO plots are produced, but now, each cell is an aggregation over the results of all pair-wise interactions between individuals in the two populations at the generations corresponding to that cell. This is clearly more expensive, and a number of memory policies are introduced for reducing time complexity.

The unifying idea of these techniques is to determine whether an algorithm is making at least some sort of “local” progress away from the starting point.

All the metrics described so far require performing additional evaluations, rather than using the ones already performed by the algorithm. Funes and Pollack (2000) introduced a technique that took the latter approach. Their metric, called “relative strength” is still subjective, as the value it returns for a particular individual depends on the current history of all interactions, and this history grows with time. The metric is based on paired comparison statistics and is applicable to domains with two symmetric roles.

4.1.3 Test-Bed Analysis

A means of judging both an algorithm’s ability to make progress as well as its expected performance on some problem(s) is to test the algorithm on artificial domains for which true quality can be computed. The domains are constructed such that they exhibit properties thought to be present in real domains or such that they expose a range of different behaviors of the algorithm. The hope is that the results observed on the artificial domains will transfer to real domains.

A class of domains introduced to this effect are *number games* (Watson and Pollack 2001). These are domains with two roles, a common entity set consisting of real-valued vectors and a binary-valued interaction function (based on comparisons between vector values). The roles can be interpreted as symmetric or asymmetric. The solution concept is Pareto dominance. What makes these problems easier to analyze is the fact that the interaction function is designed such that a genotype can be used as-is to represent its own true quality. There are a handful of such interaction functions in the literature and many progress-focused algorithm comparisons have been performed using them as a test bed (De Jong 2004a, c, 2005, 2007; Bucci and Pollack 2003a; Funes and Pujals 2005).

Popovici and De Jong (2005b) introduced a set of four domains with two asymmetric roles, a common entity set consisting of single real values and a real-valued interaction function given in closed form. These domains were used to showcase the biases of symmetric last elite opponent evaluation, and how this technique generally does not optimize expected utility. The closed form of the interaction function allowed for easy computation of true quality.

A number of domains of the same kind (two asymmetric roles, real-valued entities, closed-form real-valued interaction function) have been introduced specifically for studying algorithms targeted at the best worse case solution concept (Jensen 2001; Branke and Rosenbusch 2008; Stuermer et al. 2009), which has been shown to be non-monotonic and difficult in terms of achieving even very weak forms of progress (Popovici and De Jong 2009).

Test-bed analysis has also been performed for algorithms targeted at the ideal team solution concept, yet with a different motivation (e.g., studying problem properties and algorithm design choices) since in this case assessing global quality is not an issue. Examples are given in  Sect. 4.2.4.

4.1.4 Comparing Performance

As emphasized throughout the previous section, instrumenting an algorithm from a progress-making perspective or, indeed, designing an algorithm such that it guarantees progress relies

rather heavily on the fact that the amount of information seen by the algorithm increases over time. How an algorithm uses this information can be key to instrumenting/achieving progress. Therefore, the theoretical work on monotonicity has so far had nothing to say about how to compare algorithms since there is no connection between the information that different algorithms see. This line of research provided no generic advice on how to design efficient algorithms.

While test-bed analysis is useful for determining whether an algorithm makes progress or not, when global quality can be computed, one can also use test-beds to compare algorithms based on the expected rate of progress or expected level of solution quality over multiple runs. This kind of performance analysis, especially when backed by dynamics analysis of the kind reviewed in [Sect. 4.2.5](#), has provided insight into the biases of design choices by providing a means of comparison; however, it is unclear how well the results transfer to real-world problems, so the generality of such results is unclear.

From traditional optimization research, the famous no free lunch (NFL) theorem has approached the issue of performance generality (Wolpert and Macready [1997](#); Schumacher et al. [2001](#)). This result has led to a welcome shift in EC research toward identifying classes of problems where NFL does not hold and designing algorithms targeted at a specific class (Igel and Toussaint [2003](#)). This is of particular current interest in co-optimization/co-evolution (Wolpert and Macready [2005](#)) since recent work states the opposite, namely, that free lunches do exist for certain solution concepts.

Wolpert and Macready extended their formalism of traditional optimization algorithms to co-optimization algorithms (albeit without using this terminology) as having two components: a search heuristic and an output selection function. The search heuristic takes as input the sequence of interactions assessed so far and returns a new interaction to be assessed. The output selection function takes the same input and returns a potential solution. The need to explicitly model the output selection function comes from the algorithms' lack of access to a global quality function: in traditional optimization, output selection is based on the quality values seen so far (usually picking the best of them); in co-optimization, what an algorithm sees are values of the interaction function, which provide incomplete information about global quality. The use of archives in CoEAs is often targeted at implementing the output selection function, thus relieving the populations of this duty so they can focus on the search component.

Wolpert and Macready showed that aggregate performance advantages (free lunches) can be obtained both via the output selection function and via the search heuristic. The former means that if the search heuristic is fixed, some output functions perform better than others. The optimal output function is the so-called Bayes output function, which outputs the potential solution with the highest estimated global quality, estimated over all possible problems consistent with the measurements of the interactions seen so far. The latter was showcased in the context of the best worst case solution concept: fixing the output selection function to Bayes, two specific search heuristics were shown to have different average performance.

This line of work was extended by Service and Tauritz ([2008b](#)), which showed that the same two algorithms from Wolpert and Macready ([2005](#)) exhibit different aggregate performance for the maximization of all outcomes solution concept.

Generally, such free lunches are believed to exist in co-optimization whenever the performance of an algorithm depends on information not available in the algorithm's trace of interactions; in other words, it explicitly depends on the interaction function (as opposed to

implicitly, via the values in the trace). Therefore, the ideal team solution concept does not exhibit free lunches since the interaction set is actually the same as the potential solution set and the interaction function serves as the potential solution quality function.

The existence or lack of free lunches can be seen as a property of solution concepts, which has been named “bias” (Service 2009). Since monotonicity is also a solution concept property, the natural question that arises is how these two properties relate to one another. Service (2009) showed they are orthogonal to one another: solution concepts exist in all four classes combining monotonic/non-monotonic with biased/unbiased.

Especially for monotonic and biased solution concepts, the next natural question is whether the algorithms that can achieve monotonicity (known to exist) are also the best performing. This question was investigated by Popovici and De Jong (2009), who showed that the output selection function used by monotonic algorithms is not equivalent to the best-performing Bayes, thus exposing a potential trade-off in algorithm design between monotonicity and performance.

4.2 Analysis of CoEA Biases: A Survey

The previous half of this section discussed the difficulties of measuring and achieving algorithmic progress for co-optimization problems, and reviewed the state of the art in understanding and dealing with these issues. However, when applying coevolutionary algorithms to such problems, an additional concern is understanding and harnessing the biases these algorithms have as dynamical systems. This section surveys the analytical research that contributed such understanding.

4.2.1 Model Analysis

A common algorithm analysis method is to *model* an algorithm mathematically, then to study the properties of that model. The advantage is that certain mathematical facts can sometimes be precisely learned from these models, facts that help to understand system behavior in ways that are simply not possible via traditional approaches. The disadvantage is that the model is *not* a real algorithm, and it is not always clear what can be transferred from model to algorithm. This step is typically bridged by empirical study of some kind.

The simplest way to approach modeling evolutionary computation as a dynamical system is the so-called “infinite population model” (Vose 1999). One assumes that a population is infinitely large and focuses on how the distribution of genotypes within the population changes over time. Questions about the existence (and sometimes the location) of stable, attracting fixed points, or their nature and quality, can sometimes be answered by applying traditional dynamical systems methods to such a model.

Evolutionary game theory (EGT) (Hofbauer and Sigmund 1998; Weibull 1992; Maynard Smith 1982) is an appealing dynamical systems model of biological systems that is well-suited for studying coevolution. In EGT, interactions between genotypes are treated as a *stage game*, and the payoff from that game informs how the distribution is altered for the next play of the game (*replicator dynamics*). Because of its incorporation of game theory, a number of game-theoretic properties of such systems become useful points of consideration (e.g., Nash

equilibria). Additionally, EGT has received a great deal of attention by the economics community (Friedman 1998). Consequently, quite a bit is already known about the dynamical properties of these mathematical models under different conditions.

For the most part, EGT analysis has focused on the dynamics of CoEAs under selection only; however, variational operators in CoEAs can be (and have been) modeled as is typically done in dynamical systems models of traditional EAs, by constructing a matrix that transforms the probability distribution resulting from the selection operator into one that reflects the properties of the population after crossover and mutation are applied, the so-called *mixing matrix* (Vose 1999). To specify this, one must be explicit about representation. Despite these simplifications, some interesting things are known about the system that inform our understanding of CoEAs.

In single population EGT models of CoEAs, for example, stable attracting *polymorphic* equilibria can exist when the underlying stage game is not a constant sum game. That is, aside from the stochastic effects of genetic drift, selection alone can drive the systems to states where the population is a mixture of different kinds of genotypes, and these stable, mixed strategies correspond with a Nash equilibrium of the stage game. Moreover, examining this system using other selection methods has uncovered some fascinating results. Common EC selection methods (e.g., truncation, (μ, λ) , linear rank, Boltzmann) can sometimes lead to systems in which there are stable cycles and even chaotic orbits (Ficici and Pollack 2000b). Indeed, the dynamics of CoEAs can be much more complex than traditional EAs in that stable attractors of infinite population models can be quite unrelated to the specific values in the game payoff matrix itself. Empirical experiments on real algorithms verify that these fundamental dynamical pressures of the selection operator do, in fact, affect how real CoEAs perform on real problems.

Additionally, by considering certain game-theoretic properties of the stage game (a particular variation of the notion of constant-sum), one can show that the behavior of certain single population CoEAs using nonparametric selection operators will, in principle, behave dynamically like a comparable EA on some unknown fitness function (Luke and Wiegand 2003), while this cannot be said for other systems *even when the payoff is fully transitive*. This reinforces the idea that certain co-optimization problems are fundamentally different from traditional optimization problems.

EGT-based models have also been used to study two-population compositional CoEAs, showing that any pure Nash equilibrium of the underlying payoff game is a stable, attracting fixed point. This suggests selection alone can drive compositional CoEAs toward points that are globally very suboptimal in terms of their actual payoff value. Experimentation confirms this is so in real, analogous compositional CoEAs even to the extent that highly inferior local optima draw significantly more populations than do the true global optima. Indeed, complete mixing implies a solution concept wherein “optimality” is defined solely based on how individual strategies do on average over all possible partners. If one is using such methods to try to find a globally maximal payoff value, this can be seen as a kind of pathology (*relative generalization*). This dynamic may well be consistent with obtaining solutions that meet certain specific definitions of *robustness* (Wiegand and Potter 2006).

But most compositional CoEAs use best-of-generation interaction methods, not complete mixing, and Panait et al. (2006b) noted that these more common methods bias compositional coevolutionary search of projections of the payoff space to favor aggregate projections that

suit identifying the global optimum. Further, by incorporating archival notions that help retain *informative* partners in terms of these projection estimates, the optimization potential of compositional CoEAs can be further improved. These analytical lessons have been applied to both real compositional CoEAs, as well as other RL-based co-optimization processes (Panait 2006). Additionally, Vo et al. (2009) showed a certain infinite population EGT model of two-population compositional CoEAs to be equivalent to the distribution model for univariate estimation-of-distribution algorithms, thus arguing for the potential of cross-fertilization between these two subdisciplines of evolutionary computation. This work also provides a generalization of a subset of compositional CoEA models to n -populations.

These studies have shown that interaction and aggregation have profound impacts on the dynamical influences of selection in compositional coevolution. More specifically, the complete mixing interaction pattern may be ill-suited for problems with an ideal team solution concept. Another, altogether different dynamical systems approach by Subbu and Sanderson (2004) provides convergence results for a particular class of distributed compositional CoEAs to ideal team style solutions under conditions where complete mixing is not even possible.

In addition to the infinite population models described above, several works have focused on finite populations. EGT itself has been applied to finite population situations (Maynard Smith 1982; Ficici and Pollack 2000a), but typically under very constrained circumstances (two-strategy games). Outside of using EGT-based methods, there have been attempts to model the dynamics of finite, two-population CoEAs using Markov models (Liekens et al. 2003). While these methods also restrict the size of the problem spaces for obvious reasons, they have important implications for infinite population models in the sense that they clearly demonstrate that the long-term behavior of these systems can differ quite significantly from infinite population models.

4.2.2 Runtime Analysis

EC has made good use of classical algorithm analysis methods to find bounds on the expected number of evaluations necessary before a global optimum has been obtained as a function of the size of the (typically, but not always, combinatorial) genotype space (Oliveto et al. 2007). These methods have the advantage of providing precise and correct performance bounds on real algorithms for real problems, but their chief disadvantage is that it is often difficult to gain crucial insight into the behavior of an algorithm. This concern is typically addressed by selecting problem classes that help identify important properties in the problem and falsify hypotheses regarding how EAs address them.

Indeed, runtime analysis has been quite useful in understanding how simple compositional coevolutionary algorithms approach problems with ideal team solution concepts. A common hypothesis about such algorithms is that they tend to perform better when the representation choices result in a linear separation of problem components with respect to the objective function. Runtime analysis of simple (1 + 1) (Jansen and Wiegand 2003a, b) and steady-state-like (Jansen and Wiegand 2004) variants of compositional CoEAs revealed this hypothesis to be conclusively false: though nonlinear relationships between components can have profound effects on the relative performance differences between a coevolutionary algorithm and a similarly structured EA, separability itself is neither a sufficient nor

necessary property with which to predict problem difficulty (Jansen and Wiegand 2004). Additionally, there exist certain problem classes that essentially *cannot be solved* by a large class of compositional coevolutionary algorithms. Finally, this work suggests that compositional CoEAs optimize by leveraging the decomposition to partition the problem *while* increasing the focus of the explorative effects of the variational operators (so-called *partitioning and focusing* effect).

Runtime analysis of other CoEAs is more challenging and has not yet been done. Fortunately, recent research provides several needed pieces to overcome these challenges. First, there has been a great deal of recent progress in analyzing increasingly more complex multi-objective methods (Laumanns et al. 2004), and much of this may help study CoEAs that employ multi-objective mechanisms (e.g., archives). Second, the formalisms for explicit definition of solution concepts and the guarantee of monotonicity in certain concepts provide clear global goals needed for runtime analysis.

4.2.3 Probabilistic/Convergence Analysis

Another way to investigate algorithms theoretically, is to consider the dynamical behaviors of the real algorithms (as opposed to abstractly modeling them). Typically, such analyses are focused on providing convergence guarantees, but often careful consideration of problem and algorithm properties can provide greater insight.

Schmitt (2003a, b) provides a strong type of convergence analysis. For the maximization of all outcomes solution concept, the proposed algorithm is guaranteed not only to find a solution, but to have its populations converge to containing only genotypes corresponding to the solution(s). Markov chain analysis is used for the proofs. Bounds on convergence speed are not given; however, the work provides some useful general advice for how to apply rates of genetic operators to ensure such convergence.

Additionally, Funes and Pujals (2005) conduct a probability-theory-based investigation into the relationships between problem properties, algorithm properties, and system dynamics. This work focuses on test-based CoEAs, and demonstrates that the interplay of the considered properties cause trajectories through the domain-role sets to either make progress or resemble random-walk behaviors.

4.2.4 Empirical Black-Box Analysis

Analysis of coevolution can be approached empirically by viewing the system as a black box whose inputs are the algorithm and the problem, and the output is observed performance (e.g., quality of potential solution outputted given a certain amount of time). The algorithm and/or the problem are then varied and the output of the system re-observed, with the goal of determining the rules governing the dependency between inputs and outputs. Such studies are performed for problems with computable global quality, such as ideal team problems or test-beds of the kind reviewed in  Sect. 4.1.3.

Two different approaches can be distinguished within the black-box analysis category. One approach focuses on *properties* of the algorithms, of the problems, or of both. When algorithm properties are involved, this approach has been referred to as component analysis

(Wiegand 2004). The other approach varies the algorithm and/or the problem and compares the resulting performance without a clear isolation of the properties responsible for the differences in performance.

This latter approach is reviewed first. Most works introducing new algorithms are of this kind, including some of the early research comparing CoEAs with traditional EAs, both for test-based problems (Hillis 1990; Angeline and Pollack 1993) and for compositional problems (Potter and De Jong 1994). Later on, the approach was used to compare CoEAs among themselves, usually “enhanced” algorithms with basic ones. Examples include works for test-based problems (Ficici and Pollack 2001, 2003; Ficici 2004; De Jong and Pollack 2004; De Jong 2004a, c, 2005) and compositional problems (Bucci and Pollack 2005; Panait et al. 2006a; Panait and Luke 2006; Panait 2006). Some of the algorithms introduced by these works are backed up by theoretical results (De Jong 2004a, 2005; De Jong and Pollack 2004).

Property-focused approaches address problem properties, algorithm properties, or the interactions between them. These cases are discussed in order.

Some of the CoEA-versus-EA comparisons were extended to determine the classes of problems for which one type of algorithm would provide performance advantages over the other. Such work was performed for compositional problems by Potter (1997) and Wiegand and Potter (2006). The scrutinized problem properties were *separability* (Wiegand 2004), *inter-agent epistasis* (Bull 1998), *dimensionality* (here dimensionality refers to the number of roles), *noise*, and *relative sizes of basins of attraction of local optima*.

For test-based problems, empirical black-box analysis was used to investigate problem properties such as *role asymmetry* (Olsson 2001), *intransitivity* (De Jong 2004b), and *dimensionality* (De Jong and Bucci 2006). (Here dimensionality refers to the number of underlying objectives implicitly defined by the test role(s).) These works introduced specialized algorithms intended to target the respective problem property.

Studies focusing only on algorithm properties investigated either the mechanism for *selecting interactions* (Panait et al. 2006c; Panait and Luke 2005; Parker and Blumenthal 2003; Blumenthal and Parker 2004) or the mechanism for *aggregating* the result of those interactions (Panait et al. 2003; Wiegand et al. 2001; Cartlidge and Bullock 2002). All but the last cited work were concerned with compositional problems.

Finally and perhaps most importantly, some black-box empirical studies analyzed the effects on performance of the interdependency between algorithm properties and problem properties. Most studied has been the mechanism for selecting interactions, especially in the context of compositional problems (Potter 1997; Wiegand et al. 2001, 2002; Wiegand 2004; Bull 1997, 2001). The launched hypothesis was that the performance effects of the interaction method are tightly related to the (previously mentioned) problem property separability and the kind of *cross-population epistasis* created when representation choices split up inseparable pieces of the problem. This dependency however proved not to be as straightforward as thought, as it was later shown in (Popovici and De Jong 2005a) via empirical dynamics analysis. For test-based problems, two different methods for selecting interactions were analyzed by Panait and Luke (2002), suggesting that the amount of noise in the problem affects their influence on performance.

For compositional problems, Panait et al. (2004) extended their previous work on the mechanism for aggregating results from multiple interactions by studying how its performance effects are affected by the problem’s local optima and their basins of attraction. Wiegand and Sarma (2004) studied the potential benefits of *spatially distributed schemes for selecting interactions and/or selecting parents* on domains with role asymmetry.

Bull (1998) studied the effects of mutation and crossover in the context of “mixed-payoff” domains Kauffman and Johnson’s (1991) NKC landscapes) and found them to be sensitive to inter-agent epistasis.

4.2.5 Empirical Dynamics Analysis

While black-box analysis can provide some heuristics for improving performance, it cannot explain the causes for the observed dependencies between the inputs and the outputs of a CoEC setup. To understand these causes, one needs to observe the system while running and track some of its time-varying properties. Instead, dynamics analysis is used to explain the connection between algorithm and problem properties on one side and performance on the other side. Individual studies connect different subsets of the pieces, analyzing:

- Dynamics in isolation (Cliff and Miller 1995; Stanley and Miikkulainen 2002b; Bader-Natal and Pollack 2004; Floreano and Nolfi 1997; Axelrod 1989; Cartlidge and Bullock 2004b);
- Problem properties and dynamics (Bull 2005a);
- Problem properties, algorithm properties, and dynamics (Bull 2005b; Kauffman and Johnson 1991);
- Dynamics and performance (Pagie and Mitchell 2002; Juillé and Pollack 1998; Ficici and Pollack 1998; Miller 1996; Funes and Pollack 2000; Paredis 1997);
- Problem properties, dynamics, and performance (Watson and Pollack 2001);
- Algorithm properties, dynamics, and performance (Williams and Mitchell 2005; Pagie and Hogeweg 2000; Rosin and Belew 1995, 1997; Cartlidge and Bullock 2003 2004a; Bucci and Pollack 2003a);
- How the combination of algorithm properties and problem properties influences dynamics and how dynamics influences performance (Popovici 2006).

These analyses used some of the progress metrics described in [Sect. 4.1.2](#), and/or various techniques for tracking genotypic/phenotypic changes. Studies involving the latter were performed for domains and representations that are easily amenable to visualization.

Cliff and Miller (1995) were also the first to use techniques for tracking genotypic changes in order to analyze the run-time behavior (dynamics) of CoEAs. They introduced “elite bitmaps,” “ancestral Hamming plots,” and “consensus distance plots” as tools complementary to CIAO plots. They all work on binary representations. Elite bitmaps simply display the genotype of best-of-generation individuals next to each other in temporal sequence. Some additional processing can reveal interesting patterns. Ancestral Hamming plots display the Hamming distance between elite individuals from different generations. Consensus distance plots monitor the genotypic make up of the whole population through the distribution of Hamming distances from the genotype of each individual to the population’s “consensus sequence.” All three techniques are applicable to EAs in general, not just CoEAs.

Popovici (2006) extensively used best-of-generation trajectory plots for understanding the effects of various CoEA design choices and exposed best-response curves as a problem property that is a strong driver of coevolutionary algorithm behavior.

It is via the empirical dynamics analysis approach that most knowledge about the pros and cons of using CoEAs for co-optimization was generated. And just as traditional canonical

evolutionary algorithms can be used for optimization, yet they are not intrinsically optimizers (De Jong 1992), canonical coevolutionary algorithms may be useful in co-optimization, yet they are not intrinsically co-optimizers.

5 The Future of Coevolution

The bulk of this chapter has focused on how coevolutionary algorithms, co-search problems, and interactive domains have been studied. This section is dedicated to the future of coevolution, including current lines of work, research questions, and results with intriguing but so-far-untapped potential. We end with some concluding remarks.

5.1 Open Issues

Recent research has helped establish a firm, formal foundation for coevolutionary problem solvers, and it has provided some principled ways to think about, design, and apply such systems. Nevertheless, many issues remain quite unresolved. These are grouped into two categories: those relating disparate theoretical analyses and those relating the analytical understanding of CoEAs to their practical application.

5.1.1 Relating Theories

Dynamical systems analysis of compositional coevolution has demonstrated that certain robustness properties in the problem can attract populations (Wiegand and Potter 2006). But there has yet to be any attempt to formalize this idea as a solution concept, and it remains to be seen whether such a concept is monotonic. Similarly, Lothar Schmitt's convergence analysis (Schmitt 2003b) provides useful feedback about how to manage the parameters of operators; however, it makes very specific assumptions about the solution concept, and no work has yet approached the task of generalizing his ideas to include CoEAs as applied to problems with different concepts of solution. Runtime analysis for compositional approaches has yielded precise bounds for those algorithms on certain optimization problems (Jansen and Wiegand 2004), and there has been a lot of success using similar methods to analyze the performance of EAs on multi-objective optimization problems. This suggests it should be possible to apply such analyses to CoEAs for test-based problems, due to their underlying multi-objective nature.

Though in a handbook of natural computing, this chapter has tried to present coevolutionary algorithms in the broader context of co-optimization. As pointed out in [Sect. 4.1](#), some of the difficulties CoEAs face, such as the difficulty to monitor and achieve progress, are specific to the nature of certain co-optimization problems and not the evolutionary aspect of these algorithms. Any other algorithms attempting to solve such problems will have to deal with these issues. Formalizing co-optimization algorithms as having two components, a search heuristic and an output selection function, allows for easy transfer of results pertaining to monotonicity and performance to algorithms using non-evolutionary search heuristics, such as the ones introduced in ([Service and Tauritz 2008a](#)).

Last but not least, many research questions on how algorithms should be designed have been opened by the recent confluence of the two generic co-optimization theories concerning monotonicity and performance (Service 2009; Popovici and De Jong 2009). This is expounded below.

5.1.2 Relating Theory to Practice

As discussed earlier in the chapter, the notion of monotonic solution concept not only provides a way to characterize problems but also allows engineers to construct principled algorithms for which consistent progress toward a solution can be expected (Ficici 2004). Still, on a practical level, several questions remain. Many algorithms that implement monotonic solution concepts have unrealistic memory requirements. Researchers are only beginning to try to understand what the impact of practical limits on mechanisms such as archive size have on performance and theoretical guarantees about progress. Similarly, hidden in the free lunch proofs of Wolpert and Macready (2005) was advice on how to obtain some performance advantage, such as avoiding unnecessary evaluations, exploiting biases introduced by the solution concept, and using a Bayes output selection function. Still, determining how to use this advice in real algorithms for real problems is very much an open research issue. Further, the design choices required by monotonicity can conflict with those required for performance (Popovici and De Jong 2009) and additional research is needed to determine when/if this trade-off can be avoided or at least how to minimize it. Indeed, recent work relating monotonicity, solution concepts, and the NFL suggest this relationship could be quite complex (Service 2009; Popovici and De Jong 2009).

The same types of questions can be raised with respect to methods used for the discovery of informative dimensions of co-optimization problems: Can heuristic methods be developed that can approximate this search process in the general case, and what are the precise trade-offs of such an approximation? Finally, under certain conditions, it is theoretically possible for some CoEAs to prefer components considered later in the run, which in turn will tend to require larger supporting test sets (Bucci 2007). This implies the possibility of a kind of bloating that practitioners may need to address in practical application.

5.2 Discovery of Search Space Structure

De Jong and Pollack (2004) present empirical results suggesting that a Pareto coevolutionary algorithm could find what were dubbed the *underlying objectives* of a problem. These are hypothetical objectives that determine the performance of components without the need to have them interact with all possible tests. De Jong and Pollack (2004) apply a two-population Pareto coevolutionary algorithm, DELPHI, to instances of a class of abstract interactive domains. Figures 13 and 15 of that work suggest that evaluator individuals (what we have called tests) evolve in a way that tracks the underlying objectives of the domain. The results suggest that the algorithm is sensitive to the presence of underlying objectives even though it is not given explicit information about those objectives, exhibiting what has been called an “emergent geometric organization” of test individuals (Bucci 2007). Bucci and Pollack (2003a) make a similar observation, also empirical though using a different algorithm; Fig. 5 of that work suggests a similar sensitivity to underlying objectives. In both cases, clusters

of individuals, rather than single individuals, move along or collect around the known objectives of the domain. The problems considered, namely, numbers games (Watson and Pollack 2001), were designed to have a known and controllable number of objectives, but the algorithms used in these two studies did not rely on that fact. The work therefore raises the question of whether underlying objectives exist in all domains, and whether algorithms can discover them.

A partial answer to this question is found in the notion of *coordinate system* defined in (Bucci et al. 2004). Coordinate systems, which were defined for a class of test-based problems (specifically, problems with a finite number of components and a finite number of binary-outcome tests), can be viewed as a formalization of the empirically observed underlying objectives of De Jong and Pollack (2004). To elaborate, a coordinate system consists of several *axes*. Each axis is a list of tests ordered in such a way that any component can be placed somewhere in the list. A component's placement is such that it does well against all tests before that spot and does poorly against all tests after it. For this reason, an axis can be viewed as measuring some aspect of a component's performance: A component that places high on an axis is “better than” one that sits lower in the sense that it does well against more tests (more of the tests present in the axis, not more tests of the domain as a whole). Formally, an axis corresponds to a numerical function over the components, in other words to an objective function. It can be proven (Bucci et al. 2004) that every domain of the considered class possesses at least one coordinate system, meaning it has a decomposition into a set of axes. In short, every such domain has some set of objective functions associated with it, one for each axis in a coordinate system for the problem.

Besides defining coordinate systems formally, Bucci et al. (2004) give an algorithm that finds a coordinate system for an interactive domain in polynomial time. The algorithm, though fast, is not guaranteed to produce the smallest-possible coordinate system for the domain, however. Finite domains must have a minimal coordinate system, but in general, even finite domains can have distinct coordinate systems of different sizes. The algorithm is not coevolutionary per se, as it examines the outcomes of tests on components. It is therefore applicable to the entire class of test-based problems and agnostic about the specific algorithm used to solve the problem.

The above work leaves open the question of whether underlying objectives or coordinate systems are simply mathematical curiosities with no practical or theoretical utility. The work of De Jong and Bucci (2007) addresses that question in two ways. First, it provides an exact coordinate system extraction algorithm that, unlike the approximate algorithm of Bucci et al. (2004), is slow but is guaranteed to return a minimal-sized coordinate system for a finite interactive domain. (Interactive domains can, in general, have more than one minimal-sized coordinate system. Hence *a* and not *the*.) Second, it applies the exact extraction algorithm to several small instances of the game of Nim, and observes that for all instances tested, the axes of extracted coordinate systems contain significant information about how a strategy performs at the game. Thus, far from being theoretical peculiarities, at least in this case (minimal) coordinate systems intuitively relate to real structure in the domain.

Given this last fact and that certain coevolutionary algorithms seem to be sensitive to the underlying dimensions of a domain, one is led to an intriguing possibility: that appropriately designed coevolutionary algorithms could discover and extract meaningful structure from interactive domains in the process of solving problems over them. Besides solving problems, algorithms might be able to simultaneously output useful knowledge about domains. For complex, ill-understood domains, the knowledge output might be at least as useful as a solution.

5.3 Open-Ended Evolution and Novelty

An important possibility that should not be lost to our solution-based, problem-oriented view is the idea that the goal of evolutionary systems need not necessarily be to find a particular solution (be it a set of points or otherwise) at all. In particular, in cases where a representation space is not explicitly limited, evolutionary systems can be seen as dynamical methods to explore “interesting” spaces. Here “interesting” might correspond with some problem-oriented view or relate to aspects of some simulation that involves the evolutionary process itself, but it may also simply correspond with notions of novelty.

In traditional, non-coevolutionary systems, an objective fitness measure in conjunction with selection methods will tend to drive the systems in particular directions. Explicit construction of fitness functions that encourage some direct notion of novelty can certainly be developed; however, coevolution offers some very natural and interesting possibilities here. Indeed, some of the earliest forms of artificial evolution involved game playing as means of exploring some of basic mechanisms of evolution and self-maintenance themselves (Barricelli 1963).

Many of the concepts discussed above (e.g., *informativeness* and *distinctions*) are very natural measures of novelty, and algorithms that literally function by moving in directions that work to maintain these are compelling for this purpose. Even more compelling is the idea just discussed above: Since CoEAs can be used to discover geometries of comparisons, to construct and report dimensions of informativeness in a space, they are very appealing mechanisms for novelty search in open-ended evolution because they may be capable of providing much more than new, unusual individuals – they may well be able to place such individuals in relationship with things already seen.

6 Concluding Remarks

The great promise of coevolutionary algorithms to find sorting networks, teams, 3-D creatures, game-playing strategies, or other complicated entities given only information about how these interact with other entities is far from realized. However, the theoretical tools and empirical methodologies presented in this chapter help in getting closer to that goal. One noteworthy outcome of theoretical investigations is that free lunch is possible: algorithms that provably perform better than others across all problem instances. This observation is relatively new and unexplored, but represents an exciting direction in coevolutionary algorithms research. It offers the possibility that a coevolutionary algorithm could not only find interesting or capable entities, but do so faster than random search would. We hope that the principled introduction to co-optimization and coevolutionary computation given in this chapter will contribute to the realization of these goals, by helping future efforts leverage the strong foundation laid by prior research.

References

- Angeline PJ, Pollack JB (1993) Competitive environments evolve better solutions for complex tasks. In: Proceedings of the 5th international conference on genetic algorithms ICGA-1993. Urbana-Champaign, IL, pp 264–270
- Axelrod R (1989) The evolution of strategies in the iterated prisoner’s dilemma. In: Davis L (ed) Genetic algorithms and simulated annealing. Morgan Kaufmann, San Francisco, CA, pp 32–41

- Bader-Natal A, Pollack JB (2004) A population-differential method of monitoring success and failure in coevolution. In: Proceedings of the genetic and evolutionary computation conference, GECCO-2004. Lecture notes in computer science, vol 3102. Springer, Berlin, pp 585–586
- Barbosa H (1999) A coevolutionary genetic algorithm for constrained optimization. In: Proceedings of the congress on evolutionary computation, CEC 1999. IEEE Press, Washington, DC
- Barricelli N (1963) Numerical testing of evolution theories. Part II. Preliminary tests of performance. Symbiogenesis and terrestrial life. *Acta Biotheor* 16(3–4):99–126
- Blumenthal HJ, Parker GB (2004) Punctuated anytime learning for evolving multi-agent capture strategies. In: Proceedings of the congress on evolutionary computation, CEC 2004. IEEE Press, Washington, DC
- Branke J, Rosenbusch J (2008) New approaches to coevolutionary worst-case optimization. In: Parallel problem solving from nature, PPSN-X, Lecture notes in computer science, vol 5199. Springer, Berlin, pp 144–153
- Bucci A (2007) Emergent geometric organization and informative dimensions in coevolutionary algorithms. Ph.D. thesis, Michtom School of Computer Science, Brandeis University, Waltham, MA
- Bucci A, Pollack JB (2002) Order-theoretic analysis of coevolution problems: coevolutionary statics. In: Langdon WB et al. (eds) Genetic and evolutionary computation conference workshop: understanding coevolution. Morgan Kaufmann, San Francisco, CA
- Bucci A, Pollack JB (2003a) Focusing versus intransitivity: geometrical aspects of coevolution. In: Cantú-Paz E et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2003. Springer, Berlin
- Bucci A, Pollack JB (2003b) A mathematical framework for the study of coevolution. In: De Jong KA et al. (eds) Foundations of genetic algorithms workshop VII. Morgan Kaufmann, San Francisco, CA, pp 221–235
- Bucci A, Pollack JB (2005) On identifying global optima in cooperative coevolution. In: Beyer HG et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2005. ACM Press, New York
- Bucci A, Pollack JB, De Jong ED (2004) Automated extraction of problem structure. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2004, Lecture notes in computer science, vol 3102. Springer, Berlin, pp 501–512
- Bull L (1997) Evolutionary computing in multi-agent environments: partners. In: Baeck T (ed) Proceedings of the 7th international conference on genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 370–377
- Bull L (1998) Evolutionary computing in multi-agent environments: operators. In: Wagen D, Eiben AE (eds) Proceedings of the 7th international conference on evolutionary programming. Springer, Berlin, pp 43–52
- Bull L (2001) On coevolutionary genetic algorithms. *Soft Comput* 5(3):201–207
- Bull L (2005a) Coevolutionary species adaptation genetic algorithms: a continuing saga on coupled fitness landscapes. In: Capcarrere M et al. (eds) Proceedings of the 8th European conference on advances in artificial life, ECAL 2005. Springer, Berlin, pp 845–853
- Bull L (2005b) Coevolutionary species adaptation genetic algorithms: growth and mutation on coupled fitness landscapes. In: Proceedings of the congress on evolutionary computation, CEC 2005. IEEE Press, Washington, DC
- Cartlidge J, Bullock S (2002) Learning lessons from the common cold: how reducing parasite virulence improves coevolutionary optimization. In: Proceedings of the congress on evolutionary computation, CEC 2002. IEEE Press, Washington, DC, pp 1420–1425
- Cartlidge J, Bullock S (2003) Caring versus sharing: how to maintain engagement and diversity in coevolving populations. In: Banzhaf W et al. (eds) Proceedings of the 7th European conference on advances in artificial life, ECAL 2003, Lecture notes in computer science, vol 2801. Springer, Berlin, pp 299–308
- Cartlidge J, Bullock S (2004a) Combating coevolutionary disengagement by reducing parasite virulence. *Evolut Comput* 12(2):193–222
- Cartlidge J, Bullock S (2004b) Unpicking tartan CIAO plots: understanding irregular co-evolutionary cycling. *Adapt Behav* 12(2):69–92
- Chellapilla K, Fogel DB (1999) Evolving neural networks to play checkers without expert knowledge. *IEEE Trans Neural Networks* 10(6):1382–1391
- Cliff D, Miller GF (1995) Tracking the red queen: measurements of adaptive progress in co-evolutionary simulations. In: Proceedings of the 3rd European conference on advances in artificial life, ECAL 1995. Lecture notes in computer science, vol 929. Springer, Berlin, pp 200–218
- De Jong KA (1992) Genetic algorithms are not function optimizers. In: Whitley LD (ed) Foundations of genetic algorithms II. Morgan Kaufmann, San Francisco, CA, pp 5–17
- De Jong ED (2004a) The incremental Pareto-coevolution archive. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2004, Lecture notes in computer science, vol 3102. Springer, Berlin, pp 525–536

- De Jong ED (2004b) Intransitivity in coevolution. In: Yao X et al. (eds) Parallel problem solving from nature, PPSN-VIII, Birmingham, UK, Lecture notes in computer science, vol 3242. Springer, Berlin, pp 843–851
- De Jong ED (2004c) Towards a bounded Pareto-coevolution archive. In: Proceedings of the congress on evolutionary computation, CEC 2004. IEEE Press, Washington, DC, pp 2341–2348
- De Jong ED (2005) The MaxSolve algorithm for coevolution. In: Beyer HG et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2005. ACM Press, New York
- De Jong ED (2007) Objective fitness correlation. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2007. ACM Press, New York, pp 440–447
- De Jong ED, Bucci A (2006) DECA: dimension extracting coevolutionary algorithm. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2006. ACM Press, New York
- De Jong ED, Bucci A (2007) Objective set compression: test-based problems and multiobjective optimization. In: Multiobjective problem solving from nature: from concepts to applications, Natural Computing Series. Springer, Berlin
- De Jong ED, Pollack JB (2004) Ideal evaluation from coevolution. *Evolut Comput* 12(2):159–192
- Ficici SG (2004) Solution concepts in coevolutionary algorithms. Ph.D. thesis, Department of Computer Science, Brandeis University, Waltham, MA
- Ficici SG, Pollack JB (1998) Challenges in coevolutionary learning: arms-race dynamics, open-endedness, and mediocre stable states. In: Adami C et al. (eds) Artificial life VI proceedings. MIT Press, Cambridge, MA, pp 238–247
- Ficici SG, Pollack JB (2000a) Effects of finite populations on evolutionary stable strategies. In: Whitley D et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2000. Morgan Kaufmann, San Francisco, CA, pp 880–887
- Ficici SG, Pollack JB (2000b) Game-theoretic investigation of selection methods used in evolutionary algorithms. In: Whitley D (ed) Proceedings of the 2000 congress on evolutionary computation, IEEE Press, Washington, DC, pp 880–887
- Ficici SG, Pollack JB (2001) Pareto optimality in coevolutionary learning. In: Proceedings of the 6th European conference on advances in artificial life, ECAL 2001. Springer, London, pp 316–325
- Ficici SG, Pollack JB (2003) A game-theoretic memory mechanism for coevolution. In: Cantú-Paz E et al. (eds) Genetic and evolutionary computation conference, GECCO 2003. Springer, Berlin, pp 286–297
- Floreano D, Nolfi S (1997) God save the red queen! competition in co-evolutionary robotics. In: Koza JR et al. (eds) Proceedings of the 2nd genetic programming conference, GP 1997. Morgan Kaufmann, San Francisco, CA, pp 398–406
- Friedman D (1998) On economic applications of evolutionary game theory. *J Evol Econ* 8:15–43
- Funes P, Pollack JB (2000) Measuring progress in coevolutionary competition. In: From animals to animats 6: Proceedings of the 6th international conference on simulation of adaptive behavior. MIT Press, Cambridge, MA, pp 450–459
- Funes P, Pujals E (2005) Intransitivity revisited coevolutionary dynamics of numbers games. In: Proceedings of the conference on genetic and evolutionary computation, GECCO 2005. ACM Press, New York, pp 515–521
- Hillis WD (1990) Co-evolving parasites improve simulated evolution as an optimization procedure. In: CNLS '89: Proceedings of the 9th international conference of the center for nonlinear studies on self-organizing, collective, and cooperative phenomena in natural and artificial computing networks on emergent computation. North-Holland Publishing Co., Amsterdam, pp 228–234
- Hofbauer J, Sigmund K (1998) Evolutionary games and population dynamics. Cambridge University Press, Cambridge
- Horn J (1995) The nature of niching: genetic algorithms and the evolution of optimal, cooperative populations. Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL
- Husbands P, Mill F (1991) Simulated coevolution as the mechanism for emergent planning and scheduling. In: Belew R, Booker L (eds) Proceedings of the fourth international conference on genetic algorithms, Morgan Kaufmann, San Francisco, CA, pp 264–270
- Igel C, Toussaint M (2003) On classes of functions for which no free lunch results hold. *Inform Process Lett* 86(6):317–321
- Jansen T, Wiegand RP (2003a) Exploring the explorative advantage of the CC (1+1) EA. In: Proceedings of the 2003 genetic and evolutionary computation conference. Springer, Berlin
- Jansen T, Wiegand RP (2003b) Sequential versus parallel cooperative coevolutionary (1+1) EAs. In: Proceedings of the congress on evolutionary computation, CEC 2003. IEEE Press, Washington, DC
- Jansen T, Wiegand RP (2004) The cooperative coevolutionary (1+1) EA. *Evolut Comput* 12(4):405–434
- Jaśkowski W, Wieloch B, Krawiec K (2008) Fitnessless coevolution. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2008. ACM Press, New York, pp 355–365
- Jensen MT (2001) Robust and flexible scheduling with evolutionary computation. Ph.D. thesis, Department of Computer Science, University of Aarhus, Denmark

- Juillé H, Pollack JB (1998) Coevolving the ideal trainer: application to the discovery of cellular automata rules. In: Koza JR et al. (eds) *Proceedings of the 3rd genetic programming conference, GP 1998*. Morgan Kaufmann, San Francisco, CA, pp 519–527
- Kauffman S, Johnson S (1991) Co-evolution to the edge of chaos: coupled fitness landscapes, poised states and co-evolutionary avalanches. In: Langton C et al. (eds) *Artificial life II proceedings*, vol 10. Addison-Wesley, Reading, MA, pp 325–369
- Laumanns M, Thiele L, Zitzler E (2004) Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Trans Evolut Comput* 8(2):170–182
- Liekens A, Eikelder H, Hilbers P (2003) Finite population models of co-evolution and their application to haploidy versus diploidy. In: Cantú-Paz E et al. (eds) *Proceedings of the genetic and evolutionary computation conference, GECCO 2003*. Springer, Berlin, pp 344–355
- Luke S, Wiegand RP (2003) Guaranteeing coevolutionary objective measures. In: De Jong KA et al. (eds) *Foundations of genetic algorithms VII*, Morgan Kaufmann, San Francisco, CA, pp 237–251
- Maynard-Smith J (1982) Evolution and the theory of games. Cambridge University Press, Cambridge
- Miller JH (1996) The coevolution of automata in the repeated prisoner's dilemma. *J Econ Behav Organ* 29(1):87–112
- Monroy GA, Stanley KO, Miikkulainen R (2006) Coevolution of neural networks using a layered Pareto archive. In: *Proceedings of the genetic and evolutionary computation conference, GECCO 2006*. ACM Press, New York, pp 329–336
- Noble J, Watson RA (2001) Pareto coevolution: using performance against coevolved opponents in a game as dimensions for Pareto selection. In: Spector L et al. (eds) *Proceedings of the genetic and evolutionary computation conference, GECCO 2001*. Morgan Kaufmann, San Francisco, CA, pp 493–500
- Oliehoek FA, De Jong ED, Vlassis N (2006) The parallel Nash memory for asymmetric games. In: *Proceedings of the genetic and evolutionary computation conference, GECCO 2006*. ACM Press, New York, pp 337–344
- Oliveto P, He J, Yao X (2007) Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. *Int J Autom Comput* 4(3): 281–293
- Olsson B (2001) Co-evolutionary search in asymmetric spaces. *Inform Sci* 133(3–4):103–125
- Osborne MJ, Rubinstein A (1994) A course in game theory. MIT Press, Cambridge, MA
- Pagie L, Hogeweg P (2000) Information integration and red queen dynamics in coevolutionary optimization. In: *Proceedings of the congress on evolutionary computation, CEC 2000*. IEEE Press, Piscataway, NJ, pp 1260–1267
- Pagie L, Mitchell M (2002) A comparison of evolutionary and coevolutionary search. *Int J Comput Intell Appl* 2(1):53–69
- Panait L (2006) The analysis and design of concurrent learning algorithms for cooperative multiagent systems. Ph.D. thesis, George Mason University, Fairfax, VA
- Panait L, Luke S (2002) A comparison of two competitive fitness functions. In: Langdon WB et al. (eds) *Proceedings of the genetic and evolutionary computation conference, GECCO 2002*. Morgan Kaufmann, San Francisco, CA, pp 503–511
- Panait L, Luke S (2005) Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In: AAAI fall symposium on coevolutionary and coadaptive systems. AAAI Press, Menlo Park, CA
- Panait L, Luke S (2006) Selecting informative actions improves cooperative multiagent learning. In: *Proceedings of the 5th international joint conference on autonomous agents and multi agent systems, AAMAS 2006*. ACM Press, New York
- Panait L, Wiegand RP, Luke S (2003) Improving coevolutionary search for optimal multiagent behaviors. In: Gottlob G, Walsh T (eds) *Proceedings of the 18th international joint conference on artificial intelligence, IJCAI 2003*. Morgan Kaufmann, San Francisco, CA, pp 653–658
- Panait L, Wiegand RP, Luke S (2004) A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. In: *Proceedings of the genetic and evolutionary computation conference, GECCO 2004. Lecture notes in computer science*, vol 3102. Springer, Berlin, pp 573–584
- Panait L, Luke S, Harrison JF (2006a) Archive-based cooperative coevolutionary algorithms. In: *Proceedings of the genetic and evolutionary computation conference, GECCO 2006*. ACM Press, New York
- Panait L, Luke S, Wiegand RP (2006b) Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Trans Evolut Comput* 10(6):629–645
- Panait L, Sullivan K, Luke S (2006c) Lenience towards teammates helps in cooperative multiagent learning. In: *Proceedings of the 5th international joint conference on autonomous agents and multi agent systems, AAMAS 2006*. ACM Press, New York
- Paredis J (1997) Coevolving cellular automata: be aware of the red queen. In: Bäck T (ed) *Proceedings of the 7th international conference on genetic algorithms, ICGA 1997*. Morgan Kaufmann, San Francisco, CA
- Parker GB, Blumenthal HJ (2003) Comparison of sample sizes for the co-evolution of cooperative agents. In: *Proceedings of the congress on evolutionary computation, CEC 2003*. IEEE Press, Washington, DC

- Popovici E (2006) An analysis of two-population coevolutionary computation. Ph.D. thesis, George Mason University, Fairfax, VA
- Popovici E, De Jong KA (2004) Understanding competitive co-evolutionary dynamics via fitness landscapes. In: Luke S (ed) AAAI fall symposium on artificial multiagent learning. AAAI Press, Menlo Park, CA
- Popovici E, De Jong KA (2005a) A dynamical systems analysis of collaboration methods in cooperative co-evolution. In: AAAI fall symposium series co-evolution workshop. AAAI Press, Menlo Park, CA
- Popovici E, De Jong KA (2005b) Relationships between internal and external metrics in co-evolution. In: Proceedings of the congress on evolutionary computation, CEC 2005. IEEE Press, Washington, DC
- Popovici E, De Jong KA (2005c) Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In: Beyer HG et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2005. ACM Press, New York
- Popovici E, De Jong KA (2009) Monotonicity versus performance in co-optimization. In: Foundations of genetic algorithms X. ACM Press, New York
- Potter M (1997) The design and analysis of a computational model of cooperative coevolution. Ph.D. thesis, Computer Science Department, George Mason University
- Potter M, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from nature, PPSN-III, Jerusalem, Israel. Springer, Berlin, pp 249–257
- Potter MA, De Jong KA (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolut Comput* 8(1):1–29
- Rosin CD (1997) Coevolutionary search among adversaries. Ph.D. thesis, University of California, San Diego, CA
- Rosin CD, Belew RK (1995) Methods for competitive coevolution: finding opponents worth beating. In: Proceedings of the 6th international conference on genetic algorithms, ICGA 1995. Morgan Kaufmann, San Francisco, CA, pp 373–381
- Rosin CD, Belew RK (1997) New methods for competitive coevolution. *Evolut Comput* 5(1):1–29
- Schmitt LM (2003a) Coevolutionary convergence to global optima. In: Cantú-Paz E et al. (eds) Genetic and evolutionary computation conference, GECCO 2003. Springer, Berlin, pp 373–374
- Schmitt LM (2003b) Theory of coevolutionary genetic algorithms. In: Guo M et al. (eds) International symposium on parallel and distributed processing and applications, ISPA 2003. Springer, Berlin, pp 285–293
- Schumacher C, Vose M, Whitley L (2001) The no free lunch and description length. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2001. Morgan Kaufmann, San Francisco, CA, pp 565–570
- Service TC (2009) Unbiased coevolutionary solution concepts. In: Foundations of genetic algorithms X. ACM Press, New York
- Service TC, Tauritz DR (2008a) Co-optimization algorithms. In: Keijzer M et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2008. ACM Press, New York, pp 387–388
- Service TC, Tauritz DR (2008b) A no-free-lunch framework for coevolution. In: Keijzer M et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2008. ACM Press, New York, pp 371–378
- Sims K (1994) Evolving 3D morphology and behaviour by competition. In: Brooks R, Maes P (eds) Artificial life IV proceedings. MIT Press, Cambridge, MA, pp 28–39
- Spears W (1994) Simple subpopulation schemes. In: Proceedings of the 1994 evolutionary programming conference. World Scientific, Singapore
- Stanley KO (2004) Efficient evolution of neural networks through complexification. Ph.D. thesis, The University of Texas at Austin, Austin, TX
- Stanley KO, Miikkulainen R (2002a) Continual coevolution through complexification. In: Langdon WB et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2002. Morgan Kaufmann, San Francisco, CA, pp 113–120
- Stanley KO, Miikkulainen R (2002b) The dominance tournament method of monitoring progress in coevolution. In: Langdon WB et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2002. Morgan Kaufmann, San Francisco, CA
- Stanley KO, Miikkulainen R (2004) Competitive coevolution through evolutionary complexification. *J Artif Intell Res* 21:63–100
- Stuermer P, Bucci A, Branke J, Funes P, Popovici E (2009) Analysis of coevolution for worst-case optimization. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2009. ACM Press, New York
- Subbu R, Sanderson A (2004) Modeling and convergence analysis of distributed coevolutionary algorithms. *IEEE Trans Syst Man Cybern B Cybern* 34(2):806–822
- Vo C, Panait L, Luke S (2009) Cooperative coevolution and univariate estimation of distribution algorithms. In: Foundations of genetic algorithms X. ACM Press, New York
- Vose M (1999) The simple genetic algorithm. MIT Press, Cambridge, MA
- Watson RA, Pollack JB (2001) Coevolutionary dynamics in a minimal substrate. In: Spector L et al. (eds) Proceedings of the genetic and evolutionary computation conference, GECCO 2001. Morgan Kaufmann, San Francisco, CA, pp 702–709

- Watson RA (2002) Compositional evolution: interdisciplinary investigations in evolvability, modularity, and symbiosis. Ph.D. thesis, Brandeis University, Waltham, Massachusetts
- Weibull J (1992) Evolutionary game theory. MIT Press, Cambridge, MA
- Wiegand RP (2004) An analysis of cooperative coevolutionary algorithms. Ph.D. thesis, George Mason University, Fairfax, VA
- Wiegand RP, Potter M (2006) Robustness in cooperative coevolution. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2006. ACM Press, New York
- Wiegand RP, Sarma J (2004) Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In: Yao X et al. (eds) Parallel problem solving from nature, PPSN-VIII. Springer, Birmingham, UK, pp 912–921
- Wiegand RP, Liles W, De Jong KA (2001) An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In: Spector L (ed) Proceedings of the genetic and evolutionary computation conference, GECCO 2001. Morgan Kaufmann, San Francisco, CA, pp 1235–1242, errata available at <http://www.tesseract.org/paul/papers/gecco01-cca-errata.pdf>
- Wiegand RP, Liles WC, De Jong KA (2002) The effects of representational bias on collaboration methods in cooperative coevolution. In: Proceedings of the 7th conference on parallel problem solving from nature. Springer, Berlin, pp 257–268
- Williams N, Mitchell M (2005) Investigating the success of spatial coevolution. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2005. ACM Press, New York, pp 523–530
- Wolpert D, Macready W (1997) No free lunch theorems for optimization. *IEEE Trans Evolut Comput* 1(1):67–82
- Wolpert D, Macready W (2005) Coevolutionary free lunches. *IEEE Trans Evolut Comput* 9(6):721–735

32 Niching in Evolutionary Algorithms

Ofer M. Shir

Department of Chemistry, Princeton University, NJ, USA

oshir@princeton.edu

1	<i>Introduction</i>	1036
2	<i>Background: From Speciation to Ecological Optima</i>	1037
3	<i>Population Diversity in Evolutionary Algorithms</i>	1041
4	<i>Evolutionary Algorithms Niching Techniques</i>	1044
5	<i>Experimental Methodology</i>	1055
6	<i>The Niche-Radius Problem</i>	1060
7	<i>Discussion and Outlook</i>	1064

Abstract

Niching techniques are the extension of standard evolutionary algorithms (EAs) to multimodal domains, in scenarios where the location of multiple optima is targeted and where EAs tend to lose population diversity and converge to a solitary basin of attraction. The development and investigation of EA niching methods have been carried out for several decades, primarily within the branches of genetic algorithms (GAs) and evolution strategies (ES). This research yielded altogether a long list of algorithmic approaches, some of which are bio-inspired by various concepts of organic speciation and ecological niches, while others are more computational-oriented. This chapter will lay the theoretical foundations for niching, from the perspectives of biology as well as optimization, provide a summary of the main contemporary niching techniques within EAs, and discuss the topic of experimental methodology for niching techniques. This will be accompanied by the discussion of specific case-studies, including the employment of the popular covariance matrix adaptation ES within a niching framework, the application to real-world problems, and the treatment of the so-called niche radius problem.

1 Introduction

Evolutionary Algorithms (EAs) have the tendency to lose diversity within their population of feasible solutions and to converge into a single solution (Bäck 1994, 1996; Mahfoud 1995a), even if the search landscape has multiple globally optimal solutions.

Niching methods, the extension of EAs to finding multiple optima in multimodal optimization within one population, address this issue by maintaining the diversity of certain properties within the population. Thus, they aim at obtaining parallel convergence into multiple attraction basins in the multimodal landscape within a single run.

The study of niching is challenging both from the theoretical point of view and from the practical point of view. The theoretical challenge is twofold – maintaining diversity within a population-based stochastic algorithm from the computational perspective, and also gaining an insight into *speciation* theory or *population genetics* from the evolutionary biology perspective. The practical aspect provides a real-world incentive for this problem – there is an increasing interest of the *applied optimization community* in providing decision makers with multiple solutions that ideally represent different conceptual designs, for single-criterion or multi-criterion search spaces (Avigad et al. 2004, 2005). The concept of “*going optimal*” is often extended nowadays into the aim of “*going multi optimal*”: **obtaining optimal results but also providing the decision maker with a variety of different options**. (The *Second Toyota Paradox* (Cristiano et al. 2001), which is often reviewed in management studies, promotes the consideration of multiple candidate solutions during the car production process: “Delaying decisions, communicating ambiguously, and pursuing an excessive number of prototypes, can produce better cars faster and cheaper”).

Among the three conventional streams of EAs (Bäck 1996) – *genetic algorithms* (GAs) (Goldberg 1989), *evolution strategies* (ES) (Beyer and Schwefel 2002), and evolutionary programming (EP) (Fogel 1966) – niching methods have been studied in the past four decades almost exclusively within the framework of GAs. However, niching methods have been mostly

a by-product of studying *population diversity*, and were hardly ever at the front of the evolutionary computation (EC) research as an independent subfield.

This chapter thus aims at achieving three main goals:

1. Laying the theoretical foundations for niching, both from the biological as well as the computational perspectives
2. Providing a summary of the main contemporary niching techniques within evolutionary algorithms
3. Discussing the topic of experimental methodology for niching techniques, and proposing a framework for it

We shall begin by providing the natural computing motivation for niching by means of the biological background for the evolutionary process of *speciation*, given in [Sect. 2](#), where it will also be linked to the domain of function optimization. The important topic of *population diversity* within evolutionary algorithms is discussed in detail in [Sect. 3](#), focusing on the two canonical streams of GAs and ES. This is followed by an overview of the existing niching techniques within EAs, in [Sect. 4](#). A case study of a specific niching technique with the popular covariance matrix adaptation evolution strategy (CMA-ES) receives special attention by means of a detailed description in the end of that section. The topic of experimental methodology for research on niching is discussed in [Sect. 5](#), where proposed synthetic test functions as well as recommended performance criteria are outlined. Niching with the CMA-ES is revisited in that section, as an experimental observation case study. [Section 6](#) is dedicated to a crucial and challenging issue for niching methods, the so-called *niche radius problem*. Finally, [Sect. 7](#) concludes this chapter and proposes directions for future research in this domain.

A Note on the Scope It should be stressed that there exist niching methods in other natural computing frameworks, such as in particle swarm optimization (PSO) or in ant colony optimization (ACO) (for general field reviews see Kennedy and Eberhart ([2001](#)) and Engelbrecht ([2005](#)) for PSO or Dorigo and Stützle ([2004](#)) and Blum ([2005](#)) for ACO). These frameworks exceed the scope of this chapter, which solely focuses on evolutionary algorithms. We refer the reader who wishes to learn about niching techniques in PSO to Brits et al. ([2002](#)) and Parrott and Li ([2006](#)), or to Angus ([2006](#)) in ACO.

A Note on Notation Upon consideration of optimization problems, this chapter will assume *minimization*, without loss of generality, unless specified otherwise.

2 Background: From Speciation to Ecological Optima

This section constitutes the introduction to niching, and thus covers a diverse set of interdisciplinary topics. It will review the *biological* elementary concepts that correspond to the core of niching methods, such as *population diversity* and *speciation*, while mainly relying on Freeman and Herron ([2003](#)). It will then make the linkage to computing, by shifting to the optimization arena and discussing the equivalent to ecological niches: *basins of attraction*.

A Note on Terminology A species is defined as the *smallest evolutionary independent unit*. The term *niche*, however, stems from ecology, and it has several different definitions. It is sometimes referred to as the collective environmental components which are favored by a specific species, but could also be considered as the ecosystem itself which hosts individuals of

various species. Most definitions would typically also consider the *hosting capacity* of the niche, which refers to the limited available resources for sustaining life in its domain. In the context of function optimization, *niche* is associated with a *peak*, or a *basin of attraction*, whereas a *species* corresponds to the subpopulation of individuals occupying that *niche*.

2.1 Preliminary: Organic Evolution and Genetic Drift

Organic evolution can be broken down into four defining fundamental mechanisms: *natural selection*, *mutation*, *migration* or *gene flow*, and *genetic drift*. The latter, which essentially refers to *sampling errors in finite populations*, was overlooked by Darwin, who had not been familiar with Mendelian genetics, and thus did not discuss this effect in his *Origin of Species* (Darwin 1999). We assume that the reader is familiar with the first three fundamental elements, that is, selection–mutation–migration, and would like to elaborate on the drift effect, as it is the least intuitive mechanism among the four aforementioned evolutionary forces. In short, *genetic drift* (Fisher 1922; Wright 1931; Kimura 1983) is a stochastic process in which the diversity is lost in finite populations. A distribution of genetic properties is transferred to the following generation in a limited manner, due to the finite number of generated offspring, or equivalently, the limited statistical sampling of the distribution. As a result, the distribution is likely to approach an *equilibrium distribution*, for example, fixation of specific alleles when subject to equal fitness. This is why *genetic drift* is often considered as a *neutral effect*. The smaller the population, the faster and stronger this effect occurs. An analogy is occasionally drawn between genetic drift to *Brownian motion* of particles in mechanics.

The *genetic drift* effect had been originally recognized by Fisher (1922) (referred to as *random survival*), and was explicitly mentioned by Wright when studying Mendelian populations (Wright 1931). It was, however, revisited and given a new interpretation in the *Neutral Theory of Molecular Evolution* of Kimura (1983). The *neutral theory* suggested that the *random genetic drift* effect is the main driving force within molecular evolution, rather than the *nonrandom natural selection* mechanism. *Natural selection* as well as *genetic drift* are considered nowadays, by the contemporary evolutionary biology community, as the combined driving force of organic evolution. Moreover, the importance of the *neutral theory* is essentially in its being a **null hypothesis model** for the *natural selection theory*, by definition.

2.2 Organic Diversity

Diversity among individuals or populations in nature can be attributed to different evolutionary processes which occur at different levels. We distinguish here between variations that are observed within a single species to a *speciation* process, during which a new species arises, and both of them are reviewed shortly.

2.2.1 Variations Within a Species

Diversity of organisms within a single species stems from a variance at the genotypic level, referred to as *genetic diversity*, or from the existence of a spectrum of phenotypic realizations to a specific genotype. These effects are quantified and are usually associated with *genotypic variance* and *phenotypic variance*, respectively. Several hypotheses explaining *genetic diversity*

have been proposed within the discipline of *population genetics*, including the *neutral evolution theory*. It should be noted that genetic diversity is typically considered to be advantageous for survival, as it may allow better adaptation of the population to environmental changes, such as climate variations, diseases, etc.

Phenotypic variance is measured on a continuous spectrum, also known as quantitative variation. Roughly speaking, the main sources of quantitative variations (Freeman and Herron 2003; Scheiner and Goodnight 1984) are outlined here:

1. Genes have *multiple loci*, and hence are mapped onto a large set of phenotypes.
2. *Environmental effects* have direct influence on natural selection; fitness is time-dependent, and thus phenotypic variations in the outcome of selection are expected to occur.
3. *Phenotypic plasticity* is the amount in which the *genotypic expression* vary in different environments. (Bradshaw (1965) gave the following qualitative definition to phenotypic plasticity: “The amount by which the expressions of individual characteristics of a genotype are changed by different environments is a measure of the plasticity of these characters.”) And it is a direct source of variation at the phenotypic level.
4. The plastic response of the genotype to the environment, that is, the joint effect of genetic and environmental elements, also affects the selection of a specific phenotype, and thus can lead to variations. This effect is known as *genotype–environment interaction* (“G-by-E”).

Thus, *quantitative variations* are mainly caused by genotypic and phenotypic realizations and their interaction with the environment. The ratio between *genetic variance* to total *phenotypic variance* is defined as *heritability* (Wright 1931).

2.2.2 Speciation

Speciation, on the other hand, is the process during which a new species arises. In this case, statistical disassociation, which is the trigger to speciation, originates from gradually decreasing physical linkage.

The essence of the speciation process is **lack of gene flow**, where physical isolation often plays the role of the barrier to gene flow. Lack of gene flow is only one of the necessary conditions for speciation. Another necessary condition for speciation to occur is that the reduction of gene flow will be followed by a phase of **genetic divergence**, by means of *mutation, selection, or drift*. Finally, the completion or elimination of divergence can be assessed via the so-called *secondary contact* phase: interbreeding between the parental populations would possibly fail (offspring is less fit), succeed (offspring is fitter), or have a neutral outcome (offspring has the same fitness). This would correspond, respectively, to increasing, decreasing, or stabilizing the differentiation between the two arising species. Note that the speciation can occur *de facto*, without the actual secondary contact taking place; the latter is for observational assessment purposes.

In organic evolution, four different levels of *speciation* are considered, corresponding to four levels of physical linkage between the subpopulations:

1. **Allopatric speciation** The split in the population occurs only due to complete geographical separation, for example, migration or mountain building. It results in two geographically isolated populations.

2. **Peripatric speciation** Species arise in small populations, which are not geographically separated but rather isolated in practice; the effect occurs mainly due to the *genetic drift* effect.
3. **Parapatric speciation** The geographical separation is limited, with a physical overlap between the two zones where the populations split from each other.
4. **Sympatric speciation** The two diverging populations coexist in the same zone, and thus the speciation is strictly non-geographical. This is observed in nature in parasite populations, that are located in the same zone, but associated with different plant or animal hosts (McPheron et al. 1988).

These four modes of speciation correspond to four levels of geographically decreasing linkages. Generally speaking, *statistical association* of genetic components in nature, such as *loci*, typically results from *physical linkage*. In this case, we claim that statistical disassociation, which is the trigger to speciation, originates from gradually decreasing physical linkage.

In summary, speciation typically occurs throughout three steps:

1. Geographic isolation or reduction of gene flow
2. Genetic divergence (mutation, selection, drift)
3. Secondary contact (observation/assessment)

2.3 “Ecological Optima”: Basins of Attraction

We devote this section to the definition of basins of attraction. The task of defining a *generic basin of attraction* seems to be one of the most difficult problems in the field of *global optimization*, and there have only been few attempts to treat it theoretically. (Intuitively, and strictly metaphorically speaking, we may think of a *region of attraction* of \mathbf{x}_L as the region, where if water is poured, it will reach \mathbf{x}_L . Accordingly, we may then think of the basin of \mathbf{x}_L as the maximal region that will be covered when the cavity at \mathbf{x}_L is filled to the lowest part of its rim (Törn and Zilinskas 1987).)

Rigorously, it is possible to define the basin by means of a *local optimizer*. In particular, consider a gradient descent algorithm starting from \mathbf{x}_0 , which is characterized by the following dynamics:

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla f(\mathbf{x}(t)) \quad (1)$$

with the initial condition $\mathbf{x}(0) = \mathbf{x}_0$. Now, consider the set of points for which the limit exists:

$$\Upsilon = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}(0) = \mathbf{x} \wedge \mathbf{x}(t) \mid_{t \geq 0} \text{satisfies Eq. 1} \wedge \lim_{t \rightarrow \infty} \mathbf{x}(t) \text{ exists} \right\} \quad (2)$$

Definition 1 The *region of attraction* $A(\mathbf{x}_L)$ of a local minimum, \mathbf{x}_L , is

$$A(\mathbf{x}_L) = \left\{ \mathbf{x} \in \Upsilon \mid \mathbf{x}(0) = \mathbf{x} \wedge \mathbf{x}(t) \mid_{t \geq 0} \text{satisfies Eq. 1} \wedge \lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}_L \right\} \quad (3)$$

The *basin* of \mathbf{x}_L is the **maximal level set** that is fully contained in $A(\mathbf{x}_L)$.

In the case of several disconnected local minima with the same function value, it is possible to define the region of attraction as the union of the nonoverlapping connected sets.

2.4 Classification of Optima: The Practical Perspective

On a related note to the theoretical definition of the basin, the practical perspective for the *classification of optima shapes*, also referred to as global topology, is worth mentioning. This topic is strongly related to the emerging subfield of *robustness study* (see, e.g., Tsui 1992), which aims at attaining optima of high quality with large basins (i.e., low partial derivative values in the proximity of the peak). Moreover, yet visited from a different direction, another approach was introduced recently by Lunacek and Whitley for classifying different classes of multimodal landscapes with respect to algorithmic performance (Lunacek and Whitley 2006). The latter defines the *dispersion metric* of a landscape as the degree to which the local optima are globally clustered near one another. Landscapes with low dispersion have their best local optima clustered together in a single *funnel*. (We deliberately avoid the definition of a funnel as it is rather vague. We refer the reader to Doye et al. (2004).) This classification of low dispersion and high dispersion may be associated with the algorithmic trade-off between exploration of the landscape and exploitation of local structures. Upon considering landscapes with multiple funnels, a recent study (Lunacek et al. 2008) investigated the impact of a global structure of two uneven funnels on the evolutionary search. It concluded that EAs tend to converge into the larger funnel, even if it is of suboptimal quality, and thus put their effectiveness in global exploration in multi-funnel landscapes into question.

3 Population Diversity in Evolutionary Algorithms

The term *population diversity* is commonly used in the context of Evolutionary Algorithms, but it rarely refers to a rigorous definition. Essentially, it is associated both with *genetic diversity* as well as with *speciation* – the two different concepts from organic evolution that were discussed in the previous section. This is simply due to the fact that the differences between the two concepts do not have any practical effect on the evolutionary search nor on the goal of maintaining diversity among the evolving candidate solutions. In the well-known trade-off between *exploration* and *exploitation* of the landscape during a search, *maintaining population diversity* is a driving force in the *exploration front*, and thus it is an important component. However, among EC researchers, population diversity is primarily considered as a component due to play a role in the important exploration of the landscape for the sake of obtaining a single solution, while its role in obtaining multiple solutions is typically considered as a secondary one.

Mahfoud's Formalism

Mahfoud constructed a formalism for characterizing *population diversity* in the framework of Evolutionary Algorithms (see Mahfoud 1995a, pp. 50–59). Mahfoud's formal framework was based on the partitioning of the search space into equivalence classes (set to *minima* in the search landscape), a descriptive relation (typically, *genotypic* or *phenotypic* mappings), and the measurement of distance between the current distribution of subpopulations to some predefined *goal distribution*.

Let P be a discrete distribution describing the current partitioning of the population into subpopulations, and let Q be the goal distribution of the population with respect to the defined sites. The formalism focuses in defining the *directed divergence*, or distance

of distribution P to distribution Q . Several well-known metrics follow this formalism by satisfying its various criteria, such as Shannon's information entropy, standard distance metrics, etc. (Mahfoud 1995a).

Diversity Loss

Subject to the complex dynamics of the various forces within an evolutionary algorithm, population diversity is typically lost, and the search is likely to converge into a single basin of attraction in the landscape. *Population diversity loss* within the population of solutions is the fundamental effect which niching methods aim to treat. In fact, from the historical perspective, the quest for diversity-promoting techniques was the main goal within the EC community for some time, and niching methods were merely obtained as *by-products*, so to speak, of that effort.

Due to the fundamental differences between GAs and ES, we choose to describe the effect of population diversity loss for each one of them separately.

3.1 Diversity Loss Within Genetic Algorithms

Mahfoud devoted a large part of his doctoral dissertation to studying population diversity within GAs (Mahfoud 1995a). He concluded that three main components can be attributed to the effect of population diversity loss within GAs.

Selection Pressure The traditional GA applies a probabilistic selection mechanism, namely, the *roulette-wheel selection* (RWS). This mechanism belongs to a broad set of selection mechanisms, which follow the fitness-proportionate selection principle. Selection pressure is thus associated with the first *moment of the selection operator*. It has been demonstrated by Mahfoud (1995a) that the selection pressure, or equivalently the nonzero expectation of the selection operator, prevents the algorithm from converging in parallel into more than a single attractor.

Selection Noise Selection noise is associated with the second *moment of the selection operator*, or its *variance*. Mahfoud (1995a) demonstrated that the high variance of the RWS, as well as of other selection mechanisms, is responsible for the fast convergence of a population into a single attractor, even when there exists a set of equally fit attractors. This effect can be considered as a *genetic drift* in its broad definition, that is, sampling error of a distribution.

Operator Disruption Evolutionary operators in general, and the *mutation* and *recombination* operators in particular, boost the evolution process toward exploration of the search space. In that sense, they have a constructive effect on the process, since they allow locating new and better solutions. However, their action also has a destructive effect. This is due to the fact that, by applying them, good solutions that have been previously located might be lost. Therefore, they may eliminate competition between highly fit individuals, and "assist" some of them to take over. The mutation operator usually has a small effect, since it acts in small steps – low mutation probability in the traditional GA, which means infrequent occurrence of bit flips. Thus, the mutation operator can be considered to have a negligible disruption. The recombination operator, on the other hand, has a more considerable effect. In the GA field, where the *crossover* operator is in use (single-point, two-point or n -point crossovers), the latter has been shown to have a disruptive nature by breaking desired patterns within the population (the well-known *schema theorem* discusses the schema disruption by the crossover operator

and states that schemata with high defining length will most likely be disrupted by the crossover operator; see, for example, Goldberg (1989).

Corollary 1 *The traditional GA, which employs the standard set of operators, is exposed to statistical as well as disruptive effects that are responsible for the loss of population diversity. This outcome is likely to occur due to the first and second moments of the RWS operator, as well as to the disruptive nature of the crossover operator. We conclude that the traditional GA is expected to lose diversity among its candidate solutions.*

3.2 Diversity Loss Within Evolution Strategies

The defining mechanism of ES is strongly dictated by the mutation operator as well as by the deterministic selection operator. As defining operators, they have a direct influence on the diversity property of the population. The recombination operator, nevertheless, does not play a critical role in the ES mechanism.

We attribute two main components to the *population diversity loss* within ES: fast *take-over*, which is associated with the *selection* operator, and *genetic drift* (or *neutrality effect*), which is associated both with the *selection* and the *recombination* operators.

3.2.1 Selective Pressure: Fast Take-Over

Evolution strategies have a strictly deterministic, rank-based approach to selection. In the two traditional approaches (Bäck 1996) – (μ, λ) and $(\mu + \lambda)$ – the best individuals are selected, implying, rather intuitively, high *selective pressure*. Due to the crucial role of the selection operator within the evolution process, its impact within the ES field has been widely investigated.

Goldberg and Deb introduced the important concept of *take-over time* (Deb and Goldberg 1989), which gives a quantitative description of selective pressure **with respect only to the selection operator**.

Definition 2 The *take-over time*, τ^* , is the minimal number of generations until repeated application of the selection operator yields a uniform population filled with copies of the best individual.

The selective pressure has been further investigated by Bäck (1994), who analyzed all the ES selection mechanisms also with respect to take-over times. He concluded that upon employing the typical selection mechanisms, very short *take-over times* are yielded. This result implies that ES are typically subject to high *selective pressures*.

3.2.2 ES Genetic Drift

We consider two different ES neutral effects that could be together ascribed as a general ES genetic drift: *recombination drift*, and *selection drift*. We argue that these two components are responsible for the loss of population diversity within ES.

Recombination Drift

Beyer explored extensively the so-called *mutation-induced speciation by recombination* (MISR) principle (see, e.g., Beyer 1999). According to this important principle, repeated application of the mutation operator, subject to a dominant recombination operator, would lead to a stable distribution of the population, which resembles a species or a cloud of individuals. When fitness-based selection is applied, this cloud is likely to move together toward fitter regions of the landscape. Furthermore, Beyer managed to prove analytically (Beyer 1999) that the MISR principle is indeed universal when finite populations are employed, subject to sampling-based recombination. The latter was achieved by analyzing the ES dynamics without fitness-based selection, deriving the expected population variance, and showing that it is reduced with random sampling in finite populations. This result was also corroborated by numerical simulations. This study provides one with an analytical result that a sampling-based recombination is subject to genetic drift, and leads to loss of population diversity.

Selection Drift

A recent study on the extinction of subpopulations on a simple *bimodal equi-fitness* model investigated the drift effect of the selection operator (Schönemann et al. 2004). It considered the application of *selection* on finite populations, when the fitness values of the different attractors were equal (i.e., eliminating the possibility of a *take-over effect*), and argued that a neutral effect (*drift*) would occur, pushing the population into a single attractor. The latter study, indeed, demonstrated this effect of *selection drift* in ES, which resulted in a convergence to an equilibrium distribution around a single attractor. It was also shown that the time of extinction increases proportionally with μ . The analysis was conducted by means of Markov chain models, supported by statistical simulations.

Corollary 2 *Evolution Strategies that employ finite populations are typically underposed to several effects that are responsible for the loss of population diversity. It has been shown that the standard selection mechanisms may lead to a fast take-over effect. In addition, we argued that both the recombination and the selection operators experience their own drift effects that lead to population diversity loss. We concluded that an evolution strategy with a small population is likely to encounter a rapid effect of diversity loss.*

4 Evolutionary Algorithms Niching Techniques

Despite the fact that the motivation for multimodal optimization is beyond doubt, and the biological inspiration is real, there is no unique definition of the goal statement for *niching techniques*. There have been several attempts to provide a proper definition and functional specification for niching; we review here some of them:

1. Mahfoud (1995a) chose to put emphasis on locating as well as maintaining good optima, and formulated the following:
 - The litmus test for a niching method, therefore, will be whether it possesses the capability to find multiple, final solutions within a reasonable amount of time, and to maintain them for an extended period of time.

2. Beyer et al. (2002) put forward also the actual maintenance of population diversity:
 - *Niching*: process of separation of individuals according to their states in the search space or maintenance of diversity by appropriate techniques, for example, local population models, fitness sharing, or distributed EA.
3. Preuss (2006) considered the two definitions mentioned above, and proposed a third:
 - Niching in EAs is a two-step procedure that (a) concurrently or subsequently distributes individuals onto distinct basins of attraction and (b) facilitates approximation of the corresponding (local) optimizers.

We choose to adopt Preuss' mission statement and define **the challenge in niching as follows:**

- **Attaining the optimal interplay between partitioning the search space into niches occupied by stable subpopulations, by means of population diversity preservation, to exploiting the search in each niche by means of a highly efficient optimizer with local-search capabilities.**

Next, we shall provide an overview of existing niching techniques. Special attention will be given to a specific niching method which is based on a state-of-the-art evolution strategy, the so-called covariance matrix adaptation ES (CMA-ES). The latter is considered to be particularly efficient for high-dimensional continuous optimization, and will be described in greater detail.

4.1 GA Niching Methods

Niching methods within genetic algorithms have been studied during the past few decades, initially triggered by the necessity to promote *population diversity* within EAs. The research has yielded a variety of different methods, which are the vast majority of existing work on niching in general.

The remainder of this section will focus on GA niching techniques, by providing a short survey of the main known methods, with emphasis on the important concepts of *sharing* and *crowding*. This survey is mainly based on Mahfoud (1995a) and Singh and Deb (2006).

4.1.1 Fitness Sharing

The *sharing* concept was one of the pioneering niching approaches. It was first introduced by Holland (1975), and later implemented as a niching technique by Goldberg and Richardson (1987). This strong approach of **considering the fitness as a shared resource** has essentially become an important concept in the broad field of evolutionary algorithms, and laid the foundation for various successful niching techniques for multimodal function optimization, mainly within GAs. A short description of the *fitness sharing* mechanism follows.

The basic idea of *fitness sharing* is to consider the fitness of the landscape as a resource to be shared among the individuals, in order to decrease redundancy in the population. Given a similarity metric of the population, which can be *genotypic* or *phenotypic*, the *sharing function* is defined as follows:

$$sh(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\rho}\right)^{\alpha_{sh}} & \text{if } d_{i,j} < \rho \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $d_{i,j}$ is the distance between individuals i and j , ρ (traditionally noted as σ_{sh}) is the fixed radius of every niche, and $\alpha_{sh} \geq 1$ is a control parameter, typically set to 1. Using the *sharing function*, the *niche count* is given by

$$m_i = \sum_{j=1}^N sh(d_{i,j}) \quad (5)$$

where N is the number of individuals to be considered in the selection phase.

Let an individual raw fitness be denoted by f_i , then the *shared fitness* is defined by:

$$f_i^{sh} = \frac{f_i}{m_i} \quad (6)$$

assuming that the fitness is *strictly positive* and subject to *maximization*. The evaluation of the shared fitness is followed by the selection phase, which is typically based on the RWS operator (Goldberg 1989); The latter takes into consideration the shared fitness. Thus, the *sharing mechanism* practically penalizes individuals that have similar members within the population via their fitness, and by that it aims at reducing redundancy in the gene pool, especially around the peaks of the fitness landscape.

One important auxiliary component of this approach is the *niche radius*, ρ . Essentially, this approach makes a strong assumption concerning the fitness landscape, stating that the optima are far enough from one another with respect to the *niche radius*, which is estimated for the given problem and remains fixed during the course of evolution. Furthermore, it is important to note that the formulas for determining the value of ρ , which will be given in [Sect. 4.3](#), are dependent on q , the number of peaks of the target function. Hence, a second assumption is that q can be estimated. In practice, an accurate estimation of the expected number of peaks q in a given domain may turn out to be extremely difficult. Moreover, peaks may vary in shape, and this would make the task of determining ρ rather complicated. The aforementioned assumptions pose the so-called *niche radius problem*, to be discussed in [Sect. 6](#).

In the literature, several GA niching *sharing-based* techniques, which implement and extend the basic concept of sharing, can be found (Mahfoud 1995a; Goldberg 1987; Yin and Germany 1993; Jelasity 1998; Miller and Shaw 1996; Petrowski 1996; Cioppa et al. 2004). Furthermore, the concept of sharing was successfully extended to other “yields of interest,” such as *concept sharing* (Avigad et al. 2004).

4.1.2 Dynamic Fitness Sharing

In order to improve the *sharing mechanism*, a dynamic approach was proposed. The *dynamic niche sharing* method (Miller and Shaw 1996), which extended the *fitness sharing* technique, aimed at dynamically recognizing the q peaks of the forming niches, and based on that information, classified the individuals as either members of one of the niches, or as members of the “non-peaks domain.”

Explicitly, let us introduce the *dynamic niche count*:

$$m_i^{\text{dyn}} = \begin{cases} n_j & \text{if individual } i \text{ is within dynamic niche } j \\ m_i & \text{otherwise (non-peak individual)} \end{cases} \quad (7)$$

where n_j is the size of the j th dynamic niche (i.e., the number of individuals which were classified to niche j), and m_i is the standard *niche count*, as defined in [Eq. 5](#).

The shared fitness is then defined as follows:

$$f_i^{\text{dyn}} = \frac{f_i}{m_i^{\text{dyn}}} \quad (8)$$

The identification of the dynamic niches can be carried out by means of a *greedy* approach, as proposed in Miller and Shaw (1996) as the dynamic peak identification (DPI) algorithm (see [Algorithm 1](#)). As in the original *fitness sharing* technique, the *shared fitness evaluation* is followed by the selection phase, typically implemented with the RWS operator. Thus, this technique does not fixate the peak individuals, but rather provides them with an advantage in the selection phase, which is probability-based within GAs.

4.1.3 Clearing

Another variation to the *fitness sharing* technique, called *clearing*, was introduced by Petrowski (1996) at the same time as the *dynamic fitness sharing* (Miller and Shaw 1996). The essence of this mechanism is the “*winner takes it all*” principle, and its idea is to designate a specific number of individuals per niche, referred to as *winners*, which could enjoy the resources of that niche. This is equivalent to the introduction of a “*death penalty*” to the *losers* of the niche, the individuals of each niche that lose the generational competition to the actual peak individuals. Following a *radius-based* procedure of identifying the winners and losers of each niche in each generation, the winners are assigned with their raw-fitness values, whereas all the other individuals are assigned with zero fitness (*maximization* was assumed). This is called the *clearing phase*. The selection phase, typically based on the RWS operator, considers, *de facto*, only the winners of the different niches. The allowed number of winners per niche, also referred to as the *niche capacity*, is a control parameter that reflects the degree of elitism. In any case, as in the previous techniques, the peak individuals are never fixated, and are subject to the probabilistic selection of the GA. This method was shown to outperform the *fitness sharing* technique on a specific set of low-dimensional test problems (Petrowski 1996).

Algorithm 1 Dynamic Peak Identification

```

 input: population Pop, number of niches q, niche radius ρ
1: Sort Pop in increasing fitness order {minimization}
2: i := 1
3: NumPeaks := 0
4: DPS := ∅ {Set of peak elements in population}
5: while NumPeaks ≠ q and i ≤ popSize do
6:   if Pop[i] is not within sphere of radius ρ around peak in DPS then
7:     DPS := DPS ∪ {Pop[i]}
8:     NumPeaks := NumPeaks + 1
9:   end if
10:  i := i + 1
11: end while
output: DPS

```

4.1.4 Crowding

Crowding was one of the pioneering methods in this field, as introduced by De Jong (1975). It considered, and to some extent generalized, *preselection schemes*, which had been investigated in the doctoral dissertation of Cavicchio (1970). The latter had showed that certain preselection schemes boosted the preservation of population diversity. The *crowding* approach aimed at reducing changes in the population distribution between generations, in order to prevent *premature convergence*, by means of *restricted replacement*. Next, we will describe the method in more detail.

Given the traditional GA, a proportion G of the population is selected in each generation via fitness-proportionate selection to undergo variations (i.e., *crossover* and *mutation*) – out of which a part is chosen to die and to be replaced by the new offspring. Each offspring finds the individuals it replaces by taking a random sample of CF (referred to as **crowding factor**) individuals from the population, and replacing the **most similar individual** from the sample. An appropriate *similarity metric* should be chosen.

The crucial point of this niching mechanism is the calculation of the so-called *crowding distance between parents and offspring*, in order to control the *change rate* between generations. A different use of the *crowding distance*, applied among individuals of the same generation and assigned with reversed ranking, is widely encountered in the context of evolutionary multiobjective optimization (EMOA) (Deb 2001; Coello Coello et al. 2007). In the joint context of niching and EMOA see also Deb’s “omni-optimizer” (Deb and Tiwari 2005).

Mahfoud, who analyzed the *crowding* niching technique (1995a), concluded that it was subject to disruptive effects, mainly *drift*, which prevented it from maintaining more than two peaks. He then proposed a mechanism called *deterministic crowding*, as an improvement to the original *crowding* scheme. The proposed procedure applies variation operators to pairs of individuals in order to generate their offspring, which are all then evaluated with respect to the crowding distance, and undergo *replacement selection* (see  [Algorithm 2](#), which assumes *maximization*).

4.1.5 Clustering

The application of *clustering* for niching is very intuitive from the computational perspective, as well as straightforward in its implementation. Yin et al. (1993) proposed a clustering

Algorithm 2 Deterministic Crowding: Replacement Selection (*maximization*)

- 1: Select two parents, p_1 and p_2 , randomly, without replacement
- 2: Generate two variations, c_1 and c_2
- 3: **if** $d(p_1, c_1) + d(p_2, c_2) \leq d(p_1, c_2) + d(p_2, c_1)$ **then**
- 4: **if** $f(c_1) > f(p_1)$ **then** replace p_1 with c_1
- 5: **if** $f(c_2) > f(p_2)$ **then** replace p_2 with c_2
- 6: **else**
- 7: **if** $f(c_2) > f(p_1)$ **then** replace p_1 with c_2
- 8: **if** $f(c_1) > f(p_2)$ **then** replace p_2 with c_1
- 9: **end if**

framework for niching with GAs, which we describe here briefly. A clustering algorithm, such as the *K-Means* algorithm (Haykin 1999), first partitions the population into niches, and then considers the *centroids*, or center points of mass, of the newly partitioned subpopulations.

Let d_{ic} denote the distance between individual i and its *centroid*, and let f_i denote the raw fitness of individual i . Assuming that there are n_c individuals in the niche of individual i , its fitness is defined as:

$$f_i^{\text{Clustering}} = \frac{f_i}{n_c \cdot (1 - (d_{ic}/2d_{\max})^\alpha)} \quad (9)$$

where d_{\max} is the maximal distance allowed between an individual and its niche centroid, and α is a defining parameter. It should be noted that the clustering algorithm uses an additional parameter, d_{\min} , for determining the minimal distance allowed between centroids, playing an equivalent role to the *niche radius* ρ of the *sharing*-based schemes.

This method is often subject to criticism for its strong dependency on a relatively large number of parameters. However, this *clustering* technique has become a popular kernel for niching with EAs, and its application was reported in various studies (see, e.g., Schönemann et al. 2004, Hanagandi and Nikolaou 1998, Branke 2001, Gan and Warwick 2001, Aichholzer et al. 2000, Streichert et al. 2003, and Ando et al. 2005).

4.1.6 The Sequential Niche Technique

A straightforward approach of *iteration* can be used to sequentially locate multiple peaks in the landscape, by means of an *iterative local search* (Ramalhinho-Lourenco et al. 2000). This procedure is blind to any information gathered in previous searches, and sequentially restarts stochastic search processes, hoping to hit a different peak every run. Obviously, it is likely to encounter *redundancy*, and the number of expected iterations is then increased by a factor. A **redundancy factor** can be estimated if the peaks are of equal height (equi-fitness landscape), that is, the probability to converge into any of the q peaks is equal to $1/q$:

$$R = \sum_{i=1}^q \frac{1}{i}$$

For $q > 3$, this can be approximated by:

$$R \approx \gamma + \ln(q) \quad (10)$$

where $\gamma \approx 0.577$ is the Euler–Mascheroni constant. This *redundancy factor* remains reasonably low for any practical value of q , but is expected to considerably increase if all optima are not likely to be found equal.

On a related note, we would like to mention a multirestart with an increasing population size approach, that was developed with the CMA-ES algorithm (Auger and Hansen 2005a). The latter aims at attaining the global minimum, while possibly visiting local minima along the process and restarting the algorithm with a larger population size and a modified initial step-size. It is not defined as a niching technique and does not target optima other than the global minimum, but it can capture suboptimal minima during its search.

Beasley et al. extended the naive *iteration* approach, and developed the so-called *Sequential Niche* technique (Beasley et al. 1993). This method, in contrast to the other niching methods presented earlier, does not modify the genetic operators nor any characteristics of the

traditional GA, but rather creates a general search framework suitable for locating multiple solutions. By means of this method the search process turns into a sequence of independent runs of the traditional GA, where the basic idea is to suppress the fitness function at the observed optimum that was obtained in each run, in order to prevent the search from revisiting that optimum.

In further detail, the traditional GA is run multiple times sequentially: given the best solution of each run, it is first stored as a possible final solution and, second, the fitness function is artificially suppressed in all the points within the neighborhood of that optimum up to a desired radius. This modification is done immediately after each run. Its purpose is to discourage the following runs from revisiting these optima, and by that to encourage the exploration of other areas of the search landscape – aiming at obtaining all its optima. It should be noted that each function modification might yield artificial discontinuities in the fitness landscape. This method focuses only on locating multiple optima of the given search problem, without considering the concepts of parallel evolution and formation of subpopulations. In that sense, it has been claimed that it could not be considered as a niching method, but rather as a modified iterated search.

4.1.7 The Islands Model

This is probably the most intuitive niching approach from the biological perspective, directly inspired by organic evolution. Also referred to as the *regional population model*, this approach (see, e.g., Grosso 1985; Adamidis 1994; Martin et al. 1997) simulates the evolution of subpopulations on remote computational units (independent processors), aiming at achieving a speciation effect by **monitoring the gene flow**. The population is divided into multiple subpopulations, which evolve independently for a fixed number of generations, called *isolation period*. This is followed by a phase of controlled gene flow, or *migration*, when a portion of each subpopulation migrates to other nodes.

The genetic diversity and the amount of information exchange between subpopulations are determined by the following parameters – the number of exchanged individuals, the *migration rate*, the selection method of the individuals for migration (uniformly at random, or elitist fitness-based approach), and the scheme of migration, for example, complete net topology, ring topology, or neighborhood topology.

4.1.8 Other GA-Based Methods

Tagging (see, e.g., Spears 1994 and Deb and Spears 1997) is a mechanism that aims at improving the distance-based methods of *fitness sharing* and *crowding*, by labeling individuals with tag-bits. Rather than carrying out distance calculations, the tag-bits are employed for identifying the subpopulations, enforcing *mating restrictions*, and then implementing the *fitness sharing* mechanism. An individual is classified to a subpopulation by its genetic inheritance, so to speak, which is subject to generational variations, rather than by its actual spatial state. This concept simplifies the classification process, and obviously reduces the computational costs per generation, and at the same time it introduces a new bio-inspired approach into niching: individuals belong to a species because their parents did, and not because they are currently adjacent to a “peak individual,” for instance. This technique was shown in Spears (1994) to be a rather efficient implementation of the *sharing* concept.

A complex subpopulation differentiation model, the so-called **multinational evolutionary algorithm**, was presented by Ursem (1999). This original technique considers a world of “nations,” “governments,” and “politicians,” with dynamics dictated by migration of individuals, merging of subpopulations, and selection. Additionally, it introduces a topology-based auxiliary mechanism of *sampling*, which detects whether feasible solutions share the same basin of attraction. Due to the *curse of dimensionality*, this sampling-based mechanism is expected to lose its efficiency in high-dimensional landscapes.

Stoean et al. (2005) constructed the so-called **elitist generational genetic chromodynamics algorithm**. The idea behind this radius-based technique was the definition of a *mating region*, a *replacement region*, and a *merging region* — with appropriate mating, replacement, and merging radii — which dictates the dynamics of the genetic operations.

4.1.9 Miscellaneous: Mating Schemes

It has been observed that once the niche formation process starts, that is, when the population converges into the multiple basins of the landscape, crossbreeding between different niches is likely to fail in producing good offspring. In biological terms, this is the elimination of the divergence, by means of *hybridization*, in the **secondary contact phase**, as discussed in [Sect. 2.2](#).

Deb and Goldberg (1989) proposed a so-called *mating restriction scheme*, which poses a limitation on the choice of partners in the reproduction phase and prevents recombination between competing niches. They employed a distance measure, subject to a distance threshold, which was set to the niche radius, and showed that it could be used to improve the *fitness sharing* algorithm.

Mahfoud (1995a) proved that the mating restriction scheme of Deb and Goldberg was not sufficient, *per se*, in maintaining the population diversity in GA niching. A different approach of Smith and Bonacina (2003), however, considered an evolutionary computation multi-agent system, as opposed to the traditional *centralized EA*, and did manage to show that the same mating restriction scheme in an agent-based framework was capable of maintaining diversity and converging with stability into the desired peaks.

From the biological perspective, the mating restriction scheme is obviously equivalent to keeping the geographical isolation, or the barrier to gene flow, in order to allow the completion of the speciation phase. As discussed earlier, the geographical element in organic evolution is the crucial component which creates the conditions for speciation, and it is not surprising that artificial niching techniques choose to enforce it, by means of mechanisms such as the niche radius or the mating restriction scheme.

4.2 ES Niching Methods

Researchers in the field of evolution strategies initially showed no particular interest in the topic of niching, leaving it essentially for genetic algorithms. An exception would be the employment of island models. Generally speaking, classical niching schemes such as *fitness sharing*, which redefine the selection mechanism, are likely to interfere with the core of evolution strategies – the *self-adaptation mechanism* – and thus doomed to fail in a straightforward implementation. Any manipulation of the fitness value is usually not suitable for evolution strategies, as in the case of constraints handling: death penalty is typically the chosen

approach for a violation of a constraint in ES, rather than a continuous penalty as used in other EAs, in order to avoid the introduction of disruptive effects to the self-adaptation mechanism (see, e.g., Coello Coello 1999; Kramer and Schwefel 2006). Therefore, niching with evolution strategies would have to be addressed from a different direction. Moreover, the different nature of the ES dynamics, throughout the *deterministic selection* and the *mutation operator*, suggests that an alternative treatment is required here.

There are several, relatively new, niching methods that have been proposed within ES, mostly clustering-based (Schönemann et al. 2004; Aichholzer et al. 2000; Streichert et al. 2003). In addition, niching was also introduced to the mixed-integer ES framework (Li et al. 2008). A different approach, based on derandomized evolution strategies (DES) (for the latter see, e.g., Ostermeier et al. (1993, 1994) and Hansen et al. (1995)), was presented by Shir and Bäck (2008). One of its variants, which employs the popular covariance matrix adaptation evolution strategy (CMA-ES), will receive special attention here, and will be set as a detailed case study of niching techniques, to be outlined in the following section.

4.2.1 Case Study: Niching with CMA-ES

A niching framework for $(1 + \lambda)$, derandomized ES (DES) kernels subject to a fixed niche radius has been introduced recently (see, e.g., Shir and Bäck 2008). Following the *mission statement* presented earlier, the aim was the construction of a generic niching framework, which offers the combination of population diversity preservation and local search capabilities. Thus, DES were considered as an excellent choice for that purpose, as EAs with local search characteristics. Furthermore, DES typically employ small populations, which was shown to be a potential advantage for a niching technique, as it can boost the speciation effect (Shir 2008).

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The CMA-ES (Hansen and Ostermeier 2001), is a DES variant that has been successful in treating correlations among object variables by efficiently learning matching mutation distributions. Explicitly, given an initial search point $\mathbf{x}^{(0)}$, λ offspring are generated by means of normally distributed variations:

$$\mathbf{x}^{(g+1)} \sim \mathcal{N}\left(\langle \mathbf{x} \rangle_W^{(g)}, \sigma^{(g)^2} \mathbf{C}^{(g)}\right) \quad (11)$$

Here, $\mathcal{N}(\mathbf{m}, \mathbf{C})$ denotes a normally distributed random vector with mean \mathbf{m} and a covariance matrix \mathbf{C} . The best μ search points out of these λ offspring undergo weighted recombination and become the parent of the next generation, denoted by $\langle \mathbf{x} \rangle_W$. The covariance matrix \mathbf{C} is initialized as the *unity matrix* and is learned during the course of evolution, based on cumulative information of successful past mutations (the *evolution path*). The global step-size, $\sigma^{(g)}$, is updated based on information extracted from *principal component analysis* of $\mathbf{C}^{(g)}$ (the *conjugate evolution path*). For more details, we refer the reader to Hansen and Ostermeier (2001).

An elitist sibling to the CMA comma strategy was also introduced (Igel et al. 2006), based upon the classical $(1+1)$ -ES (Bäck 1996).

A Detailed Description of the Algorithm

This niching technique is based upon interacting search processes, which simultaneously perform a derandomized $(1, \lambda)$ or $(1 + \lambda)$ search in different locations of the space. In case of

multimodal landscapes these search processes are meant to explore different attractor basins of local optima.

An important point in this approach is to strictly enforce the fixed allocation of the population resources, that is, number of offspring per niche. The idea is thus to prevent a scenario of a take-over, where a subpopulation located at a fitter optimum can generate more offspring. The biological idea behind this fixed allocation of resources stems from the concept of limited *hosting capacities* of given ecological niches, as previously discussed.

The *speciation interaction* occurs every generation when all the offspring are considered together to become niches' representatives for the following iteration, or simply the next search points, based on the rank of their fitness and their location with respect to higher-ranked individuals. The focus here is on a simple framework without recombination ($\mu = 1$).

Given q , the estimated/expected number of peaks, $q + p$ "D-sets" are initialized, where a D-set is defined as the collection of all the dynamically adapted strategy as well as decision parameters of the CMA algorithm, which uniquely define the search at a given point of time. (A D-set originally referred to the *derandomized* set of strategy parameters. When the CMA kernel is in use, it is sometimes referred to in the literature as a CMA-set.) These parameters are the current search point, the covariance matrix, the step-size, as well as other auxiliary parameters. At every point in time the algorithm stores exactly $q + p$ D-sets, which are associated with $q + p$ search points: q for the peaks and p for the "non-peaks domain." The $(q+1)th \dots (q+p)th$ D-sets are associated with individuals, which are randomly re-generated every *epoch*, that is, a cycle of κ generations, as potential candidates for niche formation. This is basically a *quasi-restart* mechanism, which allows new niches to form dynamically. Setting the value of p should reflect the trade-off between applying a wide restart approach for exploring further the search space to exploiting computational resources for the existing niches. In any case, due to the *curse of dimensionality*, p loses its significance as the dimension of the search space increases.

Until the stopping criterion is met, the following procedure takes place. Each search point samples λ offspring, based on its evolving D-set. After the fitness evaluation of the new $\lambda \cdot (q + p)$ individuals, the classification into niches of the entire population is obtained in a *greedy* fashion, by means of the DPI routine (Miller and Shaw 1996) (Algorithm 1). The latter is based on the fixed niche radius ρ . The peaks then become the new search points, while their D-sets are inherited from their parents and updated according to the CMA defining equations.

A pseudocode for a single iteration in this *niching routine* is presented as Algorithm 3.

Natural Interpretation: Alpha-Males Competition

We would like to point out the nature of the subpopulations dynamics in the aforementioned niching scheme. Due to the *greedy* classification to niches, which is carried out every generation, some niches can merge in principle, while all the individuals, except for the *peak individual*, die out in practice. Following the posed principle of fixed resources per niche, only the peak individual will be sampled λ times in the following generation. In socio-biological terms, the peak individual could be then associated with an **alpha-male**, which wins the local competition and gets all the sexual resources of its ecological niche. The algorithm as a whole can be thus considered as a competition between $q + p$ alpha-males, each of which is fighting for one of the available q "computational resources," after winning its local competition at the "ecological optimum" site. The domination battles take place locally every cycle, as dictated by the DPI scheme. An elitist-CMA kernel will allow aging alpha-males to keep participating in the ongoing competitions, whereas a comma strategy will force their replacement by fresh

Algorithm 3 $(1+\lambda)$ -CMA-ES Niching with Fixed Niche Radius (A Single Iteration)

```

1: for  $i = 1 \dots (q + p)$  search points do
2:   Generate  $\lambda$  samples based on the D-set of  $i$ 
3: end for
4: Evaluate fitness of the population
5: Compute the Dynamic Peak Set with the DPI Algorithm
6: for all elements of  $DPS$  do
7:   Set peak as a search point
8:   Inherit the D-set and update it respectively
9: end for
10: if  $N_{DPS}$ =size of  $DPS < q$  then
11:   Generate  $q - N_{DPS}$  new search points, reset D-sets
12: end if
13: if  $gen \bmod \kappa \equiv 0$  then
14:   Reset the  $(q + 1)th \dots (q + p)th$  search points
15: end if

```

blood of new alpha-males. At the global level, the value of p determines the selection pressure of alpha-males; setting $p = 0$ will then eliminate global competition and will grant automatically λ offspring to each alpha-male in the following generation, respectively.

4.3 Niche-Radius Calculation

The original formula for ρ for *phenotypic sharing* in GAs was derived by Deb and Goldberg (1989). Analogously, by considering the decision parameters as the decoded parameter space of the GA, the same formula can be applied, using the Euclidean metric, to ES. Given q , the number of target peaks in the solution space, every niche is considered to be surrounded by an n -dimensional hypersphere with radius ρ , which occupies $\frac{1}{q}$ of the entire volume of the space. The volume of the hypersphere which contains the entire space is

$$V = cr^n \quad (12)$$

where c is a constant, given explicitly by:

$$c = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2} + 1)} \quad (13)$$

with $\Gamma(n)$ as the Gamma function. Given lower and upper bound values $x_{k,\min}$, $x_{k,\max}$ of each coordinate in the decision parameters space, r is defined as follows:

$$r = \frac{1}{2} \sqrt{\sum_{k=1}^n (x_{k,\max} - x_{k,\min})^2} \quad (14)$$

If we divide the volume into q parts, we may write

$$c\rho^n = \frac{1}{q} cr^n \quad (15)$$

which yields

$$\rho = \frac{r}{\sqrt[n]{q}} \quad (16)$$

Hence, by applying this niche radius approach, two assumptions are made:

1. The number of target peaks, q , is given or can be estimated.
2. All peaks are at least at distance 2ρ from each other, where ρ is the fixed radius of every niche.

5 Experimental Methodology

Since the topic of niching has not drawn considerable attention from the mainstream EC community, there have been no constructive attempts to generate a generalized experimental framework, for testing niching methods to be agreed upon. This section will focus on that, and propose an experimental methodology for EA niching algorithms. It will present a suite of synthetic test functions, discuss possible performance criteria, and conclude by revisiting the niching-CMA algorithm and some of its experimental observations.

In the broad context of function optimization, EA niching techniques are obviously within the general framework of stochastic algorithms, and as such should be treated carefully upon reporting their experiments. We recommend following Bartz-Beielstein (2006) when conducting experiments, and especially following the 7-points scheme of Preuss (2007) when reporting them.

5.1 Multimodal Test Functions

The choice of a numerical testbed for evaluating the performance of search or optimization methods is certainly one of the core issues among the scholars in the community of algorithms and operations research.

In a benchmark article, Whitley et al. (1996) criticized the commonly tested artificial landscapes in the evolutionary algorithms community, and offered general guidelines for constructing test problems. A remarkable effort was made almost a decade after that document, when a large group of scholars in the EC community joined their efforts and compiled an agreed test suite of single-objective artificial landscapes (Suganthan et al. 2005), to be tested in an open performance competition reported at the 2005 IEEE Congress on Evolutionary Computation (CEC) (Auger and Hansen 2005b). The latter also included multimodal functions.

The issue of developing a multimodal test suite received even less attention, likely due to historical reasons. Since multimodal domains were mainly treated by GA-based niching methods, their corresponding test suites were limited to low-dimensional continuous landscapes, typically with two decision parameters to be optimized ($n = 2$) (see, e.g., Goldberg and Richardson 1987; Mahfoud 1995a).

When compiling our proposed test suite, we aim at following Whitley's guidelines, and to include some traditional GA-niching test functions as well as functions from the 2005 CEC inventory (Suganthan et al. 2005). Some of the landscapes have symmetric or equal distributions of minima, and some do not. Some of the functions are *separable*, that is, they can be optimized by solving n 1-dimensional problems separately (Whitley et al. 1996), while some of them are non-separable.

Table 1

Test functions to be *minimized* and initialization domains. For some of the non-separable functions, we apply translation and rotation: $\mathbf{y} = \mathcal{O}(\mathbf{x} - \mathbf{r})$ where \mathcal{O} is an orthogonal rotation matrix, and \mathbf{r} is a shifting vector

Separable:			
Name	Function	Init	Niches
\mathcal{M}	$\mathcal{M}(\mathbf{x}) = -\frac{1}{n} \sum_{i=1}^n \sin^\alpha(5\pi x_i)$	$[0, 1]^n$	100
\mathcal{A} [Ackley]	$\mathcal{A}(\mathbf{x}) = -c_1 \cdot \exp\left(-c_2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) \\ - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(c_3 x_i)\right) + c_1 + e$	$[-10, 10]^n$	$2n+1$
\mathcal{L}	$\mathcal{L}(\mathbf{x}) = -\prod_{i=1}^n \sin^k(l_1 \pi x_i + l_2) \cdot \exp\left(-l_3 \left(\frac{x_j - l_4}{l_5}\right)^2\right)$	$[0, 1]^n$	$n+1$
\mathcal{R} [Rastrigin]	$\mathcal{R}(\mathbf{x}) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i))$	$[-1, 5]^n$	$n+1$
\mathcal{G} [Griewank]	$\mathcal{G}(\mathbf{x}) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-10, 10]^n$	5
\mathcal{S} [Shekel]	$\mathcal{S}(\mathbf{x}) = -\sum_{i=1}^{10} \frac{1}{k_i(\mathbf{x}-\mathbf{a}_i)(\mathbf{x}-\mathbf{a}_i)^T + c_i}$	$[0, 10]^n$	8
\mathcal{V} [Vincent]	$\mathcal{V}(\mathbf{x}) = -\frac{1}{n} \sum_{i=1}^n \sin(10 \cdot \log(x_i))$	$[0.25, 10]^n$	50
Non-separable:			
Name	Function	Init	Niches
\mathcal{F} [Fletcher-Powell]	$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^n (A_i - B_i)^2 \\ A_i = \sum_{j=1}^n (a_{ij} \cdot \sin(\alpha_j) + b_{ij} \cdot \cos(\alpha_j)) \\ B_i = \sum_{j=1}^n (a_{ij} \cdot \sin(x_j) + b_{ij} \cdot \cos(x_j)) \\ a_{ij}, b_{ij} \in [-100, 100]; \quad \alpha \in [-\pi, \pi]^n$	$[-\pi, \pi]^n$	10
\mathcal{R}_{SR} [S.R. Rastrigin]	$\mathcal{R}_{SR}(\mathbf{x}) = 10n + \sum_{i=1}^n (y_i^2 - 10 \cos(2\pi y_i))$	$[-5, 5]^n$	$n+1$
\mathcal{G}_{SR} [S.R. Griewank]	$\mathcal{G}_{SR}(\mathbf{x}) = 1 + \sum_{i=1}^n \frac{y_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{y_i}{\sqrt{i}}\right)$	$[0, 600]^n$	5

We propose for consideration a set of multimodal test functions, as indicated in [Table 1](#). The table summarizes the proposed unconstrained multimodal test functions as well as their initialization intervals and the number of target niches. Next, we provide the reader with an elaborate description of the test functions, corresponding to the notation of [Table 1](#):

- \mathcal{M} is a basic hypergrid multimodal function with uniformly distributed minima of equal function value of -1 . It is meant to test the stability of a particularly large number of niches: in the interval $[0, 1]^n$ it has 5^n minima. We set $\alpha = 6$.
- The well-known Ackley function has one global minimum, regardless of its dimension n , which is surrounded isotropically by $2n$ local minima in the first hypersphere, followed by an exponentially increasing number of minima in successive hyperspheres. Ackley's function has been widely investigated in the context of *evolutionary computation* (see, e.g., Bäck [1996](#)). We set $c_1 = 20$, $c_2 = 0.2$, and $c_3 = 2\pi$.

- \mathcal{L} – also known as $F2$, as originally introduced by Goldberg and Richardson (1987) – is a sinusoid trapped in an exponential envelope. The parameter k determines the sharpness of the peaks in the function landscape; We set it to $k = 6$. \mathcal{L} has one global minimum, regardless of n and k . It has been a popular test function for GA niching methods. We set $l_1 = 5.1$, $l_2 = 0.5$, $l_3 = 4 \cdot \ln(2)$, $l_4 = 0.0667$ and $l_5 = 0.64$.
- The Rastrigin function (Törn and Zilinskas 1987) has one global minimum, surrounded by a large number of local minima arranged in a lattice configuration.

We also propose its shifted-rotated variant (Suganthan et al. 2005), with a linear transformation matrix of condition number 2 as the rotation operator.

- The Griewank function (Törn and Zilinskas 1987) has its global minimum ($f^* = 0$) at the origin, with several thousand local minima in the area of interest. There are four suboptimal minima $f \approx 0.0074$ with $\mathbf{x}^* \approx (\pm\pi, \pm\pi\sqrt{2}, 0, 0, 0, \dots, 0)$.

We also propose its shifted-rotated variant (Suganthan et al. 2005), with a linear transformation matrix of condition number 3 as the rotation operator.

- The Vincent function is a sine function with a decreasing frequency. It has 6^n global minima in the interval $[0.25, 10]^n$.
- The Shekel function, suggested by Törn and Zilinskas (1987), introduces a landscape with a dramatically uneven spread of minima. It has one global minimum, and seven ordered local minima.
- The function after Fletcher and Powell (Bäck 1996) is a non-separable *nonlinear parameter estimation problem*, which has a nonuniform distribution of 2^n minima. It has non-isotropic attractor basins.

5.2 Performance Criteria

The performance criteria of niching methods include numerous possible attractive options and, generally speaking, vary in different experimental reports. The common ground is typically the examination of the population in its final stage, while aiming to observe the algorithm's ability to capture as well as *Maintain* peaks of high-quality. Stability of good niches is thus considered as one of the implicit criteria.

Studies of traditional GA niching methods had been strongly interested in the distribution of the final population compared to a goal-distribution, as formalized by Mahfoud (1995a). While Mahfoud's formalism introduced a generic theoretical tool, being derived from information theory, other studies considered, *de facto*, specific performance calculations. For example, a very popular niching performance measurement, which satisfies Mahfoud formalism's criteria, is the *Chi-square-like performance statistic* (see, e.g., Deb and Goldberg 1989). The latter estimates the deviation of the actual distribution of individuals N_i from an ideal distribution (characterized by mean μ_i and variance σ_i^2) in all the $i = 1 \dots q + 1$ subspaces (q peak subspaces and the non-peak subspace):

$$\chi^2 = \sqrt{\sum_{i=1}^{q+1} \left(\frac{N_i - \mu_i}{\sigma_i} \right)^2} \quad (17)$$

where the ideal-distribution characteristic values are derived per function.

Most of the existing studies have focused on the ability to identify global as well as local optima, and to converge in these directions through time, with no particular interest in the

distribution of the population. One possible performance criterion is often defined as the *success-rate* of the niching process, which refers to the percentage of optima attained by the end of the run with respect to the target peaks, as defined *a priori*. Furthermore, as has been employed in earlier studies of GA niching (Miller and Shaw 1996), another performance criterion is called the *maximum peak ratio* (MPR) statistic. This metric measures the quality as well as the number of optima given as a final result by the evolutionary algorithm. Explicitly, assuming a *minimization problem*, given the fitness values of the subpopulations in the final population $\{\tilde{f}_i\}_{i=1}^q$, and the fitness values of the real optima of the objective function $\{\hat{\mathcal{F}}_i\}_{i=1}^q$, the *maximum peak ratio* is defined as follows:

$$\text{MPR} = \frac{\sum_{i=1}^q \hat{\mathcal{F}}_i}{\sum_{i=1}^q \tilde{f}_i} \quad (18)$$

where all values are assumed to be *strictly positive*. If this is not the case in the original parametrization of the landscape, the latter should be scaled accordingly with an additive constant for the sake of this calculation. Also, given a maximization problem, the MPR is defined as the sum of the obtained optima divided by the sum of the real optima. A drawback of this performance metric is that the real optima need to be known *a priori*. However, for many artificial test problems these can be derived analytically, or tight numerical approximations to them are available. We adopt the MPR performance criterion and recommend it for reporting experimental results of EA niching techniques.

5.2.1 Another Perspective: MPR Versus Time

Although the MPR metric was originally derived to be analyzed by means of its saturation value in order to examine the niches' stability, a new perspective was introduced by Shir and Bäck (2005a). That study investigated the MPR as a function of time, focusing on the early stages of the run, in addition to the saturation value. It was shown experimentally that the time-dependent MPR data fits a theoretical function, the *logistic curve*:

$$y(t) = \frac{a}{1 + \exp\{c(t - T)\}} \quad (19)$$

where a is the saturation value of the curve, T is its time shift, and c (in this context always negative) determines the shape of the exponential rise. This equation, known as the *logistic equation*, describes many processes in nature. All those processes share the same pattern of behavior: growth with *acceleration*, followed by *deceleration* and then a *saturation* phase. In the context of evolutionary niching methods, it was argued (Shir and Bäck 2005a) that the logistic parameters should be interpreted in the following way: T as the *learning period* of the algorithm, and the absolute value of c as its *niching formation acceleration*. a is clearly the MPR saturation value.

5.3 Experimental Observation Examples: Niching-CMA Revisited

We revisit the niching-CMA algorithm with three examples of experimental observations.

5.3.1 MPR Time-Dependent Analysis: Reported Observations

The MPR time-dependent analysis was applied in Shir and Bäck (2005a) to two ES-based niching techniques: niching with the CMA-ES and niching with the Standard-ES according to the Schwefel-approach (Shir and Bäck 2005b). In short, the latter method applies the same niching framework as the niching-CMA (Sect. 4.2.1) except for one conceptual difference: It employs a (μ, λ) strategy in each niche, subject to *restricted mating*. Otherwise, it applies the standard ES operators (Bäck 1996).

We outline some of the conclusions of that study:

1. The **niching formation acceleration**, expressed as the absolute value of c , had larger values for the CMA-ES kernel for all the observed test cases. That implied stronger niching acceleration and faster convergence.
2. A trend concerning the absolute value of c as a function of the dimensionality was observed: The higher the dimensionality of the search space, the lower the absolute value of c , that is, the slower the niching process.
3. The **learning period**, expressed as the value of T in the curve fitting, has negative as well as positive values. Negative values mean that the niches' formation process, expressed as the exponential rise of the MPR, started immediately from generation zero.
4. The averaged **saturation value a** , that is, the MPR saturation value, was larger in all of the test cases for the CMA-ES mechanism. In that respect, the CMA kernel outperformed the standard-ES on the tested landscapes.

That study concluded with the claim that there was a clear *trade-off*. Either a long learning period followed by a high niching acceleration (CMA-ES), or a short learning period followed by a low niching acceleration (Standard-ES). A hypothesis concerning the existence of a general trade-off between the learning period T and the niching acceleration, c , was numerically assessed in a following study (Shir and Bäck 2008). It was shown that this trade-off stands for various DES niching variants on two synthetic landscapes (one separable, one non-separable) in a large spectrum of search space dimensions.

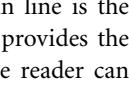
5.3.2 Performance on the Synthetic Multimodal Test-Suite

The proposed CMA-ES niching framework has been successfully applied to a suite of synthetic multimodal *high-dimensional* continuous landscapes, as reported by Shir and Bäck (2008), which in principle followed the recommended test functions proposed earlier in this chapter. That study addressed various *research questions* concerning the generalized DES niching framework, such as *which DES variant captures and maintains most desired optima* (i.e., best saturation MPR values), *what are the differences in the niching formation acceleration among the DES variants*, and others. The CMA-ES kernels were observed to perform very well among the DES variants, with the (1+10) kernel performing best, while typically obtaining most of the desired basins of attraction of the various landscapes at different dimensions. Furthermore, upon carrying out behavioral analysis of the simulations, e.g., MPR analysis, some characteristic patterns for the different algorithmic kernels were revealed. For instance, it was observed that the elitist-CMA has consistently the lowest niching acceleration. A straightforward and rather intuitive explanation for the excellent behavior of the elitist-CMA variant would be its tendency to maintain convergence in any basin of attraction, versus a higher

probability for the comma strategy to escape them. Moreover, another argument for the advantage of an elitist strategy for niching was suggested. The niching problem can be considered as an optimization task with constraints, that is, the formation of niches that restricts competing niches and their optimization routines from exploring the search space freely. It has been suggested in previous studies (see, e.g., Kramer and Schwefel 2006) that ES self-adaptation in constrained problems would tend to fail with a comma strategy, and thus an elitist strategy is preferable for such problems.

We choose to omit here specific details concerning the numerical simulations, and refer the reader to Shir and Bäck (2008) and Shir (2008).

5.3.3 Real-World Application: Quantum Control

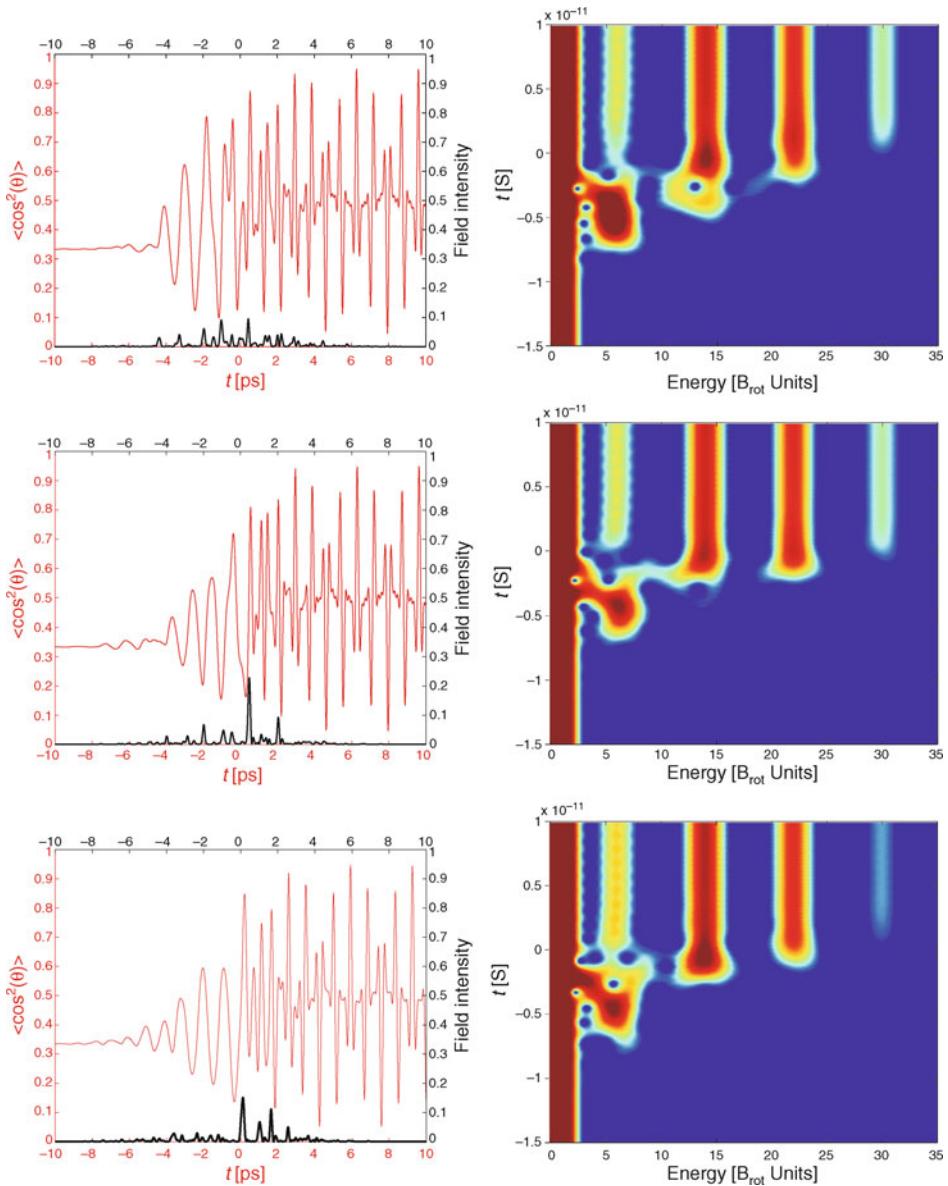
As was, furthermore, reported by Shir and Bäck (2008), the proposed DES niching framework was successfully applied to a real-world landscape from the field of Quantum Control. (Symbolically, this interdisciplinary study forms a *closed natural computing circle*, where biologically oriented investigation of organic evolution and speciation helps to develop methods for solving applications in physics in general, and in quantum control in particular. By our reckoning, this symbolism is even further strengthened upon considering the stochastic nature of evolutionary algorithms. This process can be thus considered as throwing dice in order to solve quantum mechanics, sometimes referred to as the *science of dice*.) Namely, the dynamic molecular alignment problem (for a review see Shir et al. 2008). In short, the goal of this application is the maximization of the alignment of diatomic molecules after the interaction with an electric field arising from a laser source. The black-box noise-free simulator, which is designed in a lab-oriented fashion, provides a reliable physics simulation with a duration of 35 s per trial solution. The 80 object variables constitute a parametrized curve, to be learned, that undergoes spectral transformation for the construction of the electric field. This challenging high-dimensional application required the definition of a tailor-made diversity measure, due to certain invariance properties of the control function that stem from the spectral transformations. The resulting metric employed was the Euclidean distance in the second derivative space of the control function. DES niching variants were shown to perform well, and to obtain different pulse shapes of high quality, representing different conceptual designs. Also in this case, the elitist-CMA kernel was observed to perform best. While referring the reader to Shir and Bäck (2008) and Shir (2008) for more details on this particular study, we would like to conclude with a visualization of the resulting niching process. Three laser-pulse niches, obtained in a typical elitist-CMA run, are plotted for illustration in  Fig. 1 (left column): The thick line describes the attained pulse shape, while the thin line is the molecular reaction, also referred to as the *revival structure*. The right column provides the equivalent quantum pictures, in terms of population of the energy levels: The reader can observe three different modes of quantum energy-level population for the different attained niches.

6 The Niche-Radius Problem

While the motivation and usefulness of niching cast no doubt, the relaxation of assumptions and limitations concerning the hypothetical landscape is much needed if niching methods are

Fig. 1

Experimental results for niching-CMA with a fixed niche-radius on the real-world problem of dynamic molecular alignment (quantum control). *Left column: alignment and revival-structure of the three niches obtained by the (1+10)-CMA. Thin red line: alignment; thick black line: intensity of the laser pulse.* *Right column: Quantum picture of the solutions; a Fourier transform applied to the revival structures of the optimal solutions (the thin red alignment curves).* The values are log scaled, and represent how high the rotational levels of the molecules are populated as a function of time. Note that the quality of the laser pulse cannot be measured in those plots.



to be valid in a broader range of applications. In particular, consider the assumptions made in [Sect. 4.3](#) concerning the fitness landscape, stating that the optima are far enough from one another with respect to the *niche radius*, which is estimated for the given problem and remains fixed during the course of evolution. Most prominently, the niche radius is used in the *sharing function*, which penalizes fitness values of individuals whose distance to other individuals is below that threshold ([Sect. 4.1.1](#)). Obviously, there are landscapes for which this assumption is not applicable, and where this approach is most likely to fail (see [Fig. 2](#) and [3](#) for illustrations). This topic is directly linked to the task of defining a generic basin of attraction, which was discussed in [Sect. 2.3](#).

6.1 Treating the Niche Radius Problem: Existing Work

There were several GA-oriented studies which addressed this so-called *niche radius problem*, aiming to relax the assumption specified earlier, or even to drop it completely. Jelasity (1998) suggested a cooling-based mechanism for the niche-radius, also known as the UEGO, which adapts the global radius as a function of time during the course of evolution. Gan and Warwick (2001) introduced the so-called dynamic niche clustering, to overcome the radius problem by using a clustering mechanism. A complex subpopulation differentiation model, the so-called multinational evolutionary algorithm, was presented by Ursem (1999). It introduces a topology-based auxiliary mechanism of sampling, which detects whether feasible solutions share the same basin of attraction. A recent study by Stoean et al. (2007) considered

Fig. 2

The Shekel function (see, e.g., Törn and Zilinskas 1987) in a 2D decision space: introducing a dramatically uneven spread of optima.

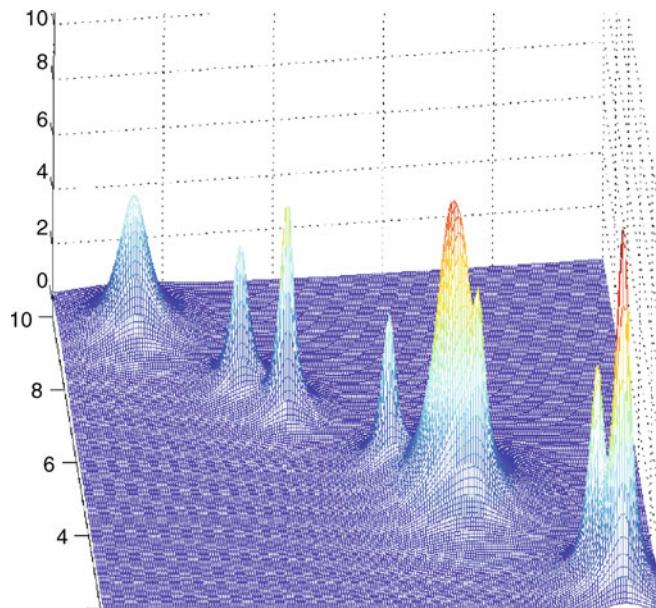
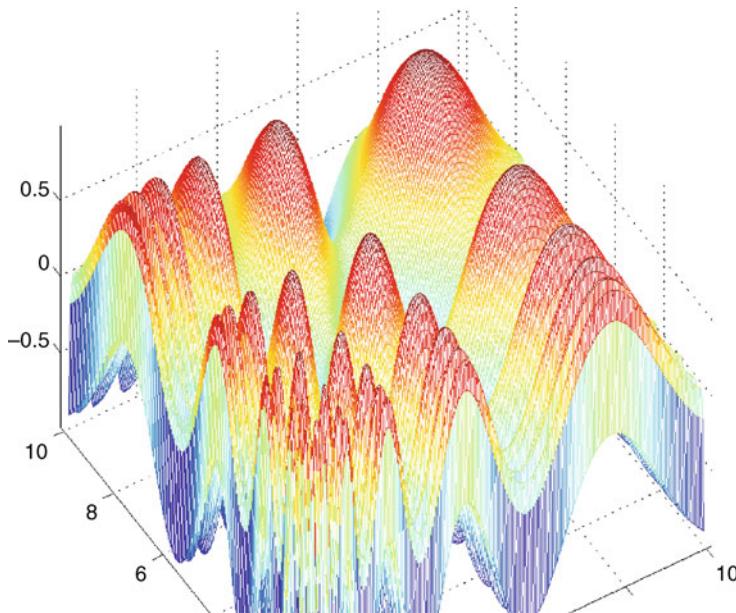


Fig. 3

The Vincent function in a 2D decision space: a sine function with a decreasing frequency.



the hybridization of the latter with a radius-based niching method proposed in Stoean et al. (2005). An iterative statistical-based approach was introduced (Cioppa et al. 2004) for learning the optimal niche radius, without *a priori* knowledge of the landscape. It considers the *fitness sharing* strategy, and optimizes it as a function of the population size and the niche radius, without relaxing the landscape assumption specified earlier – that is, the niches are eventually obtained using a single fixed niche radius. Finally, two ES-based niche-radius adaptation niching algorithms were proposed recently (Shir and Bäck 2006; van der Goes 2008). They both considered individual niche radii for the evolving population members, based upon original learning schemes. While Shir and Bäck (2006) relied on coupling the niche radius to the niche's global step size in combination with a secondary selection scheme, van der Goes et al. (2008) introduced a pure self-adaptive niche-radius approach, independent of any coupling to strategy parameters. The latter considered inner and outer niche counts, and applied the original sharing function for the sake of niches classification. Both approaches were shown to successfully tackle landscapes with challenging distributions as well as shapes of attraction basins.

6.2 Learning Niche Shapes: Exploiting CMA Information

As an extension to the previously discussed niching-CMA case study (Sect. 4.2.1), a self-adaptive niche-shaping approach was derived in a recent study (Shir et al. 2010). The latter introduced the *Mahalanobis distance metric* into the niching mechanism, aiming to allow a

more accurate spatial classification of niches by considering rotatable ellipsoid shapes as classification regions. The shapes of these ellipsoids were obtained from the covariance matrix of the evolving multivariate normal distribution adapted by the CMA mutation scheme, replacing the default classification by means of Euclidean hyperspheres. This approach was implemented in a straightforward manner using the Mahalanobis metric as a replacement for the Euclidean, due to the fact that the former metric, the hyperspheres of which are ellipsoids when viewed in the Euclidean space, is parametrized by a covariance matrix that is obtained without additional cost from the CMA. The proposed approach was tested on high-dimensional artificial landscapes at several levels of difficulty, and was shown to be robust and to achieve satisfying results.  *Figure 4* provides an illustration for the numerical results of this self-adaptive approach. It presents a snapshot gallery of the niching algorithm with the elitist-CMA kernel, employing the Mahalanobis distance, performing on the 2-dimensional Fletcher-Powell landscape.

7 Discussion and Outlook

Niching studies, following somehow various *mission statements*, introduce a large variety of approaches, some of which are more biologically inspired, whereas others are multimodal-optimization oriented. In both cases, those techniques were mainly tested on *low-dimensional synthetic landscapes*, and the application of these methods to real-world landscapes was hardly ever reported to date. We claim that niching methods should be implemented also for attaining multiple solutions in high-dimensional real-world problems, serving decision makers by providing them with the choice of optimal solutions, and representing well evolutionary algorithms in multimodal domains. By our reckoning, the *multimodal front* of real-world applications, that is, multimodal real-world problems, which pose the demand for multiple optimal solutions, should also enjoy the powerful capabilities of evolutionary algorithms, as other fronts do, for example, multi-objective or constrained domains.

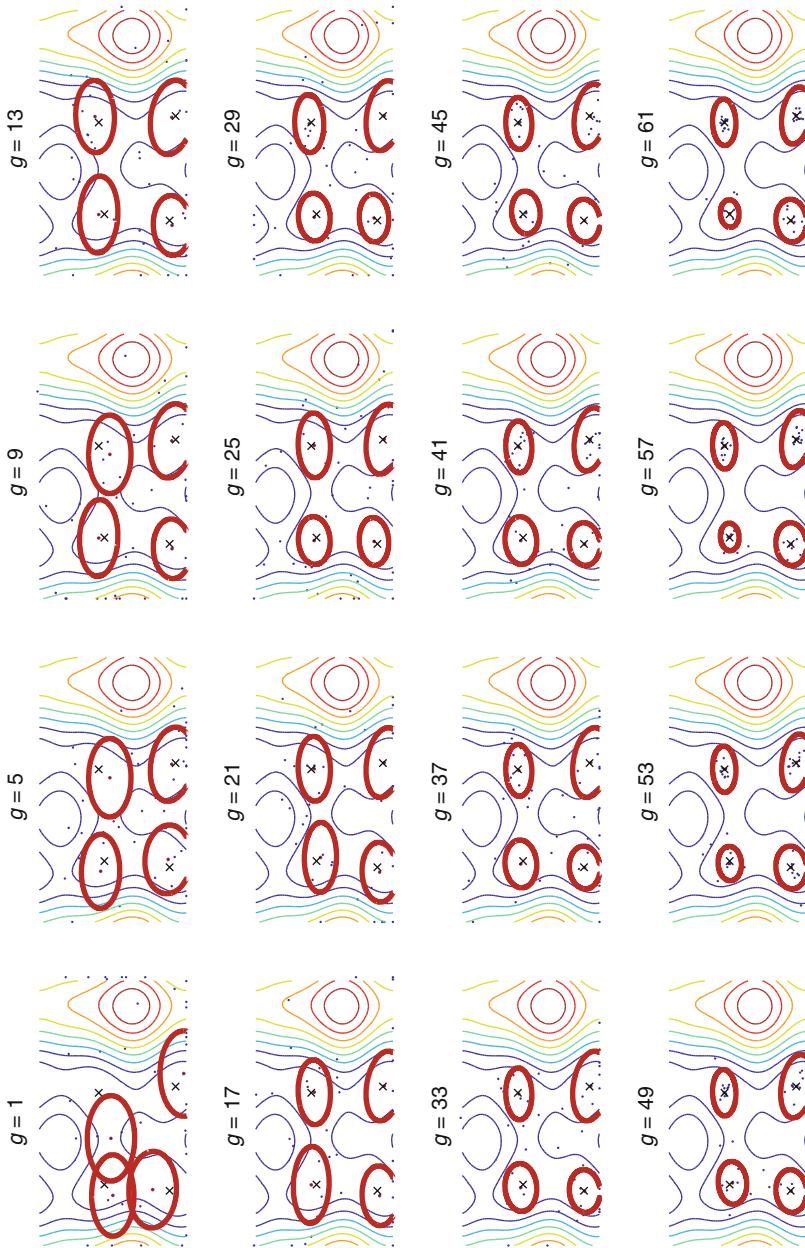
On a different note, Preuss, in an important paper (Preuss 2006), raised the question: “*Under what conditions can niching techniques be faster than iterated local search algorithms?*” Upon considering a simplified model, and assuming the existence of an efficient basin identification method, he managed to show that it does pay off to employ EA niching techniques on landscapes whose basins of attraction vary significantly in size. However, the original question in its general form remained open. Mahfoud (1995b) drew a comparison of *parallel* versus *sequential* niching methods, while considering *fitness sharing*, *deterministic crowding*, *sequential niching*, and *parallel hillclimbing*. Generally speaking, he concluded that parallel niching GAs outperform parallel hillclimbers on a hard set of problems, and that *sequential niching* is always outperformed by the parallel approaches.

Obviously, there is *no free lunch*, and there is no best technique, especially in the domain of multimodal search spaces. In this respect, *local search* capabilities should not be underestimated, and *population diversity preservers* should not be overestimated. We claim that like any other complex component in organic as well as artificial systems, the success of niching is about the subtle interplay between the different, sometime conflicting, driving effects.

In this chapter, we have introduced *niching* as an algorithmic framework following the evolutionary process of *speciation*. Upon providing the practical motivation for such a framework, in terms of conceptual designs for better decision making, we outlined in detail the essential biological background, as well as the computational perspective of multimodal function

Fig. 4

A snapshot gallery: the adaptation of the *classification-ellipses*, subject to the Mahalanobis metric with the updating covariance matrix, in the elitist-CMA strategy for a 2D Fletcher-Powell problem. Images are taken in the box $[-\pi, \pi]^2$. Contours of the landscape are given as the background, where the Xs indicate the real optima, the dots are the evolving individuals, and the ellipses are plotted centered about the peak individual. A snapshot is taken every four generations (i.e., every 160 function evaluations), as indicated by the counter.



optimization. This was followed by a survey of existing EA niching techniques, mainly from the GA field, and a detailed description of a specific case-study technique, based on the CMA-ES. We highlighted the *natural computing* aspects of these techniques, especially the biologically oriented components in each scheme, e.g., hosting capacity, gene flow, alpha-males, etc. We then proposed an experimental framework for testing niching methods, and presented some experimental observations of the CMA-ES case-study technique, including results from a real-world problem. We would like to use this opportunity to encourage other scholars to apply the proposed experimental framework, and at the same time to consider ways to join forces for the construction of a general framework to be agreed upon. Finally, we discussed the important topic of the niche-radius problem, gave an overview of existing methods to treat it, and revisited the CMA-ES case study for an extended self-adaptive scheme, which employs the Mahalanobis metric for obtaining niches with more complex geometrical shapes. We conclude that even though the assumptions made for radius-based niching techniques are problematic, there are good proposed solutions that treat the problem at different levels.

We would like to propose possible directions for future research in this domain:

- Transferring existing niching algorithms into additional real-world applications in general, and into experimental optimization in particular
- Proceeding with the effort to tackle the niche radius problem, in order to develop state-of-the-art niching techniques, which are not subject to the niche radius assumptions
- Extending the study of niching to environments with uncertainty
- Developing theoretical frameworks for the investigation of niching, for example, by means of simplified modeling

References

- Adamidis P (1994) Review of parallel genetic algorithms bibliography. Tech. rep., Automation and Robotics Lab., Dept. of Electrical and Computer Eng., Aristotle University of Thessaloniki, Greece
- Aichholzer O, Aurenhammer F, Brandstätter B, Ebner T, Krasser H, Magele C (2000) Niching evolution strategy with cluster algorithms. In: Proceedings of the 9th biennial IEEE conference on electromagnetic field computations. IEEE Press, New York
- Ando S, Sakuma J, Kobayashi S (2005) Adaptive isolation model using data clustering for multimodal function optimization. In: Proceedings of the 2005 conference on genetic and evolutionary computation, GECCO 2005. ACM, New York, pp 1417–1424
- Angus D (2006) Niching for population-based ant colony optimization. In: Second international conference on e-science and grid technologies (e-science 2006), December 4–6, 2006, Amsterdam, The Netherlands, IEEE Computer Society, p 115
- Auger A, Hansen N (2005a) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 congress on evolutionary computation CEC 2005. IEEE Press, Piscataway, NJ, pp 1769–1776
- Auger A, Hansen N (2005b) Performance evaluation of an advanced local search evolutionary algorithm. In: Proceedings of the 2005 congress on evolutionary computation CEC 2005. IEEE Press, Piscataway, NJ, pp 1777–1784
- Avigad G, Moshaiov A, Brauner N (2004) Concept-based interactive brainstorming in engineering design. *J Adv Comput Intell Intell Informatics* 8(5): 454–459
- Avigad G, Moshaiov A, Brauner N (2005) Interactive concept-based search using MOEA: the hierarchical preferences case. *Int J Comput Intell* 2 (3):182–191
- Bäck T (1994) Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: Michalewicz Z, Schaffer JD, Schwefel HP, Fogel DB, Kitano H (eds) Proceedings of the first IEEE conference on evolutionary computation (ICEC'94). Orlando FL. IEEE Press, Piscataway, NJ, pp 57–62
- Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, New York

- Bartz-Beielstein T (2006) Experimental research in evolutionary computation – the new experimentalism. Natural computing series. Springer, Berlin
- Beasley D, Bull DR, Martin RR (1993) A sequential niche technique for multimodal function optimization. *Evolut Comput* 1(2):101–125
- Beyer HG (1999) On the dynamics of GAs without selection. In: Banzhaf W, Reeves C (eds) Foundations of genetic algorithms 5. Morgan Kaufmann, San Francisco, CA, pp 5–26
- Beyer HG, Schwefel HP (2002) Evolution strategies a comprehensive introduction. *Nat Comput Int J* 1(1):3–52
- Beyer HG, Brucherseifer E, Jakob W, Pohlheim H, Sendhoff B, To TB (2002) Evolutionary algorithms – terms and definitions. <http://ls11-www.cs.uni-dortmund.de/people/beyer/EA-glossary/>
- Blum C (2005) Ant colony optimization: introduction and recent trends. *Phys Life Rev* 2:353–373
- Bradshaw A (1965) Evolutionary significance of phenotypic plasticity in plants. *Adv Genet* 13:115–155
- Branke J (2001) Evolutionary optimization in dynamic environments. Kluwer, Norwell, MA
- Brits R, Engelbrecht AP, Bergh FVD (2002) A niching particle swarm optimizer. In: The fourth Asia-Pacific conference on simulated evolution and learning (SEAL2002). Singapore, pp 692–696
- Cavicchio D (1970) Adaptive search using simulated evolution. Ph.D. thesis, University of Michigan, Ann Arbor, MI
- Cioppa AD, Stefano CD, Marcelli A (2004) On the role of population size and niche radius in fitness sharing. *IEEE Trans Evolut Comput* 8(6):580–592
- Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems. Springer, Berlin
- Coello Coello CA (1999) A survey of constraint handling techniques used with evolutionary algorithms. Tech. Rep. Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada. Xalapa, Veracruz, México
- Cristiano JJ, White CC, Liker JK (2001) Application of multiattribute decision analysis to quality function deployment for target setting. *IEEE Trans Syst Man Cybern Part C* 31(3):366–382
- Darwin CR (1999) The origin of species: by means of natural selection or the preservation of favoured races in the struggle for life. Bantam Classics, New York
- De Jong KA (1975) An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
- Deb K, Goldberg DE (1989) An investigation of niche and species formation in genetic function optimization. In: Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 42–50
- Deb K, Spears WM (1997) Speciation methods. In: Bäck T, Fogel D, Michalewicz Z (eds) The handbook of evolutionary computation. IOP Publishing and Oxford University Press, Bristol
- Deb K, Tiwari S (2005) Omni-optimizer: a procedure for single and multi-objective optimization. In: Evolutionary multi-criterion optimization, third international conference, EMO 2005, Lecture notes in computer science, vol 3410. Springer, Guanajuato, Mexico, pp 47–61
- Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge, MA
- Doye J, Leary R, Locatelli M, Schoen F (2004) Global optimization of Morse clusters by potential energy transformations. *INFORMS J Comput* 16(4): 371–379
- Engelbrecht A (2005) Fundamentals of computational swarm intelligence. New York
- Fisher RA (1922) Darwinian evolution of mutations. *Eugen Rev* 14:31–34
- Fogel LJ (1966) Artificial intelligence through simulated evolution. Wiley, New York
- Freeman S, Herron JC (2003) Evolutionary analysis. Benjamin Cummings, 3rd edn. Redwood City, CA
- Gan J, Warwick K (2001) Dynamic niche clustering: a fuzzy variable radius niching technique for multimodal optimisation in GAs. In: Proceedings of the 2001 congress on evolutionary computation CEC2001, IEEE Press, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, pp 215–222
- van der Goes V, Shir OM, Bäck T (2008) Niche radius adaptation with asymmetric sharing. In: Parallel problem solving from nature – PPSN X, Lecture notes in computer science, vol 5199. Springer, pp 195–204
- Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
- Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the second international conference on genetic algorithms and their application. Lawrence Erlbaum, Mahwah, NJ, pp 41–49
- Grosso PB (1985) Computer simulations of genetic adaptation: parallel subcomponent interaction in a multilocus model. Ph.D. thesis, University of Michigan, Ann Arbor, MI
- Hanagandi V, Nikolaou M (1998) A hybrid approach to global optimization using a clustering algorithm in a genetic search framework. *Comput Chem Eng* 22(12):1913–1925

- Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evolut Comput* 9(2):159–195
- Hansen N, Ostermeier A, Gawelczyk A (1995) On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation. In: Proceedings of the sixth international conference on genetic algorithms (ICGA6). Morgan Kaufmann, San Francisco, CA, pp 57–64
- Haykin S (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, NJ, USA
- Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, MI
- Igel C, Suttorp T, Hansen N (2006) A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2006. ACM, New York, pp 453–460
- Jelasity M (1998) UEGO, an abstract niching technique for global optimization. In: Parallel problem solving from nature - PPSN V, Lecture notes in computer science, vol 1498. Springer, Amsterdam, pp 378–387
- Kennedy J, Eberhart R (2001) Swarm intelligence. Morgan Kaufmann, San Francisco, CA
- Kimura M (1983) The neutral theory of molecular evolution. Cambridge University Press, Cambridge
- Kramer O, Schwefel HP (2006) On three new approaches to handle constraints within evolution strategies. *Nat Comput Int* 5(4):363–385
- Li R, Eggemont J, Shir OM, Emmerich M, Bäck T, Dijkstra J, Reiber J (2008) Mixed-integer evolution strategies with dynamic niching. In: Parallel problem solving from nature - PPSN X, Lecture notes in computer science, vol 5199. Springer, pp 246–255
- Lunacek M, Whitley D (2006) The dispersion metric and the CMA evolution strategy. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2006. ACM, New York, pp 477–484
- Lunacek M, Whitley D, Sutton A (2008) The impact of global structure on search. In: Parallel problem solving from nature - PPSN X, Lecture notes in computer science, vol 5199. Springer, pp 498–507
- Mahfoud SW (1995a) Niching methods for genetic algorithms. Ph.D. thesis, University of Illinois at Urbana Champaign, IL
- Mahfoud SW (1995b) A comparison of parallel and sequential niching methods. In: Eshelman L (ed) Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 136–143
- Martin W, Lienig J, Cohoon J (1997) Island (migration) models: evolutionary algorithms based on punctuated equilibria. In: Bäck T, Fogel DB, Michalewicz Z (eds) *Handbook of evolutionary computation*. Oxford University Press, New York, and Institute of Physics, Bristol, pp C6.3:1–16
- McPheron BA, Smith DC, Berlocher SH (1988) Genetic differences between host races of *Rhagoletis pomonella*. *Nature* 336:64–66
- Miller B, Shaw M (1996) Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: Proceedings of the 1996 IEEE international conference on evolutionary computation (ICEC'96). New York, pp 786–791
- Ostermeier A, Gawelczyk A, Hansen N (1993) A derandomized approach to self adaptation of evolution strategies. Tech. rep., TU Berlin
- Ostermeier A, Gawelczyk A, Hansen N (1994) Step-size adaptation based on non-local use of selection information. In: Parallel problem solving from nature - PPSN III, Lecture notes in computer science, vol 866. Springer, Berlin, pp 189–198
- Parrott D, Li X (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *Evolut Comput, IEEE Trans* 10(4): 440–458, doi: 10.1109/TEVC.2005.859468
- Petrowski A (1996) A clearing procedure as a niching method for genetic algorithms. In: Proceedings of the 1996 IEEE international conference on evolutionary computation (ICEC'96). New York, pp 798–803
- Preuss M (2006) Niching prospects. In: Proceedings of the international conference on bioinspired optimization methods and their applications, BIOMA 2006. Jožef Stefan Institute, Slovenia, pp 25–34
- Preuss M (2007) Reporting on experiments in evolutionary computation. Tech. Rep. CI-221/07, University of Dortmund, SFB 531
- Ramalhinho-Lourenco H, Martin OC, Stützle T (2000) Iterated local search. Economics Working Papers 513, Department of Economics and Business, Universitat Pompeu Fabra
- Scheiner SM, Goodnight CJ (1984) The comparison of phenotypic plasticity and genetic variation in populations of the grass *Danthonia spicata*. *Evolution* 38(4):845–855
- Schönemann L, Emmerich M, Preuss M (2004) On the extinction of sub-populations on multimodal landscapes. In: Proceedings of the international conference on bioinspired optimization methods and their applications, BIOMA 2004. Jožef Stefan Institute, Slovenia, pp 31–40
- Shir OM (2008) Niching in derandomized evolution strategies and its applications in quantum control. Ph.D. thesis, Leiden University, The Netherlands
- Shir OM, Bäck T (2005a) Dynamic niching in evolution strategies with covariance matrix adaptation. In: Proceedings of the 2005 congress on evolutionary computation CEC-2005. IEEE Press, Piscataway, NJ, pp 2584–2591

- Shir OM, Bäck T (2005b) Niching in evolution strategies. Tech. Rep. TR-2005-01, LIACS, Leiden University
- Shir OM, Bäck T (2006) Niche radius adaptation in the CMA-ES niching algorithm. In: Parallel problem solving from nature - PPSN IX, Lecture notes in computer science, vol 4193. Springer, pp 142–151
- Shir OM, Bäck T (2008) Niching with derandomized evolution strategies in artificial and real-world landscapes. *Nat Comput Int J* (2008), doi: 10.1007/s11047-007-9065-5
- Shir OM, Beltrani V, Bäck T, Rabitz H, Vrakking MJ (2008) On the diversity of multiple optimal controls for quantum systems. *J Phys B At Mol Opt Phys* 41(7):(2008). doi: 10.1088/0953-4075/41/7/074021
- Shir OM, Emmerich M, Bäck T (2010) Adaptive niche-radii and niche-shapes approaches for niching with the CMA-ES. *Evolut Comput* 18(1):97–126. doi: 10.1162/evco.2010.18.1.18104
- Singh G, Deb K (2006) Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proceedings of the 2006 annual conference on genetic and evolutionary computation, GECCO 2006. ACM Press, New York, pp 1305–1312
- Smith RE, Bonacina C (2003) Mating restriction and niching pressure: results from agents and implications for general EC. In: Proceedings of the 2003 conference on genetic and evolutionary computation, GECCO 2003, Lecture notes on computer science, vol 2724. Springer, Chicago, IL, pp 1382–1393
- Spears WM (1994) Simple subpopulation schemes. In: Proceedings of the 3rd annual conference on evolutionary programming, World Scientific. San Diego, CA, Singapore, pp 296–307
- Stoean C, Preuss M, Gorunescu R, Dumitrescu D (2005) Elitist generational genetic chromodynamics – a new radii-based evolutionary algorithm for multi-modal optimization. In: Proceedings of the 2005 congress on evolutionary computation (CEC'05). IEEE Press, Piscataway NJ, pp 1839–1846
- Stoean C, Preuss M, Stoean R, Dumitrescu D (2007) Disburdening the species conservation evolutionary algorithm of arguing with radii. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2007. ACM Press, New York, pp 1420–1427
- Streichert F, Stein G, Ulmer H, Zell A (2003) A clustering based niching EA for multimodal search spaces. In: Proceedings of the international conference évolution artificielle, Lecture notes in computer science, vol 2936. Springer, Heidelberg, Berlin, pp 293–304
- Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University, Singapore
- Törn A, Zilinskas A (1987) Global optimization, Lecture notes in computer science, vol 350. Springer, Berlin
- Tsui K (1992) An overview of Taguchi method and newly developed statistical methods for robust design. *IIE Trans* 24:44–57
- Ursem RK (1999) Multinational evolutionary algorithms. In: Proceedings of the 1999 congress on evolutionary computation (CEC 1999). IEEE Press, Piscataway NJ, pp 1633–1640
- Whitley D, Mathias KE, Rana SB, Dzubera J (1996) Evaluating evolutionary algorithms. *Artif Intell* 85(1–2):245–276
- Wright S (1931) Evolution in Mendelian populations. *Genetics* 16:97–159
- Yin X, Germany N (1993) A fast genetic algorithm with sharing using cluster analysis methods in multimodal function optimization. In: Proceedings of the international conference on artificial neural nets and genetic algorithms, Innsbruck. Austria, 1993, Springer, pp 450–457

Section IV

Molecular Computation

Lila Kari

33 DNA Computing — Foundations and Implications*

Lila Kari¹ · Shinnosuke Seki² · Petr Sosík^{3,4}

¹Department of Computer Science, University of Western Ontario,
London, Canada

lila@csd.uwo.ca

²Department of Computer Science, University of Western Ontario,
London, Canada

sseki@csd.uwo.ca

³Institute of Computer Science, Silesian University in Opava,
Czech Republic

petr.sosik@fpf.slu.cz

⁴Departamento de Inteligencia Artificial, Universidad Politécnica de
Madrid, Spain

1	<i>Introduction</i>	1074
2	<i>A Computer Scientist's Guide to DNA</i>	1075
3	<i>The First DNA Computing Experiment</i>	1084
4	<i>Beyond Unaided Human Computation</i>	1086
5	<i>DNA Complementarity and Formal Language Theory</i>	1088
6	<i>Vectorial Models of DNA-Based Information</i>	1104
7	<i>DNA Complementarity and Combinatorics on Words</i>	1118
8	<i>Conclusions</i>	1123

* This work was supported by The Natural Sciences and Engineering Council of Canada Discovery Grant and Canada Research Chair Award to L.K.

Abstract

DNA computing is an area of natural computing based on the idea that molecular biology processes can be used to perform arithmetic and logic operations on information encoded as DNA strands. The first part of this review outlines basic molecular biology notions necessary for understanding DNA computing, recounts the first experimental demonstration of DNA computing (Adleman's 7-vertex Hamiltonian Path Problem), and recounts the milestone wet laboratory experiment that first demonstrated the potential of DNA computing to outperform the computational ability of an unaided human (20 variable instance of 3-SAT).

The second part of the review describes how the properties of DNA-based information, and in particular the Watson–Crick complementarity of DNA single strands, have influenced areas of theoretical computer science such as formal language theory, coding theory, automata theory and combinatorics on words. More precisely, we describe the problem of DNA encodings design, present an analysis of intramolecular bonds, define and characterize languages that avoid certain undesirable intermolecular bonds, and investigate languages whose words avoid even imperfect bindings between their constituent strands. We also present another, vectorial, representation of DNA strands, and two computational models based on this representation: sticker systems and Watson–Crick automata. Lastly, we describe the influence that properties of DNA-based information have had on research in combinatorics on words, by enumerating several natural generalizations of classical concepts of combinatorics of words: pseudopalindromes, pseudoperiodicity, Watson–Crick conjugate and commutative words, involutively bordered words, pseudoknot bordered words. In addition, we outline natural extensions in this context of two of the most fundamental results in combinatorics of words, namely Fine and Wilf's theorem and the Lyndon–Schutzenberger result.

1 Introduction

DNA computing is an area of natural computing based on the idea that molecular biology processes can be used to perform arithmetic and logic operations on information encoded as DNA strands. The aim of this review is twofold. First, the fundamentals of DNA computing, including basics of DNA structure and bio-operations, and two historically important DNA computing experiments, are introduced. Second, some of the ways in which DNA computing research has impacted fields in theoretical computer science are described.

The first part of this chapter outlines basic molecular biology notions necessary for understanding DNA computing (► Sect. 2), recounts the first experimental demonstration of DNA computing (► Sect. 3), as well as the milestone wet laboratory experiment that first demonstrated the potential of DNA computing to outperform the computational ability of an unaided human (► Sect. 4).

The second part of the chapter describes how the properties of DNA-based information, and, in particular, the Watson–Crick (WK) complementarity of DNA single strands have influenced areas of theoretical computer science such as formal language theory, coding theory, automata theory, and combinatorics on words. More precisely, ► Sect. 5 summarizes notions and results in formal language theory and coding theory arising from the problem of design of optimal encodings for DNA computing experiments: ► Section 5.1 describes the problem of DNA encodings design, ► Sect. 5.2 consists of an analysis of intramolecular bonds

(bonds within a given DNA strand), [Sect. 5.3](#) defines and characterizes languages that avoid certain undesirable intermolecular bonds (bonds between two or more DNA strands), and [Sect. 5.4](#) investigates languages whose words avoid even imperfect bindings between their constituent strands.

[Sect. 6](#) contains another representation (vectorial) of DNA strands and two computational models based on this representation: sticker systems and Watson–Crick automata. After a brief description of the representation of DNA partial double strands as two-line vectors, and of the sticking operation that combines them, [Sect. 6.1](#) describes basic sticker systems, sticker systems with complex structures ([Sect. 6.1.1](#)), and observable sticker systems ([Sect. 6.1.2](#)). [Section 6.2](#) investigates the accepting counterpart of the generative sticker systems devices: Watson–Crick automata and their properties.

[Section 7](#) describes the influence that properties of DNA-based information have had on research in combinatorics on words, by enumerating several natural generalizations of classical concepts of combinatorics of words: pseudo-palindromes, pseudo-periodicity, Watson–Crick conjugate and commutative words, involutively bordered words, pseudoknot bordered words. In addition, this section outlines natural extensions in this context of two of the most fundamental results in combinatorics of words, namely Fine and Wilf’s theorem and the Lyndon–Schützenberger result.

[Section 8](#) presents general thoughts on DNA-based information, bioinformation, and biocomputation.

2 A Computer Scientist’s Guide to DNA

In this section, a brief description of the basic molecular biology notions of DNA structure and DNA-based bio-operations used in DNA computing is given. For further details, the reader is referred to Turner et al. (2000); Drlica (1996); Calladine and Drew (1997); Gonick and Wheelis (1991); and Lewin (2007).

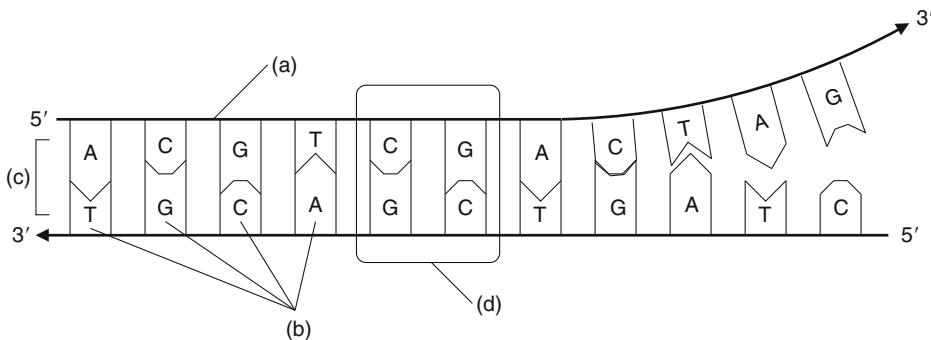
A **DNA (deoxyribonucleic acid)** molecule is a linear polymer. The monomer units of DNA are nucleotides (abbreviated *nt*), and the polymer is known as a “polynucleotide.” There are four different kinds of nucleotides found in DNA, each consisting of a nitrogenous *base* (Adenine, *A*, Cytosine, *C*, Guanine, *G*, or Thymine, *T*), and a sugar-phosphate unit. The abbreviation *N* stands for any nucleotide. The bases are relatively flat molecules and can be divided into purine bases (adenine and guanine) that have two carbon-containing rings in their structure, and smaller pyrimidine bases (cytosine and thymine) that have one carbon-containing ring in their structure.

A sugar-phosphate unit consists of a deoxyribose sugar and one to three phosphate groups. Together, the base and the sugar comprise a *nucleoside*. The sugar-phosphate units are linked together by strong *covalent bonds* to form the backbone of the DNA single strand (ssDNA). A DNA strand consisting of *n* nucleotides is sometimes called an *n-mer*. An *oligonucleotide* is a short DNA single strand, with 20 or fewer nucleotides. Since nucleotides may differ only by their bases, a DNA strand can be viewed simply as a word over the four-letter alphabet $\{A, C, G, T\}$.

A DNA single strand has an orientation, with one end known as the 5' end, and the other end known as the 3' end, based on their chemical properties. By convention, a word over the DNA alphabet represents the corresponding DNA single strand in the 5' to 3' orientation, that is, the word ACGTCGACTAC stands for the DNA single strand 5'-ACGTCGACTAC-3'.

Fig. 1

DNA structure: (a) DNA's sugar-phosphate backbone, (b) DNA bases, (c) Watson–Crick (WK) complementarity between bases A and T of two DNA single strands of opposite orientation, (d) Watson–Crick complementarity between bases C and G of two DNA single strands of opposite orientation.



A crucial feature of DNA single strands is their **Watson–Crick (WK) complementarity**: A (a purine) is complementary to T (a pyrimidine), and G (a purine) is complementary to C (a pyrimidine). Two complementary DNA single strands with opposite orientation bind to each other by weak *hydrogen bonds* between their individual bases as follows: A binds with T through two hydrogen bonds, while G binds with C through three hydrogen bonds. Thus, two Watson–Crick complementary single strands form a stable DNA double strand (dsDNA) resembling a helical ladder, with the two backbones at the outside, and the bases paired by hydrogen bonding and stacked on each other, on the inside. For example, the DNA single strand 5'-ACGTCGACTAC- 3' will bind to the DNA single strand 5'-GTAGTCGACGT-3' to form the 11 base-pair long (abbreviated as 11bp) double strand

$$5' - \text{ACGTCGACTAC} - 3'$$

$$3' - \text{TGCAGCTGATG} - 5'.$$

❷ *Figure 1* schematically illustrates this DNA double strand, omitting the double helix structure, for clarity. If v denotes a DNA single strand over the alphabet $\{A, C, G, T\}$, then by \overline{v} we will denote its Watson–Crick complement.

Another molecule that can be used for computation is *RNA, ribonucleic acid*. RNA is similar to DNA, but differs from it in three main aspects: RNA is usually single-stranded while DNA is usually double-stranded, RNA nucleotides contain the sugar ribose, while DNA nucleotides contain the sugar deoxyribose, and in RNA the base Uracil, U, substitutes for thymine, T, which is present in DNA.

The *genome* of an organism is the totality of its genetic information encoded in DNA. It consists of *chromosomes*, which, in turn, consist of genes. A *gene* is a segment of DNA that holds the information encoding a coherent set of functions necessary to build and maintain cells, and pass genetic traits to offspring. A gene comprises *coding* subsequences (*exons*) that determine what the gene does, and *noncoding* subsequences (*introns*). When a gene is active, the coding and noncoding sequences are copied in a process called *transcription*, producing an RNA copy of the gene's information. This piece of RNA can then direct the *translation* of the catenation of the coding sequences of this gene into *proteins* via the *genetic code*. The genetic

code maps each 3-letter RNA segment (called *codon*) into an amino acid. Several designated triplets, the start codon (*AUG*), and the stop (*UAA*, *UAG*, *UGA*) codons, signal the initiation, respectively, the termination of a translation. There are 20 different standard amino acids. Some of them are encoded by one codon, while others are encoded by several “synonymous” codons. A protein is a sequence over the 20-letter alphabet of amino acids. Proteins are essential parts of organisms and participate in every process within cells having, for example, catalytical, structural, or mechanical functions.

To encode, for example, English text using DNA, one can choose an encoding scheme mapping the Latin alphabet onto strings over $\{A, C, G, T\}$, and proceed to synthesize the obtained information-encoding strings as DNA single strands. In a hypothetical example, one could encode the letters of the English alphabet as $A \rightarrow ACA$, $B \rightarrow ACCA$, $C \rightarrow ACCCA$, $D \rightarrow AC^4A$, etc., wherein the i th letter of the alphabet is represented by AC^iA , that is, a single strand of DNA consisting of i repetitions of C flanked by one A at the beginning and another A at the end. Under this encoding, the text “To be or not to be” becomes the DNA single strand represented by



that can be readily synthesized.

Indeed, **DNA synthesis** is the most basic **bio-operation** used in DNA computing. DNA solid-state synthesis is based on a method by which the initial nucleotide is bound to a solid support, and successive nucleotides are added step by step, from the 3' to the 5' direction, in a reactant solution (☞ Fig. 2).

While the above encoding example is purely hypothetical, DNA strands of lengths of up to 100 nucleotides can be readily synthesized using fully automated *DNA synthesizers*. The result is a small test tube containing a tiny, dry, white mass of indefinite shape containing a homogeneous population of DNA strands that may contain 10^{18} identical molecules of DNA. In bigger quantities, dry DNA resembles tangled, matted white thread.

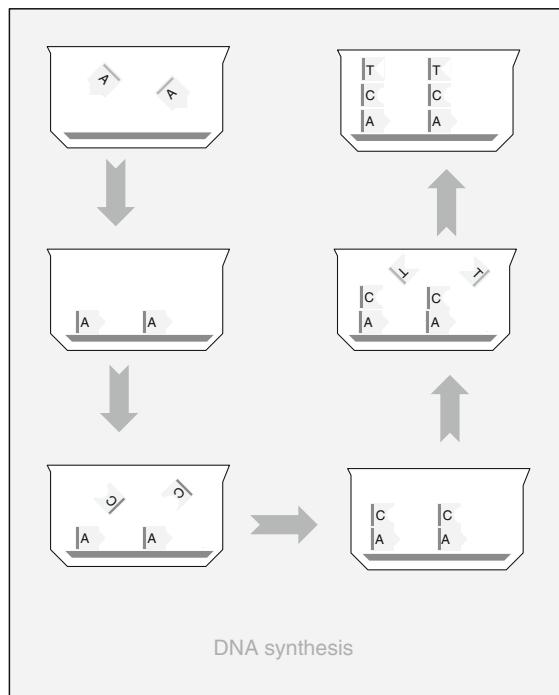
Using this or other DNA synthesis methods, one can envisage encoding any kind of information as DNA strands. There are several reasons to consider such a DNA-based memory as an alternative to all the currently available implementations of memories. The first is the extraordinary information-encoding density that can be achieved by using DNA strands. According to Reif et al. (2002), 1 g of DNA, which contains 2.1×10^{21} DNA nucleotides, can store approximately 4.2×10^{21} bits. Thus, DNA has the potential to store data on the order of 10^{10} more compactly than conventional storage technologies. In addition, the robustness of DNA data ensures the maintenance of the archived information over extensive periods of time (Bancroft et al. 2001; Cox 2001; Smith et al. 2003).

For the purposes of DNA computing, after encoding the input data of a problem on DNA strands, DNA bio-operations can be utilized for computations, see Kari (1997), Păun et al. (1998), and Amos (2005). The bio-operations most commonly used to control DNA computations and DNA robotic operations are described below.

DNA single strands with opposite orientation will join together to form a double helix in a process based on the Watson–Crick complementarity and called **base-pairing** (**annealing**, **hybridization**, **renaturation**), illustrated in ☞ Fig. 3. The reverse process – a double-stranded helix coming apart to yield its two constituent single strands – is called **melting** or **denaturation**. As the name suggests, melting is achieved by raising the temperature, and annealing by lowering it. Each DNA double-strand denaturates at a specific temperature, called its *melting*

Fig. 2

DNA solid-state synthesis. The initial nucleotide is bound to a solid support, and successive nucleotides are added step by step, from the 3' to the 5' direction, in a reactant solution. (From Kari 1997.)



temperature. The melting temperature is defined as the temperature at which half of the DNA strands are double-stranded, and half are single-stranded. It depends on both the length of the DNA sequence, and its specific base-composition (SantaLucia 1998).

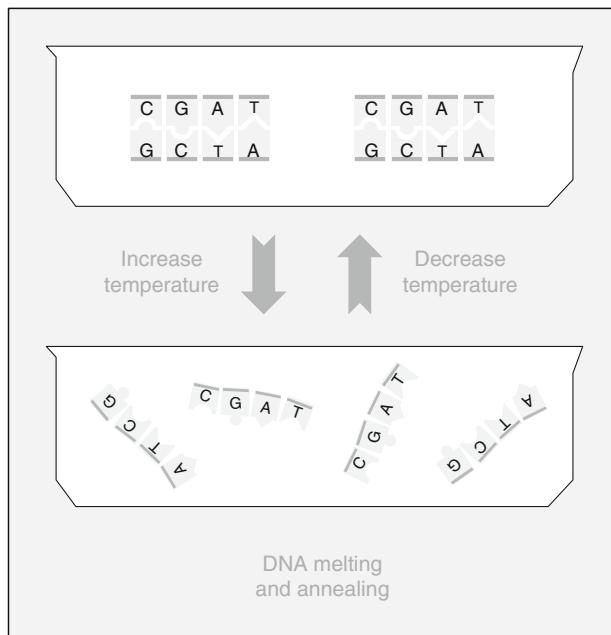
Cutting DNA double strands at specific sites can be accomplished with the aid of specific enzymes, called *restriction enzymes* (*restriction endonucleases*). Each restriction enzyme recognizes a specific short sequence of DNA, known as a *restriction site*. Any double-stranded DNA that contains the restriction site within its sequence is cut by the enzyme at that location, according to a specific pattern. Depending on the enzyme, the cutting operation leaves either two “blunt-ended” DNA double strands or, more often, two DNA strands that are double-stranded but have single-stranded overhangs known as “sticky-ends,” (Fig. 4). Hundreds of restriction enzymes are now known, and a large number are commercially available. They usually recognize sites ranging in size from 4 to 8 bp, the recognition sites are often pseudo-palindromic (see Sect. 7 for a definition of pseudopalindrome).

Another enzyme, called *DNA ligase*, can repair breaks in a double-stranded DNA backbone, and can covalently rejoin annealed complementary ends in the reverse of a restriction enzyme reaction, to create new DNA molecules. The process of thus pasting together compatible DNA strands is called **ligation**.

Separation of DNA strands by size is possible by using a technique called *gel electrophoresis*. The DNA molecules, which are negatively charged, are placed in “wells” situated at one

Fig. 3

Melting (separating DNA double strands into their constituent single strands), and annealing (the reformation of double-stranded DNA from thermally denatured DNA). Raising the temperature causes a DNA double strand to “melt” into its constituent Watson–Crick complementary single strands. Decreasing the temperature has the opposite effect: two DNA single strands that are Watson–Crick complementary will bind to each other to form a DNA double strand. (From Kari 1997.)



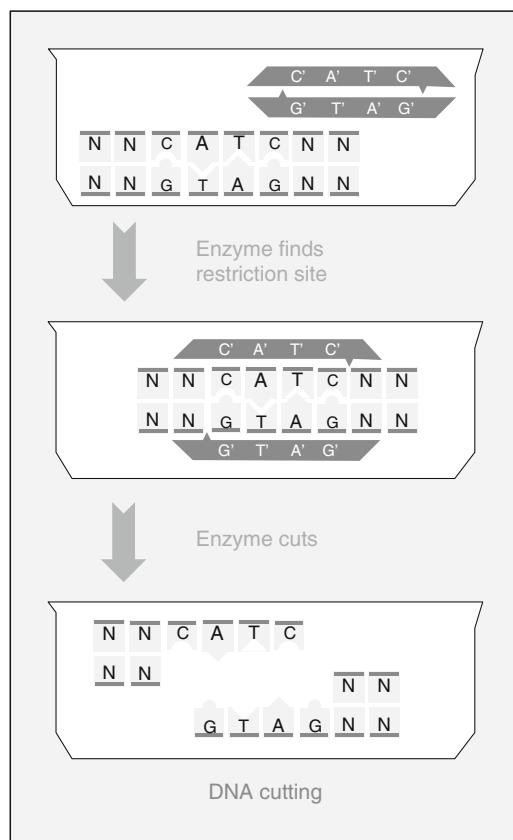
side of an agarose or polyacrylamide gel. Then an electric current is applied to the gel, with the negative pole at the side with the wells, and the positive pole at the opposite side. The DNA molecules will be drawn toward the positive pole, with the larger molecules traveling more slowly through the gel. After a period, the molecules will spread out into distinct bands according to size, **Fig. 5**. The gel method at constant electric field is capable of separating by size DNA molecules as long as 50,000 base pairs, with a resolution of better than 1% the size of the DNA.

Extraction of DNA single strands that contain a target sequence, v , from a heterogeneous solution of DNA single strands, can be accomplished by *affinity purification*, **Fig. 6**. A DNA probe is a single-stranded DNA molecule used in laboratory experiments to detect the presence of a complementary sequence among a mixture of other single-stranded DNA molecules.

The extraction process begins by synthesizing probes, that is, strands \vec{v} , Watson–Crick complementary to v , and attaching them to a solid support, for example, magnetic beads. Then, the heterogeneous solution of DNA strands is passed over the beads. Those strands containing v are “detected” by becoming annealed to \vec{v} , and are thus retained. Strands not containing v pass through without being retained. The solid medium, for example, magnetic

Fig. 4

DNA cutting (digestion) by a restriction enzyme. A hypothetical restriction enzyme (dark gray) recognizes the restriction site CATC on a DNA double-strand (light gray), and cuts the two backbones of the DNA strand, as shown. Under most conditions, Watson–Crick complementarity of four base-pairs is not sufficient to keep the strands together, and the DNA molecule separates into two fragments. The result of the digestion is thus a *CATC*—3' sticky end overhang, and a 3'—*GTAG* sticky end overhang. The reverse process of ligation can restore the original strand by bringing together strands with compatible sticky ends by means of Watson–Crick base-pairing, and using the enzyme ligase that repairs the backbone breaks (nicks) that had been introduced by the restriction endonuclease. (From Kari 1997.)



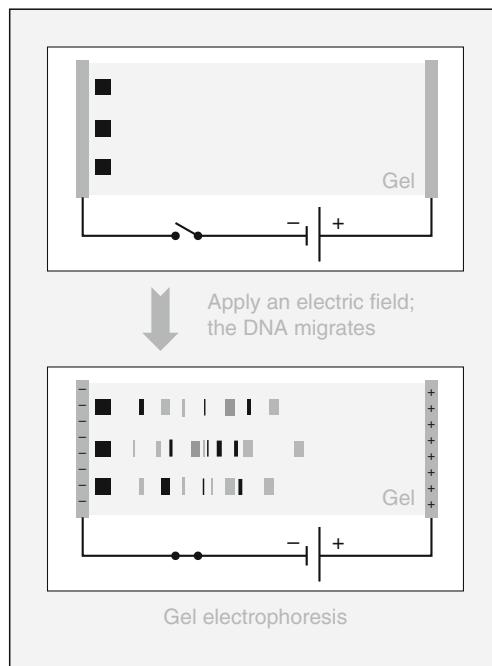
beads, can then be removed from the mixture, washed, and the target DNA molecules can be released from the entrapment.

DNA replication is accomplished by a process called *polymerase chain reaction*, or PCR, that involves the *DNA polymerase* enzyme, [Fig. 7](#).

The PCR replication reaction requires a guiding DNA single strand called *template*, and an oligonucleotide called *primer*, that is annealed to the template. The primer is required to initiate the synthesis reaction of the polymerase enzyme. The DNA polymerase enzyme catalyzes DNA synthesis by successively adding nucleotides to one end of the primer.

Fig. 5

Separation of DNA strands by size (length) by using gel electrophoresis. The DNA samples are placed in *wells* (slots) near one end of the gel slab. The power supply is switched on and the DNA, which is highly negatively charged, is allowed to migrate toward the positive electrode (right side of the gel) in separate *lanes* or *tracks*. DNA fragments will move through the gel at a rate which is dependent on their size and shape. After a while, the DNA molecules spread out into distinct bands and the use of a control “ladder” that contains DNA strands of incremental lengths allows the determination of the lengths of the DNA molecules in the samples. (From Kari 1997.)



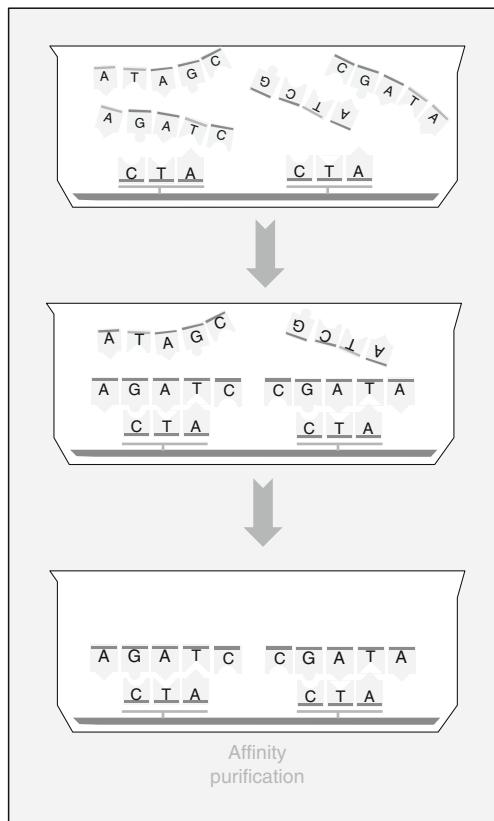
The primer is thus extended at its 3' end, in the direction 5' to 3' only, until the desired strand is obtained that starts with the primer and is complementary to the template. (Note that DNA chemical synthesis and enzymatic DNA replication proceed in opposite directions, namely 3' → 5' and 5' → 3', respectively).

If two primers are used, the result is the exponential multiplication of the subsequence of the template strand that is flanked by the two primers, in a process called *amplification*, schematically explained below. For the purpose of this explanation, if x is a string of letters over the DNA alphabet $\{A, C, G, T\}$, then \bar{x} will denote its simple complement, for example, $AACCTTGG = TTGGAACC$.

One can now assume that one desires to amplify the subsequence between x and y from the DNA double strand ${}^{5'}-\alpha x \beta y \delta-{}^{3'}$, where α, x, β, y , and δ are DNA segments. Then one uses as primers the strand x and the Watson–Crick complement \bar{y}' of y . After heating the solution and thus melting the double-stranded DNA into its two constituent strands, the solution is cooled and the Watson–Crick complement of y anneals to the “top” strand, while x anneals to the “bottom” strand. The polymerase enzyme extends the 3' ends of both primers into the 5' to 3'

Fig. 6

Extraction of strands that contain a target sequence, $v = 5' - GAT - 3'$, by affinity purification. (The 3nt pattern GAT is for illustration purposes only, in practice a longer DNA subsequence would be needed for the process of extraction to work.) The Watson–Crick complement of v , namely $3' - CTA - 5'$, is attached to a solid support, for example, magnetic beads. The heterogeneous solution of DNA strands is poured over. The DNA strands that contain the target sequence v as a subsequence will attach to the complement of the target by virtue of Watson–Crick complementarity. Washing off the solution will result in retaining the beads with the attached DNA strands containing the target sequence. (From Kari 1997.)

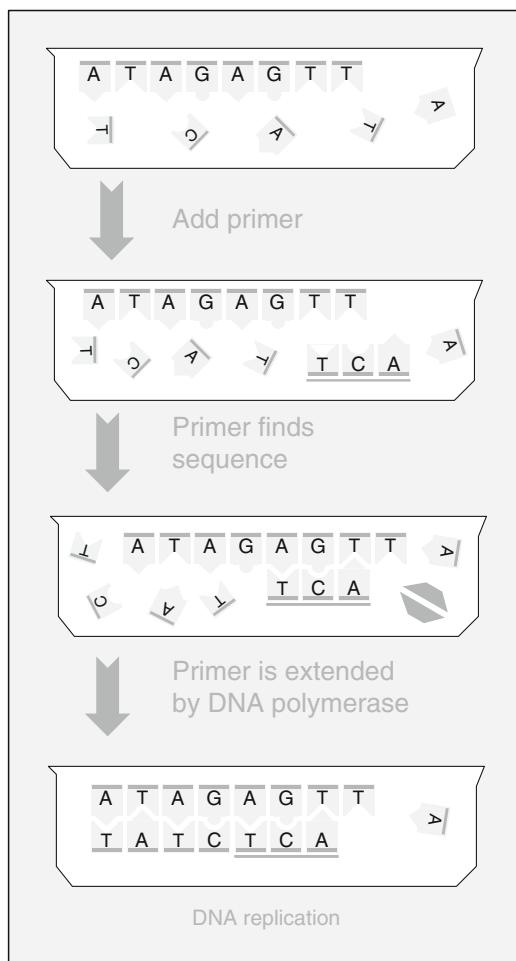


direction, producing partially double-stranded molecules $5' - \alpha x \beta y \delta - 3'$ and $5' - x \beta y \delta - 3'$. In a similar fashion, the next heating–cooling cycle will result in the production of the additional strands $5' - x \beta y - 3'$ and $3' - \bar{x} \bar{\beta} \bar{y} - 5'$. These strands are Watson–Crick complementary and will, from now on, be produced in excess of the other strands, since both are replicated during each cycle. At the end, an order of 2^n copies of the desired subsequences flanked by x and y will be present in solution, where n is the number of the heating–cooling cycles, typically 20 to 30.

A biocomputation consists of a succession of *bio-operations* (Daley and Kari 2002), such as the ones described in this section. The DNA strands representing the output of the biocomputation can then be **sequenced (read out)** using an automated sequencer. One sequencing method uses special chemically modified nucleotides (dideoxyribonucleoside

Fig. 7

DNA strand replication using one primer and the enzyme DNA polymerase. Given a template DNA strand 5' – ATAGAGTT – 3' to replicate, first a primer 3' – TCA – 5' (a short DNA sequence, usually 10–15 nt long, that is complementary to a portion of the template) is added to the solution. DNA polymerase (dark gray) extends the primer in the 5' to 3' direction, until the template becomes fully double stranded. Repeating the process by heating the solution to denature the double-strands and then cooling it to allow annealing of the primer, will thus lead to producing many copies of the portion of the complement of the template strand that starts with the primer. The idea is used in polymerase chain reaction (PCR) that uses two primers, DNA polymerase, dNTPs, and thermal cycling to produce exponentially many copies of the subsequence of a template strand that is flanked by the primers. dNTP, *deoxyribonucleoside triphosphate*, stands for any of the nucleotides dATP, dTTP, dCTP, and dGTP. Each nucleotide consists of a base, plus sugar (which together form a *nucleoside*), plus triphosphate. dNTPs are the building blocks from which the DNA polymerases synthesizes a new DNA strand. (From Kari 1997.)



triphosphates – ddNTPs), that act as “chain terminators” during PCR, as follows. A sequencing primer is annealed to the DNA template that one wishes to read. A DNA polymerase then extends the primer. The extension reaction is split into four tubes, each containing a different chain terminator nucleotide, mixed with standard nucleotides. For example, tube C would contain chemically modified C (ddCTP), as well as the standard nucleotides (dATP, dGTP, dCTP, and dTTP). Extension of the primer by the polymerase then produces all prefixes ending in G of the complement of the original strand. A separation of these strands by length using gel electrophoresis allows the determination of the position of all Gs (complements of Cs). Combining the results obtained in this way for all four nucleotides allows the reconstruction of the original sequence.

Some of the novel features of DNA-encoded information and bio-operations have been used for the first time for computational purposes in the breakthrough proof-of-principle experiment in DNA computing reported by Adleman in 1994.

3 The First DNA Computing Experiment

The practical possibilities of encoding information in a DNA sequence and of performing simple bio-operations were used by Leonard Adleman in 1994 (Adleman 1994) to perform the first experimental DNA computation that solved a seven-vertex instance of an NP-complete problem, namely the directed Hamiltonian path problem (HPP).

A directed graph G with designated vertices v_{start} and v_{end} is said to have a Hamiltonian path if and only if there exists a sequence of compatible directed edges e_1, e_2, \dots, e_z (i.e., a directed path) that begins at v_{start} , ends at v_{end} and enters every other vertex exactly once.

The following (nondeterministic) algorithm solves the problem:

Input. A directed graph G with n vertices and designated vertices v_{start} and v_{end} .

Step 1. Generate random paths through the graph.

Step 2. Keep only those paths that begin with v_{start} and end with v_{end} .

Step 3. Keep only those paths that enter exactly n vertices.

Step 4. Keep only those paths that enter all of the vertices of the graph at least once.

Output. If any paths remain, output “YES”; otherwise output “NO.”

Below, we describe Adleman’s bio-algorithm which solves the seven-vertex instance of the HPP illustrated in Fig. 8, where $v_{\text{start}} = 0$ and $v_{\text{end}} = 6$.

To encode the *input* to the problem, that is the vertices and directed edges of the graph, each vertex of the graph was encoded into a carefully chosen 20-mer single strand of DNA. Then, for each oriented edge of the graph, a DNA sequence was designed and synthesized, consisting of the second half of the sequence encoding the source vertex and the first half of the sequence encoding the target vertex. Exceptions were the edges that started with v_{start} , and the ones that ended in v_{end} , for which the DNA encoding consisted of the full sequence of the source vertex followed by the first half of the target vertex, respectively, the second half of the source vertex followed by the full sequence of the target vertex.

For example, the DNA sequences for the vertex 3 and the oriented edges $2 \rightarrow 3$ and $3 \rightarrow 4$ were encoded respectively as

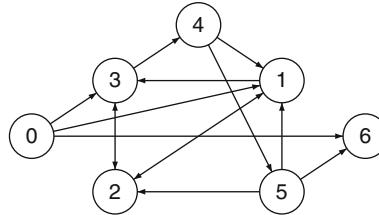
$$O_3 = 5' - \text{GCTATTGAGCTTAAAGCTA} - 3'$$

$$O_{2 \rightarrow 3} = 5' - \text{GTATATCCGAGCTATTGAG} - 3'$$

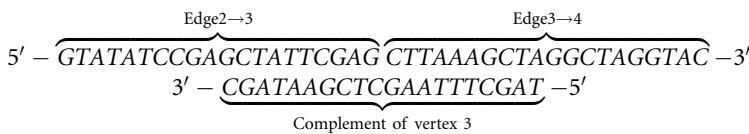
$$O_{3 \rightarrow 4} = 5' - \text{CTTAAAGCTAGGCTAGGTAC} - 3'.$$

Fig. 8

The seven-vertex instance of the Hamiltonian path problem (HPP) solved by Adleman, who used solely molecular biology tools to obtain the answer. This was the first ever experimental evidence that DNA computing is possible. (From Adleman 1994.)



To implement *Step 1*, one mixed together in a test tube multiple copies of each of the encodings of the Watson–Crick complements $\overrightarrow{O_i}$ of all the vertices, together with the encodings for all the directed edges, for a ligation reaction. The complements of the vertices served as splints and brought together sequences associated to compatible edges. For example, the edges $O_{2 \rightarrow 3}$ and $O_{3 \rightarrow 4}$ were brought together by the Watson–Crick complement $\overrightarrow{O_3}$ as follows:



Hence, the Watson–Crick complementarity and the combined ligation reaction resulted in the formation of DNA molecules encoding random paths through the graph. Out of these, the next steps had to find and discard the paths that were not Hamiltonian.

To implement *Step 2* (keep only paths that start with v_{start} and end with v_{end}), the product of *Step 1* was amplified by PCR with primers O_0 and $\overrightarrow{O_6}$. Thus, only those molecules encoding paths that begin with the start vertex 0 and end with the end vertex 6 were amplified.

For implementing *Step 3* (keep only paths of the correct length), gel electrophoresis was used, allowing separation of DNA strands by length. Since any Hamiltonian path, if it exists, has to pass through all the seven vertices of the graph, only DNA double strands of length $7 \times 20 = 140$ were retained.

Step 4 (keep only paths that pass through each vertex at least once) was accomplished by iteratively using affinity purification. After generating single-stranded DNA from the double-stranded product of the preceding step, one attached a sequence $\overrightarrow{O_i}$ to magnetic beads, and the heterogeneous solution of “candidate paths” was passed over the beads. Those strands containing O_i annealed to the complementary sequence and were hence retained. These strands represented paths that pass through the vertex i . This process was repeated successively with $\overrightarrow{O_1}$, $\overrightarrow{O_2}$, $\overrightarrow{O_3}$, $\overrightarrow{O_4}$, and $\overrightarrow{O_5}$.

To obtain the *Output* (are there any paths left?), the presence of a molecule encoding a Hamiltonian path was checked. This was done by amplifying the result of *Step 4* by PCR with primers O_0 and $\overrightarrow{O_6}$, and then reading out the DNA sequence of the amplified molecules. Note that the final PCR was not required for computational purposes: First, it was employed to make sure that, after several filtering steps, the amount of DNA was above the detection threshold, and second to verify the correctness of the answer.

The entire computation required approximately 7 days of wet lab work, and was carried out in approximately 1/50 of a teaspoon of solution (Adleman 1998). It was the first proof-of-concept experiment that DNA computation was possible. From a practical point of view, Adleman's approach had both advantages and disadvantages. On one hand, *Step 1*, which is the cause of the time-complexity exponential blowup in the classical electronic implementation of the algorithm, took only one time-step in Adleman's bio-algorithm. On the other hand, the amount of space needed for the generation of the full solution space grows exponentially relative to the problem size. Indeed, a scaled-up input of size $n = 200$ for the HPP would require, in this brute force approach, an amount of DNA whose weight would be greater than that of the Earth (Hartmanis 1995).

The construction of a DNA pool that contains the full-solution space has been avoided in subsequent bio-algorithms proposed for solving HPP. For example, in Morimoto et al. (1997) a bio-algorithm was proposed that stepwise generated only the possible paths. In this approach, all paths were stepwise extended from v_{start} to v_{end} as follows. After letting many molecules representing v_{start} attach to a surface, $n + 1$ repetitions of the following step found the Hamiltonian path: "Extend each path by one adjacent vertex; Remove paths which contain the same vertex twice." In addition to being space and time efficient (each extension step could be as quick as 30 min), this bio-algorithm avoided the laborious step of separation of strands by length.

4 Beyond Unaided Human Computation

We present another significant milestone in DNA computing research, the first experiment that demonstrated that DNA computing devices can exceed the computational power of an unaided human. Indeed, in 2002, an experiment was reported (Braich et al. 2002), that solved a 20-variable instance of the NP-complete 3-SAT problem, wherein the answer to the problem was found after an exhaustive search of more than 1 million (2^{20}) possible solution candidates.

The input to a 3-SAT problem is a Boolean formula in three-conjunctive-normal-form (3-CNF), that is, a Boolean formula that consists of conjunctions of disjunctive clauses, where each disjunctive clause is the disjunction of at most three literals (a literal is either a Boolean variable or its negation). This formula is called *satisfiable* if there exists a truth value assignment to its variables that satisfies it, that is, that makes the whole formula true. Thus, the output to the 3-SAT problem is "yes" if such a satisfying truth value assignment exists, and "no" otherwise.

The input formula for this experiment was the 20-variable, 24-clause, 3-CNF formula:

$$\begin{aligned} \Phi = & (\overline{x_3} \vee \overline{x_{16}} \vee x_{18}) \wedge (x_5 \vee x_{12} \vee \overline{x_9}) \wedge (\overline{x_{13}} \vee \overline{x_2} \vee x_{20}) \wedge (x_{12} \vee \overline{x_9} \vee \overline{x_5}) \\ & \wedge (x_{19} \vee \overline{x_4} \vee x_6) \wedge (x_9 \vee x_{12} \vee \overline{x_5}) \wedge (\overline{x_1} \vee x_4 \vee \overline{x_{11}}) \wedge (x_{13} \vee \overline{x_2} \vee \overline{x_{19}}) \\ & \wedge (x_5 \vee x_{17} \vee x_9) \wedge (x_{15} \vee x_9 \vee \overline{x_{17}}) \wedge (\overline{x_5} \vee \overline{x_9} \vee \overline{x_{12}}) \wedge (x_6 \vee x_{11} \vee x_4) \\ & \wedge (\overline{x_{15}} \vee \overline{x_{17}} \vee x_7) \wedge (\overline{x_6} \vee x_{19} \vee x_{13}) \wedge (\overline{x_{12}} \vee \overline{x_9} \vee x_5) \wedge (x_{12} \vee x_1 \vee x_{14}) \\ & \wedge (x_{20} \vee x_3 \vee x_2) \wedge (x_{10} \vee \overline{x_7} \vee \overline{x_8}) \wedge (\overline{x_5} \vee x_9 \vee \overline{x_{12}}) \wedge (x_{18} \vee \overline{x_{20}} \vee x_3) \\ & \wedge (\overline{x_{10}} \vee \overline{x_{18}} \vee \overline{x_{16}}) \wedge (x_1 \vee \overline{x_{11}} \vee \overline{x_{14}}) \wedge (x_8 \vee \overline{x_7} \vee \overline{x_{15}}) \wedge (\overline{x_8} \vee x_{16} \vee \overline{x_{10}}) \end{aligned}$$

where, for a Boolean variable x_i , $\overline{x_i}$ denotes the negation of x_i , $1 \leq i \leq 20$. The formula Φ was designed so as to have a unique satisfying truth assignment, namely $x_1 = F$, $x_2 = T$, $x_3 = F$, $x_4 = F$,

$x_5 = F, x_6 = F, x_7 = T, x_8 = T, x_9 = F, x_{10} = T, x_{11} = T, x_{12} = T, x_{13} = F, x_{14} = F, x_{15} = T, x_{16} = T, x_{17} = T, x_{18} = F, x_{19} = F, x_{20} = F$.

The DNA computing experiment that solved the problem (Braich et al. 2002), was based on the following nondeterministic algorithm.

Input: A Boolean formula Φ in 3-CNF.

Step 1: Generate the set of all possible truth value assignments.

Step 2: Remove the set of truth value assignments that make the first clause false.

Step 3: Repeat Step 2 for all the clauses of the input formula.

Output: The remaining (if any) truth value assignments.

To implement this algorithm, the input data was encoded as follows. Every variable $x_k, k = 1, \dots, 20$, was associated with two distinct 15-mer DNA single strands. One of them, denoted by X_k^T , represented true (T), while the second, denoted by X_k^F , represented false (F).

Below are some examples of the particular 15-mer sequences – none of which contained the nucleotide G – synthesized and used in the experiment:

$$X_2^T = \text{ATT TCC AAC ATA CTC}, \quad X_2^F = \text{AAA CCT AAT ACT CCT},$$

$$X_3^T = \text{TCA TCC TCT AAC ATA}, \quad X_3^F = \text{CCC TAT TAA TCA ATC}.$$

Using these 15-mer encodings for the two truth values of all the 20 variables, the library consisting of all possible 2^{20} truth assignments was assembled using the mix-and-match combinatorial synthesis technique of Faulhammer et al. (2000). In brief, oligonucleotides for X_{20}^T and X_{20}^F were synthesized separately, then mixed together. The mixture was divided in half and the result put in two separate test tubes. The synthesis was restarted separately, with sequences X_{19}^T and X_{19}^F , respectively. In principle, the process can be repeated until the desired library is obtained. In practice, two half-length libraries were created separately, and then linked together using a polymerase-chain extension similar to that in Stemmer et al. (1995). Each *library strand* encoding a truth assignment was thus represented by a 300-mer DNA strand consisting of the ordered catenation of twenty 15-mer value sequence, one for each variable, as follows:

$$X_1 X_2 \dots X_{20}, \text{ where } \alpha_i \in \{X_i^T, X_i^F\}, 1 \leq i \leq 20$$

The biocomputation *wet-ware* essentially consisted of a glass module filled with a gel containing the library, as well as 24 glass *clause modules*, one for each of the 24 clauses of the formula. Each clause module was filled with gel containing probes (immobilized DNA single strands) designed to bind only library strands encoding truth assignments satisfying that clause.

The strands were moved between the modules with the aid of gel electrophoresis, that is, by applying an electric current that resulted in the migration of the negatively charged DNA strands through the gel.

The protocol started with the library passing through the first clause module, wherein library strands containing the truth assignments satisfying the first clause (i.e., library strands containing sequences X_3^F , or X_{16}^F , or X_{18}^T) were captured by the immobilized probes, while library strands that did not satisfy the first clause (i.e., library strands containing sequences X_3^T , and X_{16}^T , and X_{18}^F) continued into a buffer reservoir. The captured strands were then released by raising the temperature, and used as input to the second clause module, etc. At the end, only the strand representing the truth assignment that satisfied all 24 clauses remained.

The output strand was PCR amplified with primer pairs corresponding to all four possible true–false combinations of assignments for the first and last variable x_1 and x_{20} . None except

the primer pair $(X_1^F, \overrightarrow{X_{20}^F})$ showed any bands, thus indicating two truth values of the satisfying assignment, namely $x_1 = F$ and $x_{20} = F$. The process was repeated for each of the variable pairs (x_1, x_k) , $2 \leq k \leq 19$, and, based on the lengths of the bands observed, value assignments were given to the variables. These experimentally derived values corresponded to the unique satisfying assignment for the formula Φ , concluding thus the experiment.

One of the remarkable features of this benchmark DNA computing experiment was that the sole bio-operation that was used (except during input and output) was Watson–Crick complementarity-based annealing and melting.

Generally, it is believed that DNA computers that use a brute-force search algorithm for SAT are limited to 60 to 70 variables (Lipton 1995). Several other algorithms that do not use brute force, such as the breadth-first search algorithm (Yoshida and Suyama 2000), and random walk algorithms (Liu et al. 2005; Diaz et al. 2001) have been proposed. With the breadth-first search algorithm, the capacity of a DNA computer can be theoretically increased to about 120 variables (Yoshida and Suyama 2000). A recent example of this approach that avoids the generation of the full solution space is a solution to the SAT problem using a DNA computing algorithm based on ligase chain reaction (LCR) (Wang et al. 2008). This bio-algorithm can solve an n -variable m -clause SAT problem in m steps, and the computation time required is $O(3m + n)$. Instead of generating the full-solution DNA library, this bio-algorithm starts with an empty test tube and then generates solutions that partially satisfy the SAT formula. These partial solutions are then extended step by step by the ligation of new variables using DNA ligase. Correct strands are amplified and false strands are pruned by a ligase chain reaction (LCR) as soon as they fail to satisfy the conditions.

The two DNA computing experiments described in [Sects. 3](#) and [4](#) are historically significant instances belonging to a vast and impressive array of often astonishing DNA computing experiments, with potential applications to, for example, nanorobotics, nanocomputing, bioengineering, bio-nanotechnology, and micromedicine. Several of these significant experiments are described in the chapters [Molecular Computing Machineries — Computing Models and Wet Implementations](#), [Bacterial Computing and Molecular Communication](#), [DNA Memory](#), and [Engineering Natural Computation by Autonomous DNA-Based Biomolecular Devices](#) of this handbook.

5 DNA Complementarity and Formal Language Theory

The preceding two sections described two milestone DNA computing experiments whose main “computational engine” was the Watson–Crick complementarity between DNA strands. Watson–Crick complementarity can now be studied from a theoretical point of view, and the impact this notion has had on theoretical computer science will be described. This section focuses mainly on the formal language theoretical and coding theoretical approach to DNA-encoded information, and highlights some of the theoretical concepts and results that emerged from these studies.

The idea of formalizing and investigating DNA or RNA molecules and their interactions by using the apparatus of formal language theory is a natural one. Indeed, even though the processes involving DNA molecules are driven by complex biochemical reactions, the primary information is encoded in DNA sequences. Therefore, even without the inclusion of all the thermodynamic parameters that operate during DNA–DNA interactions, formal language theory and coding theory are capable of providing a uniform and powerful framework for an effective study of DNA-encoded information.

To briefly describe the problem of encoding information as DNA strands for DNA computing experiments, one of the main differences between electronic information and DNA-encoded information is that the former has a fixed address and is reusable, while the latter is not. More precisely, DNA strands float freely in solution in a test tube and, if one of the input strands for a bio-operation has become involved in another, unprogrammed, hybridization, that input strand is unavailable for the bio-operation at hand, compromising thus the final result of the experiment. Thus, a considerable effort has been dedicated to finding “optimal” DNA sequences for encoding the information on DNA so as to prevent undesirable interactions and favor only the desirable programmed ones.

Standard biological methods evaluating and predicting hybridization between DNA molecules conditions rely on thermodynamical parameters such as the free energy ΔG (intuitively, the energy needed to melt DNA bonds), and the melting temperature of DNA molecules. For the calculation of these quantities, not only the WK complementarity between single bases is important, but also the WK complementarity (or lack thereof) of their neighboring bases with their counterparts: the *nearest-neighbor model* (SantaLucia 1998) has been frequently used for this purpose. However, in many natural processes, the WK complementarity alone plays a crucial role. Furthermore, together with various similarity metrics, the WK complementarity has been often used to obtain an approximate characterization of DNA hybridization interactions.

This section is devoted to describing methods of discrete mathematics and formal language theory that allow for rapid and mathematically elegant characterization of (partially) complementary DNA molecules, their sets, and set-properties relevant for potential cross-hybridizations. For other approaches to this problem and to the problem of design of molecules for DNA computing, the reader is referred to the chapter [DNA Memory](#) in this book. The section is organized as follows. [Section 5.1](#) describes the problem of optimal encoding of information for DNA computing experiments that was the initial motivation for this research, and also introduces the basic definitions and notation. [Section 5.2](#) describes the problem of intramolecular hybridization (hybridization within one molecule, resulting, e.g., in hairpin formation) and results related to hairpin languages and hairpin-free languages. [Section 5.3](#) describes the problem of intermolecular hybridization (interaction between two or more DNA molecules) and the theoretical concepts and results motivated by this problem. [Section 5.4](#) investigates properties of languages that guarantee that even undesirable imperfect bonds between DNA strands are avoided.

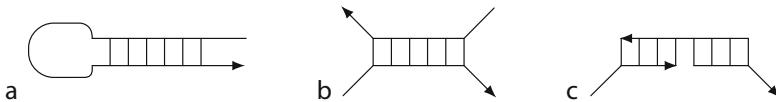
5.1 DNA Encoding: Problem Setting and Notation

In the process of designing DNA computational experiments, as well as in many general laboratory techniques, special attention is paid to the design of an initial set of “good” DNA strands. Mauri and Ferretti (2004), Sager and Stefanovic (2006), and others distinguish two elementary subproblems of the encoding sequence design:

- *Positive* design problem: Design a set of input DNA molecules such that there is a sequence of reactions that produces the correct result.
- *Negative* design problem: Design a set of input DNA molecules that do not interact in undesirable ways, that is, do not produce incorrect outputs, and/or do not consume molecules necessary for other, programmed, interactions.

Fig. 9

Types of undesired (a) intramolecular and (b), (c) intermolecular hybridizations.



The positive design problem is usually highly related to a specific experiment and it is reported to be hard to find a general framework for its solution. In contrast, the negative design problem can be solved on a general basis by construction of a library of molecules that do not allow for undesired mutual hybridizations. According to Sager and Stefanovic (2006), the following conditions must be guaranteed: (1) no strand forms any undesired secondary structure such as hairpin loops (Fig. 9a), (2) no string in the library hybridizes with any string in the library, and (3) no string in the library hybridizes with the complement of any string in the library (Fig. 9b or c). Many laboratory techniques stress the importance of a unified framework for the negative design. An example is the multiplex PCR in which a set of PCR primers is used simultaneously in a single test tube and mutual bonds between primers must be prevented.

A related issue often studied together with the problem of unwanted hybridization is the uniqueness of the oligonucleotides used in experiments. More precisely, one requires that individual oligonucleotides in a mixture differ substantially from each other such that they (and eventually also longer sequences produced by their catenation) can be easily distinguished. Such a property in the mathematical sense is typical for *codes*, hence many authors adopted this naming convention for sets of oligonucleotides, usually of a fixed length, whose elements are then called *DNA codewords*.

A wide spectrum of methods devoted to the DNA encoding design exists. Thermodynamical methods such as those in Deaton et al. (2003), and Dirks and Pierce (2004) provide the most precise results but are computationally the most expensive. Experimental studies trying to construct DNA codes in vitro with the help of the PCR operation can be found, for example, in Chen et al. (2006), and Deaton et al. (2006). An opposite approach relying solely on the WK complementarity allows for fastest but least-precise methods (see, e.g., Head (2000), Jonoska and Mahalingam (2004), and Kari et al. (2005b)). Approximation methods trying to capture key aspects of the nearest neighbor thermodynamic model represent an intermediate step between these two methodologies. Various discrete metrics based often on Hamming or Levenshtein distance have been studied, for example, in Dyachkov et al. (2006, 2008), and Garzon et al. (1997, 2006). The reader is referred to monographs Amos (2005), Ignatova et al. (2008), and Păun et al. (1998) for an overview.

In the sequel, we focus on the characterization of DNA hybridization and unwanted bonds by means of the concepts of formal language and coding theory. Simple examples of construction of DNA codes are also given.

To describe DNA bonds formally, we represent the single-stranded DNA molecules by strings over the *DNA alphabet* $\Delta = \{A, C, T, G\}$, and their mutual reactions can be reduced to formal manipulation of these strings. Therefore, some formal language prerequisites are necessary. For further details the reader is referred to Hopcroft and Ullman (1979), Choffrut and Karhumäki (1997), and Salomaa (1973).

An *alphabet* is a finite and nonempty set of symbols. In the sequel a fixed non-singleton alphabet Σ shall be used as a generalization of the natural DNA alphabet Δ . The set of all words

over Σ is denoted by Σ^* . This set includes the *empty word* λ . The length of a word $w \in \Sigma^*$ is denoted by $|w|$. For an $x \in \Sigma^+$, $|w|_x$ denotes the number of occurrences of x within w . For a nonnegative integer n and a word w , we use w^n to denote the word that consists of n concatenated copies of w . A word v is a *subword* of w if $w = xvy$ for some words x and y . In this case, if $|x| + |y| > 0$ then v is a *proper subword*. By $\text{Sub}(w)$ we denote the set of all subwords of w . For a positive integer k , we use $\text{Sub}_k(w)$ to denote the set of subwords of length k of w . We say that $u \in \Sigma^*$ is a prefix of a word $v \in \Sigma^*$, and denote it by $u \leq v$, if $v = ut$ for some $t \in \Sigma^*$. Two words u, v are said to be *prefix comparable*, denoted by $u \sim_p v$ if one of them is a prefix of the other. In a similar manner, u is said to be a suffix of v if $v = su$ for some $s \in \Sigma^*$. By $\text{Pref}(u)$ ($\text{Suff}(u)$), we denote the sets of all prefixes (respectively suffixes) of u .

The relation of *embedding order* over words is defined as follows: $u \leq_e w$ iff

$$u = u_1 u_2 \cdots u_n, \quad w = v_1 v_2 v_3 \cdots v_n v_n v_{n+1}$$

for some integer n with $u_i v_j \in \Sigma^*$.

A language L is a set of words, or equivalently a subset of Σ^* . A language is said to be λ -free if it does not contain the empty word. If n is a nonnegative integer, we write L^n for the language consisting of all words of the form $w_1 \dots w_n$ such that each w_i is in L , and $L^{\geq n}$ for the language consisting of all catenations of at least n words from L . We also write L^* for the language $L^0 \cup L^1 \cup L^2 \cup \dots$, and L^+ for the language $L^* \setminus \{\lambda\}$. The set $\text{Sub}(L) = \bigcup_{w \in L} \text{Sub}(w)$ we call the set of all subwords of L . The families of regular, linear, context-free, and recursively enumerable languages are denoted by REG, LIN, CF, and RE, respectively.

Many approaches to the construction of DNA encoding are based on the assumption that the set of molecules in a test tube (*tube language*) L is equal to, or a subset of, K^+ , where K is a finite language whose elements are called *codewords*. In general, K might contain codewords of different lengths. In many cases, however, the set K consists of words of a certain fixed length l . In this case, we shall refer to K as a *code of length l*.

A mapping $\alpha : \Sigma^* \rightarrow \Sigma^*$ is called a *morphism* (*antimorphism*) of Σ^* if $\alpha(uv) = \alpha(u)\alpha(v)$ (respectively $\alpha(uv) = \alpha(v)\alpha(u)$) for all $u, v \in \Sigma^*$. Note that a morphism or an antimorphism of Σ^* are completely defined if we define their values on the letters of Σ .

If for a morphism α , $\alpha(a) \neq \lambda$ for each $a \in V$, then α is said to be λ -*free*. A *projection* associated to Σ is a morphism $pr_\Sigma : (V \cup \Sigma)^* \rightarrow \Sigma^*$ such that $pr_\Sigma(a) = a$ for all $a \in \Sigma$ and $pr_\Sigma(b) = \lambda$ otherwise. A morphism $h : V^* \rightarrow \Sigma^*$ is called a *coding* if $h(a) \in \Sigma$ for all $a \in V$ and *weak coding* if $h(a) \in \Sigma \cup \{\lambda\}$ for all $a \in V$. For a family of languages FL , one can denote by $\text{Cod}(FL)$ (respectively $\text{wcod}(FL)$) the family of languages of the form $h(L)$, for $L \in FL$ and h a coding (respectively weak coding).

The *equality set* of two morphisms $h_1, h_2 : V^* \rightarrow \Sigma^*$ is defined as

$$\text{EQ}(h_1, h_2) = \{w \in V^* \mid h_1(w) = h_2(w)\}$$

An *involution* $\theta : \Sigma \rightarrow \Sigma$ of Σ is a mapping such that θ^2 is equal to the identity mapping, that is, $\theta(\theta(x)) = x$ for all $x \in \Sigma$. It follows then that an involution θ is bijective and $\theta = \theta^{-1}$. The identity mapping is a trivial example of an involution. An involution of Σ can be extended to either a morphism or an antimorphism of Σ^* . For example, if the identity of Σ is extended to a morphism of Σ^* , we obtain the identity involution of Σ^* . However, if the identity of Σ is extended to an antimorphism of Σ^* we obtain instead the mirror-image involution of Σ^* that maps each word u into u^R where

$$u = a_1 a_2 \dots a_k, \quad u^R = a_k \dots a_2 a_1, \quad a_i \in \Sigma, \quad 1 \leq i \leq k$$

A word w that is equal to its reverse w^R is called a *palindrome*. If we consider the DNA alphabet Δ , then the mapping $\tau : \Delta \rightarrow \Delta$ defined by $\tau(A) = T, \tau(T) = A, \tau(C) = G, \tau(G) = C$ can be extended in the usual way to an antimorphism of Δ^* that is also an involution of Δ^* . This involution formalizes the notion of Watson–Crick complementarity and will therefore be called the *DNA involution* (Kari et al. 2002).

5.2 Intramolecular Bond (Hairpin) Analysis

In this section, we focus on mathematical properties of DNA hairpins and their importance in DNA encodings. A *DNA hairpin* is a particular type of DNA secondary structure illustrated in [Fig. 10](#).

Hairpin-like secondary structures play an important role in insertion/deletion operations with DNA. Hairpin-freeness is crucial in the design of primers for the PCR reaction. Among numerous applications of hairpins in DNA computing only the Whiplash PCR computing techniques (Rose et al. 2002) and the DNA RAM (see the chapter [DNA Memory](#) in this handbook) are mentioned. The reader is referred, for example, to Kijima and Kobayashi (2006), Kobayashi (2005), and Mauri and Ferretti (2004) for a characterization and design of tube languages with or without hairpins. Coding properties of hairpin-free languages were studied in Jonoska et al. (2002) and Jonoska and Mahalingam (2004). A language-theoretical characterization of hairpins and hairpin languages was given in Păun et al (2001). Hairpins have also been studied in the context of bio-operations occurring in single-celled organisms. For example, the operation of hairpin inversion was defined as one of the three molecular operations that accomplish gene assembly in ciliates (Daley et al. 2003, 2004; Ehrenfeucht et al. 2004). Applications of hairpin structures in biocomputing and bio-nanotechnology are discussed in the chapter [Molecular Computing Machineries — Computing Models and Wet Implementations](#) in this handbook.

The following definition formally specifies hairpin as a structure described in [Fig. 10](#), whose stem consists of at least k base pairs. This condition is motivated by the fact that a hairpin with shorter stem is less stable. An oligonucleotide that does not satisfy this condition is said to be hairpin-free.

Definition 1 (Jonoska et al. 2002; Kari et al. 2006) Let θ be a (morphic or antimorphic) involution of Σ^* and k be a positive integer. A word $u \in \Sigma^*$ is said to be θ - k -hairpin-free or simply $hp(\theta, k)$ -free if $u = xvy\theta(v)z$ for some $x, v, y, z \in \Sigma^*$ implies $|v| < k$.

We denote by $hpf(\theta, k)$ the set of all $hp(\theta, k)$ -free words in Σ^* . The complement of $hpf(\theta, k)$ is the set of all hairpin-forming words over Σ and is denoted by $hp(\theta, k) = \Sigma^* \setminus hpf(\theta, k)$. Observe that $hp(\theta, k+1) \subseteq hp(\theta, k)$ for all $k > 0$. A language L is said to be θ - k -hairpin-free or simply $hp(\theta, k)$ -free if $L \subseteq hpf(\theta, k)$.

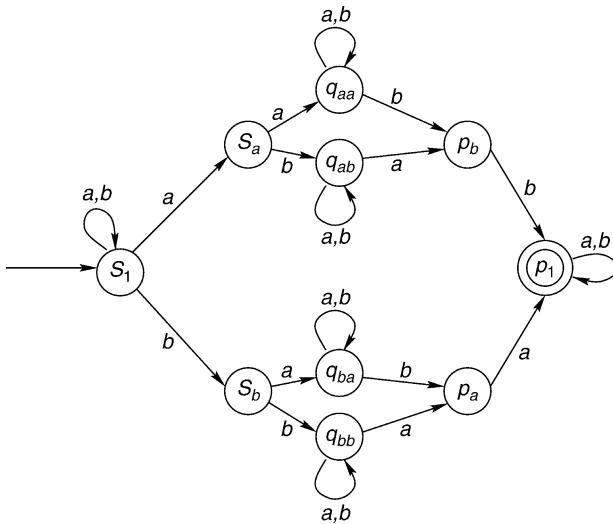
[Fig. 10](#)

A single-stranded DNA molecule forming a hairpin loop.



Fig. 11

An NFA accepting the language $hp(\theta, 2)$ over the alphabet $\{a, b\}$, where the antimorphism is defined as $\theta(a) = b$ and $\theta(b) = a$.



Example 1 Let $\theta = \tau$, the DNA involution over Δ^* . Then

$$hpf(\theta, 1) = \{A, C\}^* \cup \{A, G\}^* \cup \{T, C\}^* \cup \{T, G\}^*$$

In the version of Definition 1 given in Jonoska et al. (2002), a θ - k -hairpin-free language was called θ -subword- k -code. The authors focused on their coding properties and relations to other types of codes. A restriction on the length of the loop of a hairpin was also considered: $1 \leq |y| \leq m$ for some $m \geq 1$. Most of the results mentioned in this section remain valid if this additional restriction is considered.

Theorem 1 (Păun et al. 2001) *The languages $hp(\theta, k)$ and $hpf(\theta, k)$, $k \geq 1$, are regular.*

Figure 11 illustrates a nondeterministic finite automaton (NFA) accepting the language $hp(\theta, 2)$ over the alphabet $\{a, b\}$ where $\theta(a) = b$ and $\theta(b) = a$. Given the above characterization of $hp(\theta, k)$ and $hpf(\theta, k)$, the following result is rather immediate.

Theorem 2 (Kari et al. 2006) *The following problem is decidable in linear (or cubic, respectively) time with respect to (w.r.t.) $|M|$:*

Input: A nondeterministic regular (pushdown, respectively) automaton M
Output: Yes/No depending on whether $L(M)$ is $hp(\theta, k)$ -free

The *maximality problem* of hairpin-free languages is stated as follows: can a given language $L \subseteq \Sigma^*$, satisfying a certain property (e.g., to be a hairpin-free language), be still extended without loss of this property? Formally, a language $L \subseteq \Sigma^*$ satisfying a property \mathcal{P} is said to be

maximal with respect to \mathcal{P} iff $L \cup \{w\}$ does not satisfy \mathcal{P} for any $w \in M \setminus L$, where M is a fixed library of available strands.

Theorem 3 (Kari et al. 2006) *The following problem is decidable in time $\Theta(|M_1| \cdot |M_2|)$ (or $\Theta(|M_1| \cdot |M_2|^3)$, respectively):*

Input: A positive integer k , a deterministic finite (pushdown, respectively) automaton M_1 accepting a $hp(\theta, k)$ -free language, and an NFA M_2

Output: Yes/No depending on whether there is a word $w \in L(M_2) \setminus L(M_1)$ such that $L(M_1) \cup \{w\}$ is $hp(\theta, k)$ -free

For hairpin-free languages, it is relatively straightforward to solve the *optimal negative design problem*: to construct a set of hairpin-free DNA words of a given size, where the words can be chosen from a certain library. All the hairpin-free sets are subsets of $hpf(\theta, k)$. For example, if the length of the desired DNA words equals a constant ℓ , the optimal hairpin-free set is simply $hpf(\theta, k) \cap \Sigma^\ell$. Due to Theorem 1, the set $hpf(\theta, k)$ is regular and hence can be accepted by a finite automaton. The size of the automaton, however, grows exponentially with respect to k .

Theorem 4 (Kari et al. 2006) *Consider the DNA alphabet $\Delta = \{A, C, T, G\}$ and the DNA involution τ .*

- (i) *The size of a minimal NFA accepting $hp(\tau, k)$ is at most 15×4^k . The number of its states is between 4^k and 3×4^k .*
- (ii) *The number of states of either a minimal deterministic finite automaton (DFA) or an NFA accepting $hpf(\tau, k)$ is between $2^{2^{k-1}}$ and $2^{3 \times 2^k}$.*

The reader is referred to Kari et al. (2006) for a generalization of the above theorem for the case of an arbitrary alphabet and arbitrary involution. The construction of the automaton is illustrated in Fig. 11 for the case of alphabet $\{a, b\}$ and an antimorphism θ , where $\theta(a) = b$ and $\theta(b) = a$.

Problems analogous to Theorems 1–4 have been studied also in the case of *scattered hairpins* and *hairpin frames* that represent more complex but rather common types of intramolecular hybridization. The definition of scattered hairpins covers structures like the one described in Fig. 12.

Definition 2 (Kari et al. 2006) *Let θ be an involution of Σ^* and let k be a positive integer. A word $u = wy$, for $u, w, y \in \Sigma^*$, is θ - k -scattered-hairpin-free or simply $shp(\theta, k)$ -free if for all $t \in \Sigma^*$, $t \leq_e w$, $\theta(t) \leq_e y$ implies $|t| < k$.*

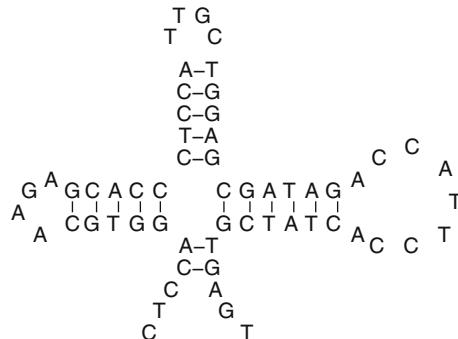
Fig. 12

An example of a scattered hairpin – a word in $shp(\tau, 11)$.



Fig. 13

An example of a hairpin frame.



Similarly, the following definition of hairpin frames characterizes secondary structures containing several complementary sequences such as that in Fig. 13.

Definition 3 (Kari et al. 2006) The pair $(v, \theta(v))$ in a word u of the form $u = xvy\theta(v)z$, for $x, v, y, z \in \Sigma^*$, is called an hp-pair of u . The sequence of hp-pairs $(v_1, \theta(v_1)), (v_2, \theta(v_2)), \dots, (v_j, \theta(v_j))$ of the word u in the form:

$$u = x_1 v_1 y_1 \theta(v_1) z_1 x_2 v_2 y_2 \theta(v_2) z_2 \cdots x_j v_j y_j \theta(v_j) z_j$$

is called an hp-frame of degree j of u or simply an $\text{hp}(j)$ -frame of u .

Several other studies have been published, focusing on formal-language aspects of the hairpin formation. A complete characterization of the syntactic monoid of the language consisting of all hairpin-free words over a given alphabet was given in Kari et al. (2007). Manea et al. (2009) described formal language operations of hairpin completion and reduction and studied their closure and other mathematical properties. DNA trajectories – a new formal tool for description of scattered hairpins – were presented in Domaratzki (2007), where also complexity of the set of hairpin-free words described by a set of DNA trajectories and closure properties of hairpin language classes were studied. Hairpin finite automata with the ability to apply the hairpin inversion operation to the remaining part of the input were introduced in Bordihn et al. (2007). The authors studied the power of hairpin-inspired operations and the resulting language classes. Finally, Kari and Seki (2009) focused on a related secondary structure based on intramolecular bonds – *pseudoknots* (see also Dirks and Pierce (2004) for more information on pseudoknots). The authors provided mathematical formalization of pseudoknots and obtained several properties of pseudoknot-bordered and -unbordered words.

5.3 How to Avoid DNA Intermolecular Bonds

In this section, several properties of a tube language $L \subseteq \Sigma^+$, which prohibit various types of undesired hybridizations between two DNA strands, are formally characterized. Many authors

assume, for simplicity, that hybridization occurs only between those parts of single-stranded DNA molecules that are perfectly complementary. The following language properties have been considered in Hussini et al. (2003) and Kari et al. (2002, 2003).

- (A) **θ -Nonoverlapping:** $L \cap \theta(L) = \emptyset$
- (B) **θ -Compliant:** $\forall w \in L, x, y \in \Sigma^*, w, x\theta(w)y \in L \Rightarrow xy = \lambda$
- (C) **θ -p-Compliant:** $\forall w \in L, y \in \Sigma^*, w, \theta(w)y \in L \Rightarrow y = \lambda$
- (D) **θ -s-Compliant:** $\forall w \in L, y \in \Sigma^*, w, y\theta(w) \in L \Rightarrow y = \lambda$
- (E) **Strictly θ -compliant:** both θ -compliant and θ -nonoverlapping
- (F) **θ -Free:** $L^2 \cap \Sigma^+ \theta(L) \Sigma^+ = \emptyset$
- (G) **θ -Sticky-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, wx, y\theta(w) \in L \Rightarrow xy = \lambda$
- (H) **θ -3'-Overhang-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, wx, \theta(w)y \in L \Rightarrow xy = \lambda$
- (I) **θ -5'-Overhang-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, xw, y\theta(w) \in L \Rightarrow xy = \lambda$
- (J) **θ -Overhang-free:** both θ -3'-overhang-free and θ -5'-overhang-free

For convenience, we agree to say that a language L containing the empty word has one of the above properties if $L \setminus \{\lambda\}$ has that property. Observe that (F) avoids situations like Fig. 9c, while other properties exclude special cases of Fig. 9b.

In Jonoska and Mahalingam (2004), a θ -nonoverlapping language was said to be *strictly θ* . Generally, if any other property holds in conjunction with (A), we add the qualifier *strictly*. This notation has already been used for the property (E). Both *strict* and *non-strict* properties turn out to be useful in certain situations.

For example, a common way to check for the presence of a certain single-stranded molecule w is to add to the solution its complement $\tau(w)$, and use enzymes to destroy any molecules that are not fully double stranded. Simultaneously, we want to prevent all other hybridizations except w and $\tau(w)$. This condition is equivalent to testing whether the whole solution, including w and $\tau(w)$, is non-strictly bond-free (exact matches are allowed).

Further properties have been defined in Jonoska and Mahalingam (2004) for a language L . Observe that the property (K) below avoids bonds like those in Fig. 9a, with a restricted length of the loop part:

- (K) **$\theta(k, m_1, m_2)$ -Subword compliant:** $\forall u \in \Sigma^k, \Sigma^* u \Sigma^m \theta(u) \Sigma^* \cap L = \emptyset$ for $k > 0, m_1 \leq m \leq m_2$
- (L) **θ -k-Code:** $\text{Sub}_k(L) \cap \text{Sub}_k(\theta(L)) = \emptyset, k > 0$

The property (L) was also considered implicitly in Baum (1998) and Feldkamp et al. (2002). In particular, Baum (1998) considered tube languages of the form $(sZ)^+$ satisfying (L), where s is a fixed word of length k and Z is a code of length k – the notation sZ represents the set of all words sz such that z is in Z .

The following property was defined for $\theta = I$, the identity relation, in Head (2000). A language L is called

- (M) **solid** if
 1. $\forall x, y, u \in \Sigma^*, u, xuy \in L \Rightarrow xy = \lambda$ and
 2. $\forall x, y \in \Sigma^*, u \in \Sigma^+, xu, uy \in L \Rightarrow xy = \lambda$

L is *solid relative* to a language $M \subseteq \Sigma^*$ if 1 and 2 above must hold only when there are $p, q \in \Sigma^*$ such that $pxuyq \in M$. L is called *comma-free* if it is solid relative to L^* . Solid languages were also used in Kari et al. (2003) as a tool for constructing error-detecting tube languages that were invariant under bio-operations.

Fig. 14

Classes of tube languages free of certain types of undesired hybridization.

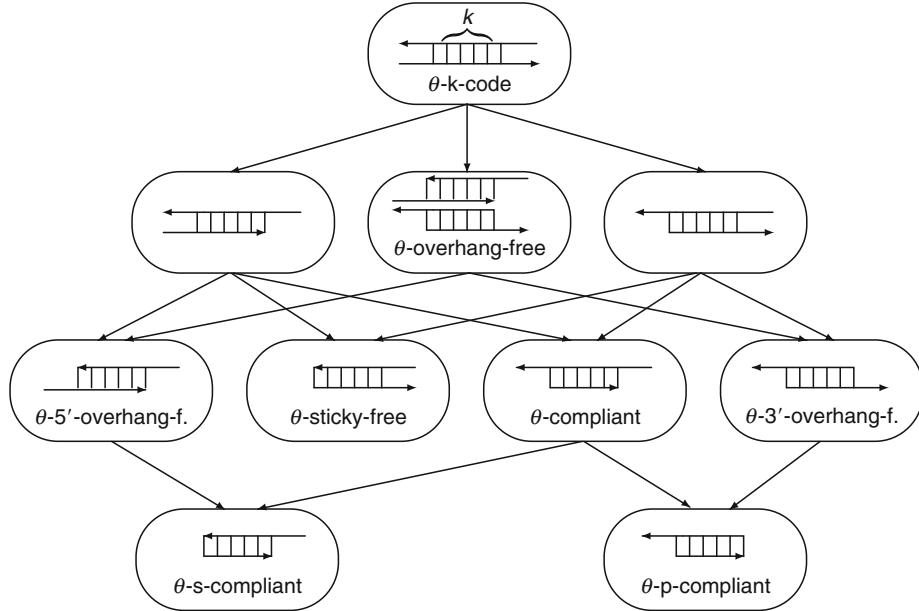


Figure 14 shows the hierarchy of some of the above language properties. Arrows stand for inclusion relations among language classes satisfying these properties.

Example 2 (Kari et al. 2005b) Consider the language $L = \{A^nT^n \mid n \geq 1\} \subset \Delta^+$, and the DNA involution τ . Observe that $\tau(L) = L$. We can deduce that L is:

- Neither τ -nonoverlapping, nor τ - k -code for any $k \geq 1$
- Not τ -compliant, as for $w = A^nT^n$, $x = A$, $y = T$ we have $w, xt(w)y \in L$
- τ - p -Compliant, as $w, \theta(w)y \in L$ implies $w = A^nT^n$, $y = \lambda$; similarly, L is τ - s -compliant
- Not τ -free, as $A^nT^nA^mT^m$, $n, m > 1$, is both in L^2 and in $\Delta^+L\Delta^+$
- Not τ -sticky-free, as for $w = y = A^n$. $x = T^n$ we have $wx, \tau(w)y \in L$
- τ - $3'$ -Overhang-free, as $wx, \tau(w)y \in L$ implies $w = A^nT^n$, $x = T^{n-m}$, $y = T^{m-n}$ and hence $xy = \lambda$; similarly, L is τ - $5'$ -overhang-free and hence τ -overhang-free
- Not $\theta(k, m_1, m_2)$ -subword compliant for any k, m_1, m_2

For further details and relations between the above listed DNA language properties the reader is referred to Jonoska and Mahalingam (2004) and Kari et al. (2003, 2005b).

To establish a common framework allowing us to handle various types of unwanted hybridization in a uniform way, it is necessary to introduce the generalizing concept of word operations on trajectories. Consider a *trajectory alphabet* $V = \{0, 1\}$ and assume $V \cap \Sigma = \emptyset$. We call any string $t \in V^*$ a *trajectory*. A trajectory is essentially a syntactical condition that specifies how a binary word operation is applied to the letters of its two operands. Let $t \in V^*$ be a trajectory and let α, β be two words over Σ .

Definition 4 (Mateescu et al. 1998) The shuffle of α with β on a (fixed) trajectory t , denoted by $\alpha \sqcup \sqcup_t \beta$, is the following binary word operation:

$$\alpha \sqcup \sqcup_t \beta = \{ \alpha_1 \beta_1 \dots \alpha_k \beta_k \mid \alpha = \alpha_1 \dots \alpha_k, \beta = \beta_1 \dots \beta_k, t = 0^{i_1} 1^{j_1} \dots 0^{i_k} 1^{j_k}, \\ \text{where } |\alpha_m| = i_m \text{ and } |\beta_m| = j_m \text{ for all } m, 1 \leq m \leq k \}.$$

Example 3 Let $\alpha = a_1 a_2 \dots a_8$, $\beta = b_1 b_2 \dots b_5$ and assume that $t = 0^3 1^2 0^3 1 0 1 0 1$. The shuffle of α and β on the trajectory t is:

$$\alpha \sqcup \sqcup_t \beta = \{ a_1 a_2 a_3 b_1 b_2 a_4 a_5 a_6 b_3 a_7 b_4 a_8 b_5 \}$$

Observe that the result of the operation is generally a set of words, though in the case of shuffle on trajectory it is always a singleton or the empty set. Notice also that $\alpha \sqcup \sqcup_t \beta = \emptyset$ if $|\alpha| \neq |t|_0$ or $|\beta| \neq |t|_1$.

A set of trajectories is any set $T \subseteq V^*$. The shuffle of α with β on the set T , denoted by $\alpha \sqcup \sqcup_T \beta$, is

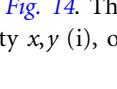
$$\alpha \sqcup \sqcup_T \beta = \bigcup_{t \in T} \alpha \sqcup \sqcup_t \beta \quad (1)$$

The shuffle on (sets of) trajectories generalizes several traditional word operations. Let, for example, $T = 0^* 1^*$. Then $\sqcup \sqcup_T = \cdot$, the operation of catenation.

To characterize the properties of tube languages described above, we define formally a *property* \mathcal{P} as a mapping $\mathcal{P} : 2^{\Sigma^*} \rightarrow \{\text{true, false}\}$. We say that a language L has (or satisfies) the property \mathcal{P} if $\mathcal{P}(L) = \text{true}$. The next definition introduces a general concept of *bond-free property* that covers surprisingly many types of undesired bonds studied in the literature.

Definition 5 (Kari et al. 2005b) Consider a language property \mathcal{P} . Let there be binary word operations \diamond_{lo} , \diamond_{up} and an involution θ such that for an arbitrary $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \text{true}$ iff

- (i) $\forall w \in \Sigma^+, x, y \in \Sigma^* ((w \diamond_{\text{lo}} x) \cap L \neq \emptyset, (\theta(w) \diamond_{\text{up}} y) \cap L \neq \emptyset) \Rightarrow xy = \lambda$, then \mathcal{P} is called a bond-free property (of degree 2)
- (ii) $\forall w, x, y \in \Sigma^* ((w \diamond_{\text{lo}} x) \cap L \neq \emptyset, (\theta(w) \diamond_{\text{up}} y) \cap L \neq \emptyset) \Rightarrow w = \lambda$, then \mathcal{P} is called a strictly bond-free property (of degree 2)

If not stated otherwise, we assume in the sequel that $\diamond_{\text{lo}} = \sqcup \sqcup_{T_{\text{lo}}}$ and $\diamond_{\text{up}} = \sqcup \sqcup_{T_{\text{up}}}$ for some sets of trajectories $T_{\text{lo}}, T_{\text{up}}$. Intuitively, w and $\theta(w)$ represent two complementary oligonucleotides. Then $w \sqcup \sqcup_{T_{\text{lo}}} x$ and $\theta(w) \sqcup \sqcup_{T_{\text{up}}} y$ represent two DNA single strands, which could form a double-stranded DNA molecule with blunt ends, depicted in  Fig. 14. The bond-free property \mathcal{P} guarantees that $w \sqcup \sqcup_{T_{\text{lo}}} x$ and $\theta(w) \sqcup \sqcup_{T_{\text{up}}} y$ with nonempty x, y (i), or w (ii), cannot simultaneously exist in L .

Theorem 5 (Kari et al. 2005b)

- (i) The language properties (B), (C), (D), (G), (H), (I), (M.1), (M.2) are bond-free properties.
- (ii) The language properties (A), strictly (B)–(D), strictly (G)–(I), (L), strictly (L) are strictly bond-free properties.

Moreover, in both cases, the associated sets of trajectories $T_{\text{lo}}, T_{\text{up}}$ are regular.

Proof Let θ be an antimorphism and let the sets of trajectories $T_{\text{lo}}, T_{\text{up}}$ corresponding to the listed bond-free properties be

- (A) $T_{\text{lo}} = T_{\text{up}} = 0^+$
- (B) $T_{\text{lo}} = 0^+, T_{\text{up}} = 1^*0^+1^*$
- (C) $T_{\text{lo}} = 0^+, T_{\text{up}} = 0^+1^*$
- (D) $T_{\text{lo}} = 0^+, T_{\text{up}} = 1^*0^+$
- (G) $T_{\text{lo}} = 0^+1^*, T_{\text{up}} = 1^*0^+$
- (H) $T_{\text{lo}} = 0^+1^*, T_{\text{up}} = 0^+1^*$
- (I) $T_{\text{lo}} = 1^*0^+, T_{\text{up}} = 1^*0^+$
- (L) $T_{\text{lo}} = T_{\text{up}} = 1^*0^k1^*$
- (L) strictly: $T_{\text{lo}} = T_{\text{up}} = 1^*0^k1^* \cup 0^+$

If θ is a morphism, one can similarly define

- (M.1) $T_{\text{lo}} = 0^*, T_{\text{up}} = 1^*0^*1^*$
- (M.2) $T_{\text{lo}} = 1^*0^+, T_{\text{up}} = 0^+1^*$

Consider, for example, the property (H), θ -3'-overhang-freedom. Then $w \sqcup \sqcup_{T_{\text{lo}}} x = \{wx\}$ and $\theta(w) \sqcup \sqcup_{T_{\text{up}}} y = \{\theta(w)y\}$. The relations in [Definition 5](#) (i) take the form $wx \in L, \theta(w)y \in L$ that corresponds to the definition of (H) above. The proofs of the other mentioned properties are analogous.

Observe that $T_{\text{lo}}, T_{\text{up}}$ for a certain property correspond to the “shape” of the bonds prohibited in languages satisfying the property. The above theorem allows for a general characterization of bond-free properties via a solution of an *unique* language inequation in Kari et al. ([2005b](#)). As a consequence, the following result can be obtained.

Theorem 6 (Kari et al. [2005b](#)) *Let \mathcal{P} be a (strictly) bond-free property associated with regular sets of trajectories $T_{\text{lo}}, T_{\text{up}}$. Then the following problem is decidable in quadratic time with respect to $|A|$:*

Input: an NFA A

Output: Yes/No depending on whether $L(A)$ satisfies \mathcal{P}

By [Theorem 5](#), the above result applies to the properties (A) – (D), (G) – (J), (M), strictly (B) – strictly (D), strictly (G) – strictly (J), (L), strictly (L), in the case of regular languages, on one hand. On the other hand, for a given context-free language L it is undecidable whether it satisfies certain bond-free properties, for example, (B) and (F).

Theorem 7 (Hussini et al. [2003](#)) *The following problem is undecidable.*

Input: A bond-free property \mathcal{P} associated with regular sets of trajectories $T_{\text{lo}}, T_{\text{up}}$, and a context-free language L

Output: Yes/No depending on whether $\mathcal{P}(L) = \text{true}$

An important problem studied in the literature is the *optimal negative design problem*: How to construct a non-crosshybridizing set (i.e., a set of single-stranded molecules which do not mutually hybridize) of a certain required size, given a fixed library of available molecules. In general, even to decide whether such a *finite* set exists is an NP-complete problem and its equivalence with the maximal independent set problem can be easily shown

(Deaton et al. 2003). Various heuristics were used to find a nearly optimal solution (Phan and Garzon 2008). Here we focus on a similar but in some cases easier *maximality problem* defined formally in [Sect. 5.2](#).

Theorem 8 (Kari et al. 2005b) *Let $M \subseteq \Sigma^+$ be a regular set of words, and $L \subseteq M$ a regular language satisfying a bond-free property \mathcal{P} .*

- (a) *Let θ be an antimorphism and let \mathcal{P} be one of the properties (B), (C), (D), (G), strictly (B) – strictly (D), strictly (G), (L), strictly (L), or*
- (b) *Let θ be a morphism and let \mathcal{P} be one of the properties (B), (C), (D), (H), (I), strictly (B) – strictly (D), strictly (H), strictly (I), (L), strictly (L).*

Then there is an algorithm deciding whether there is a $w \in M \setminus L$ such that $L \cup \{w\}$ satisfies \mathcal{P} .

Algorithms deciding the maximality of these properties can require an exponential time with respect to the size of an NFA accepting L . The same holds when we want to construct a maximal regular set of DNA strands satisfying these bond-free properties. In two important cases, however, a polynomial time can be achieved (Kari et al. 2005b): (1) for maximality of *regular* nonoverlapping sets satisfying the property (A), and (2) for maximality of *finite θ -compliant* sets satisfying the property (B).

5.4 Preventing Imperfect DNA–DNA Bonds

In [Sects. 5.2](#) and [5.3](#), properties of DNA codes are presented based on the assumption that hybridization binds only two perfectly WK complementary single-stranded DNA molecules. In reality, however, thermodynamical laws allow for hybridization also in cases of some “roughly” complementary molecules with certain irregularities in the WK complementarity sense. This section describes properties of languages that ensure that even such imperfect bindings can be described and eventually prevented.

As already mentioned, the negative design problem is rather computationally expensive when using thermodynamical methods. Various approximative methods using discrete similarity metrics have been therefore studied. In Marathe et al. (2001) and Tulpan et al. (2003), the authors considered codes K of length k satisfying the following property ($H(u, v)$ is the Hamming distance, i.e., the number of mismatches at corresponding positions, between words u and v of the same length).

X[d, k]: If u and v are any codewords in K then $H(u, \tau(v)) > d$.

In fact the above property is studied in conjunction with the uniqueness property $H(K) > d$ – here $H(K)$ is the smallest Hamming distance between any two different words in K .

Garzon et al. (1997) introduced the H -measure based on Hamming distance for two words x and y of length k and explained how this measure can be used to encode instances of the HPP. This measure was also used in Garzon et al. (2006) to search optimal codes for DNA computing using the shuffle operation on DNA strands. A similar measure was defined in Arita (2004) and Arita and Kobayashi (2002) extending the work of Frutos et al. (1997) and was applied to codes of length k whose words can be concatenated in arbitrary ways. Thus, the tube language here was $L = K^+$. The code K satisfied certain uniqueness conditions. In particular, the tube language $L = K^+$ satisfied the following property.

Y[d, k]: If x is a subword of L of length k and v is a codeword in K then $H(x, \tau(v)) > d$.

This property was considered also in Reif et al. (2002) for tube languages of the form $K_1 K_2 \dots K_m$ where each K_i is a certain code of length k .

Finally, Kari et al. (2005a) introduced the following property of a tube language L , motivated by the fact that the above defined properties \mathbf{X} , \mathbf{Y} still allow for certain types of undesired bonds between DNA codewords.

$\mathbf{Z}[d,k]$: If x and y are any subwords of L of length k then $H(x, \tau(y)) > d$.

The reader can observe that the property \mathbf{X} is a generalization of (A) from the previous section. Similarly, \mathbf{Y} is a generalization of (B) and \mathbf{Z} generalizes (L). Note that any set L satisfying property $\mathbf{Z}[d,k]$ also satisfies $\mathbf{Y}[d,k]$. Further relations among different bond-free properties using similarity measures were studied in Kari et al. (2005a).

The choice of the Hamming distance in the condition $H(x, \tau(y)) \leq d$ for *similarity* between words is a very natural one and has attracted a lot of interest in the literature. One might argue, however, that parts of two DNA molecules could form a stable bond even if they have different lengths. In Fig. 15, for example, the bound parts of the two molecules have lengths 10 and 9. Such hybridizations (and even more complex ones) were addressed in Andronescu et al. (2003). Based on this observation, the condition for two subwords x and y to bind together should be

$$|x|, |y| \geq k \quad \text{and} \quad \text{Lev}(x, \tau(y)) \leq d$$

The symbol $\text{Lev}(u, v)$ denotes the *Levenshtein distance* between the words u and v – this is the smallest number of substitutions, insertions, and deletions of symbols required to transform u into v .

To establish a general framework that would cover both the similarity measure H , Lev and possibly also others, Kari et al. (2005a) considered a general binary relation γ on words over Σ , that is, a subset of $\Sigma^* \times \Sigma^*$. The expression “ (u, v) is in γ ” can be rephrased as “ $\gamma(u, v)$ is true” when γ is viewed as a logic predicate. A binary relation is called *rational* if it can be realized by a finite transducer.

Definition 6 (Kari et al. 2005a) A binary relation sim is called a *similarity relation* with parameters (t, l) , where t and l are nonnegative integers, if the following conditions are satisfied.

- (i) If $\text{sim}(u, v)$ is true then $\text{abs}(|u| - |v|) \leq t$, where abs is the absolute value function.
- (ii) If $\text{sim}(u, v)$ is true and $|u|, |v| > l$ then there are proper subwords x and y of u and v , respectively, such that $\text{sim}(x, y)$ is true.

We can interpret the above conditions as follows: (1) the lengths of two similar words cannot be too different and (2) if two words are similar and long enough, then they contain two similar proper subwords.

The notation $H_{d,k}$ shall be used for the relation “ $|u|, |v| \geq k$ and $H(u, v) \leq d$ ”, and $\text{Lev}_{d,k}$ for “ $|u|, |v| \geq k$ and $\text{Lev}(u, v) \leq d$ ”. It is evident that the $H_{d,k}$ is an example of a rational similarity

Fig. 15

Two DNA molecules in which the parts 5' – AAGCGTTCGA – 3' and 5' – TCGGACGTT – 3' bind together although these parts have different lengths.



relation with parameters $(0, k)$. It is also easy to show that $Lev_{d,k}$ is a rational similarity relation as well, with parameters $(d, d + k)$.

Based on the above definition, for any similarity relation $\text{sim}(\cdot, \cdot)$ between words and for every involution θ , we define the following property of a language L with strong mathematical properties.

P[θ , sim]: If x and y are any nonempty subwords of L then $\text{sim}(x, \theta(y))$ is false.

Any language satisfying **P[θ , sim]** is called a (θ, sim) -bond-free language. Although this property seems to be quite general and covering many possible situations, it can be shown that it is only a special case of the strict bond-freeness defined in [Sect. 5.3](#).

Theorem 9 (Kari et al. 2005a) **P[θ , sim]** is a strictly bond-free property.

Proof The mappings sim_L and sim_R are defined as follows:

$$\begin{aligned}\text{sim}_L(y) &= \{x \in \Sigma^* \mid \text{sim}(x, y)\} \\ \text{sim}_R(x) &= \{y \in \Sigma^* \mid \text{sim}(x, y)\}\end{aligned}$$

Recall that a language L is (θ, sim) -bond-free iff

$$\begin{aligned}\forall x_1, y_1, x_2, y_2 \in \Sigma^*, w_1, w_2 \in \Sigma^+ \\ (x_1 w_1 y_1, x_2 w_2 y_2 \in L) \Rightarrow \text{not sim}(w_1, \theta(w_2)) \quad \text{iff} \\ \forall x_1, y_1, x_2, y_2 \in \Sigma^*, w_1, w_2 \in \Sigma^+ \\ (x_1 w_1 y_1, x_2 \theta(w_2) y_2 \in L) \Rightarrow \text{not sim}(w_1, w_2) \quad \text{iff} \\ \forall x_1, w_1, y_1, x_2, w_2, y_2 \in \Sigma^* \\ (x_1 w_1 y_1, x_2 \theta(w_2) y_2 \in L, w_2 \in \text{sim}_R(w_1)) \Rightarrow (w_1 = \lambda \text{ or } w_2 = \lambda) \quad \text{iff} \\ \forall x_1, y_1, x_2, y_2, w \in \Sigma^* \\ (\{x_1 w y_1\} \cap L \neq \emptyset, \{x_2\} \cdot \theta(\text{sim}_R(w) \cap \Sigma^+) \cdot \{y_2\} \cap L \neq \emptyset) \Rightarrow w = \lambda \quad \text{iff} \\ \forall x, y, w \in \Sigma^* \\ (w \sqcup_T x \cap L \neq \emptyset, \theta(\text{sim}_R(w)) \sqcup_T y \cap L \neq \emptyset) \Rightarrow w = \lambda\end{aligned}$$

where $T = 1^* 0^+ 1^*$.

Therefore, an expression corresponding to the definition of strictly bond-free property has been obtained. Notice that results analogous to [Theorem 9](#) could be proved also for the properties $X[d,k]$, $Y[d,k]$, and $Z[d,k]$. Observe, furthermore, that the operation on words w and y defined as $\theta(\text{sim}_R(w)) \sqcup_T y$ is “almost” \sqcup_T , and hence some results from [Sect. 5.3](#) are applicable in the case of (θ, sim) -bond-free languages, provided that the relation sim is “reasonable.”

Theorem 10 (Kari et al. 2005a) Let sim be a rational relation. The following problem is decidable in quadratic time with respect to $|A|$.

Input: An NFA A

Output: YES/NO, depending on whether $L(A)$ is a (θ, sim) -bond-free language

For the case where sim is one of the similarity relations $H_{d,k}$ or $Lev_{d,k}$ the algorithm runs at time $\mathcal{O}(dk|A|^2)$ (or $\mathcal{O}(dk^2|A|^2)$, respectively). The (θ, sim) -bond-freeness remains decidable

even in the case of context-free tube languages, although the existence of a polynomial-time algorithm cannot be guaranteed.

Two problems related to the design of large sets of bond-free molecules, the *optimal negative design problem* and the *maximality problem* have been addressed, too. Both were formally specified in previous sections. The optimal negative design problem remains NP complete even for *finite* tube languages in the case of various similarity metrics. The problem of maximality of *regular* (θ, sim) -bond-free languages has been shown decidable in Kari et al. (2005a), although the existence of a polynomial-time algorithm cannot be generally guaranteed. However, rather surprisingly, in the important Hamming case such an algorithm exists. One can consider languages that are subsets of $(\Sigma^k)^+$, for some positive integer k . We call such languages *k-block languages*. Naturally, any regular k -block language can be represented by a special type of lazy DFA (Wood 1987), which we call a *k-block DFA*.

Theorem 11 (Kari et al. 2005a) *Let d be fixed to be either 0 or 1. The following problem is computable in a polynomial time.*

Input: k -block DFA A such that $L(A)$ is a $(\theta, H_{d,k})$ -bond-free subset of $(\Sigma^k)^+$

Output: YES/NO, depending on whether the language $L(A)$ is maximal with that property.

Moreover, if $L(A)$ is not maximal, output a minimal-length word $w \in (\Sigma^k)^+ \setminus L(A)$ such that $L(A) \cup \{w\}$ is a $(\theta, H_{d,k})$ -bond-free subset of $(\Sigma^k)^+$.

In particular, the time complexity $t(|A|)$ is bounded as follows:

$$t(|A|) = \begin{cases} O(k|A|^3), & \text{if } k \text{ is odd and } d = 0 \\ O(|A|^6), & \text{if } k \text{ is even and } d = 0 \\ O(k^3|A|^6), & \text{if } d = 1 \end{cases}$$

The concept of (θ, sim) -bond-free languages is quite general and could cover also subtler similarity measures than the Hamming or Levenshtein distance. Consider the original nearest-neighbor thermodynamical approach to the hybridization problem (SantaLucia 1998). The calculation of ΔG_{\min} , the minimum free energy among the free energies of all possible secondary substructures that may be formed by the examined DNA sequences, is frequently used to determine the most likely secondary structure that will actually form. Assume secondary substructures of a size limited from above (say, of at most 25 bp), which is reasonable from the practical point of view. One can consider two DNA sequences *similar* if and only if they contain subsequences satisfying the condition $\Delta G_{\min} \geq B$, where B is a threshold value for hybridization energy (the “all or nothing” hybridization model). Such a similarity relation obviously fulfills the conditions of \blacktriangleright Definition 6 and, furthermore, it is rational (even finite). Therefore, for a fixed value of B , the hybridization analysis can possibly benefit from the above mentioned results and rapid algorithms.

We conclude with two examples of a construction of DNA codes in the Hamming case. The first example is based on the method of *k-templates* proposed originally in Arita and Kobayashi (2002). This method allows us to produce codes that are subsets of $(\Sigma^k)^+$.

Theorem 12 (Kari et al. 2005a) *Let I be a nonempty subset of $\{1, \dots, k\}$ of cardinality $m = \lfloor k/2 \rfloor + 1 + \lfloor (d + (k \bmod 2))/2 \rfloor$. Then the language K^+ is $(\tau, H_{d,k})$ -bond-free, where*

$$K = \{v \in \Sigma^k \mid \text{if } i \in I \text{ then } v[i] \in \{A, C\}\}$$

Observe that the size of the code K is $2^m 4^{k-m}$. An advantage of the method of [Theorem 12](#) is that we can construct $(\tau, H_{d,k})$ -bond-free languages with a large ratio d/k .

Another method is based on the operation of *subword closure* K^\otimes of a set $K \subseteq \Sigma^k$

$$K^\otimes = \{w \in \Sigma^* \mid |w| \geq k, \text{Sub}_k(w) \subseteq K\}$$

Denote further $K^\oplus \stackrel{\text{def}}{=} K^\otimes \cap (\Sigma^k)^+$. The following theorem characterizes all maximal $(\tau, H_{d,k})$ -bond-free subsets of $(\Sigma^k)^+$ and $\Sigma^k \Sigma^*$.

Theorem 13 (Kari et al. 2005a) *The class of all maximal $(\tau, H_{d,k})$ -bond-free subsets of $(\Sigma^k)^+$ is finite and equal to*

$$\{K^\oplus \mid K \text{ is a maximal } (\tau, H_{d,k})\text{-bond-free subset of } \Sigma^k\}.$$

The above theorem holds also for subsets of $\Sigma^k \Sigma^*$ if one replaces K^\oplus with K^\otimes . As a consequence, if one constructs a maximal finite subset K of Σ^k satisfying $\tau(K) \cap H_d(K) = \emptyset$, then the language K^\oplus is a maximal $(\tau, H_{d,k})$ -bond-free subset of $(\Sigma^k)^+$. Another implication of [Theorem 13](#) is that all maximal $(\tau, H_{d,k})$ -bond-free subsets of $(\Sigma^k)^+$ or $\Sigma^k \Sigma^*$ are regular.

To conclude, in [Sect. 5](#) several basic types of DNA interaction based on Watson–Crick complementarity are characterized, using the apparatus of formal language and automata theory. Besides its value as a contribution to theoretical computer science, the main application of this research is the description and construction of sets of DNA molecules (called also DNA codes) that are free of certain types of unwanted binding interactions. These codes are especially useful in DNA computing and many other laboratory techniques, which assume that an undesirable hybridization between DNA molecules does not occur. It has been shown that a uniform mathematical characterization exists for many types of DNA bonds, both perfect and imperfect with respect to the WK complementarity principle. This characterization, in turn, implies the existence of effective algorithms for manipulation and construction of these DNA codes.

6 Vectorial Models of DNA-Based Information

In [Sect. 5](#), we saw that formal language theory is a natural tool for modeling, analyzing, and designing “good” DNA codewords that control DNA strand inter- and intramolecular interactions based on Watson–Crick complementarity. This was achieved by formalizing a DNA single-strand in the 5'- 3' orientation as a linear word over the DNA alphabet {A, C, G, T}. A limitation of this representation of DNA single strand as words is that it does not model double-stranded DNA molecules, partially double-stranded DNA molecules (such as DNA strands with sticky ends), or interactions between DNA single strands that may lead to double strands.

This section offers an alternative natural way of representing DNA strands, namely as vectors. A (single, partially double-stranded, or fully double-stranded) DNA molecule is modeled namely by a vector whose first component is a word over the DNA alphabet representing the “top” strand, and the second component is a word over the DNA alphabet representing the “bottom” strand. Using this representation, one can naturally model the operations of annealing and ligation.

We now introduce two computational models that use this representation of DNA strands: a language generating device called *sticker system* ([Sect. 6.1](#)) (Freund et al. 1999; Kari et al.

1998; Păun and Rozenberg 1998), and its automata counterpart, *Watson–Crick automata* (Sect. 6.2) (Freund et al. 1999). A summary of essential results on these topics can be found in Păun et al. (1998).

We start by introducing a vectorial formalism of the notions of DNA single strand, double strand, and partial double strand, as well of the bio-operations of annealing (hybridization) and ligation. Other notions and notations that are used here were defined in Sect. 5.1.

The notion of Watson–Crick complementarity is formalized by a symmetric relation. A relation $\rho \subseteq \Sigma \times \Sigma$ is said to be *symmetric* if for any $a, b \in \Sigma$, $(a, b) \in \rho$ implies $(b, a) \in \rho$. In order to define a symmetric relation ρ , it suffices to specify one of (a, b) and (b, a) as long as we explicitly note that ρ is symmetric.

DNA strands are modeled by 2×1 vectors, wherein the first row corresponds to the “top” DNA strand and the second row corresponds to the “bottom” DNA strand. In this formalism, DNA double strands are modeled as 2×1 vectors in square brackets where the top word is in relation ρ with the bottom word, while DNA single strands are modeled as 2×1 vectors in round brackets where one of the rows is the empty word. More concretely, we write $\begin{bmatrix} a \\ b \end{bmatrix}_\rho$ if two letters a, b are in the relation ρ . Whenever ρ is clear from context, the subscript ρ is omitted. We now define an alphabet of double-stranded columns

$$\Sigma_d = \left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_\rho \cup \left(\begin{array}{c} \Sigma \\ \lambda \end{array} \right) \cup \left(\begin{array}{c} \lambda \\ \Sigma \end{array} \right)$$

where $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_\rho = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \middle| a, b \in \Sigma, (a, b) \in \rho \right\}$, $\left(\begin{array}{c} \Sigma \\ \lambda \end{array} \right) = \left\{ \begin{pmatrix} a \\ \lambda \end{pmatrix} \middle| a \in \Sigma \right\}$, and $\left(\begin{array}{c} \lambda \\ \Sigma \end{array} \right) = \left\{ \begin{pmatrix} \lambda \\ b \end{pmatrix} \middle| b \in \Sigma \right\}$.

The *Watson–Crick domain* associated to Σ and ρ is the set $\text{WK}_\rho(\Sigma)$ defined as $\text{WK}_\rho(\Sigma) = \left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_\rho^*$. An element $\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \dots \begin{bmatrix} a_n \\ b_n \end{bmatrix} \in \text{WK}_\rho(\Sigma)$ can be written as $\begin{bmatrix} a_1 a_2 \dots a_n \\ b_1 b_2 \dots b_n \end{bmatrix}$ succinctly. Note that $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ means no more than a pair of words w_1, w_2 , whereas $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ imposes that $|w_1| = |w_2|$ and their corresponding letters are complementary in the sense of the relation ρ . Elements of $\text{WK}_\rho(\Sigma)$ are called *complete double-stranded sequences* or *molecules*. Moreover, we denote $\text{WK}_\rho^+(\Sigma) = \text{WK}_\rho(\Sigma) \setminus \{(\lambda, \lambda)\}$.

Note that elements of $\text{WK}_\rho(\Sigma)$ are fully double-stranded. In most DNA computing experiments, for example, Adleman’s first experiment, partially double-stranded DNA, that is, DNA strands with sticky ends, are essential. To introduce sticky ends in the model, let $S(\Sigma) = \left(\begin{array}{c} \lambda \\ \Sigma^* \end{array} \right) \cup \left(\begin{array}{c} \Sigma^* \\ \lambda \end{array} \right)$ be the set of sticky ends. Then we define a set $W_\rho(\Sigma)$ whose elements are molecules with sticky ends at both sides as $W_\rho(\Sigma) = L_\rho(\Sigma) \cup R_\rho(\Sigma) \cup LR_\rho(\Sigma)$, where

$$L_\rho(\Sigma) = S(\Sigma) \text{WK}_\rho(\Sigma)$$

$$R_\rho(\Sigma) = \text{WK}_\rho(\Sigma) S(\Sigma)$$

$$LR_\rho(\Sigma) = S(\Sigma) \text{WK}_\rho^+(\Sigma) S(\Sigma)$$

Note that unlike an element in $L_\rho(\Sigma)$ or $R_\rho(\Sigma)$, elements of $LR_\rho(\Sigma)$ must have at least one “column” $\begin{bmatrix} a \\ b \end{bmatrix}$. Any element of $W_\rho(\Sigma)$ with at least a position $\begin{bmatrix} a \\ b \end{bmatrix}$, $a \neq \lambda$, $b \neq \lambda$, is called a *well-started (double stranded) sequence*. Thus, $LR_\rho(\Sigma)$ is equivalent to the set of all well-started sequences.

Annealing and ligation of DNA molecules can be modeled as a partial operation among elements of $W_\rho(\Sigma)$. A well-started molecule can be prolonged to the right or to the left with another molecule, provided that their sticky ends match. We define “sticking y to the right of x ” operation, denoted by $\mu_r(x, y)$. In a symmetric way, $\mu_\ell(y, x)$ (sticking y to the left of x) is defined. Let $x \in LR_\rho(\Sigma)$ and $y \in W_\rho(\Sigma)$. Being well-started, $x = x_1x_2x_3$ for some $x_1, x_3 \in S(\Sigma)$, $x_2 \in WK_\rho^+(\Sigma)$. Then $\mu_r(x, y)$ is defined as follows (also see Fig. 16):

Case A If y is single-stranded, that is, $y \in S(\Sigma)$, we have the following cases: for $r, p \geq 0$,

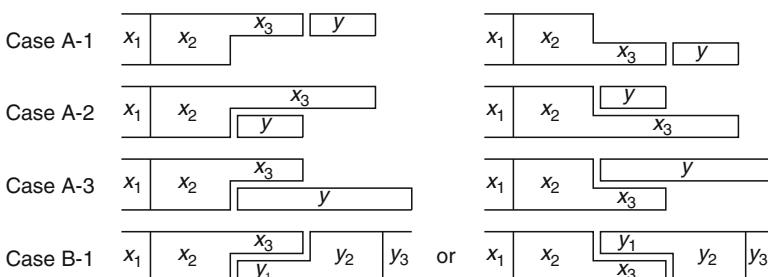
1. If $x_3 = \begin{pmatrix} a_1 \cdots a_r \\ \lambda \end{pmatrix}$ and $y = \begin{pmatrix} a_{r+1} \cdots a_{r+p} \\ \lambda \end{pmatrix}$, then $\mu_r(x, y) = x_1x_2 \begin{pmatrix} a_1 \cdots a_r \\ a_{r+1} \cdots a_{r+p} \end{pmatrix}$.
2. If $x_3 = \begin{pmatrix} a_1 \cdots a_r \\ \lambda \end{pmatrix} \begin{pmatrix} a_{r+1} \cdots a_{r+p} \\ \lambda \end{pmatrix}$, $y = \begin{pmatrix} \lambda \\ b_1 \cdots b_r \end{pmatrix}$, and $(a_i, b_i) \in \rho$ for $1 \leq i \leq r$, then $\mu_r(x, y) = x_1x_2 \begin{pmatrix} a_1 \cdots a_r \\ b_1 \cdots b_r \end{pmatrix} \begin{pmatrix} a_{r+1} \cdots a_{r+p} \\ \lambda \end{pmatrix}$.
3. If $x_3 = \begin{pmatrix} a_1 \cdots a_r \\ \lambda \end{pmatrix}$, $y = \begin{pmatrix} \lambda \\ b_1 \cdots b_r \end{pmatrix} \begin{pmatrix} \lambda \\ b_{r+1} \cdots b_{r+p} \end{pmatrix}$, and $(a_i, b_i) \in \rho$ for $1 \leq i \leq r$, then $\mu_r(x, y) = x_1x_2 \begin{pmatrix} a_1 \cdots a_r \\ b_1 \cdots b_r \end{pmatrix} \begin{pmatrix} \lambda \\ b_{r+1} \cdots b_{r+p} \end{pmatrix}$.
4. The counterparts of cases A1 – A3 where the roles of upper and lower strands are reversed.

Case B If y is well-started (partially double-stranded), that is, $y = y_1y_2y_3$ for some $y_1, y_3 \in S(\Sigma)$ and $y_2 \in WK_\rho^+(\Sigma)$: for $r \geq 0$,

1. If $x_3 = \begin{pmatrix} a_1 \cdots a_r \\ \lambda \end{pmatrix}$, $y_1 = \begin{pmatrix} \lambda \\ b_1 \cdots b_r \end{pmatrix}$, and $(a_i, b_i) \in \rho$ for $1 \leq i \leq r$, then $\mu_r(x, y) = x_1x_2 \begin{pmatrix} a_1 \cdots a_r \\ b_1 \cdots b_r \end{pmatrix} y_2y_3$.
2. The counterpart of case B1 when the roles of upper and lower strands are reversed.

Fig. 16

Sticking operations: Prolongation of a well-started molecule to the right.



If none of these cases applies, $\mu_r(x, y)$ is undefined. Note that in all cases, r can be 0, that is, the system is allowed to prolong the blunt ends of molecules. Moreover, we have $\mu_r\left(x, \binom{\lambda}{\lambda}\right) = \mu_\ell\left(\binom{\lambda}{\lambda}, x\right) = x$ for any $x \in LR_\rho(\Sigma)$.

Note also that this vectorial model of DNA molecules allows – unlike its linear counterpart presented in the previous section – the differentiation between single-stranded DNA molecules, double-stranded DNA molecules, and DNA molecules with sticky ends, as well as for modeling of DNA–DNA interactions such as annealing and ligation.

6.1 Sticker Systems

Sticker systems are formal models of molecular interactions occurring in DNA computing, based on the sticking operation. Several variants of sticker systems have been defined in the literature. In this section, the simple regular sticker system, which is the most realistic variant but which is weak in terms of generating capacity, is described. We also introduce two practical ways to strengthen this variant: by using some complex DNA structures (☞ Sect. 6.1.1), and by observing the sticker system externally (☞ Sect. 6.1.2). Both enhance the computational power of the sticker system to Turing universality.

A sticker system prolongs given well-started DNA molecules, both to the left and to the right, by using the sticking operation, so as to turn them into complete double-stranded molecules. This system is an appropriate model of molecular interaction occurring, for example, in Adleman’s 1994 experiment, and was proposed under the name *bidirectional sticker system* (Freund et al. 1998).

A (bidirectional) *sticker system over a relation* ρ is a 4-tuple $\gamma = (\Sigma, \rho, A, P)$, where Σ is an alphabet endowed with the symmetric relation $\rho \subseteq \Sigma \times \Sigma$, $A \subseteq LR_\rho(\Sigma)$ is a finite subset of well-started sequences (axioms), and P is a finite subset of $W_\rho(\Sigma) \times W_\rho(\Sigma)$. Starting from an axiom in A , the system prolongs it with using a pair in P as follows:

$$x \Rightarrow_\gamma w \text{ iff } w = \mu_r(\mu_\ell(y, x), z) \text{ for some } (y, z) \in P$$

In other words, $x \Rightarrow_\gamma w$ iff sticking y to the left and z to the right of x results in w . The reflexive and transitive closure of \Rightarrow_γ is denoted by \Rightarrow_γ^* . A sequence $x_1 \Rightarrow_\gamma x_2 \Rightarrow_\gamma \dots \Rightarrow_\gamma x_k$, $x_1 \in A$, is called a *computation* in γ (of length $k - 1$). A computation as above is *complete* if $x_k \in WK_\rho^+(\Sigma)$.

The set of all molecules over Σ generated by complete computations in γ is defined as $LM(\gamma) = \{w \in WK_\rho^+(\Sigma) \mid x \Rightarrow_\gamma^* w, x \in A\}$. We can also consider the sticker systems as a generator of languages of strings rather than double-stranded molecules. To this aim, the following language is associated with $LM(\gamma)$

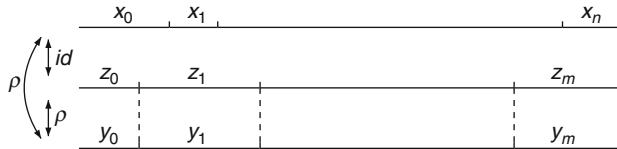
$$L(\gamma) = \left\{ w \in \Sigma^* \mid \begin{bmatrix} w \\ w' \end{bmatrix}_\rho \in LM(\gamma) \text{ for some } w' \in \Sigma^* \right\}$$

A language L is called a *sticker language* if there exists a sticker system γ such that $L(\gamma) = L$.

A sticker system $\gamma = (\Sigma, \rho, A, P)$ is said to be *simple* (respectively *regular*) if for each pair $(y, z) \in P$, $y, z \in S(\Sigma)$ (respectively $y = \lambda$). A simple regular system extends either the upper or lower strand, one at a time (hence the attribute “simple”), and only to the right (hence the attribute “regular”). Thus, a simple regular sticker system can be rewritten as 5-tuple $(\Sigma, \rho, A,$

Fig. 17

The idea of the proof of [Theorem 14](#). For any sticker system γ over a relation ρ , one can construct a sticker system γ' over the identity relation, such that $L(\gamma) = L(\gamma')$. The newly constructed sticker system γ' , based on id , simulates the process of γ to generate $(x, y) \in WK_{\rho}^{+}(\Sigma)$ where $x = x_0 x_1 \dots x_n$ and $y = y_0 y_1 \dots y_m$, by generating $(x, z) \in WK_{id}^{+}(\Sigma)$, where $z = z_0 z_1 \dots z_m$.



D_w and D_ℓ), where D_u and D_ℓ are finite subsets of $\binom{\Sigma^*}{\lambda}$ and $\binom{\lambda}{\Sigma^*}$, respectively. The family of languages generated by simple regular sticker systems is denoted by $SRL(n)$, where n means “no-restriction.” This variant is the most precise and realistic model of the annealing/ligation-based hybridization occurring in Adleman’s experiment. In fact, since being proposed in Kari et al. (1998), this type of sticker system has been intensively investigated.

A normal form for sticker systems with respect to the relation ρ was introduced in Hoogeboom and van Vugt (2000) for simple regular variants, and in Kuske and Weigel (2004) for general sticker systems. It says that the identity relation id suffices to generate any sticker language.

Theorem 14 (Hoogeboom and van Vugt 2000; Kuske and Weigel 2004) *For a sticker system γ over a relation ρ , one can construct a sticker system γ' over the identity relation id such that $L(\gamma) = L(\gamma')$.*

Proof The ideas proposed in Hoogeboom and van Vugt (2000) and Kuske and Weigel (2004) are essentially the same; they work for arbitrary bidirectional sticker systems. Here we present their proofs applied to *regular* (unidirectional) sticker systems to suggest the fact that the identity relation suffices also for Watson–Crick automata introduced later.

Let $\gamma = (\Sigma, \rho, A, P_r)$ be a regular sticker system. We construct a regular sticker system $\gamma' = (\Sigma, id, A', P'_r)$, [Fig. 17](#), where

$$\begin{aligned} A' &= \left\{ \begin{pmatrix} x_0 \\ z_0 \end{pmatrix} \mid \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \in A \text{ for some } y_0 \text{ such that } \begin{bmatrix} z_0 \\ y_0 \end{bmatrix}_\rho \right\} \\ P'_r &= \left\{ \begin{pmatrix} x_i \\ z_i \end{pmatrix} \mid \begin{pmatrix} x_i \\ y_i \end{pmatrix} \in P_r \text{ for some } y_i \text{ such that } \begin{bmatrix} z_i \\ y_i \end{bmatrix}_\rho \right\} \end{aligned}$$

Assume that $x \in L(\gamma)$, that is, there is a word y such that $(x, y) \in WK_{\rho}^{+}(\Sigma)$, $x = x_0 x_1 \dots x_n$ and $y = y_0 y_1 \dots y_m$, where $(x_0, y_0) \in A$, and $(x_j, y_j) \in P'_r$ ($1 \leq j \leq m$). Due to the symmetric property of ρ , x can also be written as the catenation of words z_0 and $z_1, \dots, z_m \in \Sigma^*$ such that $(z_k, y_k) \in WK_{\rho}(\Sigma)$ ($0 \leq k \leq m$). According to the definition of A' , $(x_0, z_0) \in A'$, and $(x_j, z_j) \in P'_r$ ($1 \leq j \leq m$). As a result, $\begin{bmatrix} x_0 x_1 \dots x_n \\ z_0 z_1 \dots z_m \end{bmatrix}_{id} = \begin{bmatrix} x \\ x \end{bmatrix}_{id} \in LM(\gamma')$, and hence $x \in L(\gamma')$. The proof that $L(\gamma') \subseteq L(\gamma)$ is similar.

As mentioned in the previous proof, this theorem proved to be valid for general sticker systems in Kuske and Weigel (2004). Moreover, in the paper, the authors show that an analogous result holds even for Watson–Crick automata, that is, the identity relation suffices for WK-automata. From a historical viewpoint of the theory of computation, the normal forms for grammars and acceptors have proved to be useful tools. Analogously, Theorem 14 will be useful for several proofs in the rest of this section.

The simple regular sticker system is one of the most “natural” computational models for annealing/ligation-based hybridization. Kari et al. (1998) initiated an investigation into the generative capacity of general sticker systems, including the simple regular variant, and the investigation continued in Freund et al. (1998) and Păun and Rozenberg (1998). The conclusion was that some classes of sticker systems can even characterize the recursively enumerable languages. On the contrary, the simple regular variant turned out to be quite weak.

Theorem 15 (Kari et al. 1998; Păun et al. 1998) $\text{SRSL}(n) \subsetneq \text{REG} = \text{Cod}(\text{SRSL}(n))$.

Thus, this “natural variant” of sticker systems has no more generative power than finite automata, even with the aid of encoding.

In Kari et al. (1998), the notion of *fair computation* was proposed. Let γ be a simple regular sticker system. A complete computation in γ is said to be *fair* if through the computation, the number of extensions occurring on the upper strand is equal to the number of extensions occurring on the lower strand. A language L is called a *fair sticker language* if there exists a simple regular sticker system γ such that L is the set of all words that are generated by fair computations in γ . The family of fair sticker languages is denoted by $\text{SRSL}(f)$.

It is known that $\text{REG} \subsetneq \text{Cod}(\text{SRSL}(f))$ and also that we cannot obtain characterizations of RE starting from languages in $\text{SRSL}(f)$ and using an arbitrary generalized sequential machine (*gsm*) mapping, including a coding (see Păun et al. 1998)). Hence the question of whether $\text{SRSL}(f)$ is included in CF (or even in LIN) arose. The answer to this question was obtained in Hoogeboom and van Vugt (2000). First, their example can be introduced to show that $\text{SRSL}(f) \not\subseteq \text{LIN}$.

Example 4 Let $\gamma = (\{a, b, c, d\}, \rho, A, D_u, D_\ell)$ be a simple regular sticker system, where $\rho = \{(a, a), (b, b), (b, c), (d, d)\}$, $A = \left\{ \begin{bmatrix} d \\ d \end{bmatrix} \right\}$, $D_u = \left\{ \begin{pmatrix} aa \\ \lambda \end{pmatrix}, \begin{pmatrix} b \\ \lambda \end{pmatrix} \right\}$, and $D_\ell = \left\{ \begin{pmatrix} \lambda \\ a \end{pmatrix}, \begin{pmatrix} \lambda \\ bc \end{pmatrix} \right\}$. This is a technical modification of Example 2 from Hoogeboom and van Vugt (2000) in order to make an axiom well started. Then $LM_n(\gamma) = \begin{bmatrix} d \\ d \end{bmatrix} \left\{ \begin{bmatrix} aa \\ aa \end{bmatrix}, \begin{bmatrix} bb \\ bc \end{bmatrix} \right\}^*$, and hence $L_n(\gamma) = d\{aa, bb\}^*$, $L_f(\gamma) = \{x \in L_n(\gamma) \mid \#_a(x) = \#_b(x)\}$. The pumping lemma for linear languages can be used to prove that $L_f(\gamma)$ is not linear.

Theorem 16 (Hoogeboom and van Vugt 2000) $\text{SRSL}(f) \subsetneq \text{Cod}(\text{SRSL}(f)) \subsetneq \text{CF}$.

Kari et al. (1998) imposed an additional constraint on fair computations called *coherence*, the use of which leads to a representation of RE. In the rest of this section, two new approaches are introduced to the problem of how to obtain characterizations of RE by using sticker systems augmented with more practical assumptions. Regarding the generative capacity and further topics about sticker systems, the reader is referred to the thorough summary in chapter 4 of Păun et al. (1998).

6.1.1 Sticker Systems with Complex Structures

Sakakibara and Kobayashi (2001) proposed a novel use of stickers, that involved the formation of *DNA hairpins*. In [Sect. 5](#), a hairpin was modeled as a linear word $w_1a_1\dots a_nw_2b_{n+1}\dots b_1w_3$, where $(a_i, b_i) \in \rho$, $1 \leq i \leq n$. Here we represent the same hairpin vectorially as $\begin{pmatrix} \langle w_2 \rangle \\ w_1 | w_3 \end{pmatrix} \in \begin{pmatrix} \Sigma^* \\ \Sigma^* \end{pmatrix}$ as shown in [Fig. 18](#). This hairpin-shaped molecule may stick to other molecules by its two sticky ends w_1 and w_3 or by its loop part w_2 . The sets of this type and inverted type of hairpins are denoted by $T_u(\Sigma)$ and $T_\ell(\Sigma)$, respectively, that is,

$$T_u(\Sigma) = \left\{ \begin{pmatrix} \langle w_2 \rangle \\ w_1 | w_3 \end{pmatrix} \mid w_1, w_2, w_3 \in \Sigma^* \right\}, \quad T_\ell(\Sigma) = \left\{ \begin{pmatrix} w_1 | w_3 \\ \langle w_2 \rangle \end{pmatrix} \mid w_1, w_2, w_3 \in \Sigma^* \right\},$$

The operation of “sticking a hairpin $x = \begin{pmatrix} \langle w_2 \rangle \\ w_1 | w_3 \end{pmatrix}$ onto a single-stranded molecule y ” is defined whenever $y = y_1y_2y_3$, and y_2 is complementary to w_1w_3 as follows:

$$\mu(y, x) = \begin{pmatrix} \lambda & \langle w_2 \rangle \\ \begin{pmatrix} \lambda \\ y_1 \end{pmatrix} & \begin{bmatrix} t_1 | v_1 \\ z_1 \end{bmatrix} & \begin{bmatrix} t_2 | v_2 \\ z_2 \end{bmatrix} & \cdots & \begin{bmatrix} t_n | v_n \\ z_n \end{bmatrix} & \begin{pmatrix} \lambda \\ z_{n+1}z_{n+2} \end{pmatrix} \end{pmatrix}.$$

Moreover, this operation $\mu(y, x)$ is extended to the general case; for a molecule,

$$z = \begin{pmatrix} \lambda & \langle u_1 \rangle & \langle u_2 \rangle & \cdots & \langle u_n \rangle & \lambda \\ \begin{pmatrix} \lambda \\ z_0 \end{pmatrix} & \begin{bmatrix} t_1 | v_1 \\ z_1 \end{bmatrix} & \begin{bmatrix} t_2 | v_2 \\ z_2 \end{bmatrix} & \cdots & \begin{bmatrix} t_n | v_n \\ z_n \end{bmatrix} & \begin{pmatrix} \lambda \\ z_{n+1}z_{n+2} \end{pmatrix} \end{pmatrix}$$

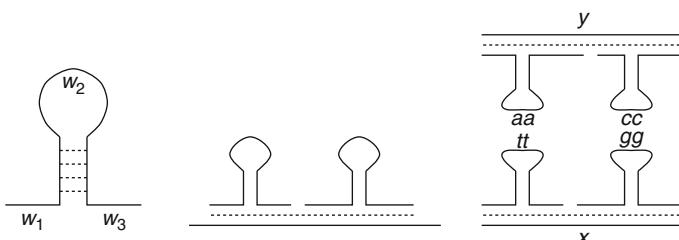
such that w_1w_3 is complementary to z_{n+1} , $\mu(z, x)$ is defined as

$$\mu(z, x) = \begin{pmatrix} \lambda & \langle u_1 \rangle & \langle u_2 \rangle & \cdots & \langle u_n \rangle & \langle w_2 \rangle & \lambda \\ \begin{pmatrix} \lambda \\ z_0 \end{pmatrix} & \begin{bmatrix} t_1 | v_1 \\ z_1 \end{bmatrix} & \begin{bmatrix} t_2 | v_2 \\ z_2 \end{bmatrix} & \cdots & \begin{bmatrix} t_n | v_n \\ z_n \end{bmatrix} & \begin{bmatrix} w_1 | w_3 \\ z_{n+1} \end{bmatrix} & \begin{pmatrix} \lambda \\ z_{n+2} \end{pmatrix} \end{pmatrix}.$$

Thus, this sticking operation forms “multiple hairpins” structures as illustrated in [Fig. 18](#) (Middle). The set of molecules with this type of structures and the set of molecules with inverted structures are denoted by $TW_u(\Sigma)$ and $TW_\ell(\Sigma)$, respectively, that is,

Fig. 18

(Left) A hairpin that the word $w_1a_1\dots a_nw_2b_{n+1}\dots b_1w_3$ may form if $(a_i, b_i) \in \rho$, $1 \leq i \leq n$; the sticky ends w_1, w_3 or the hairpin loop w_2 can bind to other molecules; (Middle) two hairpins stick to the lower strand via their sticky ends, leaving sticky ends at both ends of the lower strand; (Right) two “complete” molecules can bind together via their hairpin loops, if they are matched as shown.



$$\text{TW}_u(\Sigma) = \left(\begin{array}{c} \lambda \\ \lambda \\ \Sigma^* \end{array} \right) \left(\begin{bmatrix} \langle \Sigma^* \rangle \\ \Sigma^* \\ \Sigma^* \end{bmatrix} \right)^* \left(\begin{array}{c} \lambda \\ \lambda \\ \Sigma^* \end{array} \right), \quad \text{TW}_\ell(\Sigma) = \left(\begin{array}{c} \Sigma^* \\ \lambda \\ \lambda \end{array} \right) \left(\begin{bmatrix} \Sigma^* \\ \Sigma^* \\ \langle \Sigma^* \rangle \end{bmatrix} \right)^* \left(\begin{array}{c} \Sigma^* \\ \lambda \\ \lambda \end{array} \right).$$

Similarly, the hybridization operation $\mu(z, x)$ is defined for $z \in \text{TW}_\ell(\Sigma)$ and $x \in T_\ell(\Sigma)$. Note that $\begin{pmatrix} \lambda \\ \lambda \\ x \end{pmatrix} \in \text{TW}_\ell(\Sigma)$ or $\begin{pmatrix} x \\ \lambda \\ \lambda \end{pmatrix} \in \text{TW}_u(\Sigma)$ can be regarded as a word $x \in \Sigma^*$. Hence we will give also a word as a first argument of μ in the following. Now we can define another type of “complete” molecules, namely elements in the set $\text{TWK}(\Sigma) = \text{TWK}_u(\Sigma) \cup \text{TWK}_\ell(\Sigma)$, where

$$\text{TWK}_u(\Sigma) = \left(\begin{bmatrix} \langle \Sigma^* \rangle \\ \Sigma^* \\ \Sigma^* \end{bmatrix} \right)^*, \quad \text{TWK}_\ell(\Sigma) = \left(\begin{bmatrix} \Sigma^* \\ \Sigma^* \\ \langle \Sigma^* \rangle \end{bmatrix} \right)^*.$$

Second, we consider another sticking operation for the loop (w_2 of x) of a hairpin, which is illustrated in  Fig. 18 (Right). Consider two complete molecules $x \in \text{TWK}_u(\Sigma)$ and $y \in \text{TWK}_\ell(\Sigma)$ defined as

$$x = \left(\begin{bmatrix} \langle s_1 \rangle \\ r_1 | t_1 \\ x_1 \end{bmatrix} \quad \begin{bmatrix} \langle s_2 \rangle \\ r_2 | t_2 \\ x_2 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} \langle s_n \rangle \\ r_n | t_n \\ x_n \end{bmatrix} \right), \quad y = \left(\begin{bmatrix} \gamma_1 \\ u_1 | w_1 \\ \langle v_1 \rangle \end{bmatrix} \quad \begin{bmatrix} \gamma_2 \\ u_2 | w_2 \\ \langle v_2 \rangle \end{bmatrix} \quad \cdots \quad \begin{bmatrix} \gamma_n \\ u_n | w_n \\ \langle v_n \rangle \end{bmatrix} \right).$$

When s_i is complementary to v_i ($1 \leq i \leq n$), $\phi(x, y)$ is defined as

$$\phi(x, y) = \left(\begin{bmatrix} \gamma_1 \\ u_1 | w_1 \\ \langle v_1 \rangle \\ \langle s_1 \rangle \\ r_1 | t_1 \\ x_1 \end{bmatrix} \quad \begin{bmatrix} \gamma_2 \\ u_2 | w_2 \\ \langle v_2 \rangle \\ \langle s_2 \rangle \\ r_2 | t_2 \\ x_2 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} \gamma_n \\ u_n | w_n \\ \langle v_n \rangle \\ \langle s_n \rangle \\ r_n | t_n \\ x_n \end{bmatrix} \right).$$

Thus, for two complete molecules $x \in \text{TWK}_u(\Sigma)$ and $y \in \text{TWK}_\ell(\Sigma)$, $\phi(x, y)$ is well defined to be an element of the set

$$\text{DTWK}(\Sigma) = \left(\begin{bmatrix} \Sigma^* \\ \Sigma^* \\ \langle \Sigma^* \rangle \\ \langle \Sigma^* \rangle \\ \Sigma^* \\ \Sigma^* \end{bmatrix} \right)^*.$$

A *sticker system with complex structures* is a 4-tuple $\gamma = (\Sigma, \rho, D_u, D_\ell)$, where D_u and D_ℓ are finite subsets of $T_u(\Sigma)$ and $T_\ell(\Sigma)$, respectively. For molecules $x, y \in \text{TW}_u(\Sigma)$, we write $x \Rightarrow_u y$ if $y = \mu(x, v)$ for some $v \in D_u$. Analogously, for $x', y' \in \text{TW}_\ell(\Sigma)$, we write $x' \Rightarrow_\ell y'$ if $y' = \mu(x', v')$ for some $v' \in D_\ell$. The reflexive and transitive closures of these operations are denoted by \Rightarrow_u^* and \Rightarrow_ℓ^* .

A sequence $x_1 \Rightarrow_\alpha x_2 \Rightarrow_\alpha \cdots \Rightarrow_\alpha x_k$, with $x_i \in \Sigma^*$ and $\alpha \in \{u, \ell\}$, is called a computation in γ . (In this context, x_1 , the start of the computation, can be regarded as either a word in Σ^* or an element of $\text{TW}_\alpha(\Sigma)$.) A computation $x_1 \Rightarrow_\alpha^* x_k$ is said to be *complete* when $x_k \in \text{TWK}_\alpha(\Sigma)$.

Suppose that one has two complete computations $x \Rightarrow_u^* y$ and $x \Rightarrow_\ell^* z$ for a word $x \in \Sigma^*$. When $\phi(y, z)$ becomes a complete matching, that is, $\phi(y, z) \in \text{DTWK}(\Sigma)$, this computation process is said to be *successful*. The following language is associated with γ :

$$\begin{aligned} L(\gamma) = \{x \in (\Sigma \setminus \{\#\})^* \mid & x \# \Rightarrow_u^* y, y \in \text{TWK}_u(\Sigma) \\ & x \# \Rightarrow_\ell^* z, z \in \text{TWK}_\ell(\Sigma), \text{ and } \phi(y, z) \in \text{DTWK}(\Sigma)\}. \end{aligned}$$

Thus, one can consider this system as a language-accepting device. The family of languages accepted by sticker systems with complex structures is denoted by SLDT.

Now it can be shown that the use of hairpins enables one to characterize RE based on the following lemma.

Lemma 1 (Kari et al. 1998) *For each recursively enumerable language $L \subseteq \Sigma^*$, there exist two λ -free morphisms $h_1, h_2 : \Sigma_2^* \rightarrow \Sigma_1^*$, a regular language $R \subseteq \Sigma_1^*$, and a projection $\text{pr}_\Sigma : \Sigma_1^* \rightarrow \Sigma^*$ such that $L = \text{pr}_\Sigma(h_1(\text{EQ}(h_1, h_2)) \cap R)$.*

Theorem 17 (Sakakibara and Kobayashi 2001) *Every recursively enumerable language is the weak coding of a language in the family SLDT.*

Proof Let $L \in \text{RE}$. Due to **Lemma 1**, L can be obtained from $h_1(\text{EQ}(h_1, h_2)) \cap R$ by a projection, where $h_1, h_2 : \Sigma_2^* \rightarrow \Sigma_1^*$ are λ -free morphisms, and $R \in \text{REG}$. Hence it suffices to construct a sticker system with complex structures γ that accepts an *encoding* of $h_1(\text{EQ}(h_1, h_2)) \cap R$. Consider a complete deterministic finite automaton $M = (Q, \Sigma_1, \delta, q_0, F)$ for R , where $Q = \{q_0, q_1, \dots, q_m\}$. Any word $w = b_1 b_2 \dots b_k$ ($b_i \in \Sigma_1$) is encoded uniquely as $q_{l_0} b_1 q_{l_1} q_{l_1} b_2 q_{l_2} \dots q_{l_{k-1}} b_k q_{l_k} q_{l_k}$, where $q_{l_0} = q_0$, $q_{l_1}, \dots, q_{l_k} \in Q$ such that $\delta(q_{l_{j-1}}, b_j) = q_{l_j}$ for $1 \leq j \leq k$. So $w \in R$ iff $q_{l_k} \in F$.

Let $\Sigma_2 = \{a_1, \dots, a_n\}$, and for each a_i let $h_1(a_i) = c_1 c_2 \dots c_{k_i}$ ($c_j \in \Sigma_1$). Note that for an arbitrary state in Q , there is a unique transition on M by $h_1(a_i)$ because M is a complete deterministic automaton. Thus, a set of encodings of all such transitions is defined for each $a_i \in \Sigma_2$ as follows:

$$T_1(h_1(a_i)) = \bigcup_{q_{l_0} \in Q} \{q_{l_0} b_1 q_{l_1} \dots q_{l_{k_i-1}} b_{k_i} q_{l_{k_i}} \mid \delta(q_{l_{j-1}}, b_j) = q_{l_j}, 1 \leq j \leq k_i\}$$

Following the same idea, $T_2(h_2(a_i))$ is defined for each $a_i \in \Sigma_2$. Now γ is constructed as $(\Sigma_1 \cup \Sigma_2 \cup Q \cup \{\#\}, \text{id}, D_u \cup D_\ell)$, where

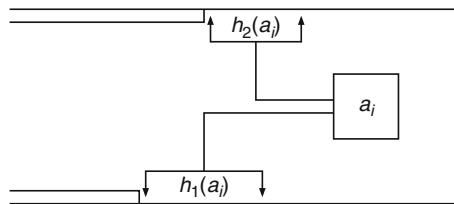
$$\begin{aligned} D_\ell &= \left\{ \left(\begin{array}{c} t_2 \\ \langle a_i \rangle \end{array} \right) \middle| t_2 \in T_2(h_2(a_i)) \right\} \cup \left\{ \left(\begin{array}{c} q_f \# \\ \langle \# \rangle \end{array} \right) \middle| q_f \in F \right\} \\ D_u &= \left\{ \left(\begin{array}{c} \langle a_i \rangle \\ t_1 \end{array} \right) \middle| t_1 \in T_1(h_1(a_i)) \right\} \cup \left\{ \left(\begin{array}{c} \langle \# \rangle \\ q_f \# \end{array} \right) \middle| q_f \in F \right\} \end{aligned}$$

Figure 19 illustrates the idea of how γ recognizes the language $h_1(\text{EQ}(h_1, h_2))$ (the encoding mentioned above for R is omitted for clarity).

By this construction, $L(\gamma)$ is the set of encodings of words $u \in R$ for which there exists a word $w \in \Sigma_2^*$ with $h_1(w) = h_2(w) = u$. Projection being a weak coding, there exists a weak coding h such that $h(L(\gamma)) = L$.

Fig. 19

A brief sketch of the proof for **Theorem 17**. Two hairpins, one with sticky end $h_1(a_i)$ and the other with sticky end $h_2(a_i)$, binding together via their matching loops, can be abstracted as a finite control (square) labeled by a_i , with two heads (bifurcated arrows). The control checks for each i , $1 \leq i \leq n$, if the single-stranded part of the upper strand begins with $h_2(a_i)$, and the lower strand begins with $h_1(a_i)$; if so, then the corresponding hairpins are stuck. This process is repeated until either this system cannot proceed in this way anymore or it generates a complete matching of two complete molecules. In fact this system is essentially equivalent to Watson-Crick automata, which will be introduced in **Sect. 6.2**.



6.1.2 Observable Sticker Systems

Another idea to strengthen a computational system is to let someone observe and report how the system works step by step. This composite system, inspired by the common practice of observing the progress of a biology or chemistry experiment, has been introduced in Cavaliere and Leupold (2004) to “observe” membrane systems. There, a finite automaton observed the change of configurations of a “computationally weak” membrane system (with context-free power). Surprisingly, this composite system proved to be universal. Following this idea, many computations were “observed”: splicing systems, derivations of grammars and string-rewriting systems, and also sticker systems. For the details of these observations as well as the formal definition of computation by observation in general, the readers are referred to Cavaliere and Leupold (2004), Cavaliere (2008), and references thereof.

The idea of observing sticker systems was introduced in Alhazov and Cavaliere (2005). Informally, an *observable sticker system* is composed of a “computationally weak” sticker system and an external observer. Observing the computation of a sticker system, starting from an axiom, the observer notes – at each computational step – the current configuration, and processes it according to its own rules, producing an output. The catenation of all the outputs thus produced by the observer during a complete computation, constitutes a word in the language of the observable sticker system. The collection of all words thus obtained is the language generated by this observable sticker system.

Formally speaking, configurations of a sticker system, which are elements of Σ_d^* , are observed so that an observer is implemented as a finite automaton that works on elements of Σ_d^* . This automaton is defined as a 6-tuple $O = (Q, \Sigma_d, \Delta, q_0, \delta, \sigma)$, where a finite set of states Q , an input alphabet Σ_d , the initial state $q_0 \in Q$, and a complete transition function $\delta : \Sigma_d \times Q \rightarrow Q$ are defined as usual for conventional finite automata: whereas Δ is an output alphabet and σ is a labeling function $Q \rightarrow \Delta \cup \{\perp, \lambda\}$, \perp being a special symbol.

An *observable (simple regular) sticker system* is a pair $\phi = (\gamma, O)$ for a simple regular sticker system γ and an observer O . For a computation $c : x_0 \Rightarrow_{\gamma} x_1 \Rightarrow_{\gamma} \dots \Rightarrow_{\gamma} x_k$ ($x_i \in R_p(\Sigma)$), $O(c)$ is defined as $O(x_0)O(x_1)\dots O(x_k)$. The language $L(\phi)$ generated by ϕ is defined as $L(\phi) = \{O(c) \mid c \text{ is a complete computation by } \gamma\}$.

Theorem 18 (Alhazov and Cavaliere 2005) *There exists an observable simple regular sticker system which generates a non-context-free language.*

This theorem shows how stronger very restricted sticker systems can get with the aid of the observer (cf. [Theorem 15](#)). A natural question that follows is of how to get universality within the framework of observable sticker systems. The next theorem proves that observers with the capability to discard any “bad” evolution endow simple regular sticker systems with universality.

The symbol $\perp \notin \Delta$ makes it possible for an observer O to distinguish bad evolutions by a sticker system γ from good ones in a way that the observation of bad evolutions leads to the output of \perp . For the observable sticker system $\phi = (\gamma, O)$, one can weed out any word that contains \perp from $L(\phi)$ by taking $\widehat{L}(\phi) = L(\phi) \cap \Delta^*$.

Theorem 19 (Alhazov and Cavaliere 2005) *For each $L \in \text{RE}$, there exists an observable simple regular sticker system $\phi = (\gamma, O)$ such that $\widehat{L}(\phi) = L$.*

Due to the fact that recursive languages are closed under intersection with regular languages, [Theorem 19](#) has the following result as its corollary.

Corollary 1 (Alhazov and Cavaliere 2005) *There exists an observable simple regular sticker system $\phi = (\gamma, O)$ such that $L(\phi)$ is a non-recursive language.*

6.2 Watson–Crick Automata

While sticker systems generate complete double-stranded molecules by using the sticking operation, their accepting counterparts, the Watson–Crick automata, parse a given complete double-stranded molecule and determine whether the input is accepted or not. A Watson–Crick automaton is equipped with a finite state machine with two heads. This machine has its heads read the respective upper and lower strands of a given complete double-stranded molecule simultaneously, and changes its state accordingly. The basic idea of how Watson–Crick automata work was described in [Fig. 19](#).

In parallel to the research on sticker systems, these biologically inspired automata have been intensively investigated within the last decade. Early studies including, for example, Freund et al. (1999) and Martin-Vide et al. (1998) investigate variants of WK-automata, relationships among them with respect to generative capacity, and universal Watson–Crick automata, topics summarized in chapter 5 of Păun et al. (1998). A more recent survey Czeizler and Czeizler (2006c) includes results on complexity measures (Păun and Păun 1999) and Watson–Crick automata systems (Czeizler and Czeizler 2006a,b). Other studies on WK-automata comprise Watson–Crick ω -automata (Petre 2003), the role of the complementarity relation (Kuske and Weigel 2004) (see [Theorem 1](#)), local testability and regular reversibility (Sempere 2007, 2008), 5' → 3' sensing Watson–Crick finite automata (Nagy 2008), and deterministic Watson–Crick automata (Czeizler et al. 2008a).

In this section, we present recent results in this field, such as studies of deterministic WK-automata, and the role of the complementarity relation. In particular, deterministic WK-automata are essential for the design of efficient molecular parsers.

A (*nondeterministic*) Watson–Crick (*finite*) automaton over a symmetric relation $\rho \subseteq \Sigma \times \Sigma$ is a 6-tuple $M = (\Sigma, \rho, Q, q_0, F, \delta)$, where Σ , Q , q_0 , and F are defined in the same manner as for finite automata. The *transition function* δ is a mapping $\delta : Q \times \Sigma^* \times \Sigma^* \rightarrow 2^Q$ such that $\delta\left(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right) \neq \emptyset$ only for *finitely many* pairs $(q, w_1, w_2) \in Q \times \Sigma^* \times \Sigma^*$. We can replace the transition function by rewriting rules, by denoting $q\left(\begin{matrix} w_1 \\ w_2 \end{matrix}\right) \rightarrow q'$ instead of $q' \in \delta\left(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right)$. Transitions in M are defined as follows. For $q, q' \in Q$ and $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \in \begin{pmatrix} \Sigma^* \\ \Sigma^* \end{pmatrix}$ such that $\begin{bmatrix} x_1 w_1 y_1 \\ x_2 w_2 y_2 \end{bmatrix} \in \text{WK}_\rho(\Sigma)$, we write $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} q\left(\begin{matrix} w_1 \\ w_2 \end{matrix}\right) \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \left(\begin{matrix} w_1 \\ w_2 \end{matrix}\right) q'\left(\begin{matrix} y_1 \\ y_2 \end{matrix}\right)$ iff $q' \in \delta\left(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right)$. If \Rightarrow^* is the reflexive and transitive closure of \Rightarrow , then the language accepted by M is

$$L(M) = \left\{ w_1 \in \Sigma^* \mid q_0 \left[\begin{matrix} w_1 \\ w_2 \end{matrix} \right] \Rightarrow^* \left[\begin{matrix} w_1 \\ w_2 \end{matrix} \right] q_f \text{ for some } q_f \in F \text{ and } w_2 \in \Sigma^* \text{ such that } \left[\begin{matrix} w_1 \\ w_2 \end{matrix} \right] \in \text{WK}_\rho(\Sigma) \right\}.$$

Other languages are also considered in Freund et al. (1999) and Păun et al. (1998) such as control words associated to computations — but they are not introduced here. By convention, as suggested also in Păun and Păun (1999), in this section we consider two languages differing only by the empty word λ as identical.

A WK-automaton $M = (\Sigma, \rho, Q, q_0, F, \delta)$ is said to be *stateless* if $Q = F = \{s_0\}$; *all-final* if $Q = F$; *simple* if for any rewriting rule $q\left(\begin{matrix} w_1 \\ w_2 \end{matrix}\right) \rightarrow q'$, either w_1 or w_2 is λ ; *1-limited* if for any transition $q\left(\begin{matrix} w_1 \\ w_2 \end{matrix}\right) \rightarrow q'$, we have $|w_1 w_2| = 1$. By AWK, NWK, FWK, SWK, and 1WK, we denote the families of languages accepted by WK-automata that are arbitrary (A), stateless (N, no-state), all-final (F), simple (S), and 1-limited (1). When two restrictions are imposed at the same time, both of the corresponding symbols are used to identify the family.

As customary in automata theory, normal forms for WK-automata are available. For example, we can convert any WK-automaton into a 1-limited one without changing the language accepted, or the symmetric relation over which the original WK-automaton is defined (Martin-Vide et al. 1998). Another normal form, standardizing the symmetric relation, is as follows:

Theorem 20 (Kuske and Weigel 2004) *For any WK-automaton M , we can construct a WK-automaton M_{id} over the identity relation id with $L(M) = L(M_{id})$.*

A 1-limited WK-automaton over the identity relation is equivalent to a *two-head finite automaton*. Therefore the next theorem follows, where TH denotes the family of languages accepted by two-head finite automata.

Theorem 21 (Păun et al. 1998) $1\text{WK} = \text{SWK} = 1\text{SWK} = \text{AWK} = \text{TH}$.

Czeizler et al. (2008a) proposed three criteria of determinism. A WK-automaton M is said to be *weakly deterministic* if at any point of computation by M , there is at most one possibility to continue the computation; and *deterministic* if for any pair of transition rules $q\left(\begin{smallmatrix} u \\ v \end{smallmatrix}\right) \rightarrow q'$ and $q\left(\begin{smallmatrix} u' \\ v' \end{smallmatrix}\right) \rightarrow q''$, we have $u \sim_p u'$ or $v \sim_p v'$. Clearly, a deterministic WK-automaton is weakly deterministic. Moreover, a deterministic WK-automaton over a symmetric relation ρ is said to be *strongly deterministic* if ρ is the identity. The families of languages accepted by *weakly deterministic*, *deterministic*, and *strongly deterministic* WK-automata are denoted by wdAWK , dAWK , and sdAWK , respectively. The symbol “A” can be replaced with N, F, S, 1, or their combination as in the nondeterministic case.

Proposition 1 (Czeizler et al. 2008a) $\text{sdAWK} \subseteq \text{dAWK} \subseteq \text{wdAWK} \subseteq \text{AWK}$.

The generative power of finite automata or Turing machines remains unchanged by bringing in determinism, while the determinism strictly weakens pushdown automata (Hopcroft and Ullman 1979). Thus, a natural question is whether the relation in (2) Proposition 1 includes some strict inclusion or all of the families are the same.

Czeizler et al. (2008a) proved that $\text{dAWK} = \text{d1WK}$ using a similar proof technique for $\text{AWK} = \text{1WK}$. As mentioned above, the technique keeps the symmetric relation unchanged. As such, $\text{sdAWK} = \text{sd1WK}$ follows immediately. Following the same reasoning to prove $\text{AWK} = \text{TH}$ (2) Theorem 21), we can see that sdAWK is equivalent to the family dTH of languages accepted by *deterministic two-head finite automata*. It is known that there exists a language in $\text{TH} \setminus \text{dTH}$, for example, $L' = \{w \in \Sigma^* \mid w \neq w^R\}$. This means that nondeterministic WK-automata are strictly more powerful than strongly deterministic ones. In fact, the following stronger result holds.

Theorem 22 (Czeizler et al. 2008a) $\text{sdAWK} \subsetneq \text{dAWK}$.

Proof It suffices to prove that $L' \in \text{dAWK}$. We prove the statement only for the case when Σ is binary, but the statement holds for arbitrary finite alphabets.

Let $M = (\Sigma \cup \{c, d_a, d_b\}, \rho, Q, q_0, F, \delta)$ be a WK-automaton, where $\rho = \{(a, a), (a, d_a), (b, b), (b, d_b), (a, c), (b, c)\}$, $Q = \{q_0, q_f, q_a, q_b\}$, $F = \{q_f\}$, and δ consists of the following rules:

$$\begin{aligned} q_0\left(\begin{smallmatrix} \lambda \\ x \end{smallmatrix}\right) &\rightarrow q_0, \quad q_0\left(\begin{smallmatrix} \lambda \\ d_x \end{smallmatrix}\right) \rightarrow q_x, \text{ with } x \in \{a, b\} \\ q_x\left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right) &\rightarrow q_x, \text{ with } x, y, z \in \{a, b\} \\ q_x\left(\begin{smallmatrix} zy \\ c \end{smallmatrix}\right) &\rightarrow q_f, \text{ with } x, y, z \in \{a, b\}, \quad x \neq y \\ q_f\left(\begin{smallmatrix} x \\ \lambda \end{smallmatrix}\right) &\rightarrow q_f, \text{ with } x \in \{a, b\} \end{aligned}$$

It is clear that M is deterministic. Let $w = w_1 w_2 \dots w_n$ with $w_i \in \Sigma$. If $w \neq w^R$, then there exists a position k in the first half of w such that $w_k \neq w_{n-k+1}$. The characters d_a, d_b are used as

a marker of this position. When M runs on the input $\binom{w_1 \cdots w_{k-1} w_k w_{k+1} \cdots w_{n-1} w_n}{w_1 \cdots w_{k-1} d_{w_k} w_{k+1} \cdots w_{n-1} c}$, it accepts w . On the other hand, M does not accept any palindrome regardless of what complement one chooses; thus $L(M) = \{w \in \{a, b\}^+ \mid w \neq w^R\}$.

This contrasts with the nondeterministic case where the complementarity relation does not play any active role (❸ [Theorem 20](#)). Though it is natural now to ask if dAWK \subseteq wdAWK is strict or not, this question remains open. Note that there exists a weakly deterministic WK-automaton that is not deterministic.

❸ [Proposition 1](#) and ❸ [Theorem 22](#) conclude that strong determinism strictly weakens WK-automata. However, deterministic WK-automata are still more powerful than finite automata. Since the construction of a strongly deterministic WK-automaton that simulates a given deterministic finite automaton is straightforward, in the following a stronger result is included.

Theorem 23 $\text{REG} \subseteq \text{sdF1WK}$.

Proof In order to simulate a deterministic finite automaton $A = (Q, \Sigma, q_0, F, \delta)$ with $Q = \{q_0, q_1, \dots, q_n\}$, we construct a strongly deterministic all-final 1-limited WK-automaton $M = (\Sigma, id, Q', q_{0,0}, Q', \delta')$, where $Q' = \{q_{i,j}, \overline{q_{i,j}} \mid 0 \leq i, j \leq n\}$, and for $a \in \Sigma, 0 \leq i, j \leq n$,

$$\begin{aligned}\delta'\left(q_{i,j}, \binom{a}{\lambda}\right) &= \begin{cases} q_{k,j} & \text{if } \delta(q_i, a) = q_k \text{ and } q_k \notin F \\ \overline{q_{k,j}} & \text{if } \delta(q_i, a) = q_k \text{ and } q_k \in F \end{cases} \\ \delta'\left(\overline{q_{i,j}}, \binom{\lambda}{a}\right) &= \begin{cases} \overline{q_{i,k}} & \text{if } \delta(q_j, a) = q_k \text{ and } q_k \notin F \\ q_{i,k} & \text{if } \delta(q_j, a) = q_k \text{ and } q_k \in F \end{cases}\end{aligned}$$

The recognition of a sequence $\left[\begin{smallmatrix} w \\ w \end{smallmatrix} \right]$ consists of two identical simulations of recognition process of w on A over the upper and lower strands. Current states of the automaton A working on upper and lower strands are recorded as first and second subscripts of states in Q' . The overline of states indicates that M is now in the simulation over the lower strand. One switches these two phases every time the simulated automaton reaches some final state of A . We can see easily that $L(M) = L(A) \cup \{\lambda\}$.

From ❸ [Theorem 23](#), it is clear that WK-automata are more powerful than finite automata. Additional results on the generative capacity of WK-automata can be found in, for example, Păun et al. (1998). Moreover, it was shown in Czeizler et al. (2008a) and Păun and Păun (1999) that WK-automata recognize some regular languages in a less space-consuming manner. Usage of space is measured by the *state complexity* (for more details, see Păun and Păun (1999) and Yu (2002)). It is well-known that the state complexity of some families of finite languages is unbounded when one considers the finite automata recognizing them. In other words, for any $k \geq 1$, there is a finite language that cannot be recognized by any finite automaton with at most $k - 1$ states. On the contrary, any finite language can be recognized by a WK-automaton with two states (Czeizler et al. 2008a).

Determinism is one of the most essential properties for the design of an efficient parser. Due to the time-space trade-off, the stronger the determinism is, the more space-consuming WK-automata get. For example, it was shown in Czeizler et al. (2008a) that in order to

recognize a finite language $L_k = \{a, aa, \dots, a^{k-1}\}$, the strongly deterministic WK-automaton needs at least k states, while for any k , two states are enough once the strong determinism is changed to determinism. Little is known about the state complexity with respect to the (strongly, weakly) deterministic WK-automata.

As a final result on this topic, the following undecidability property about determinism of WK-automata is mentioned.

Theorem 24 (Czeizler et al. 2008a) *It is undecidable whether a given WK-automaton is weakly deterministic.*

7 DNA Complementarity and Combinatorics on Words

⌚ Sections 5 and ⌚ 6 described new concepts and results in formal language theory and automata theory that attest to the influence that the notion of DNA-based information and its main aspect, the Watson–Crick complementarity, has had on formal language theory, coding theory, and automata theory. This section is devoted to describing some of the ideas, notions, and results that DNA-based information has brought to another area of theoretical computer science, namely combinatorics on words. Indeed, the “equivalence” from the informational point of view between a DNA single strand and its Watson–Crick complement led to several interesting generalizations of fundamental notions such as bordered word, palindrome, periodicity, conjugacy, and commutativity. Moreover, these generalizations led to natural extensions of two of the most fundamental results in combinatorics on words: the Fine and Wilf theorem and the Lyndon–Schützenberger equation. This section presents some of these concepts and results in combinatorics of words motivated by the Watson–Crick complementarity property of DNA-based information.

In the following, θ will denote an antimorphic involution. For further details on combinatorics on words, the reader is referred to Choffrut and Karhumäki (1997).

As mentioned in ⌚ Sect. 2, recognition sites of enzymes are often palindromic in a biological sense. Conventionally speaking, a palindrome is a word that reads the same way from the left and from the right, such as “racecar” in English. In contrast, in molecular biology terms, a palindromic DNA sequence is one that is equal to its WK complementary sequence. For example, TGGATCCA is palindromic in this sense. The biological palindromic motif is herein modeled as θ -palindrome (or, more generally, *pseudopalindrome*) defined as follows: A word w is a θ -palindrome if $w = \theta(w)$. θ -palindromes were investigated intensively from a theoretical computer science perspective, see for example, in de Luca and Luca (2006) and Kari and Mahalingam (2010).

The properties of θ -sticky-freeness and θ -overhang-freeness motivate the generalization of the notions of *border*, *commutativity*, and *conjugacy* of words into θ -*border*, θ -*commutativity*, and θ -*conjugacy* of words. A word $w \in \Sigma^+$ is *bordered* if there exists a word $v \in \Sigma^+$ satisfying $w = vx = yv$ for some $x, y \in \Sigma^+$. Kari and Mahalingam (2007) proposed an extended notion called the θ -borderedness of words as w is θ -bordered if $w = vx = y\theta(v)$ for some $v, x, y \in \Sigma^+$. Note that θ -sticky-free languages do not contain any θ -bordered word. The *pseudoknot-bordered-word* proposed in Kari and Seki (2009) is a further extension of the notion of θ -bordered word. A word $w \in \Sigma^+$ is *pseudoknot-bordered* if $w = uvx = y\theta(u)\theta(v)$. A pseudoknot-bordered word models the crossing dependency occurring in DNA and RNA

pseudoknots. A word $v \in \Sigma^+$ is said to be a *conjugate* of another word $u \in \Sigma^+$ if $v = yx$ and $u = xy$ for some $x, y \in \Sigma^*$, that is, $ux = xv$ holds. This notion was extended in Kari and Mahalingam (2008) as follows: a word $u \in \Sigma^+$ is a θ -*conjugate* of a word $v \in \Sigma^+$ if $ux = \theta(x)v$ for some $x \in \Sigma^+$. In this case, either $v = \theta(u)$ or $u = \theta(x)z$, $v = zx$ for some $z \in \Sigma^+$; hence a language that contains both u and v cannot be strictly θ -sticky-free.

In contrast with the above two notions, the θ -*commutativity of words* introduced in Kari and Mahalingam (2008) has a purely theoretical significance, being a mathematical tool for obtaining results involving WK complementarity. Two words $u, v \in \Sigma^+$ are said to *commute* if $uv = vu$ holds. For two words u, v , we say that u θ -commutes with v if $uv = \theta(v)u$. This equation is a special case of conjugacy equations. Thus, by applying a known result in combinatorics on words, one can deduce the following results.

Theorem 25 (Kari and Mahalingam 2008) *For an antimorphic involution θ , and two words $u, v \in \Sigma^+$, if $uv = \theta(v)u$ holds, then $u = r(tr)^i$, $v = (tr)^j$ for some $i \geq 0$, $j \geq 0$, and θ -palindromes $r, t \in \Sigma^*$ such that rt is primitive.*

This theorem relates the three important notions: θ -commutativity of words, θ -palindrome, and primitivity.

Another interesting perspective is the informational equivalence between the two strands of a DNA double helix. Indeed, the two constituent single strands of a DNA double strand are “equivalent” with respect to the information they encode, and thus we can say that w and $\theta(w)$ are equivalent in this sense. Practical applications of this idea include an extended Lempel–Ziv algorithm for DNA molecules proposed in Grumbach and Tahi (1993).

A word $w \in \Sigma^+$ is called *primitive* if it cannot be written as a power of another word; that is, $w = u^n$ implies $n = 1$ and $w = u$. For a word $w \in \Sigma^+$, the shortest $u \in \Sigma^+$ such that $w = u^n$ for some $n \geq 1$ is called the *primitive root* of the word w and is denoted by $\rho(w)$. It is well known that for each word $w \in \Sigma^*$, there exists a unique primitive word $t \in \Sigma^+$ such that $\rho(w) = t$, that is, $w = t^n$ for some $n \geq 1$, see for example, Choffrut and Karhumäki (1997). In Czeizler et al. (2008b), the primitivity was extended to θ -primitivity for an antimorphic involution θ . A word $u \in \Sigma^+$ is said to be θ -*primitive* if there does not exist any word $t \in \Sigma^+$ such that $u \in \{t, \theta(t)\}^{\geq 2}$. For a word $w \in \Sigma^+$, we define the θ -*primitive root* of w , denoted by $\rho_\theta(w)$, as the shortest word $t \in \Sigma^+$ such that $w \in t\{t, \theta(t)\}^*$. Note that if w is θ -primitive, then $\rho_\theta(w) = w$. Note also that θ -primitivity can be defined also for θ being a morphic involution. However, as it is meant as a model of WK complementarity, we will continue to assume that θ is antimorphic. For counterparts of the following results when θ is a morphic involution see Czeizler et al. (2008b).

We start by looking at some basic properties of θ -primitive words.

Proposition 2 (Czeizler et al. 2008b) *If a word $w \in \Sigma^+$ is θ -primitive, then it is also primitive. Moreover, the converse is not always true.*

Proof By definition, it is clear that θ -primitive words are primitive. Remark that if there exists $a \in \Sigma$ satisfying $a \neq \theta(a)$, then the word $a\theta(a)$ is not θ -primitive, but is primitive.

If a word $w \in \Sigma^+$ is in $\{t, \theta(t)\}^+$ for some word $t \in \Sigma^+$ and t in turn is in $\{s, \theta(s)\}^+$ for some word $s \in \Sigma^+$, then we have that $w \in \{s, \theta(s)\}^+$. Thus, we have the following result.

Proposition 3 (Czeizler et al. 2008b) *The θ -primitive root of a word is θ -primitive, and hence primitive.*

It is a well-known fact that any conjugate of a primitive word is primitive. This fact is heavily employed in obtaining fundamental results including the Fine and Wilf theorem and solutions to the Lyndon–Schützenberger equation. In contrast, a conjugate of a θ -primitive word need not be θ -primitive. For instance, for the DNA involution τ defined in [Sect. 5](#), the word $w = GCTA$ is τ -primitive, while its conjugate $w' = AGCT = AG\tau(AG)$ is not.

Proposition 4 (Czeizler et al. 2008b) *The class of θ -primitive words is not closed under circular permutations.*

An essential property of primitive words is that a primitive word cannot be equal to its conjugate. In other words, for a primitive word u , the equation $uu = xuy$ implies that either x or y is empty. Thus, u^i and u^j , with $i, j \geq 1$, cannot overlap non-trivially on a sequence longer than $|u|$. This is not the case when considering overlap between $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$ for some θ -primitive word $v \in \Sigma^+$. For instance, for the DNA involution τ and a τ -primitive word $v = CCGGAT$, $v^2 = CCGG \cdot \tau(v) \cdot AT$ holds. Nevertheless, an analogous result for θ -primitive words was obtained.

Theorem 26 (Kari et al. 2010) *Let $v \in \Sigma^+$ be a θ -primitive word. Neither $v\theta(v)$ nor $\theta(v)v$ can be a proper infix of a word in $\{v, \theta(v)\}^3$.*

Furthermore, Czeizler et al. completely characterized all such nontrivial overlaps with the set of all solutions of the corresponding equation (Czeizler et al. 2009).

Theorem 27 (Czeizler et al. 2009) *Let $v \in \Sigma^+$ be a θ -primitive word. The only possible proper overlaps of the form $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$ with $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$, $x, y \in \Sigma^+$ and $|x|, |y| < |v|$ are given in [Table 1](#) (modulo a substitution of v by $\theta(v)$) together with the characterization of their sets of solutions.*

Table 1

Characterization of possible proper overlaps of the form $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$. For the last three equations, $n \geq 0, m \geq 1, r, t \in \Sigma^+$ such that $r = \theta(r), t = \theta(t)$, and rt is primitive. Note that the fourth and fifth equations are the same up to the antimorphic involution θ

Equation	Solution
$v^i x = y\theta(v)^j, i \geq 1$	$v = yp, x = \theta(y), p = \theta(p)$, and whenever $i \geq 2, y = \theta(y)$
$vx = yv$	$v = (pq)^{j+1}p, x = qp, y = pq$ for some $p, q \in \Sigma^+, j \geq 0$
$v\theta(v)x = yv\theta(v),$	$v = (pq)^{j+1}p, x = \theta(pq), y = pq$, with $j \geq 0, qp = \theta(qp)$
$v^{i+1}x = y\theta(v)^j v, i \geq 1$	$v = r(tr)^{n+m}r(tr)^n, x = (tr)^m r(tr)^n, y = r(tr)^{n+m}$
$v\theta(v)^i x = yv^{i+1}, i \geq 1$	$v = (rt)^n r(rt)^{m+n}r, y = (rt)^n r(rt)^m, x = (rt)^{m+n}r$
$v\theta(v)^i x = yv^j\theta(v), i \geq 2$	$v = (rt)^n r(rt)^{m+n}r, y = (rt)^n r(rt)^m, x = (tr)^m r(tr)^n$

We now shift one's attention to extensions of two essential results in combinatorics on words. The following theorem is known as the *Fine and Wilf theorem* (Fine and Wilf 1965), in its form for words (Choffrut and Karhumäki 1997; Lothaire 1983). It illustrates a fundamental periodicity property of words. Its concise proof is available, in, for example, Choffrut and Karhumäki (1997). As usual, $\gcd(n, m)$ denotes the *greatest common divisor* of n and m .

Theorem 28 *Let $u, v \in \Sigma^*$, $n = |u|$, and $m = |v|$. If a power of u and a power of v have a common prefix of length at least $n + m - \gcd(n, m)$, then $\rho(u) = \rho(v)$. Moreover, the bound $n + m - \gcd(n, m)$ is optimal.*

A natural question is whether one can obtain an extension of this result when, instead of taking powers of two words u and v , one looks at a word in $u\{u, \theta(u)\}^*$ and a word in $v\{v, \theta(v)\}^*$. The answer is yes. Note that without loss of generality, one can suppose that the two words start with u and v because θ is an involution. Czeizler, Kari, and Seki provided extensions of Theorem 28 in two forms (Theorems 29 and 30) (Czeizler et al. 2008b). As illustrated in the following example, the bound given by Theorem 28 is not sufficient anymore.

Example 5 (Czeizler et al. 2008b) Let $\theta : \{a, b\}^* \rightarrow \{a, b\}^*$ be the mirror image mapping defined as follows: $\theta(a) = a$, $\theta(b) = b$, and $\theta(w_1 \dots w_n) = w_n \dots w_1$, where $w_i \in \{a, b\}$ for all $1 \leq i \leq n$. Obviously, θ is an antimorphic involution on $\{a, b\}^*$. Let $u = (ab)^k b$ and $v = ab$. Then, u^2 and $v^k \theta(v)^{k+1}$ have a common prefix of length $2|u| - 1 > |u| + |v| - \gcd(|u|, |v|)$. Nevertheless, u and v do not have the same θ -primitive root, that is $\rho_\theta(u) \neq \rho_\theta(v)$.

In the following, $\text{lcm}(n, m)$ denotes the *least common multiple* of n and m .

Theorem 29 (Czeizler et al. 2008b) *Let $u, v \in \Sigma^+$, and $\alpha(u, \theta(u)) \in u\{u, \theta(u)\}^*$, $\beta(v, \theta(v)) \in v\{v, \theta(v)\}^*$ be two words sharing a common prefix of length at least $\text{lcm}(|u|, |v|)$. Then, there exists a word $t \in \Sigma^+$ such that $u, v \in t\{t, \theta(t)\}^*$, i.e., $\rho_\theta(u) = \rho_\theta(v)$. In particular, if $\alpha(u, \theta(u)) = \beta(v, \theta(v))$, then $\rho_\theta(u) = \rho_\theta(v)$.*

This theorem provides us with an alternative definition of the θ -primitive root of a word.

Corollary 2 (Czeizler et al. 2008b) *For any word $w \in \Sigma^+$ there exists a unique θ -primitive word $t \in \Sigma^+$ such that $w \in t\{t, \theta(t)\}^*$, i.e., $\rho_\theta(w) = t$.*

Corollary 3 (Czeizler et al. 2008b) *Let $u, v \in \Sigma^+$ be two words such that $\rho(u) = \rho(v) = t$. Then, $\rho_\theta(u) = \rho_\theta(v) = \rho_\theta(t)$.*

Next, another bound is provided for this extended Fine and Wilf theorem, which is in many cases much shorter than the bound given in Theorem 29. As noted before, due to Proposition 4, it is intuitive that one cannot use the concise proof technique based on the fact that a conjugate of a primitive word is primitive. The proof given in Czeizler et al. (2008b) involves rather technical case analyses.

Theorem 30 (Czeizler et al. 2008b) Given two words $u, v \in \Sigma^+$ with $|u| > |v|$, if there exist two words $\alpha(u, \theta(u)) \in u\{u, \theta(u)\}^*$ and $\beta(v, \theta(v)) \in v\{v, \theta(v)\}^*$ having a common prefix of length at least $2|u| + |v| - \gcd(|u|, |v|)$, then $\rho_\theta(u) = \rho_\theta(v)$.

This section is concluded with an extension of another fundamental result related to the periodicity on words, namely the solution to the *Lyndon–Schützenberger equation*. The equation is of the form $u^\ell = v^n w^m$ for some $\ell, n, m \geq 2$ and $u, v, w \in \Sigma^+$. Lyndon and Schützenberger proved that this equation implies $\rho(u) = \rho(v) = \rho(w)$ (Lyndon and Schützenberger 1962). A concise proof when u, v, w belong to a free semigroup can be found in, for example, Harju and Nowotka (2004).

Incorporating the idea of θ -periodicity, the Lyndon–Schützenberger equation has been extended in Czeizler et al. (2009) in the following manner. Let $u, v, w \in \Sigma^+$, θ be an antimorphic involution over Σ , and ℓ, n, m be integers ≥ 2 . Let $\alpha(u, \theta(u)) \in \{u, \theta(u)\}^\ell$, $\beta(v, \theta(v)) \in \{v, \theta(v)\}^n$, and $\gamma(w, \theta(w)) \in \{w, \theta(w)\}^m$. The *extended Lyndon–Schützenberger equation* is

$$\alpha(u, \theta(u)) = \beta(v, \theta(v))\gamma(w, \theta(w))$$

In Czeizler et al. (2009), the authors investigated the problem of finding conditions on ℓ, n, m such that if the equation holds, then $u, v, w \in \{t, \theta(t)\}^+$ for some $t \in \Sigma^+$. If such t exists, we say that (ℓ, n, m) imposes θ -periodicity on u, v, w . The original condition $\ell, n, m \geq 2$ is not enough for this extension as shown below.

Example 6 (Czeizler et al. 2009) Let $\Sigma = \{a, b\}$ and θ be the mirror image. Take now $u = a^k b^2 a^{2k}$, $v = \theta(u)^l a^{2k} b^2 = (a^{2k} b^2 a^k)^l a^{2k} b^2$, and $w = a^2$, for some $k, l \geq 1$. Then, although $\theta(u)^{l+1} u^{l+1} = v^2 w^k$, there is no word $t \in \Sigma^+$ with $u, v, w \in \{t, \theta(t)\}^+$.

Example 7 (Czeizler et al. 2009) Consider again $\Sigma = \{a, b\}$ and the mirror image θ , and take $u = b^2(aba)^k$, $v = u^l b = (b^2(aba)^k)^l b$, and $w = aba$ for some $k, l \geq 1$. Then, although $u^{2l+1} = v\theta(v)w^k$, there is no word $t \in \Sigma^+$ with $u, v, w \in \{t, \theta(t)\}^+$.

Example 8 Let $\Sigma = \{a, b\}$ and θ be the mirror image. Let $v = a^{2m} b^{2j}$ and $w = aa$ for some $m \geq 1$ and $j \geq 1$. Then $v^n w^m = (a^{2m} b^{2j})^n a^{2m}$. This is θ -palindrome of even length and hence it can be written as $u\theta(u)$ for some $u \in \Sigma^+$. Clearly $\rho_\theta(w) = a$ and $\rho_\theta(v) \neq a$ because v contains b . Hence $(2, n, m)$ is not enough to impose the θ -periodicity.

Thus, once either n or m is 2, it is not always the case that there exists a word $t \in \Sigma^+$ such that $u, v, w \in \{t, \theta(t)\}^+$. On the other hand, it was proved that (ℓ, n, m) imposes θ -periodicity on u, v, w if $\ell \geq 5$, $n, m \geq 3$.

Theorem 31 (Czeizler et al. 2009) For words $u, v, w \in \Sigma^+$ and $\ell, n, m \geq 2$, let $\alpha(u, \theta(u)) \in \{u, \theta(u)\}^\ell$, $\beta(v, \theta(v)) \in \{v, \theta(v)\}^n$, and $\gamma(w, \theta(w)) \in \{w, \theta(w)\}^m$. If $\alpha(u, \theta(u)) = \beta(v, \theta(v))\gamma(w, \theta(w))$ holds and $\ell \geq 5$, $n, m \geq 3$, then $u, v, w \in \{t, \theta(t)\}^+$ for some $t \in \Sigma^+$.

This theorem requires rather complex case analyses, too, and hence the proof is omitted here. However, if ℓ, n, m are “big” enough, one can easily see that there is much room to employ the extended Fine and Wilf theorem.

Table 2**Summary of the extended Lyndon–Schützenberger equation results**

<i>l</i>	<i>n</i>	<i>m</i>	θ -Periodicity	Proved by
≥ 5	≥ 3	≥ 3	Yes	Theorem 31
4	≥ 3	≥ 3	?	
3	≥ 3	≥ 3	?	
≥ 3	2	≥ 2	No	Examples 6 and 7
≥ 3	≥ 2	2	No	
2	≥ 2	≥ 2	No	Example 8

This section is concluded with [Table 2](#), which summarizes the results obtained so far on the extended Lyndon–Schützenberger equation.

8 Conclusions

In this chapter we described how information can be encoded on DNA strands and how bio-operations can be used to perform computational tasks. In addition, we presented examples of the influence that fundamental properties of DNA-encoded information, especially the Watson–Crick complementarity, have had on various areas of theoretical computer science such as formal language theory, coding theory, automata theory, and combinatorics on words.

A few final remarks are in order regarding DNA-encoded data and the bio-operations that are used to act on it. The descriptions of DNA structure and DNA bio-operations point to the fact that DNA-encoded information is very different from electronically encoded information, and bio-operations also differ from electronic computer operations. A fundamental difference arises, for example, from the fact that in electronic computing data interaction is fully controlled, while in a test-tube DNA-computer, free-floating data-encoding DNA single strands can interact because of Watson–Crick complementarity. Another difference is that, in DNA computing, a bio-operation usually consumes both operands. This implies that, if one of the operands is either involved in an illegal binding or has been consumed by a previous bio-operation, it is unavailable for the desired computation. Yet another difference is that, while in electronic computing a bit is a single individual element, in DNA experiments, each submicroscopic DNA molecule is usually present in millions of identical copies. The bio-operations operate in a massively parallel fashion on all identical strands and this process is governed by the laws of chemistry and thermodynamics, with the output obeying statistical laws.

Differences like the ones mentioned above point to the fact that a fresh approach is needed when employing as well as when theoretically investigating bioinformation and biocomputation, and they offer a wealth of problems to explore at this rich intersection between molecular biology and computer science.

Examples of such fascinating research areas and topics include models and wet implementations of molecular computing machineries, membrane computing, DNA computing by splicing and insertion–deletion operations, bacterial computing and communication, DNA memory, DNA computing by self-assembly, and computational aspects of gene assembly in ciliates, as described in, for example, the chapter [Computational Nature of Gene Assembly in Ciliates](#) of this handbook.

References

- Adleman L (1994) Molecular computation of solutions to combinatorial problems. *Science* 266(5187): 1021–1024
- Adleman L (1998) Computing with DNA. *Sci Am* 279:54–61
- Alhazov A, Cavaliere M (2005) Computing by observing bio-systems: the case of sticker systems. In: Feretti C, Mauri G, Zandron C (eds) *Proceedings of DNA computing 10*, Milan, Italy, June 2004. Lecture notes in computer science, vol 3384. Springer-Verlag, Berlin, pp 1–13
- Amos M (2005) *Theoretical and experimental DNA computation*. Springer-Verlag, Berlin
- Andronescu M, Dees D, Slaybaugh L, Zhao Y, Condon A, Cohen B, Skiena S (2003) Algorithms for testing that sets of DNA words concatenate without secondary structure. In: Hagiya M, Ohuchi A (eds) *Proceedings of DNA computing 8*, Sapporo, Japan, June 2002. Lecture notes in computer science, vol 2568. Springer-Verlag, Berlin, pp 182–195
- Arita M (2004) Writing information into DNA. In: Jonoska N, Păun G, Rozenberg G (eds) *Aspects of molecular computing*. Lecture notes in computer science, vol 2950. Springer-Verlag, Berlin, pp 23–35
- Arita M, Kobayashi S (2002) DNA sequence design using templates. *New Generation Comput* 20:263–277
- Bancroft C, Bowler T, Bloom B, Clelland C (2001) Long-term storage of information in DNA. *Science* 293: 1763–1765
- Baum E (1998) DNA sequences useful for computation. In: Landweber L, Baum E (eds) *DNA based computers II*. DIMACS series in discrete mathematics and theoretical computer science, vol 44. American Mathematical Society, Providence, RI, pp 235–246
- Bordihn H, Holzer M, Kutrib M (2007) Hairpin finite automata. In: Harju T, Karhumäki J, Lepistö A (eds) *Developments in language theory*, Turku, Finland, July 2007. Lecture notes in computer science, vol 4588. Springer-Verlag, Berlin, pp 108–119
- Braich R, Chelyapov N, Johnson C, Rothemund P, Adleman L (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* 296:499–502
- Calladine C, Drew H (1997) *Understanding DNA: the molecule and how it works*, 2nd edn. Academic Press, London
- Cavaliere M (2008) Computing by observing: a brief survey. In: Logic and theory of algorithms. Lecture notes in computer science, vol 5028. Springer, Berlin, pp 110–119
- Cavaliere M, Leupold P (2004) Evolution and observation – a new way to look at membrane systems. In: Martin-Vide C, Mauri G, Păun G, Rozenberg G, Salomaa A (eds) *Membrane computing. Lecture notes in computer science*, vol 2933. Springer, Berlin, pp 70–87
- Chen J, Deaton R, Garzon M, Kim J, Wood D, Bi H, Carpenter D, Wang YZ (2006) Characterization of non-crosshybridizing DNA oligonucleotides manufactured in vitro. *Nat Comput* 5(2):165–181
- Choffrut C, Karhumäki J (1997) Combinatorics of words. In: Rozenberg G, Salomaa A (eds) *Handbook of formal languages*, vol 1. Springer-Verlag, Berlin-Heidelberg-New York, pp 329–438
- Cox J (2001) Long-term data storage in DNA. *Trends Biotechnol* 19:247–250
- Czeizler E, Czeizler E (2006a) On the power of parallel communicating Watson-Crick automata systems. *Theor Comput Sci* 358:142–147
- Czeizler E, Czeizler E (2006b) Parallel communicating Watson-Crick automata systems. *Acta Cybern* 17:685–700
- Czeizler E, Czeizler E (2006c) A short survey on Watson-Crick automata. *Bull EATCS* 89:104–119
- Czeizler E, Czeizler E, Kari L, Salomaa K (2008a) Watson-Crick automata: determinism and state complexity. In: DCFS'08: *Proceedings of descriptive complexity of formal systems*, University of Prince Edward Island, Charlottetown, PE, Canada, July 2008, pp 121–133
- Czeizler E, Kari L, Seki S (2008b) On a special class of primitive words. In: MFCS 2008: *Proceedings of mathematical foundations of theoretical computer science*, Torun, Poland, August 2008. Lecture notes in computer science, vol 5162. Springer, Berlin-Heidelberg, pp 265–277
- Czeizler E, Czeizler E, Kari L, Seki S (2009) An extension of the Lyndon Schützenberger result to pseudoperiodic words. In: Diekert V, Nowotka D (eds) DLT'09: *Proceedings of developments in language theory*, Stuttgart, Germany, June 2009. Lecture notes in computer science, vol 5583. Springer-Verlag, Berlin
- Daley M, Kari L (2002) DNA computing: Models and implementations. *Comments Theor Biol* 7:177–198
- Daley M, Ibarra O, Kari L (2003) Closure and decidability properties of some language classes with respect to ciliate bio-operations. *Theor Comput Sci* 306 (1):19–38
- Daley M, Kari L, McQuillan I (2004) Families of languages defined by ciliate bio-operations. *Theor Comput Sci* 320:51–69
- de Luca A, Luca AD (2006) Pseudopalindrome closure operators in free monoids. *Theor Comput Sci* 362:282–300
- Deaton R, Chen J, Bi H, Rose J (2003) A software tool for generating non-crosshybridizing libraries of DNA

- oligonucleotides. In: Hagiya M, Ohuchi A (eds) Proceedings of DNA computing 8, Sapporo, Japan, June 2002. Lecture notes in computer science, vol 2568. Springer-Verlag, Berlin, pp 252–261
- Deaton R, Chen J, Kim J, Garzon M, Wood D (2006) Test tube selection of large independent sets of DNA oligonucleotides. In: Chen J, Jonoska N, Rozenberg G (eds) Nanotechnology: science and computation, Springer-Verlag, Berlin, pp 147–161
- Diaz S, Esteban J, Ogihara M (2001) A DNA-based random walk method for solving k-SAT. In: Condon A, Rozenberg G (eds) Proceedings of DNA computing 6, Leiden, the Netherlands, June 2000. Lecture notes in computer science, vol 2054. Springer-Verlag, Berlin, pp 209–219
- Dirks R, Pierce N (2004) An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *J Comput Chem* 25:1295–1304
- Domaratzki M (online 2007) Hairpin structures defined by DNA trajectories. *Theory Comput Syst DOI* 10.1007/s00224-007-9086-6
- Drlica K (1996) Understanding DNA and gene cloning: a guide for the curious. Wiley, New York
- Dyachkov A, Macula A, Pogozelski W, Renz T, Rykov V, Torney D (2006) New t-gap insertion-deletion-like metrics for DNA hybridization thermodynamic modeling. *J Comput Biol* 13(4):866–881
- Dyachkov A, Macula A, Rykov V, Ufimtsev V (2008) RNA codes based on stem similarities between DNA sequences. In: Garzon M, Yan H (eds) Proceedings of DNA computing 13, Memphis, TN, June 2007. Lecture notes in computer science, vol 4848. Springer-Verlag, Berlin, pp 146–151
- Ehrenfeucht A, Harju T, Petre I, Prescott D, Rozenberg G (2004) Computation in living cells: gene assembly in ciliates. Natural Computing Series. Springer-Verlag, Berlin
- Faulhammer D, Cukras A, Lipton R, Landweber L (2000) Molecular computation: RNA solutions to chess problems. *Proc Nat Acad Sci U S A* 97:1385–1389
- Feldkamp U, Saghafi S, Banzhaf W, Rauhe H (2002) DNasequenceGenerator – a program for the construction of DNA sequences. In: Jonoska N, Seeman N (eds) Proceedings of DNA computing 7, Tampa, FL, June 2001. Lecture notes in computer science, vol 2340. Springer-Verlag, Berlin, pp 23–32
- Fine N, Wilf H (1965) Uniqueness theorem for periodic functions. *Proc Am Math Soc* 16(1):109–114
- Freund R, Păun G, Rozenberg G, Salomaa A (1998) Bidirectional sticker systems. In: Altman R, Dunker A, Hunter L, Klein T (eds) Pacific symposium on biocomputing, vol 98. World Scientific, Singapore, pp 535–546
- Freund R, Păun G, Rozenberg G, Salomaa A (1999) Watson-Crick finite automata. *DIMACS Ser Discrete Math Theor Comput Sci* 48:297–327
- Frutos A, Liu Q, Thiel A, Sanner A, Condon A, Smith L, Corn R (1997) Demonstration of a word design strategy for DNA computing on surfaces. *Nucleic Acids Res* 25(23):4748–4757
- Garzon M, Neathery P, Deaton R, Murphy R, Franceschetti D, Stevens Jr S (1997) A new metric for DNA computing. In: Koza J, Deb K, Dorigo M, Vogel D, Garzon M, Iba H, Riolo R (eds) Proceedings of genetic programming 1997, Stanford University, July 1997. Morgan Kaufmann, San Francisco, CA, pp 479–490
- Garzon M, Phan V, Roy S, Neel A (2006) In search of optimal codes for DNA computing. In: Mao C, Yokomori T (eds) Proceedings of DNA computing 12, Seoul, Korea, June 2006. Lecture notes in computer science, vol 4287. Springer-Verlag, Berlin, pp 143–156
- Gonick L, Wheelis M (1991) The cartoon guide to genetics, updated edn. Collins, New York
- Grumbach S, Tahí F (1993) Compression of DNA sequences. In: Proceedings of IEEE symposium on data compression. IEEE Computer Society Press, San Francisco, CA, June 1993. pp 340–350
- Harju T, Nowotka D (2004) The equation $x^i = y^j z^k$ in a free semigroup. *Semigroup Forum* 68:488–490
- Hartmanis J (1995) On the weight of computations. *Bull EATCS* 55:136–138
- Head T (2000) Relativized code concepts and multi-tube DNA dictionaries. In: Calude C, Păun G (eds) Finite versus infinite: contributions to an eternal dilemma. Springer-Verlag, London, pp 175–186
- Hoogeboom H, van Vugt N (2000) Fair sticker languages. *Acta Inform* 37:213–225
- Hopcroft J, Ullman J (1979) Introduction to automata theory, languages, and computation. Addison-Wesley, Reading, MA
- Hussini S, Kari L, Konstantinidis S (2003) Coding properties of DNA languages. *Theor Comput Sci* 290 (3):1557–1579
- Ignatova Z, Martínez-Pérez I, Zimmermann KH (2008) DNA computing models. Springer-Verlag, Berlin
- Jonoska N, Mahalingam K (2004) Languages of DNA based code words. In: Chen J, Reif J (eds) Proceedings of DNA computing 9, Madison, WI, June 2003. Lecture notes in computer science, vol 2943. Springer, Berlin, pp 61–73
- Jonoska N, Kephart D, Mahalingam K (2002) Generating DNA code words. *Congressus Numerantium* 156:99–110
- Kari L (1997) DNA computing: the arrival of biological mathematics. *Math Intell* 19(2):9–22
- Kari L, Kitto R, Thierrin G (2002) Codes, involutions and DNA encoding. In: Brauer W, Ehrig H, Karhumäki J, Salomaa A (eds) Formal and natural computing. Lecture notes in computer science, vol 2300. Springer-Verlag, Berlin, pp 376–393

- Kari L, Konstantinidis S, Losseva E, Wozniak G (2003) Sticky-free and overhang-free DNA languages. *Acta Inform* 40:119–157
- Kari L, Konstantinidis S, Sosík P (2005a) Bond-free languages: formalizations, maximality and construction methods. *Int J Foundations Comput Sci* 16 (5):1039–1070
- Kari L, Konstantinidis S, Sosík P (2005b) On properties of bond-free DNA languages. *Theor Comput Sci* 334 (1–3):131–159
- Kari L, Konstantinidis S, Losseva E, Sosík P, Thierrin G (2006) A formal language analysis of DNA hairpin structures. *Fundam Inform* 71(4):453–475
- Kari L, Mahalingam K (2007) Involutively bordered words. *Int J Foundations Comput Sci* 18(5): 1089–1106
- Kari L, Mahalingam K (2008) Watson-Crick conjugate and commutative words. In: Garzon M, Yan H (eds) *Proceedings of DNA computing 13*, Memphis, TN, June 2007. Lecture notes in computer science, vol 4848. Springer-Verlag, Berlin, pp 273–283
- Kari L, Mahalingam K (2010) Watson-Crick palindromes in DNA computing. *Natural Comput* 9(2):297–316
- Kari L, Mahalingam K, Thierrin G (2007) The syntactic monoid of hairpin-free languages. *Acta Inform* 44 (3–4):153–166
- Kari L, Masson B, Seki S (2010) Properties of pseudo-primitive words and their applications. CoRR abs/1002.4084
- Kari L, Păun G, Rozenberg G, Salomaa A, Yu S (1998) DNA computing, sticker systems, and universality. *Acta Inform* 35:401–420
- Kari L, Seki S (2009) On pseudoknot-bordered words and their properties. *J Comput Syst Sci* 75(2): 113–121
- Kijima A, Kobayashi S (2006) Efficient algorithm for testing structure freeness of finite set of biomolecular sequences. In: Carbone A, Pierce N (eds) *Proceedings of DNA 11*, London, ON, Canada, June 2005. Lecture notes in computer science, vol 3892. Springer-Verlag, Berlin, pp 171–180
- Kobayashi S (2005) Testing structure-freeness of regular sets of biomolecular sequences. In: Feretti C, Mauri G, Zandron C (eds) *Proceedings of DNA computing 10*, Milan, Italy, June 2004. Lecture notes in computer science, vol 3384. Springer-Verlag, Berlin, pp 192–201
- Kuske D, Weigel P (2004) The role of the complementarity relation in Watson-Crick automata and sticker systems. In: Calude C, Claude E, Dinneen M (eds) *DLT 2004: Proceedings of development in language theory*, Auckland, New Zealand, December 2004. Lecture notes in computer science, vol 3340. Springer-Verlag, Berlin Heidelberg, pp 272–283
- Lewin B (2007) *Genes IX*. Johns and Bartlett Publishers, Sudbury, MA
- Lipton R (1995) Using DNA to solve NP-complete problems. *Science* 268:542–545
- Liu W, Gao L, Zhang Q, Xu G, Zhu X, Liu X, Xu J (2005) A random walk DNA algorithm for the 3-SAT problem. *Curr Nanosci* 1:85–90
- Lothaire M (1983) *Combinatorics on words*. Encyclopedia of mathematics and its applications, vol 17. Addison-Wesley, Reading, MA
- Lyndon R, Schützenberger M (1962) The equation $a^m = b^n c^p$ in a free group. *Mich Math J* 9:289–298
- Manea F, Mitrana V, Yokomori T (2009) Two complementary operations inspired by the DNA hairpin formation: completion and reduction. *Theor Comput Sci* 410(4–5):417–425
- Marathe A, Condon A, Corn R (2001) On combinatorial DNA word design. *J Comput Biol* 8(3):201–220
- Martin-Vide C, Păun G, Rozenberg G, Salomaa A (1998) Universality results for finite H systems and for Watson-Crick automata. In: Păun G (ed) *Computing with bio-molecules. Theory and experiments*. Springer, Berlin, pp 200–220
- Mateescu A, Rozenberg G, Salomaa A (1998) Shuffle on trajectories: syntactic constraints. *Theor Comput Sci* 197:1–56
- Mauri G, Ferretti C (2004) Word design for molecular computing: a survey. In: Chen J, Reif J (eds) *Proceedings of DNA computing 9*, Madison, WI, June 2003. Lecture notes in computer science, vol 2943. Springer, Berlin, pp 37–46
- Morimoto N, Arita M, Suyama A (1997) Stepwise generation of Hamiltonian path with molecules. In: Lundh D, Olsson B, Narayanan A (eds) *Proceedings of bio-computing and emergent computation*, Skovde, Sweden, September 1997. World Scientific, Singapore, pp 184–192
- Nagy B (2008) On 5'→3' sensing Watson-Crick finite automata. In: Garzon M, Yan H (eds) *Proceedings of DNA computing 13*, Memphis, TN, June 2007. Lecture notes in computer science, vol 4848. Springer-Verlag, Berlin, pp 256–262
- Petre E (2003) Watson-Crick ω-automata. *J Automata, Lang Combinatorics* 8:59–70
- Phan V, Garzon M (online 2008) On codeword design in metric DNA spaces. *Nat Comput* DOI 10.1007/s11047-008-9088-6
- Păun A, Păun M (1999) State and transition complexity of Watson-Crick finite automata. In: Ciobanu G, Păun G (eds) *FCT'99*, Iasi, Romania, August–September 1999. Lecture notes in computer science, vol 1684. Springer-Verlag, Berlin Heidelberg, pp 409–420
- Păun G, Rozenberg G (1998) Sticker systems. *Theor Comput Sci* 204:183–203

- Păun G, Rozenberg G, Salomaa A (1998) DNA computing: new computing paradigms. Springer, Berlin
- Păun G, Rozenberg G, Yokomori T (2001) Hairpin languages. *Int J Foundations Comput Sci* 12 (6):837–847
- Reif J, LaBean T, Pirrung M, Rana V, Guo B, Kingsford C, Wickham G (2002) Experimental construction of very large scale DNA databases with associative search capability. In: Jonoska N, Seeman N (eds) Proceedings of DNA computing 7, Tampa, FL, June 2001. Lecture notes in computer science, vol 2340. Springer-Verlag, Berlin, pp 231–247
- Rose J, Deaton R, Hagiya M, Suyama A (2002) PNA-mediated whiplash PCR. In: Jonoska N, Seeman N (eds) Proceedings of DNA computing 7, Tampa, FL, June 2001. Lecture notes in computer science, vol 2340. Springer-Verlag, Berlin, pp 104–116
- Sager J, Stefanovic D (2006) Designing nucleotide sequences for computation: a survey of constraints. In: Carbone A, Pierce N (eds) Proceedings of DNA computing 11, London, ON, Canada. Lecture notes in computer science, vol 3892. Springer-Verlag, Berlin, pp 275–289
- Sakakibara Y, Kobayashi S (2001) Sticker systems with complex structures. *Soft Comput* 5:114–120
- Salomaa A (1973) Formal languages. Academic, New York
- SantaLucia J (1998) A unified view of polymer, dumbbell and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc Nat Acad Sci U S A* 95(4):1460–1465
- Sempere J (2007) On local testability in Watson-Crick finite automata. In: Proceedings of the international workshop on automata for cellular and molecular computing, tech report of MTA SZTAKI, Budapest, Hungary, August 2007. pp 120–128
- Sempere J (2008) Exploring regular reversibility in Watson-Crick finite automata. In: AROB 2008: Proceedings of 13th international symposium on artificial life and robotics, Beppu, Japan, January–February 2008. pp 505–509
- Smith G, Fiddes C, Hawkins J, Cox J (2003) Some possible codes for encrypting data in DNA. *Biotechnol Lett* 25:1125–1130
- Stemmer W, Crameri A, Ha K, Brennan T, Heyneker H (1995) Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides. *GENE* 164:49–53
- Tulpan D, Hoos H, Condon A (2003) Stochastic local search algorithms for DNA word design. In: Hagiya M, Ohuchi A (eds) Proceedings of DNA computing 8, Sapporo, Japan, June 2002. Lecture notes in computer science, vol 2568. Springer-Verlag, Berlin, pp 229–241
- Turner P, McLennan A, Bates A, White M (2000) Instant notes in molecular biology, 2nd edn. Garland, New York
- Wang X, Bao Z, Hu J, Wang S, Zhan A (2008) Solving the SAT problem using a DNA computing algorithm based on ligase chain reaction. *Biosystems* 91: 117–125
- Wood D (1987) Theory of computation. Harper & Row, New York
- Yoshida H, Suyama A (2000) Solution to 3-SAT by breadth-first search. In: Winfree E, Gifford D (eds) DNA based computers V. DIMACS series in discrete mathematics and theoretical computer science, vol 54. American Mathematical Society, Providence, RI, pp 9–22
- Yu S (2002) State complexity of finite and infinite regular languages. *Bull EATCS* 76:142–152

34 Molecular Computing Machineries — Computing Models and Wet Implementations

*Masami Hagiya¹ · Satoshi Kobayashi² · Ken Komiya³ · Fumiaki Tanaka⁴ ·
Takashi Yokomori⁵*

¹Department of Computer Science, Graduate School of Information
Science and Technology, The University of Tokyo, Tokyo, Japan
hagiya@is.s.u-tokyo.ac.jp

²Department of Computer Science, University of
Electro-Communications, Tokyo, Japan
satoshi@cs.uec.ac.jp

³Interdisciplinary Graduate School of Science and Engineering, Tokyo
Institute of Technology, Yokohama, Japan
komiya@dis.titech.ac.jp

⁴Department of Computer Science, Graduate School of Information
Science and Technology, The University of Tokyo, Tokyo, Japan
fumi95@is.s.u-tokyo.ac.jp

⁵Department of Mathematics, Faculty of Education and Integrated Arts
and Sciences, Waseda University, Tokyo, Japan
yokomori@waseda.jp

1	<i>Introduction</i>	1130
2	<i>Molecular Machine Models</i>	1133
3	<i>Computing Models with Structured Molecules</i>	1148
4	<i>Molecular Implementations and Applications</i>	1156

Abstract

This chapter presents both the theoretical results of molecular computing models mainly from the viewpoint of computing theory and the biochemical implementations of those models in wet lab experiments. Selected topics include a variety of molecular computing models with computabilities ranging from finite automata to Turing machines, and the associated issues of molecular implementation, as well as some applications to logical controls for circuits and medicines.

1 Introduction

Molecular computing machineries (molecular machineries, in short), the primary theme of this chapter, are novel computing devices inspired by the biochemical nature of biomolecules. In the early days it was understood that molecular machineries would replace silicon-based hardware with biological hardware through the development of molecular-based computers. However, it is now recognized that molecular machineries should not substitute for existing computers but complement them.

One may be surprised at how early the term “molecular computers” appeared in the literature. In fact, one can go back to an article of *Biofizika* in 1973 where an idea of cell molecular computers is discussed (Vaintsvaig and Liberman 1973). Since 1974, M. Conrad, one of the pioneering researchers in this area, has been conducting consistent research on information-processing capabilities using macromolecules (such as proteins) (Conrad 1985), and he edited a special issue entitled *Molecular Computing Paradigms* in Conrad (1992). These initial efforts in the short history of molecular computing have been succeeded by Head’s pioneering work on splicing systems and languages in Head (1987) whose theoretical analysis on splicing phenomena is highly appreciated, in that it was the first achievement of mathematical analysis using biochemical operations of DNA recombination. It is also worth mentioning that, prior to Head’s work, one can find some amount of study dealing with DNA/RNA molecules in the framework of formal language theory, such as Eberling and Jimenez-Montano (1980), Jimenez-Montano (1984), and Brendel and Busse (1984). Then, all of these initial works were eventually evoked by Adleman’s groundbreaking attempt in 1994 (Adleman 1994), which decisively directed this new research area of *molecular computing*.

This chapter presents both the theoretical results of molecular computing models mainly from the viewpoint of computing theory and the biochemical implementations of those models in wet lab experiments. Selection of topics included here might reflect the authors’ taste and it is intended to give concise and suggestive expositions for each chosen topic.

The chapter is organized as follows. This section ends by providing the rudiments of the theory of formal languages and computation. [Section 2](#) presents a variety of molecular computing models that are classified into two categories: the first is for molecular computing models with *Turing computability* and the second is for the molecular models of *finite automata*. Among the various ideas proposed for realizing Turing computability within the framework of molecular computing theory, four typical approaches to achieving universal computability were selected. These are based on formal grammars, equality sets, Post systems, and logical formulae. It is hoped that succinct expositions of those models may help the reader to gain some new insight into the theory of molecular computing models.

Most of the molecular computing models developed so far involve both the use of structural molecules with various complexity and sophisticated control of them.

⦿ **Section 3** is devoted to those *computing models with structured molecules* in which hairpin structures, tree structures, and more involved structures are focused on, among others.

In ⦿ **Sect. 4** attention is turned to the wet implementation issues of the molecular computing models and their applications as well. The lineup of the chosen topics is as follows:

- Enzyme-based implementation of *DNA finite automata* and its application to drug delivery
- Molecular design and implementation of *logic gates and circuits* using DNAzymes and *DNA tiles*
- Abstract models called *reaction graphs* for representing various reaction dynamics of DNA assembly pathways
- DNA implementation of finite automata with *whiplash machines*
- Hairpin-based implementation of *SAT engines* for solving the 3-SAT problem

The results mentioned above are just some successful examples in the area of molecular computing machineries that have been realized through devoted efforts and collaboration between biochemists and computer scientists. It is hoped that they will be able to provide a good springboard for advanced beginners in this area and for senior researchers in other areas.

1.1 Basic Notions and Notations

The basic notions and notations required to understand this chapter are summarized below. Then standard notions and notations are assumed from the theory of formal languages (see, e.g., Hopcroft et al. 2001, Rozenberg and Salomaa 1997, and Salomaa 1985).

For a set X , $|X|$ denotes the cardinality of X . The power set of X (i.e., the set of all subsets of X) is denoted by $\mathcal{P}(X)$.

An *alphabet* is a nonempty finite set of symbols. For an alphabet V , V^* denotes the set of all strings (of finite length) over V , where the empty string is denoted by ϵ (or λ). By V^+ one denotes the set $V^* - \{\epsilon\}$. A *language* over V is a subset of V^* . For a string $x = a_1a_2\dots a_n$ over V , the *mirror image* of $x \in V^*$, denoted by $mi(x)$ (or by x^R), is defined by $a_n \dots a_2a_1$.

A collection of languages over an alphabet V is called a *family of languages* (*language family*) or a *class of languages* (*language class*) over V .

A *phrase-structure grammar* (or *Chomsky grammar*) is a quadruple $G = (N, T, S, P)$, where N and T are disjoint alphabets of nonterminals and terminals, respectively, $S (\in N)$ is the initial symbol, and P is a finite subset of $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$. An element (u, v) of P , called a rule, is denoted by $u \rightarrow v$.

For strings $x, y \in (N \cup T)^*$, let

$$x \Rightarrow_G y \stackrel{\text{def}}{\iff} \exists x_1, x_2 \in (N \cup T)^* \text{ and } u \rightarrow v \in P \text{ such that } x = x_1ux_2, y = x_1vx_2.$$

A language generated by G is defined as $L(G) = \{x \in T^* | S \Rightarrow_G^* x\}$

A phrase-structure grammar $G = (N, T, S, P)$ is

1. *Context-sensitive* if for $u \rightarrow v \in P$ there exist $u_1, u_2 \in (N \cup T)^*$, $A \in N$, $x \in (N \cup T)^+$ such that $u = u_1Au_2$, $v = u_1xu_2$
2. *Context-free* if for $u \rightarrow v \in P$, $u \in N$
3. *(Context-free) linear* if for $u \rightarrow v \in P$, $u \in N$, and $v \in T^* NT^* \cup T^*$
4. *Regular* if for $u \rightarrow v \in P$, $u \in N$, and $v \in T \cup TN \cup \{\epsilon\}$

RE, *CS*, *CF*, *LIN*, and *REG* denote the families of languages generated by phrase-structure grammars, context-sensitive grammars, context-free grammars, (context-free) linear grammars, and regular grammars, respectively. *RE* comes from the alternative name of *recursively enumerable* language. The following inclusions (called the *Chomsky hierarchy*) are proved.

$$REG \subset LIN \subset CF \subset CS \subset RE$$

A (nondeterministic) *Turing machine* (TM) is a construct $M = (Q, V, T, p_0, F, \delta)$, where Q, V are finite sets of states and a tape alphabet, respectively. V contains a special symbol B (a blank symbol), $T(\subseteq V - \{B\})$ is an input alphabet, $p_0 (\in Q)$ is the initial state, $F (\subseteq Q)$ is a set of final states, and δ is a function from $Q \times V$ to $\mathcal{P}(Q \times V \times \{L, R\})$ acting as follows: for $p, p' \in Q, a, b \in V, d \in \{L, R\}$, if $(p', b, d) \in \delta(p, a)$, then M changes its state from p to p' , rewrites a as b , and moves the head to the left (if $d = L$) or the right (if $d = R$). M is called *deterministic* (denoted by DTM) if for each $(p, a) \in Q \times V, |\delta(p, a)| \leq 1$.

A configuration in M is a string of the form: xpy , where $x \in V^*$, $y \in V^*(V - \{B\}) \cup \{\epsilon\}$, $p \in Q$, and the head of M takes its position at the leftmost symbol of y . A binary relation \vdash on the set of configurations in M is defined by

$$\left\{ \begin{array}{l} xpay \vdash xbqy \iff (q, b, R) \in \delta(p, a) \\ xp \vdash xbq \iff (q, b, R) \in \delta(p, B) \\ xpay \vdash xqby \iff (q, b, L) \in \delta(p, a) \\ xcp \vdash xqcb \iff (q, b, L) \in \delta(p, B) \end{array} \right.$$

where $a, b, c \in V, x, y \in V^*, p, q \in Q$. Let \vdash^* be the reflexive, transitive closure of \vdash . A language $L(M)$ recognized by M is defined as

$$L(M) = \{w \in T^* \mid p_0 w \vdash^* xpy, \text{ for some } x, y \in V^*, p \in F\}$$

that is, the set of input strings such that, starting with the initial state, they drive their configurations in M to a final state. It is known that the family of languages recognized by Turing machines is equal to the family *RE*.

As subfamilies of *RE*, there are many to be noted. Among others, two families are of importance: **NP** and **P** are the families of languages recognized in polynomial-time, that is, in $f(n)$ steps by TMs and DTMs, respectively, where f is a polynomial function defined over the set of nonnegative integers and n is the size of the input string. It remains open whether **P** = **NP** or not, while the inclusion **P** ⊆ **NP** is clear from the definition. In particular, a subfamily of **NP** called *NP-complete* is of special interest, because NP-complete languages are representative of the hardness in computational complexity of **NP** in the sense that any language in **NP** is no harder than the NP-complete languages. Further, by **NSPACE**($f(n)$) and **DSPACE**($f(n)$), the families of languages recognized in $f(n)$ space (memory) are denoted by TMs and DTMs, respectively.

Due to Watson–Crick complementarity, not only single-stranded molecules but also completely hybridized double-stranded molecules can be regarded as strings over an alphabet {A, C, G, T}. Therefore, it is natural to employ the formal framework of the classical formal language theory, in order to study and formulate notions in molecular computing.

Given an alphabet Σ , a *morphism* h over Σ is a mapping defined by $h(a) \in \Sigma^*$ for each $a \in \Sigma$. Any h is extended to $h: \Sigma^* \rightarrow \Sigma^*$ by $h(\epsilon) = \epsilon$ and $h(xa) = h(x)h(a)$ for each $x \in \Sigma^*, a \in \Sigma$. A *weak coding* over Σ is a morphism h such that $h(a)$ is in $\Sigma \cup \{\epsilon\}$ for each $a \in \Sigma$.

An *involution* over Σ is a morphism h such that $h(h(a)) = a$ for each $a \in \Sigma$. Any involution h over Σ such that $h(a) \neq a$ for all $a \in \Sigma$ is called a *Watson–Crick morphism* over Σ .

In analogy to DNA molecules (where A and T (C and G) are complementary pairs), for $x, y \in \Sigma^*$, if $h(x) = y$ (and therefore, $h(y) = x$), then it is said that x and y are *complementary* to each other, and denoted by $\bar{x} = y$ ($\bar{y} = x$). An alphabet $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ is called the *complementary alphabet* of Σ .

2 Molecular Machine Models

Since Adleman's groundbreaking work on the DNA implementation of computing a small instance of the directed Hamiltonian path problem (Adleman 1994), numerous research papers in this new computation paradigm have been published. In fact, Adleman's model has been extensively studied by many researchers seeking to generalize his technique to solve larger classes of problems (Lipton 1995; Lipton and Baum 1996), provide abstract DNA computer models with Turing computability (Adleman 1996; Beaver 1995; Rothemund 1995; Winfree et al. 1996), and so forth.

In this section, the focus is on computing models of molecular machines with Turing computability and others, restricted primarily to finite automata.

2.1 Molecular Models with Turing Computability

2.1.1 Grammar Approach

The Turing machine or its grammatical equivalent (like phrase-structure grammar) is one of the most common computing models when one wants to prove the universal computability of a new computing device. Among others, most interesting is the normal form theorems proved by Geffert (1991), which tell us that each recursively enumerable language can be generated by a phrase-structure grammar with only five nonterminals $\{S, A, B, C, D\}$ (S is the initial nonterminal) and with all of the context-free rules being of the form $S \rightarrow v$ and two extra (non context-free) rules $AB \rightarrow \epsilon$ and $CD \rightarrow \epsilon$. Thus, the feature of this normal form is of great importance in that *most* of the computation processes can be carried out without checking any context-sensitivity requirement.

Geffert derived an interesting normal form theorem for phrase-structure grammars which can be conveniently rephrased here with some modifications.

Proposition 1 (Geffert 1991) *Each recursively enumerable language L over Σ can be generated by a phrase-structure grammar with $G = (\{S, A, C, G, T\}, \Sigma, R \cup \{TA \rightarrow \epsilon, CG \rightarrow \epsilon\}, S)$, where $R = R_1 \cup R_2 \cup R_3$ and*

$$R_1 = \{S \rightarrow z_a Sa \mid a \in \Sigma \text{ and } z_a \text{ is in } \{T, C\}^*\} \text{ (called Type-(i) rules)}$$

$$R_2 = \{S \rightarrow uSv \mid u \text{ is in } \{T, C\}^* \text{ and } v \text{ is in } \{A, G\}^*\} \text{ (called Type-(ii) rules)}$$

$$R_3 = \{S \rightarrow u\} \text{ (called Type-(iii) rule), where } u \text{ is in } \{T, C\}^*$$

The following properties of the phrase-structure grammar G in normal form in Proposition 1 are quite useful for deriving the computing model discussed later.

1. It is not until a Type-(iii) rule has been applied that two extra cancellation rules ($TA \rightarrow \epsilon$ and $CG \rightarrow \epsilon$) are available.

2. Mixed use of Type-(i) and Type-(ii) rules leads to sentential forms for which no chance to get terminal strings remains.
3. Any occurrence of $x (\in \Sigma)$ in any substring (except for a suffix) of a sentential form blocks a terminal string.
4. A successful derivation process can be divided into three phases: At the first phase (*Generation*), Type-(i) rules are used. The second phase (*Extension*) consists of using Type-(ii) rules and ends up with one application of the Type-(iii) rule. Finally, the third phase (*Test*) erases well-formed pairings (TA or CG) using two extra rules only.
5. There is only one occurrence of either TA or CG in any sentential form to which cancellation rules can be applied.
6. The *Test* phase can only be started up with the unique position where either TA exclusively or CG appears immediately after one application of the Type-(iii) rule.

Grammar-Based Computation of Self-Assembly: (GCS)

We can now present a way of DNA computation, GCS, which is based on the features of Geffert normal form grammars in [Proposition 1](#). GCS (this is referred to as YAC in the original paper (Yokomori 2000)), grammar-based computation of self-assembly, has the following set of basic components assembled:

1. For each Type-(ii) rule: $S \rightarrow uSv$, construct a component with the shape illustrated in [Fig. 1b](#), and let it be called a *Type-(ii)* component.
2. For the unique Type-(iii) rule: $S \rightarrow u$, construct a component with the shape illustrated in [Fig. 1c](#), where a connector is attached as an endmarker. This is called a *Type-(iii)* component.
3. Finally, for a given input string $w = b_1 \dots b_n (\in \Sigma^*)$ to be recognized, a component in the shape shown in [Fig. 1a](#) is prepared, where Z_w of the component in (a) is the binary coding of the input w and $\overline{Z_w}$ denotes the complementarity sequence of Z_w .

Under the presence of *ligase*, all these components with sufficiently high concentration in a single pot of a test tube are provided. (See [Fig. 1](#).)

Finally, the following operation is used:

- d-Detect: Given a test tube that may contain double-stranded strings, return “yes” if it contains (at least) one *completely hybridized* double-stranded string, and “no” otherwise.

[Figure 1](#) illustrates a computation schema for GCS, where a random pool is a multi-set of basic components with sufficiently high concentration that can produce all possible assembly necessary for recognizing an input string. Besides the operation d-Detect, GCS only requires two basic operations to execute its computation process: *annealing* and *melting*.

In summary, one has the following:

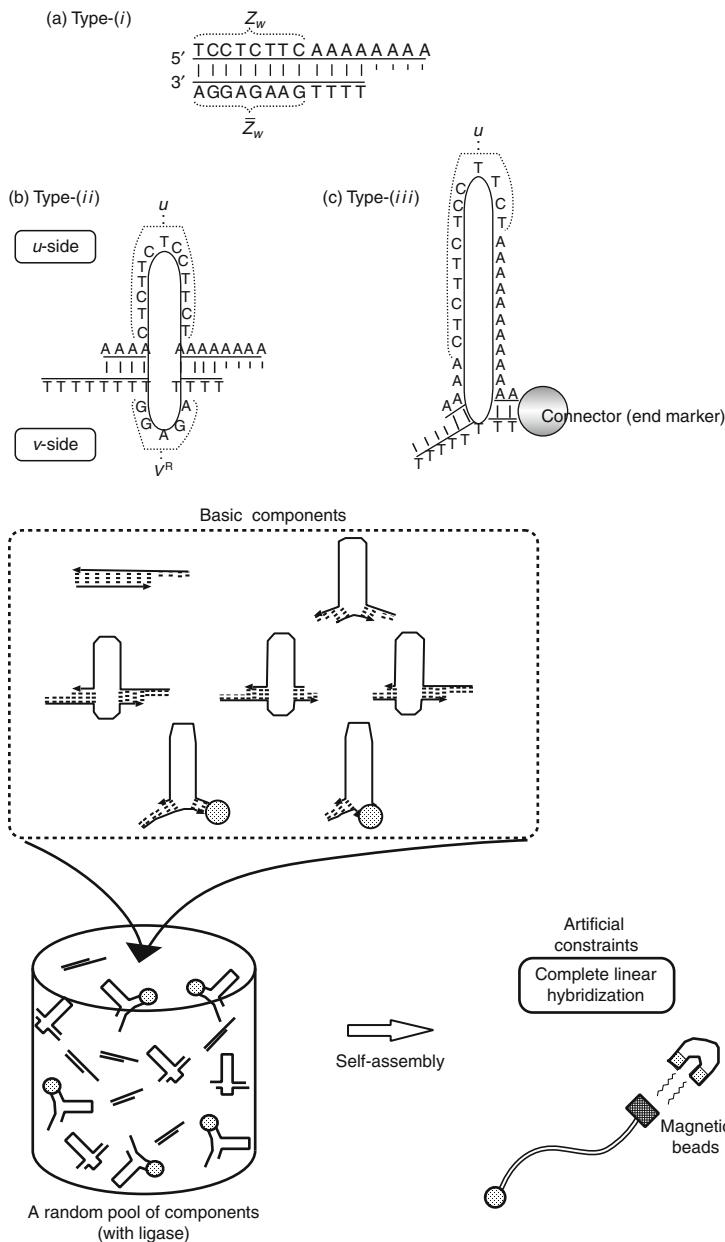
Theorem 1 (Yokomori 2000) *Any recursively enumerable language can be recognized by the GCS computation schema.*

(Note that, in contrast, GCS is based on grammatical formulation, its functional behavior is close to an acceptor rather than a generative grammar.)

The structural simplicity of computation assembly units in GCS is expected to facilitate its implementation at least in its design schema. In particular, the two groups of four DNAs (A,G:

Fig. 1

Basic components and a successful computation using the grammar-based computation process (GCS): A large quantity of molecules consisting of all basic components are put into a single test tube. Then, hybridization of those molecules produces candidate complex molecules for successful computations. By first melting the complexes and then annealing them, a desired double-stranded molecule is detected using the w -probe attached with magnetic beads when the input w is in L .



purine and C,T: pyrimidine) can be readily applied both to encoding the computational information of a language into its GCS assembly units and to checking the completeness of the computation results in the final process with its complementary property. On the other hand, as for the biomolecular feasibility of the GCS, it still remains at the conceptual level, and more molecular biological investigation is needed before the model becomes truly feasible.

2.1.2 Equality Checking Approach

In the theory of formal languages and computation, a simple computational principle called *equality checking* can often play an important role. The equality checking principle (EC principle) is simple enough for humans to understand and for machines of any kind to implement. In particular, it seems very suitable for biomolecular machines, because this EC principle only requires the task of equality checking of two memories, represented as strings of symbols, just like a completely hybridized *double-stranded DNA sequence*.

Based on the EM approach, one can present an abstract model of DNA computing, DNA-EC, which has the universal computability of Turing machines.

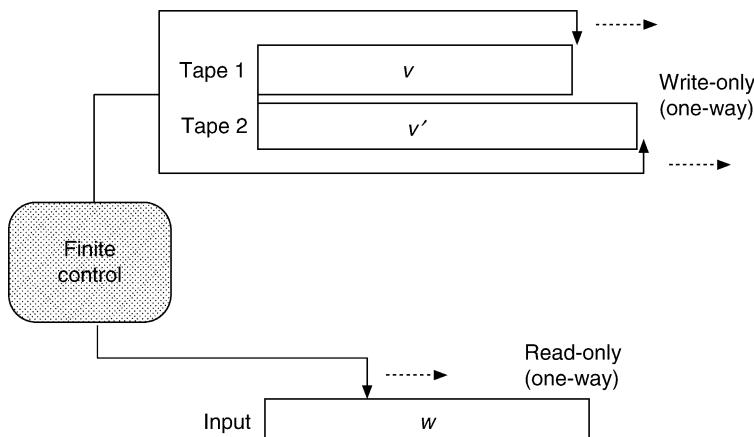
Equality Machines

An *equality machine* (Engelfriet and Rozenberg 1980) is an abstract acceptor shown in Fig. 2. Formally, an equality machine (EM) M is a structure $(Q, \Sigma, \Delta, \delta, q_0, F)$, where Q is the finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ is the input alphabet, Δ is the output alphabet, and δ is a finite set of transition rules of the form: $(p, a) \rightarrow (q, u, i)$, where $p, q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $u \in \Delta^*$, and $i \in \{1, 2\}$.

A configuration of M is of the form $(p, w, (v_1, v_2))$, where $p \in Q$, $w \in \Sigma^*$, and $v_1, v_2 \in \Delta^*$. If $(p, a) \rightarrow (q, u, i) \in \delta$, then one writes $(p, aw, (v_1, v_2)) \vdash (q, w, (v'_1, v'_2))$, where $v'_1 = v_1u$ and $v'_2 = v_2$.

Fig. 2

Equality machine EM: the machine consists of a one-way *input tape*, a *finite control*, and two *write-only* tapes for its memory. With the initial empty memory tapes, the machine starts to read the input from the left and changes its state, then extends one of the two memory tapes by concatenating an output string, according to the transition rules specified. At the end of computation, where the input string has been fully read, the machine accepts it only if the contents of the two memory tapes are identical.



(if $i = 1$), and $v'_1 = v_1$ and $v'_2 = v_2 u$ (if $i = 2$). The reflexive and transitive closure of \vdash is denoted by \vdash^* . The language accepted by M is $L(M) = \{w \in \Sigma^* \mid (q_0, w, (\epsilon, \epsilon)) \vdash^* (q, \epsilon, (v, v)) \text{ for some } q \in F \text{ and } v \in \Delta^*\}$. An EM M is *deterministic* if δ is a mapping from $Q \times \Sigma$ into $Q \times (\Delta \times \{1,2\})^*$.

The classes of languages accepted by nondeterministic and deterministic EMs are denoted by $\mathcal{L}_N(EM)$ and $\mathcal{L}_D(EM)$, respectively.

Very interesting results from Engelfriet and Rozenberg (1980) are summed up here which motivated the results in this section.

Proposition 2 (Engelfriet and Rozenberg 1980) *The following relations hold true:*

1. $\mathcal{L}_N(EM) = RE$
2. $\mathcal{L}_D(EM) \subseteq \mathbf{DSPACE}(\log n)$ and $REG \subset \mathcal{L}_D(EM) \subset \mathbf{NSPACE}(n) = CS$
3. $\mathcal{L}_D(EM)$ is incomparable to CF but not disjoint.

DNA-EC: A Model of DNA Computing

Following the notations and definitions in Rooß and Wagner (1996), the DNA computation model called DNA-EC is described, which is based on the characterization results given in the previous section.

In the *test tube computation* discussed below, T denotes a test tube containing a set of strings over some fixed alphabet Σ , and its contents are denoted by $I(T)$. Further, it is assumed that $I(T) \subseteq \Sigma^*$ unless stated otherwise.

The DNA-EC requires the following two set *test* operations:

- EM (*Emptiness Test*): Given a test tube T , return “yes” if $I(T)$ contains a string (i.e., $I(T) \neq \emptyset$) and “no” if it contains none.
- EQ (*Equivalence Test*): Given a test tube T (where $I(T) \subseteq D^*$) that may contain double-stranded strings, return “yes” if $I(T)$ contains (at least) one complete double-stranded string, and “no” otherwise. (Note that D is the alphabet for representing double strands.)

Let T, T_1, T_2 be set variables for test tubes used in DNA-EC. ▶ [Table 1](#) shows a collection of *set* operations used in DNA-EC, where $a, b, c, d \in \Sigma$, $u, v, w \in \Sigma^*$.

Let \mathcal{O} and \mathcal{T} be collections of set operations and set test operations, respectively, introduced above. Then, $\text{DNA}(\mathcal{O}, \mathcal{T})\text{-EC}$ is the class of languages recognized by DNA-EC, which allows the use of set operations from \mathcal{O} and of set test operations from \mathcal{T} .

■ **Table 1**

Set operations (from Rooß and Wagner 1996)

Name	Syntax	$I(T)$ after the Operation
UN(union)	$T := T_1 \cup T_2$	$I(T_1) \cup I(T_2)$
LC(left cut)	$T := a \setminus T_1$	$\{u \mid au \in I(T_1)\}$
LA(left adding)	$T := a \cdot T_1$	$\{a\} \cdot I(T_1)$
EX(extraction)	$T := \text{Ex}(T_1, w)$	$I(T_1) \cap \Sigma^* \{w\} \Sigma^*$
RE(replacement)	$T := \text{Re}(T_1, u, v)$	$\{xvy \mid xuy \in I(T_1)\}$
RV(reversal)	$T := T_1^R$	$\{u^R \mid u \in I(T)\}$ (u^R is the reversal of u)

EX is essentially the same as the one often referred to as *Separation* in the literature

As for the computational power of DNA-EC, it is proved that given an EM $M = (Q, \Sigma, \Delta, \delta, q_0, F)$ and $w \in \Sigma^*$, the computation process of M with w is simulated within the DNA-EC model in a variety of ways. The idea for the simulation is simply to keep only the difference between the contents of Tapes 1 and 2 in the configuration of w in the computing schema dealing with a sequence of DNAs as a single string. The hybridization property of DNA complementarity can also be readily used for checking two memories in the case of manipulating *double-stranded strings* within the schema. The obtained results include the following:

Theorem 2 (Yokomori and Kobayashi 1999) *The following equalities hold:*

1. $\text{DNA}(\{\text{UN,LA,RV,RE}\}, \{\text{EM}\})\text{-EC} = \text{DNA}(\{\text{UN,RE}\}, \{\text{EM}\})\text{-EC} = \text{RE}$
2. $\text{DNA}(\{\text{UN,LA,LC,EX}\}, \{\text{EQ}\})\text{-EC}$

2.1.3 Post System Approach

Among a variety of computing models based on string rewriting, a classical Post system will be introduced below that has a unique position, somewhere between formal grammars and logical systems. Because of the simplicity of its components and functional behavior, a Post system can sometimes provide a clear picture of the essence of the computation.

A Lesson From Post Systems

A rewriting system called the *general regular Post system* (GRP system) is a quadruple $G = (V, \Sigma, P, A)$, where V and $\Sigma (\subseteq V)$ are alphabets, P is a finite set of rules of the form of either $uX \rightarrow wX$ or $aX \rightarrow Xb$ ($X \notin V$: variable, $u, w \in V^*, a, b \in V$), $A (\subset V^+)$ is a finite set of axioms.

Given strings $\alpha, \beta \in V^*$, define a binary relation \Rightarrow as follows:

$$\alpha \Rightarrow \beta \stackrel{\text{def}}{\iff} \begin{cases} \exists uX \rightarrow wX \in P, \delta \in V^* [\alpha = u\delta, \beta = w\delta] & \text{or} \\ \exists aX \rightarrow Xb \in P, \delta \in V^* [\alpha = a\delta, \beta = \delta b] \end{cases}$$

Let \Rightarrow^* be the reflexive and transitive closure of \Rightarrow . Then, a language generated by G is defined as

$$L(G) = \{w \in \Sigma^* \mid \exists u \in A \text{ such that } u \Rightarrow^* w\}$$

It is known that the family of languages generated by GRP systems coincides with the family of recursively enumerable languages (Salomaa 1985). The source of the universal computing capability of GRP systems comes from the use of two types of rules $\alpha X \rightarrow \beta X$ (say, type 1 rule) and $aX \rightarrow Xb$ (type 2 rule). The generative capability between these two types of rule is interesting.

Theorem 3 (Post 1943) *GRP systems with type 1 rules alone can generate only regular languages, while type 1 rules together with type 2 rules enable GRP systems to generate any recursively enumerable language.* (This fact gives the base of a technique called the “rotate-simulate method” used to prove that a computing model has universal computability.)

This result seems to be useful in analyzing the computing power of rewriting systems. As seen in the subsequent discussion, this observation leads to new types of computing models with universal computability based on splicing operations.

Multi-test Tube Models

We note that Type 1 rules of GRP systems can be generally simulated by some devices of existing rewriting models of computation. Therefore, in order to achieve Turing computability, one has to answer the question: “Is there any way to carry out a rotation rule $aX \rightarrow Xb$ within the computing system?” One possible way is to consider an extended model of test tube systems studied in Csuhaj-Varju et al. (1996) and Kobayashi and Sakakibara (1998).

An *elementary formal system* (EFS) is a triple $E = (D, \Sigma, M)$, where D is a finite set of predicate symbols, Σ is a finite alphabet, and M is a set of logic formulas. Then, F in M is an *H-form* iff it is of the form:

$$P(x_1 u_1 v_1 y_2) \leftarrow Q(x_1 u_1 u_2 x_2) R(y_1 v_1 v_2 y_2) \text{ or } P(w) \leftarrow$$

where x_1, x_2, y_1, y_2 are all distinct variables, u_1, u_2, v_1, v_2 and w are in Σ^* .

An EFS $E = (D, \Sigma, M)$ is called *H-form EFS* iff every F in M is an H-form. Further, an H-form EFS with a single predicate P , that is, an H-form EFS $E = (\{P\}, \Sigma, M)$ is called a *simple H-form EFS*.

Given a predicate P and EFS $E = (\{P\}, \Sigma, M)$, define

$$L(E, P) = \{w \in \Sigma^* | P(w) \text{ is provable from } E\}$$

It is known that a simple H-form EFS is equivalent to an H-system in its computing capability.

Theorem 4 For any H-system $S = (\Sigma, R, A)$, there effectively exists a simple H-form EFS $E = (\{P\}, \Sigma, M)$ such that $L(E, P) = L(S)$. Conversely, for any simple H-form EFS $E = (\{P\}, \Sigma, M)$, one can construct an H-system $S = (\Sigma, R, A)$ such that $L(S) = L(E, P)$.

Therefore, in order to get the computing power beyond H-systems, it is necessary to consider an H-form EFS with more than one predicate symbol, which leads to the idea of “multi-splicing models” where more than one test tube is available for splicing operations.

Given an $n > 0$, by HE_n the family of languages defined by H-form EFSs is denoted with n predicate symbols. Further, let $HE_* = \bigcup_{i \geq 1} HE_i$.

Then, the following has been proved.

Theorem 5

1. $HE(FIN, FIN) = HE_1 \subset HE_2 \subseteq HE_3 \subseteq \dots \subseteq HE_{13} = \dots = HE_* = RE$.
2. HE_2 contains non-regular languages.
3. HE_3 contains non-context-free languages.

Thus, 13 test tubes are sufficient to generate any recursively enumerable language in multi-splicing models.

The way to simulate an H-form EFS $E = (D, R, A)$ by a multi-splicing model is outlined below, where $|D| = 2$, that is, two test tubes T_1 and T_2 are available (Kobayashi and Sakakibara 1998). It suffices to show how to simulate $aX \rightarrow Xb$ using splicing operations in two test tubes. For a given string $w = ax$, make $w' = BaxE$, where B and E are new symbols of marker. Then, delete a prefix Ba from w' in T_1 and transfer xE to T_2 that contains b . Then, replace E in xE with b , resulting in xb .

It remains open to solve how many test tubes is minimally necessary to generate any recursively enumerable language. See Csuhaj-Varju et al. (1996) for related discussion.

2.1.4 Logic Approach

Horn clause logic is one of the most important logical systems in mathematical logic. Its significance was first pointed out by the logician Alfred Horn (Horn 1951). Horn clause logic is a mathematical basis of a descriptive programming language, PROLOG, and is known to have Turing computability (Tarnlund 1977). Therefore, it is natural to construct a molecular computing device based on Horn clause computation (Kobayashi et al. 1997). Kobayashi proposed to use Horn clause logic as an underlying computational framework of a molecular computing machine (Kobayashi 1999). His idea is to implement a reduction process of a subclass of a Horn program, called a *simple* Horn program, with DNA molecules.

For a predicate symbol A and a sequence X_1, \dots, X_k of constants or variables, $A(X_1, \dots, X_k)$ is called a *simple atom*. If every X_k is a constant, $A(X_1, \dots, X_k)$ is called a *simple ground atom*. A *simple Horn clause* is a formula of the form $\forall X_1, \dots, X_m (F \leftarrow F_1 \wedge \dots \wedge F_n)$, where F and F_i ($1 \leq i \leq n$) are simple atoms. For a simple Horn clause $r : F \leftarrow F_1, \dots, F_n$, F and F_i ($i = 1, \dots, n$) are called a *head* and a *body* of r , respectively. A *simple Horn program* is a finite set of simple Horn clauses.

A substitution θ is a mapping from V to $V \cup C$. An *expression* means a constant, a variable, a sequence of atoms, or a clause. For an expression E and a substitution θ , $E\theta$ stands for the result of applying θ to E , which is obtained by simultaneously replacing each occurrence of a variable X in E by the corresponding variable or constant $X\theta$.

Let H be a simple Horn program and I be a set of simple ground atoms. Then, an *immediate consequence operator*, $T_H(I)$ is defined as follows:

$$T_H(I) = \{F \mid F_1, \dots, F_k \in I \wedge A \leftarrow B_1, \dots, B_k \in H \wedge \exists \theta \text{ s.t. } (B_i\theta = F_i \wedge A\theta = F)\}$$

Let Σ be a finite alphabet. A *decision problem* is a function from Σ^* to $\{0, 1\}$. Let Γ be an alphabet (possibly infinite) for representing simple Horn programs and Γ^* be the set of all strings of *finite* length over Γ . A function π from Σ^* to Γ^* is said to be an *encoding function for problem instances* if for every $w \in \Sigma^*$, $\pi(w)$ is a finite set of simple Horn clauses and π can be computed by a deterministic Turing machine in polynomial time. Let sz be a function from Σ^* to the set of all positive integers such that for every $w \in \Sigma^*$, $sz(w) \leq |w|$ and $sz(w)$ can be computed in polynomial time with respect to $|w|$ by a deterministic Turing machine. A family $\{H_i\}_{i \geq 1}$ of simple Horn programs *computes a decision problem*, f , with *encoding function*, π , and *size function*, sz , if there exists a predicate symbol A such that for every $w \in \Sigma^*$ with $sz(w) = n$, $f(w) = 1$ holds iff $A \in T_{H_{sz(w)} \cup \pi(w)}(\emptyset)$.

It is said that a function ϕ from $\{1\}^+$ to Γ^* generates a family $\{H_i\}_{i \geq 1}$ of simple Horn programs if for every positive integer i , $\phi(1^i) = H_i$ holds. A family $\{H_i\}_{i \geq 1}$ of simple Horn programs is said to be *uniform* if it is generated by some function, which can be computed by a deterministic Turing machine in polynomial time.

Then, one can show the following theorem:

Theorem 6 *For any decision problem P in NP, there exists a uniform family of simple Horn programs which computes P in polynomial steps.*

By the above theorem, efficient DNA implementation of the operation T_H for a simple Horn program enables us to construct a DNA computer that solves problems in NP in a polynomial number of biolab steps. One idea to implement T_H is given by Kobayashi, in which

the whiplash polymerase chain reaction (PCR) technique is used to (1) check the equality of arguments in a body of a clause and (2) copy an argument from a body to a head. (For details, refer to [Sect. 4.4](#) in this chapter.)

For representing ground atoms of a given simple Horn program, it is necessary to design sequences of the following oligos:

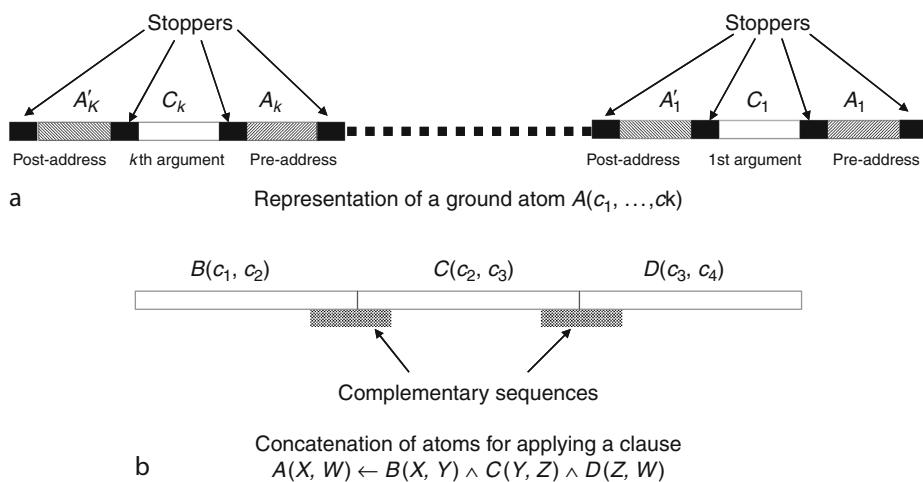
1. Oligos for representing constant symbols
2. For each predicate symbol A with k -arguments and for each i ($1 \leq i \leq k$), two different oligos A_i and A'_i , where A_i and A'_i are called a *pre-address* and a *post-address* of A at the i th position

Each ground atom $A(c_1, \dots, c_k)$ is represented as a single-stranded DNA consisting of k segments, each of which contains data c_i of each argument of A . Each segment is composed of three regions, *pre-address*, *data*, and *post-address*. The data region is an oligo representing a constant. Further, there exist stopper sequences in whiplash PCR at each of the segment ends, and at each end of the pre-address, data, and post-address regions. (See [Fig. 3a](#).)

The procedure for computing $T_H(I)$ is as follows. First, construct a set of ground instances of all possible bodies of rules in H . This is accomplished by concatenating ground atoms in DNA representation of I by using appropriate complementary sequences (see [Fig. 3b](#)). Then, check the equality constraint among arguments specified by the rule, which is implemented by using whiplash PCR with two different stopper sequences in a sophisticated manner. In this way, one can extract correct ground instances of all possible bodies of rules. Then, copy arguments from the body to the head by using whiplash PCR once again. For instance, in [Fig. 3b](#), the arguments c_1 and c_4 are copied to the head, which results in the DNA

Fig. 3

Representation of atoms and their ligation: (a) DNA representation of a ground atom $A(c_1, \dots, c_k)$. (b) Concatenation of atoms $B(c_1, c_2)$, $C(c_2, c_3)$, $D(c_3, c_4)$ for applying a clause $A(X, W) \leftarrow B(X, Y) \wedge C(Y, Z) \wedge D(Z, W)$. After checking the equivalence of corresponding arguments (the second of B and the first of C , the second of C and the first of D), one makes a new DNA sequence representing $A(c_1, c_4)$ by using the whiplash PCR technique.



representation of a ground atom $A(c_1, c_4)$. Refer to Kobayashi (1999) for details. In conclusion, from [Theorem 6](#) and the above discussion, one has:

- Claim 1**
- (1) *There effectively exists a bio-procedure based on the simple Horn clause computation model with Turing computability.*
 - (2) *For any decision problem P in NP, there effectively exists a bio-procedure of a polynomial number of steps for computing P based on the simple Horn clause computation model.*

Other approaches to the DNA implementation of Horn clause computation were also proposed by Uejima et al. (2001).

2.2 Molecular Finite State Machines

A *finite automaton* (FA) is one of the most popular concepts in theoretical computer science, and, hence, it is of great use to establish a simple implementation method for realizing an FA in the formulation of molecular computing. In particular, it is strongly encouraged to consider the problem of how simply one can construct a molecular FA using DNAs in the framework of *self-assembly computation*.

Formally, a *nondeterministic finite automaton* (NFA) is a construct $M = (Q, \Sigma, \delta, p_0, F)$, where Q is a set of states, Σ is the input alphabet, $p_0 (\in Q)$ is the initial state, $F (\subseteq Q)$ is a set of final states, and δ is defined as a function from $Q \times \Sigma$ to $\mathcal{P}(Q)$ (the power set of Q). (If δ is restricted to being a function from $Q \times \Sigma$ to Q , then M is said to be *deterministic* and is denoted by DFA.)

2.2.1 Bulge Loop Approach

The first method introduced here, proposed by Gao et al. (1999), consists of three basic phases: (1) *encoding* the transition rules of a given NFA with an input string onto DNA molecules, (2) *hybridization* and *ligation* which carry out the simulation of the state transition, and (3) *extracting* the results to decide the acceptance or rejection of the input string.

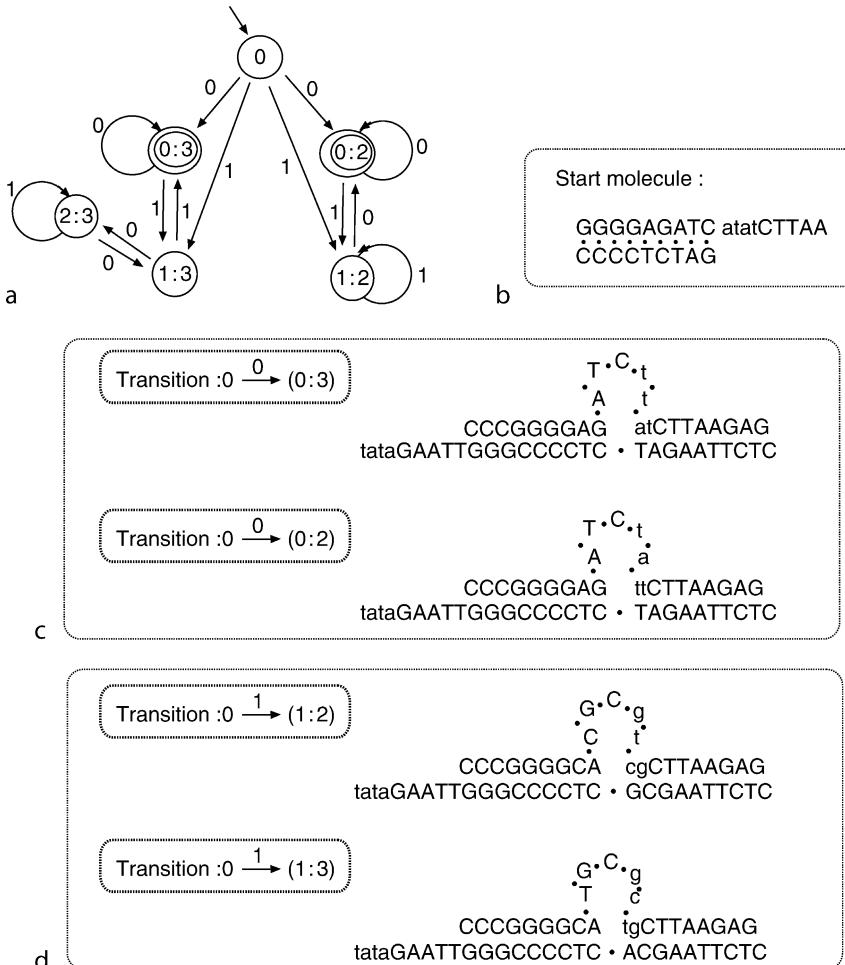
As an example, consider an NFA M with the initial state 0 and two final states (0:2) and (0:3) given as a transition graph in [Fig. 4a](#). For the phase (1), the set of six states is encoded by DNA sequences of length 4 as follows:

State 0	atat	State (0:3) ttat
State (1:3) gctg		State (2:3) ctca
State (0:2) tatt		State (1:2) gtcg

The start molecule is constructed as a double-stranded molecule with a sticky end shown in [Fig. 4b](#). Each transition rule $p \xrightarrow{a} q$ (where p, q are states, and a is in $\{0, 1\}$) is encoded as a molecule with a so-called “bulge loop” structure. For example, two transition rules: $0 \xrightarrow{0} (0:3)$ and $0 \xrightarrow{0} (0:2)$ are encoded as in [Fig. 4c](#), while two transition rules: $0 \xrightarrow{1} (1:3)$ and $0 \xrightarrow{1} (1:2)$ are encoded as in [Fig. 4d](#). All of the other transition rules have to be encoded in the same principle on structured molecules with bulge loops. (The idea behind the design principle of molecular encoding will be explained below.)

Fig. 4

NFA for divisibility by 2 or 3: Coding transition rules.



A computation in M is simulated in three steps as follows:

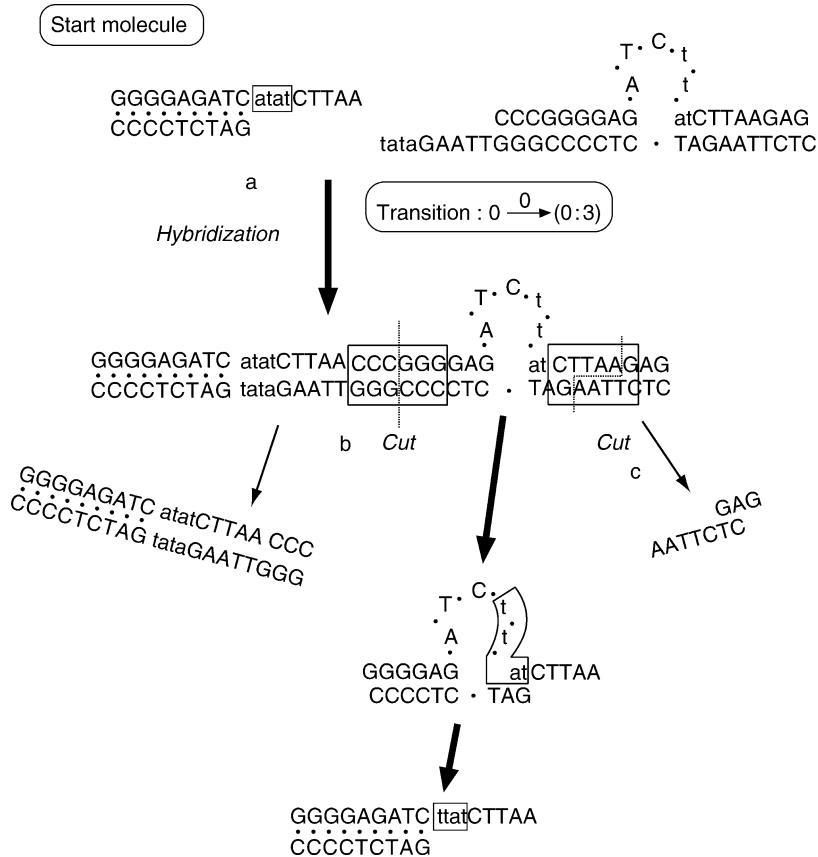
(Step 1): At the beginning of the reaction, one has the start molecule having the atatCTTAA overhang. Suppose that input 0 will be processed. Then, there are two possibilities (two molecules) that have a tata hangover, which matches atat in the hangover of the current state (of molecule). After hybridization and ligation, the remaining unreacted molecules are all washed away and collected. At this moment, the new possible states (0:3) and (0:2) from 0 with input 0 are covered by the loop part and do not show up yet. (The molecule potentially representing the state (0:3) is illustrated in Fig. 5a.)

(Step 2): Add a restriction enzyme *Sma*I which recognizes $\frac{\text{CCCCGG}}{\text{GGGCC}}$ and cut at the middle between C and G, providing two parts, one of which contains the bulge loop. (See Fig. 5b.)

(Step 3): Add a restriction enzyme *Eco*RI which recognizes $\frac{\text{CTTAAG}}{\text{GAATTTC}}$ on the portion with the bulge loop and cut it as shown in Fig. 5c, providing a molecule representing a new state (0:3), ready for the next new transition.

Fig. 5

Simulating transition with input 0 from the state 0.



The “bulge loop” structure functions to prevent the new state sequence in it from reaching the wrong blunt end reaction during ligation. The *Eco*RI recognition site is used for breaking the bulge loop to expose the next state (i.e., (0:3) or (0:2) in this example) for the subsequent reaction. That is, Steps (1)–(3) are repeated for as many input symbols as necessary. In this manner, it is seen that the nondeterministic transitions in M can be simulated by a sequence of chemical reactions with a certain chemical control.

This implementation method has several advantages: (1) The size of molecules representing “state” remains unchanged, no matter how long the computation process. (2) The reactions proceed in a cyclic way that facilitates automated implementation. (3) The manner of designing molecules representing states and transitions used in the section can be standardized in the sense that once an input alphabet and the number of states are given, a library of molecules that represent all possible transitions with inputs could be created, independent of machine specificity.

Finally, it would be interesting and instructive to compare this work with an implementation method proposed in Benenson et al. (2001) where a different type of restriction enzyme and more sophisticated encoding techniques are used; this will be described later.

2.2.2 Length-Based Approach

A new molecular implementation method based on *length-only encoding* is proposed, which leads to a very simple-molecular implementation technique to solve graph problems. Here, for example, an effective molecular implementation method for a nondeterministic finite automaton based on *one pot self-assembly* computing is demonstrated (Yokomori et al. 2002).

In the following, the focus is on ϵ -free regular languages, that is, regular languages not containing the empty string ϵ . Therefore, in any NFA M considered here, the initial state p_0 is not a final state in F of M . The following result is easily shown, but is of crucial importance to attain our objective. Given any DFA, there effectively exists an equivalent NFA M such that (1) M has the unique final state q_f , and (2) there is neither a transition into the initial state q_0 nor a transition from the final state q_f .

Thus, in what follows, one may assume that a finite automaton M is an NFA satisfying the properties (1) and (2) above.

Now, a simple molecular implementation for NFA is presented, which one may call *NFA Pot* and its computing schema is illustrated in [Fig. 6](#). NFA Pot has the following advantages:

1. In order to encode with DNAs each state transition of M , no essential design technique of encoding is required. Only the *length* of each DNA sequence involved is important.
2. Self-assembly due to hybridization of complementary DNAs is the only mechanism for carrying out the computation process.
3. An input w is accepted by M iff a completely hybridized double strand [$c(w)/\overline{c(w)}$] is detected from the pot, where $c(w)$ is a DNA sequence representing w and the “overline” version denotes its complementary sequence.

Without loss of generality, one may assume that the state set Q of M is represented as $\{0, 1, 2, \dots, m\}$ (for some $m \geq 1$) and in particular, “0” and “ m ” denote the initial and the final states, respectively.

Using an example, it is shown, how one can implement an NFA Pot for M given an NFA M .

Let one consider an NFA M (in fact, M is a DFA) in [Fig. 7](#), where an input string $w = abba$ is a string to be accepted by $M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$, where $\delta(0, a) = 1$,

Fig. 6

Finite automaton pot.

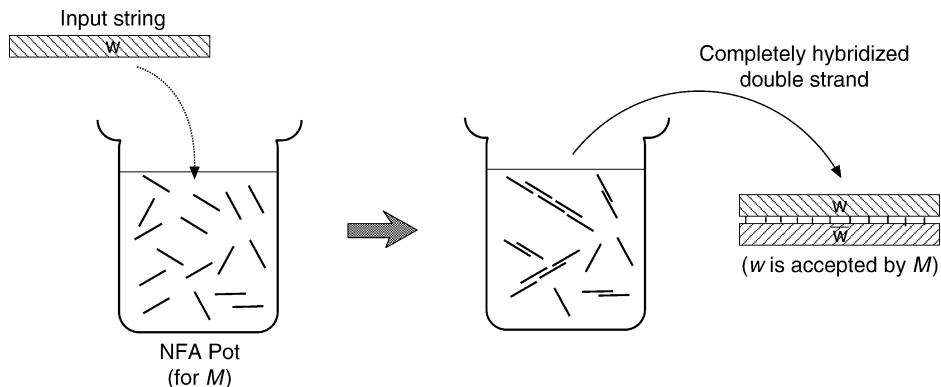
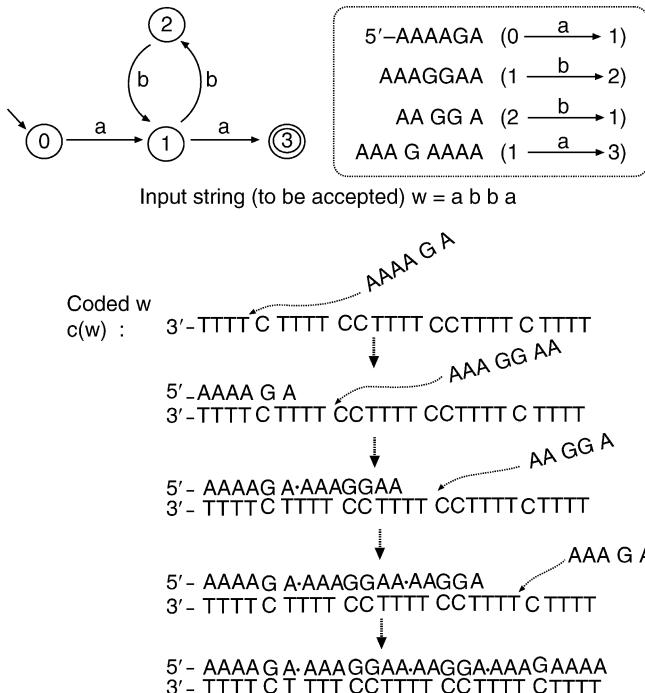


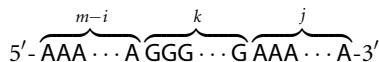
Fig. 7

FA and its DNA implementation.

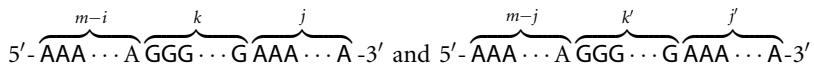


$\delta(1, b) = 2$, $\delta(2, b) = 1$, $\delta(1, a) = 3$. (Note that M satisfies properties (1) and (2) in the assumption previously mentioned and that the state 3 is the unique final state.)

In order to encode each symbol a_k from $\Sigma = \{a_1, \dots, a_n\}$, each is associated with an oligo $\text{GGG}\dots\text{G}$ of length k . Furthermore, each transition $\delta(i, a_k) = j$ is encoded as follows:



The idea is the following. Two consecutive valid transitions $\delta(i, a_k) = j$ and $\delta(j, a_{k'}) = j'$ are implemented by concatenating two corresponding encoded molecules, that is,



together make



Thus, an oligo $\overbrace{\text{AAA} \cdots \text{A}}^m$ plays the role of a “joint” between two transitions and it guarantees that the two are valid in M .

In order to get the result of the recognition task of an input w , one application of the “Detect” operation is applied to the pot, that is, the input w is accepted by M iff the operation

detects a completely hybridized double strand $[c(w)/\overline{c(w)}]$, where $c(w)$ is a DNA sequence encoding w and the “overline” version denotes its complementary sequence. Specifically, for an input string $w = a_{i_1} \dots a_{i_n}$, $c(w)$ is encoded as:

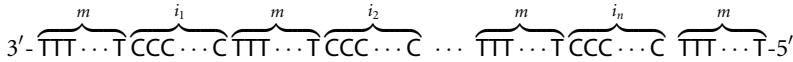


Figure 7 also illustrates the self-assembly process of computing $w = abba$ in the NFA Pot for M (although M is actually a DFA).

A method for molecular implementation is outlined to solve the directed Hamiltonian path problem (DHPP) which seems simpler than any other ever proposed. Since this implementation is also based on the “self-assembly in a one-pot computation” paradigm, one may call it *DHPP Pot*.

The DHPP Pot introduced has several advantages that are common with and similar to the implementation of NFA Pot. That is,

1. In order to encode each directed edge of a problem instance of DHPP, no essential design technique of encoding is required. Only the *length* of each DNA sequence involved is important.
2. Self-assembly due to hybridization of complementary DNAs is the only mechanism of carrying out the computation process.
3. An instance of DHPP has a solution path, p , iff a completely hybridized double strand $[c(p)/\overline{c(p)}]$ of a certain fixed length is detected from the pot, where $c(p)$ is a DNA sequence representing p and the “overline” version denotes its complementary sequence.

As seen above, the implementation methods discussed have a common distinguished feature in molecular encoding, that is, *no essential design technique is necessary* in that only the length of the DNA sequences involved in the encoding is critical. It should be remarked that this aspect of this noncoding implementation could apply to a large class of problems that are formulated in terms of graph structures. (It would be easy to list a number of problems associated with graphs.)

One can also propose a general schema for solving graph problems in the framework of the self-assembly molecular computing paradigm where the *length-only-encoding* technique may be readily applied. In fact, some of the graph-related optimization problems such as the traveling salesman problems can be handled in the schema mentioned above.

In the framework of self-assembly molecular computing, it is of great use to explore the computing capability of the self-assembly schema with “double occurrence checking (doc)” as a screening mechanism, because many of the important NP-complete problems can be formulated into this computing schema. It is shown by constructing DHPP Pot that at the sacrifice of the *nonlinear (exponential)* length increase of strands, the “doc” function can be replaced with the length-only encoding technique. It is, however, strongly encouraged to develop more practical methods for dealing with the doc function.

Finally, the biological implementations based on the method of length-only encoding (discussed here) can be found in Kuramochi and Sakakibara (2005) and in the corresponding chapter [Bacterial Computing and Molecular Communication](#).

2.3 Bibliographic Notes

Various types of abstract models for DNA computing with Turing computability have been presented. These models are classified into various principles of computation, such as a

specific grammar called “Geffert normal form” (Geffert 1991), the equality machine (or equality checking) in Engelfriet and Rozenberg (1980), Post systems (Post 1943), and logical formulations (Tarnlund 1977). In relation to the approach by equality checking discussed in [Sect. 2.1.2](#), one can find a series of intensive works on Watson–Crick automata (Freund et al. 1999b) that share the computing principle with equality machines.

Many attempts to discover grammatical DNA computation models with Turing computability have already been made (Freund et al. 1999a; Păun 1996a, b; etc.), based on the *splicing* operation (originally proposed by Head (1987) or on H-systems (refer to the chapter [DNA Computing by Splicing and by Insertion–Deletion](#) for details). In nature, there exist DNA molecules, such as bacterial DNA and plasmids, that form circular structures. This inspires a new type of splicing models of computing where “circular splicing” produces a mixture of linear and circular molecules. One can find several works on such an extended H-system, called circular splicing system, in Siromoney et al. (1992), Pixton (1995), and Yokomori et al. (1997), in which the type 2 rule of the GPS system can be simulated by using circular molecules within the framework of splicing computing.

A further extension of splicing systems is studied in which tree structural molecules are considered to be spliced (Sakakibara and Ferretti 1999). The language family generated by tree splicing systems is shown to be the family of context-free languages (refer to [Sect. 3.2](#)). Recently, CsuhaJ-Varju and Verlan proposed a new model of test tube systems with Turing computability, based on splicing, where the length-only separation of strings functions filtering in communication (CsuhaJ-Varju and Verlan 2008). More models of extended splicing systems can be found in Păun (1998), while lab experimental work on splicing systems is reported in Laun and Reddy (1997).

Kari et al. (2005) propose and investigate another type of grammatical model for DNA computation based on insertion/deletion schemes that are inspired by pioneering works on contextual grammars due to S. Marcus (1969). The reference book by Păun et al. (1998) is one of the best sources for details of many topics mentioned above (including splicing systems, Watson–Crick automata, insertion/deletion systems, and others).

Reif (1995) proposes a parallel molecular computation model called parallel associative memory (PAM), which can *theoretically* and efficiently simulate a nondeterministic Turing machine and a parallel random access machine (PRAM), where a PA-Match operation, similar to the joint operation in relational database, is employed in an effective manner. Due to the complexity of the molecular biological transformation procedure used to realize a PAM program, it seems more study is needed to fill in the technological gaps between the two computation procedures at different levels.

Roß and Wagner (1996) propose a formal framework called “DNA-Pascal” to study the computational powers of various abstract models for DNA computing and derive several interesting relationships between proposed models and known computational complexity classes within a polynomial time DNA computation. A recently published book (Ignatov et al. 2008) on DNA computing models contains broader topics in the area and partly overlaps with this chapter.

3 Computing Models with Structured Molecules

The molecules used in the first experiment in DNA computing reported in Adleman (1994) were linear non-branching structures of DNA. One of the important developments in DNA

computing is based on the use of more involved molecules, which perform specific computations essentially through the process of self-assembly.

This section, introduces some of the computing models based on such structural DNA molecules.

3.1 Computing by Hairpin Structures

Molecules with *hairpin* structure(s) form a natural extension of linear non-branched molecules, because a hairpin results from a linear molecule that folds onto itself. This structural feature of molecules is recognized as a convenient and useful tool for many applications in DNA computing based on self-assembly.

This naturally brings about the following question: what sort of problems can be computed by a schema using hairpin structures? Investigating this question leads one to consider sets of string molecules which contain complementary sequences of potential hairpin formations, that is, the set of strings containing a pair of complementary subsequences.

In Păun et al. (2001) several types of such “hairpin languages” are considered and their language theoretic complexity (using the Chomsky hierarchy) is investigated.

When forming a hairpin molecule, it is necessary that the annealed sequence is “long enough” in order to ensure the stability of the construct, so it is always imposed that this sequence is, at least, of length, k , for a given k .

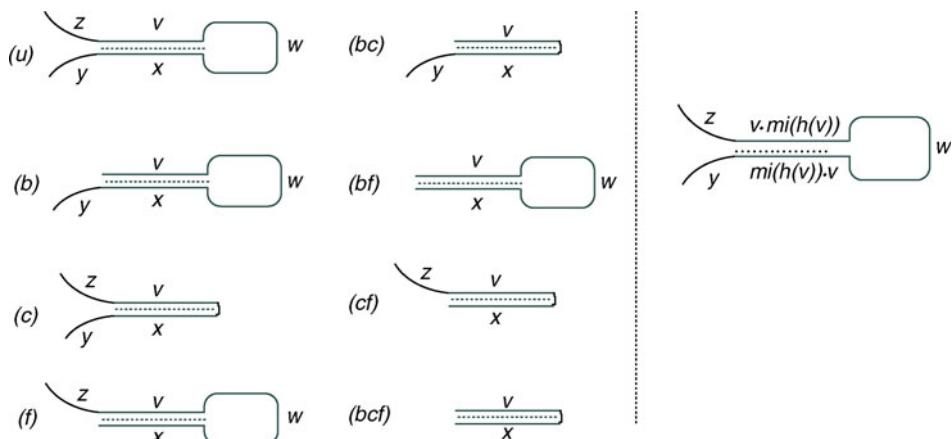
Let, then, k be a positive integer. The *unrestricted hairpin language* (of degree k) is defined by

$$uH_k = \{zvwxy \mid z, v, w, x, y \in V^*, x = mi(h(v)), \text{ and } |x| \geq k\}$$

By imposing restrictions on the location where the annealing may occur, one gets various sublanguages of uH_k (Fig. 8 illustrates the hairpin constructions corresponding to these languages):

Fig. 8

(Left) Hairpin constructions corresponding to eight languages ; (right) a hairpin formation to be removed in solving the directed Hamiltonian path problem (DHPP).



$$\begin{aligned}
 bH_k &= \{vwxy \mid v, w, x, y \in V^*, \text{ cond}(x, v; k)\}, & bfH_k &= \{vwx \mid v, w, x \in V^*, \text{ cond}(x, v; k)\}, \\
 cH_k &= \{zvxy \mid z, v, x, y \in V^*, \text{ cond}(x, v; k)\}, & cfH_k &= \{zvx \mid z, v, x \in V^*, \text{ cond}(x, v; k)\}, \\
 fH_k &= \{zvwx \mid z, v, w, x \in V^*, \text{ cond}(x, v; k)\}, & bcfH_k &= \{vx \mid v, x \in V^*, \text{ cond}(x, v; k)\}, \\
 bcH_k &= \{vxy \mid v, x, y \in V^*, \text{ cond}(x, v; k)\}, & \text{where } \text{cond}(x, v; k) \text{ means "}x = mi(h(v)) \wedge |x| \geq k\text{"}
 \end{aligned}$$

The computation schemata mentioned above is in fact embodied in the literature, for example, one subtracts a language uH_k , for some k , from a (regular) language in [Sect. 4.5](#), while one intersects a given (linear) language with bcH_1 in GCS in [Sect. 2.2.2](#). Therefore, let one consider languages of the form αH_k and $V^* - \alpha H_k$, for $\alpha \in \{u, b, c, f, bc, bf, cf, bcf\}$. As for the languages of the form αH_k , the following are obtained.

Theorem 7

- (1) All languages αH_k , for $\alpha \in \{u, b, f, c, bf\}$ and $k \geq 1$, are regular.
- (2) The languages $bcH_k, cfH_k, bcfH_k$, $k \geq 1$, are linear, but not regular

In order to see that those in (2) are not regular, one has only to consider the alphabet $V = \{a, b\}$, with a, b complementary to each other, and to intersect these languages with the regular language a^+b^+ , which leads to obtaining the following three languages: $\{a^n b^m \mid m \geq n \geq k\}$, $\{a^n b^m \mid n \geq m \geq k\}$, and $\{a^n b^n \mid n \geq k\}$ (in this order), and none of these languages is regular.

The next consequence is of interest from the viewpoint of DNA computing by hairpin removal.

Theorem 8 Let \mathcal{F} be a family of languages which is closed under intersection with regular languages. If $L \in \mathcal{F}$, then $L - \alpha H_k \in \mathcal{F}$, for all $\alpha \in \{u, b, c, f, bf\}$ and all $k \geq 1$.

3.1.1 Complements of Hairpin Languages

In turn, one can consider languages bcH_k, cfH_k and $bcfH_k$. By (2) of [Theorem 7](#), one knows already that the complements of bcH_k, cfH_k and $bcfH_k$ are not necessarily regular. However, the languages $bcfH_k, k \geq 1$, are not “very non-regular” in the sense that for each $k \geq 1$, one has $V^* - bcfH_k$ as a linear language.

For more details, the next results are obtained:

Theorem 9

- (1) If L is a regular language, then $L - bcfH_k$, $k \geq 1$, is a linear language which does not have to be regular.
- (2) If L is a context-free language, then $L - bcfH_k$, $k \geq 1$, is a context-sensitive language which does not have to be context-free.

On the other hand, for languages bcH_k and cfH_k the situation is more involved.

Theorem 10

- (1) The languages $V^* - bcH_k$, $V^* - cfH_k$, and $k \geq 1$ are not context-free.
- (2) If L is a regular language, then the languages $L - bcH_k$ and $L - cfH_k$, $k \geq 1$ are context-sensitive, but not necessarily context-free.

It would be interesting to investigate the place of the languages $L - bcH_k$ and $L - cfH_k$ with respect to families of languages intermediate between the context-free and the context-sensitive families (such as matrix or other regulated rewriting families (Dassow and Păun 1989)).

As an example of the hairpin computation schema discussed here, one can find a solution for the directed Hamiltonian path problem as well.

3.1.2 A New Hairpin-Based Operation: Hairpin Completion

Another interesting aspect of the computability involved in hairpin structures is brought about when a biological phenomenon known as *lengthening DNA by polymerases* is introduced into the DNA computation process. More specifically, consider the following hypothetical biological situation: one is given one single-stranded DNA molecule, z , such that either a prefix or a suffix of z is Watson–Crick complementary to a subword of z . Then the prefix or suffix of z and the corresponding subword of z get annealed by complementary base pairing and then z is lengthened by DNA polymerases up to a complete hairpin structure. The mathematical expression of this hypothetical situation leads to a new formal operation in DNA computing theory; starting from a single word, one can generate a set of words by this formal operation called *hairpin completion*. One can also consider the inverse operation of hairpin completion, namely hairpin reduction. That is, the *hairpin reduction* of a word x consists of all words y such that x can be obtained from y by hairpin completion. These operations are schematically illustrated in Fig. 9.

However, attention is focussed *only* on hairpin completion here.

Recall that for any $\alpha \in V^*$, $\overline{\alpha^R}$ denotes the complement of the mirror image of α . For any $w \in V^+$ and $k \geq 1$, the k -*hairpin completion* of w , denoted by $(w \rightarrow_k)$, is defined as follows.

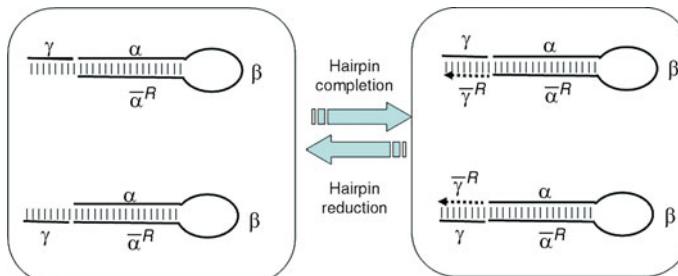
$$\begin{aligned}(w \rightarrow_k) &= \{\overline{\gamma^R}w \mid w = \alpha\beta\overline{\alpha^R}\gamma, |\alpha| = k, \alpha, \beta, \gamma \in V^+\} \\ (w \rightarrow_k) &= \{w\overline{\gamma^R} \mid w = \gamma\alpha\beta\overline{\alpha^R}, |\alpha| = k, \alpha, \beta, \gamma \in V^+\} \\ (w \rightarrow_k) &= (w \rightarrow_k) \cup (w \rightarrow_k)\end{aligned}$$

Then, the hairpin completion is naturally extended to languages by

$$(L \rightarrow_k) = \bigcup_{w \in L} (w \rightarrow_k)$$

Fig. 9

Hairpin completion and hairpin reduction: For each $k \geq 1$, hairpin completion (hairpin reduction) requires the condition of $|\alpha| = k$.



Note that all these phenomena here are considered in an idealized way. For instance, polymerase is allowed to extend in either end (3' or 5') despite that, due to the greater stability of 3' when attaching new nucleotides, DNA polymerase can act continuously only in the 5' → 3' direction. However, polymerase can also act in the opposite direction, but in short "spurts" (Okazaki fragments). Moreover, in order to have a "stable" hairpin structure, the subword α should be sufficiently long.

For a class of languages \mathcal{F} and an integer $k \geq 1$ one denotes by

$$\text{WCOD}(\mathcal{F} \rightarrow_k) = \{h(L \rightarrow_k) \mid L \in \mathcal{F} \text{ and } h \text{ is a weak coding}\}$$

the weak-coding image of the class of the hairpin completion of languages in \mathcal{F} .

The following characterizations concerning the class of linear context-free languages are interesting:

Theorem 11 (Cheptea et al. 2006) (i) For any integer $k \geq 1$, $\text{LIN} = \text{WCOD}(\text{REG} \rightarrow_k)$ holds. (Manea et al. 2009a) (ii) For any integer $k \geq 1$, $\text{WCOD}(\text{LIN} \rightarrow_k)$ is a family of mildly context-sensitive languages.

It should be remarked that this is the characterization result of a class of mildly context-sensitive languages (Joshi and Schabes 1997) that is based on a bio-inspired operation (hairpin completion) and weak codings starting from a simple class of linear languages in the Chomsky hierarchy. It remains open whether a similar result holds for the class of languages by hairpin reduction.

In contrast to its simplicity, hairpin computation as the underlying schema for DNA computing seems to be quite promising.

3.2 Computing by Tree Splicing

In their study on splicing systems, Sakakibara and Ferretti extended the notion of splicing on linear strings, proposed by Tom Head, to splicing on tree-like structures which resemble some natural molecular structures such as RNA structures at the secondary structure level (Sakakibara and Ferretti 1999).

Consider a finite alphabet V and trees over V . By V^T one denotes the set of all trees over V . A *splicing rule on tree structures over V* is given by a pair of tree substructures as in Fig. 10, where u_i 's ($i = 1, \dots, 4$) are subtrees over V , λ s are null trees, and $\#$ is a special symbol not

Fig. 10

Splicing rule on tree structures.

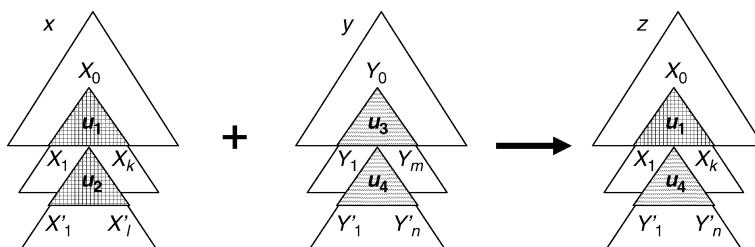
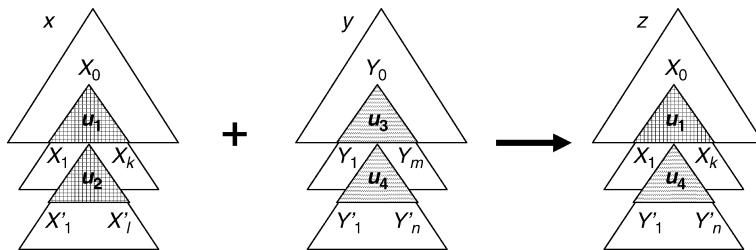


Fig. 11

Splicing on tree structures: Let x, y be trees which contain $u_1(\lambda, \dots, u_2(\lambda, \dots, \lambda), \dots, \lambda)$ and $u_3(\lambda, \dots, u_4(\lambda, \dots, \lambda), \dots, \lambda)$ as a part of the tree, respectively. Then, one can apply the rule in [Fig. 10](#) to x and y to generate a new tree z , which is obtained by replacing the subtree of x under u_2 with that of y under u_4 .



contained in the alphabet V indicating a single node in a tree. The symbol $\$$ is a separator located between the two tree substructures.

A splicing rule r takes two trees $x, y \in V^T$ as inputs and generates a tree $z \in V^T$, written $(x, y) \vdash_r z$. [Figure 11](#) illustrates an application of a splicing rule of [Fig. 10](#).

A *splicing scheme on tree structures* is a pair $\gamma = (V, R)$, where V is an alphabet and R is a set of splicing rules on tree structures over V . A pair $\mathcal{S} = (\gamma, L)$ for a splicing scheme γ and a tree language $L \subseteq V^T$ is called a *splicing scheme on tree structures* (or HT system). For an HT system $\mathcal{S} = (\gamma, L)$, the tree language generated by \mathcal{S} is defined as follows:

$$\gamma(L) = \{z \in V^T \mid (x, y) \vdash_r z \text{ for some } x, y \in L, r \in R\}$$

Iterated splicing is also defined as follows:

$$\begin{cases} \gamma^0(L) = L, \\ \gamma^{i+1}(L) = \gamma^i(L) \cup \gamma(\gamma^i(L)), \text{ and} \\ \gamma^*(L) = \bigcup_{i \geq 0} \gamma^i(L). \end{cases}$$

For a class of tree languages F_1 and a class of sets of splicing rules F_2 , the set of all tree languages $\gamma^*(L)$ is denoted by $HT(F_1 F_2)$, such that $\gamma = (V, R)$, $L \in F_1$, and $R \in F_2$.

Let FIN_T be the class of finite tree languages, and FIN_R be the class of finite sets of splicing rules. Then, $HT(FIN_T FIN_R)$ has a close relation to the class of context-free languages in the following sense.

Theorem 12 For any context-free grammar G , the set of derivation trees of G is in the class $HT(FIN_T, FIN_R)$.

3.3 Computing by PA Matching

Reif (1999) introduced an operation, called *parallel associative matching* (PA Matching), in order to provide a theoretical basis for parallel molecular computing. The operation PA Matching, denoted by \bowtie , is defined as follows.

Let Σ be a DNA alphabet $\Sigma = \{\sigma_0, \sigma_1, \dots, \sigma_{n-1}, \bar{\sigma}_0, \bar{\sigma}_1, \dots, \bar{\sigma}_{n-1}\}$ of $2n$ distinct symbols where for each i , the symbols $\sigma_i, \bar{\sigma}_i$ are called (Watson–Crick) *complements*. Note that the DNA alphabet is extended theoretically to have symbols more than four. Consider another finite alphabet A and an encoding function E from A^+ to Σ^+ . Thus, for a string $\alpha \in A^s$ of length s for some positive integer s , $E(\alpha)$ is an encoded DNA sequence of α . For a pair $\alpha, \beta \in A^s$, let $E(\alpha, \beta)$ denote an encoding of the ordered paired strings (α, β) . (For details of how to encode a pair of strings, refer to Reif (1999).)

For each $\alpha, \beta, \beta', \gamma \in A^s$, the *PA-Match* operation \bowtie is defined to be $E(\alpha, \beta) \bowtie E(\beta', \gamma) = E(\alpha, \gamma)$ when $\beta = \beta'$, and the operation \bowtie yields no value if $\beta \neq \beta'$.

Reif introduced a formal parallel computation model, called the PAM model, in which the following five operations are allowed to be executed in one step:

1. Merge: Given test tubes T_1, T_2 , produce the union $T_1 \cup T_2$.
2. Copy: Given a test tube T_1 , produce a test tube T_2 containing the same contents of T_1 .
3. Detect: Given a test tube T , say “yes” if T is not empty and say “no” otherwise.
4. Separation: Given a test tube T and a string $\alpha \in \Sigma^+$ produce a test tube T_1 consisting of all strings of T , which contain α as a substring, and also a test tube T_2 consisting of all strings of T , which do not contain α .
5. PA-Match: Given test tubes T_1, T_2 , apply the operation \bowtie to yield the test tube $T_1 \bowtie T_2$.

Reif showed a surprising simulation result of a nondeterministic Turing machine based on PAM model.

Theorem 13 *Nondeterministic Turing machine computation with space bound s and time bound $2^{O(s)}$ can be simulated by the PAM model using $O(s \log s)$ PAM operations that are not PA-Match and $O(s)$ PA-Match operations, where strings of length $O(s)$ over the alphabet size $O(s)$ (corresponding to DNA of length $O(s \log s)$ base-pairs) are used in the simulation.*

Motivated by Reif’s work, Kobayashi et al. studied the formal properties of the PA-Match operation, where the operation \bowtie was extended in order to deal with string pairs of different length (Kobayashi et al. 2001).

Informally speaking, the operation consists of cutting two strings in two segments such that the prefix of one of them matches the suffix of another, removing these two matching pieces, and pasting the remaining parts. Formally, given an alphabet V , a subset X of V^+ , and two strings $u, v \in V^+$, one defines

$$PAm_X(u, v) = \{wz \mid u = wx, v = xz, \text{ for } x \in X, \text{ and } w, z \in V^*\}$$

The operation is naturally extended to languages over V by

$$PAm_X(L_1, L_2) = \bigcup_{u \in L_1, v \in L_2} PAm_X(u, v)$$

When $L_1 = L_2 = L$, $PAm_X(L)$ is written instead of $PAm_X(L_1, L_2)$. Since one shall only deal either with finite sets X or with $X = V^+$, one uses the notation *fPAm* for finite PA-matching and the notation *PAm* for arbitrary PA-matching PAm_{V^+} .

Theorem 14

- (1) *The families REG and RE are closed under PAm.*
- (2) *The families LIN, CF, and CS are not closed under PAm.*

They further investigate the iterated version of the PA-match operation which is defined as follows. For a language $L \subseteq V^*$ and a finite set $X \subseteq V^+$, one defines:

$$\begin{cases} PAm_X^0(L) = L, \\ PAm_X^{k+1}(L) = PAm_X^k(L) \cup PAm_X(PAm_X^k(L)), \quad k \geq 0, \\ PAm_X^*(L) = \bigcup_{k \geq 0} PAm_X^k(L). \end{cases}$$

When X is finite, the iterated PA-matching operation is denoted by $fPAm^*$; in the case $X = V^*$, the corresponding operation is denoted by PAm^* . Then, one has:

Theorem 15

- (1) *The families REG and RE are closed under both $fPAm^*$ and PAm^* .*
- (2) *The family LIN is not closed under $fPAm^*$ and PAm^* .*
- (3) *The family CF is closed under $fPAm^*$ but it is not closed under PAm^* .*
- (4) *The family CS is closed neither under $fPAm^*$ nor under PAm^* .*

As shown above, if one extends the operation \bowtie to deal with string pairs of different length, then the computational power of the PA-Matching operation is rather weak, because one cannot escape from regularity when one starts from regular languages. In this sense, the length condition of \bowtie is important in Reif's results.

3.4 Bibliographic Notes

Needless to say, it is of great importance in molecular computing theory to investigate the computational properties of structural molecules. In fact, there is a variety of structural complexity of computing molecules which range from 1-D linear to 3-D structures such as a duplex with hairpin molecules (Eng 1997), molecular graphs (Jonoska et al. 1998), quite sophisticated structures like double crossover molecules (Winfree et al. 1996) and DNA tiles (LaBean et al. 2000; Lagoudakis and LaBean 2000). Among others, Jonoska et al. used the 3-D graph structure of molecules to solve two NP-complete problems (SAT and 3-vertex-colorability problems). Further, Winfree et al. investigated the computational capability of various types of linear tile assembly models (Winfree et al. 2000).

Kari et al. studied hairpin sets and characterized the complexity and decidability properties of those sets (Kari et al. 2005b). They also extended their study to *scattered* hairpin structures in which the stem allows any number of unbounded regions (Kari et al. 2005a). Furthermore, the notion of DNA trajectory is proposed to describe the bonding between two separate DNA strands, and is used to study *bond-free* properties of DNA code design (Kari et al. 2005c), while the notion is also adopted to model the scattered hairpin structures (Doramatzki 2006).

On the other hand, hairpin molecules find many applications in DNA computing. For example, the use of hairpin formation of single-stranded DNA molecules in solving 3-SAT problems is demonstrated in Sakamoto et al. (2000) (for details, refer to [Sect. 4.5](#)). Also, as previously seen in [Sect. 2.2.2](#), a molecular computing schema GCS uses the power of hairpin formation in finalizing its computation process (Yokomori 2000), and so forth. These two computation models share a common feature, which follows a general computational schema called “generation and test” strategy and is described as follows: (i) first, prepare the initial random pool of all possible structured molecules, (ii) then extract only

target molecules with (or without) hairpin formation from the pool. In fact, this paradigm is formalized and intensively studied under the formal language theoretic terms of “computing by carving” in Păun (1999).

4 Molecular Implementations and Applications

In this section, several wet lab experimental works of molecular computing machineries based on structured DNA molecules are introduced. Those molecular devices involve a variety of ideas in controlling molecular behaviors, such as utilizing specific restriction enzymes, deoxyribozymes, sophisticatedly designed DNA tiles, and so forth.

4.1 DNA Finite Automata

4.1.1 Enzyme-Based DNA Automata

An interesting wet lab experimental work has been reported from Israel’s group led by Shapiro, where an autonomous computing machine that comprises DNAs with DNA manipulating enzymes and behaves as a finite automaton was designed and implemented (Benenson et al. 2001). The hardware of the automaton consists of a restriction nuclease and ligase, while the software and input are encoded by double-stranded DNAs.

Encoding DFA onto DNA

Using an example DFA M given in Fig. 12a, it will be demonstrated how to encode state transition rules of M onto DNA molecules.

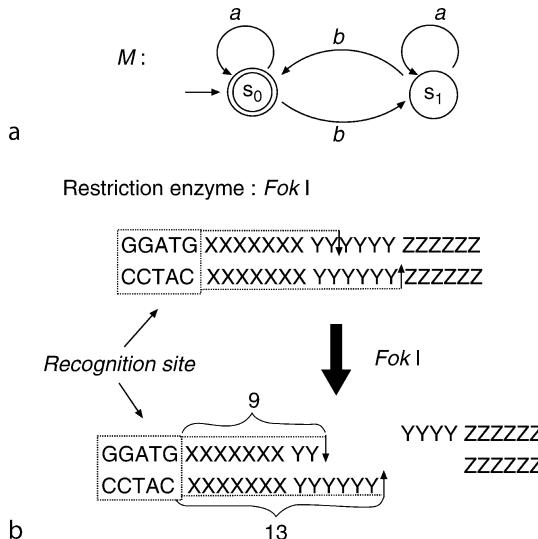
Before going into the details, however, it is important to observe how a restriction enzyme *FokI* works on double-stranded molecules to recognize the site and to cut them into two portions, which is illustrated in Fig. 12b. A restriction enzyme *FokI* has its recognition site GGATG and cuts out a double-stranded DNA molecule as shown in Fig. 12b. That is, splicing by *FokI* takes place nine DNAs to the right of the recognition site on the upper strand and 13 DNAs to the right of the recognition site on the lower strand.

A transition molecule detects the current state and symbol and determines the next state. It consists of (1) a pair (state, symbol) detector, (2) a *FokI* recognition site, and (3) a spacer; all these three together determine the location of the *FokI* cleavage site inside the next symbol encoding, which in turn defines the next state: For example, 3-base-pair spacers maintain the current state, and 5-base-pair spacers transfer s_0 to s_1 , etc. Also, a special molecule for detecting the “acceptance” of an input is prepared as shown in Fig. 13c. (Note that the details of the actual design of molecules for implementing the automaton M is not given here; rather only the principle of encoding the molecules and computation will be outlined.)

Taking this fact into account, encoding *transition molecules* for the transition rules of DFA is designed as follows. First, there are two points to be mentioned in designing transition molecules: One is that neither a state nor symbol alone is encoded but a pair (state, symbol) is taken into consideration to be encoded, and the other is that a sticky end of 6-base length encoding (state, symbol) is sophisticatedly designed such that the length 4 prefix of the sticky end represents (s_1, x) , while the length 4 suffix of the same sticky end represents (s_0, x) , where x

Fig. 12

(a) DFA M with states s_0, s_1 and input symbols a, b . The initial state is s_0 , which is the unique final state as well. M accepts the set of strings containing an even number of b 's. (b) How a restriction enzyme *Fok I* works on molecules.



is in $\{a, b, t\}$, and t is the special symbol, not in the input alphabet, for the endmarker (see [Fig. 13a](#)). Reflecting this design philosophy, the structures of the transition molecules are created as shown in [Fig. 13b](#), where molecules for two transition rules: $s_0 \xrightarrow{a} s_0$ and $s_0 \xrightarrow{b} s_1$ are selectively described. The transition molecule $m_{s_0,a}$ for $s_0 \xrightarrow{a} s_0$ consists of the recognition site of *Fok I*, the center base-pairs of length 3 and the sticky end CCGA, because its complementary sequence GGCT is designed to represent the pair (s_0, a) as seen from the table. A transition $s_0 \xrightarrow{b} s_1$ is encoded by a similar principle onto the transition molecule $m_{s_0,b}$ shown in [Fig. 13b](#), where a noteworthy difference from $m_{s_0,a}$ is that it has the center base-pairs of length 5. In this principle of designing transition molecules, a transition molecule has the center base-pairs of length 5 iff the transition changes its state (from s_0 to s_1 or vice versa). Thus the distinct length of the center base-pairs plays a crucial role.

Computing Process

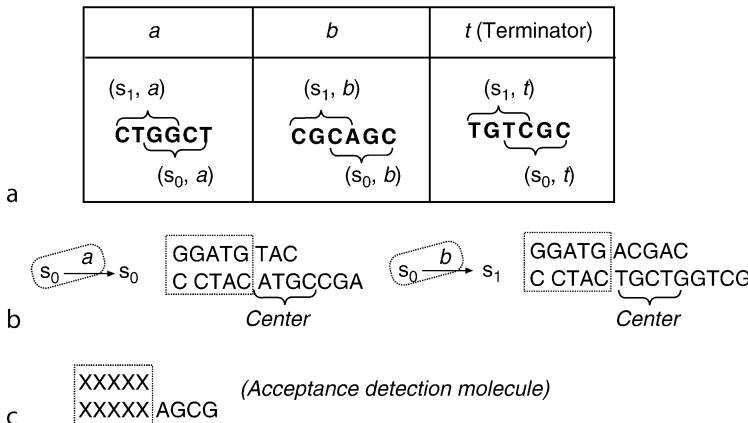
The molecular automaton processes the input as shown in [Fig. 14](#), where input w is assumed to be of the form $ab\dots(t)$, where t is the endmarker. First, the input w is encoded into the form of the molecule shown in [Fig. 14a](#), where seven X base-pairs represent a “spacer” consisting of arbitrary bases.

The input molecule is cleaved by *Fok I*, exposing a 4-base sticky end that encodes the pair of the initial state and the first input symbol, that is, (s_0, a) in the running example. The computation process is performed in a cyclic manner: In each cycle, the sticky end of an applicable transition molecule ligates to the sticky end of the (remaining) input molecule (obtained immediately after cleaved by the last transition), detecting the current state and the input symbol ([Fig. 14b](#)). (Note that it is assumed at the beginning of computation that all

Fig. 13

States and symbols encoding.

Coding a pair (state, symbol)



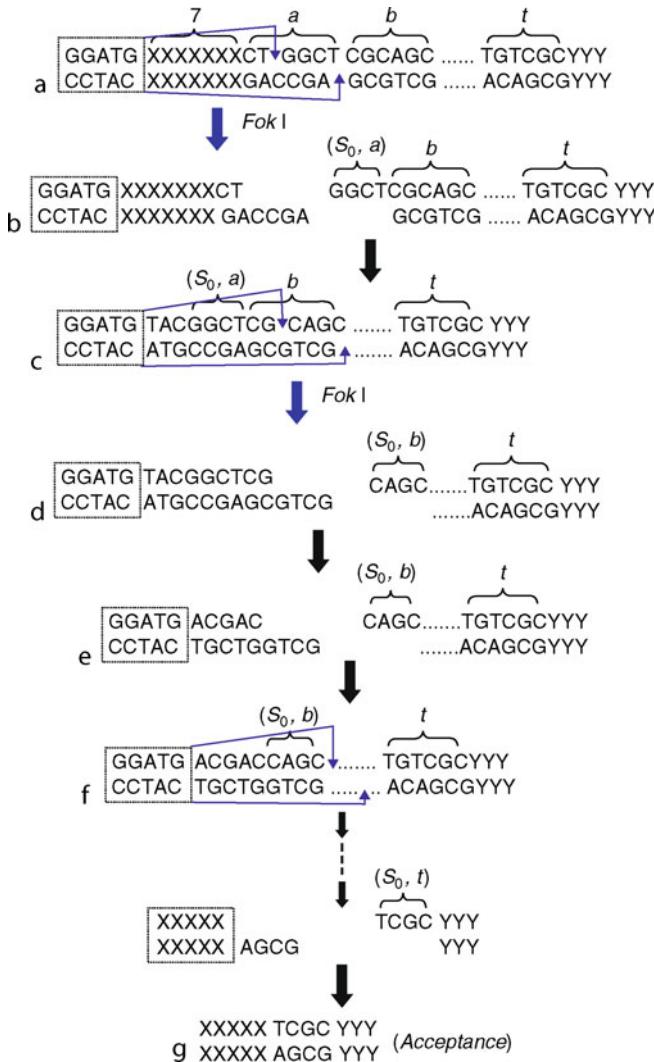
of the possible transition molecules are contained in a solution pot together with the initial molecule and the acceptance detect molecules.)

At this moment, since the transition molecule for (s_0, a) , shown in [Fig. 13b](#), can get access to the remaining input molecule on the right-hand side in [Fig. 14b](#), it produces the hybridized molecule shown in [Fig. 14c](#). Therefore, the resulting molecule is again cleaved by *FokI*, exposing a new 4-base sticky end ([Fig. 14d](#)). The design of the transition molecule ensures that the 6-base-pair long encodings of the input symbols *a* and *b* are cleaved by *FokI* at only two different “frames” as seen above, the prefix frame of 4-base-pair length for s_1 , and the suffix frame of 4-base-pair length for s_0 . Thus, the next restriction site and the next state are exactly determined by the current state and the length of the center base-pairs in an applicable transition molecule. In the example, since the sticky end of the remaining input molecule can match that of the transition molecule $m_{s_0, b}$ for (s_0, b) ([Fig. 14e](#)) and the molecule $m_{s_0, b}$ exists in the solution pot, so that these two can hybridize to form the double-stranded molecule shown in [Fig. 14f](#).

The computation process proceeds until no transition molecule matches the exposed sticky end of the rest of the input molecule, or until the special terminator symbol is cleaved, forming an output molecule having the sticky end encoding the final state (see [Fig. 14g](#)). The latter means that the input string *w* is accepted by the automaton *M*.

In summary, the implemented automaton consists of a restriction nuclease and ligase for its hardware, and the software and input are encoded by double-stranded DNA, and programming amounts to choosing the appropriate software molecules. Upon mixing solutions containing these components, the automaton processes the input molecule by a cascade cycle of restriction, hybridization, and ligation, producing a detectable output molecule that encodes the final state of the automaton.

In the implementation, a total amount of 10^{12} automata sharing the same software (“transition molecules”) run independently and parallel on inputs in $120 \mu\ell$ solution at room temperature at a combined rate of 10^9 transitions per second with a transition fidelity greater than 99.8%, consuming less than 10^{10} W. (For detailed reports of the experimental results, refer to Benenson et al. (2001).)

Fig. 14Computing process of input $w=ab(t)$.

4.1.2 Application to Drug Delivery

As an application of the molecular finite automata, Benenson et al. (2004) proposed an idea to use their molecular automata for carrying out a diagnosis *in vivo*, and demonstrated an *in vitro* experiment, which shows the feasibility of logical control of gene expression for drug delivery. An excellent review article on the work by Benenson et al. is given by Condon (2004).

In order to carry out a medical diagnosis, it is necessary to achieve *in vivo* checking of a medical statement such as “if a certain series of diagnostic conditions are true, then the drug

is administered.” Such an *if–then* mechanism is critically essential in computing models and in diagnostic process as well. It is assumed that each condition is described in the form: “a certain type of gene expression is regulated by either high or low level of concentrations of particular mRNA (called *indicator molecule*) for the gene.”

❷ [Figure 15](#) outlines the idea used in the diagnostic molecular automaton (*DMA*) experiments by Benenson et al. Suppose that a certain disease involves four conditions of gene expression to be checked before administering a drug for the disease, and each gene-*i* (for $i = 1, \dots, 4$) as its label is encoded into a structured molecule (called the *computation molecule*) with a hairpin in which the drug is enclosed (❷ [Fig. 15a](#)). Note that a diagnostic rule is of the form: if the conditions for gene-1 through gene-4 are all true, then the drug is released. The *DMA* (based on the automaton described in the previous section) can carry out the checking of each diagnostic condition in one transition step, and is constructed in such a way that it has two states “Yes” and “No” and three transitions: Yes → Yes if indicator molecules are present, Yes → No if indicator molecules are absent, and No → No in any other situation. Note that once the transition Yes → No occurs, the state of *DMA* remains unchanged, that is, it keeps staying in No no matter what the situation of the indicator molecule is. Therefore, starting with the initial state Yes, the *DMA* behaves as follows: after each transition, the state is either Yes indicating that the corresponding conditions tested *so far* are all true, or No otherwise (i.e., at least one of the conditions tested is false). Thus, if the *DMA* can successfully make four transitions, then the drug is released.

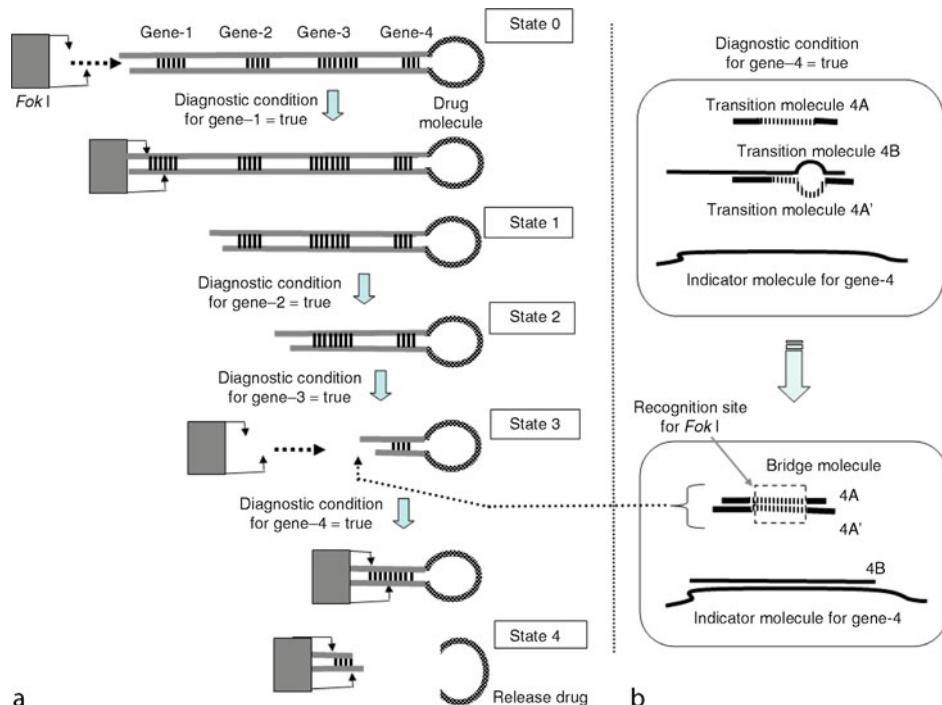
In ❷ [Fig. 15](#), at the initial state (State 0), a hairpin structural molecule (computation molecule) is designed so that the diagnostic condition is the conjunction of four logical statements on gene expression, each of which is true for gene-*i* if the level of a particular indicator molecule for gene-*i* is either high or low (depending upon the requirement on each gene expression), for each $i = 1, \dots, 4$. In order to make a transition from State $(i-1)$ to State i , the *DMA* requires a specific molecule, say a *bridge molecule*, which is supposed to be created from a team of molecules designed for each gene-*i* ($i = 1, \dots, 4$) under a certain condition.

Suppose, for example, the *DMA* is in State 3 and is going to effect a transition to State 4, which must diagnose a high level of indicator molecule for gene-4. The team of molecules to do the task comprises *transition molecules* 4A, 4B, and 4A', where 4B and 4A' initially form a binding structure, 4A and 4A' contain a sequence of oligonucleotides and its complementary one, respectively, and annealing the two can provide a bridge molecule containing the recognition site for the restriction enzyme *FokI* (❷ [Fig. 15b](#)). In addition to the team molecules, when an indicator molecule for gene-4 is present in the solution, the scenario for the *DMA* to complete a successful transition to State 4 goes as follows. Transition molecule 4B is designed so that it prefers to pair with the indicator molecule for gene-4 rather than transition molecule 4A' (current partner), because 4B shares a longer complementary region with indicator molecule 4 than 4A'. Therefore, after leaving transition molecule 4A', transition molecule 4B settles in to pair with the indicator molecule for gene-4. On the other hand, transition molecules 4A and 4A' get together forming a structure due to their mutual affinity. As a result, this structured molecule can participate as a bridge molecule in the successful transition to gene-4 which is carried out by *FokI*.

It should be noted that the *DMA* is stochastic in overall behavior, because the transition is strictly dependent upon the level of concentration of indicator molecules in each diagnostic condition, and there remain some problems to be solved before the demonstrated mechanism will work *in vivo* as well. However, without question, this is the first achievement to prove the great potential of developing medical diagnostic molecular devices for antisense therapies.

Fig. 15

DNA automaton for logical control of drug delivery: (a) At the initial state (State 0), a hairpin molecule (computation molecule) is designed so that a hairpin may maintain the drug molecule that is guarded by four specific sequences gene- i ($i = 1, \dots, 4$) each of which corresponds to one of four conjunctive logical statements in a diagnostic rule for the disease. Each State ($i - 1$) is associated with a team of molecules (given in b of this figure for $i = 4$), and in the presence of the indicator molecule for gene- i , the team can produce a specific molecule (bridge molecule) which is necessary for cleaving the guarded gene- i , resulting in a successful transition to State i . (b) A team of molecules needed for a transition from State 3 to State 4 comprises transition molecules 4A, 4B, and 4A'. In the presence of the indicator molecule for gene-4, members in the team can be relocated into the new formation in which a bridge molecule (involving the recognition site of FokI) is created to contribute to a transition to State 4.



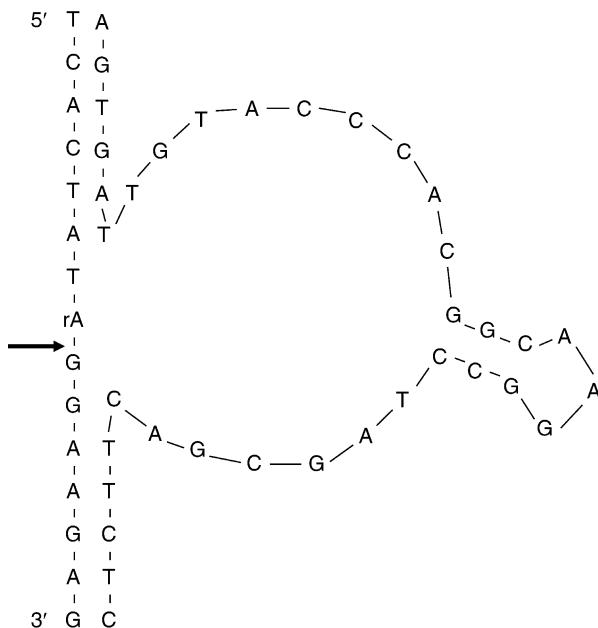
4.2 Molecular Gates and Circuits

4.2.1 DNAzymes

Stojanovic et al. established the use of DNAzymes to implement logic gates by DNA (Stojanovic et al. 2002). Just as a ribozyme refers to an enzyme made of RNA, a DNAzyme (deoxyribozyme) is a DNA molecule that has some catalytic function. **Figure 16** shows the DNAzyme named E6. The strand forming a loop structure is the DNAzyme that partially hybridizes with the target strand and cleaves it at the site shown by the arrow. The base denoted as rA is a single ribonucleotide (RNA nucleotide) inserted in the target strand. The enzyme cuts the phosphodiester bond at this site.

Fig. 16

DNAzyme named E6. The strand containing a ribonucleotide denoted as rA is cleaved at the site shown by the arrow.



After the target strand is cleaved, the partially double-stranded structure becomes unstable and the two pieces that arise from the target strand come apart. They can be regarded as outputs of the logic gate. If a fluorescent group and a quencher group are attached at the terminals of the target strand, an increase of fluorescence is observed when the strand is cleaved.

They introduced various modifications into the DNAzyme in order to control its catalytic function. Existence or non-existence of some strands affects the structure of such a modification and turns the catalytic function on or off. They can be regarded as inputs to the logic gate implemented by the modified DNAzyme.

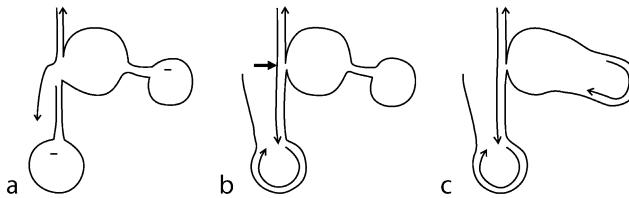
Figure 17a shows a schematic figure of the logic gate $A \wedge \neg B$. The modified enzyme implementing this gate has two more loops in addition to the loop of the original DNAzyme. It has the catalytic function if the loop denoted by $\overline{I_A}$ is opened while the loop denoted by $\overline{I_B}$ is intact. If the strand I_A which is complementary to the loop $\overline{I_A}$ is present, the loop is opened and the DNAzyme gains its catalytic function (Figure 17b). On the other hand, if the strand I_B is present, the loop $\overline{I_B}$ is opened and the catalytic function is lost because the structure of the original loop is changed (Figure 17c). Simply mixing this gate and the similar gate $B \wedge \neg A$ together results in the disjunction $(A \wedge \neg B) \vee (B \wedge \neg A)$, which is the XOR function of A and B .

Using logic gates implemented DNAzymes, Stojanovic and Stefanovic programmed a strategy to play the game of tic-tac-toe for demonstrating the expressiveness of DNAzyme-based logic circuits (Stojanovic and Stefanovic 2003).

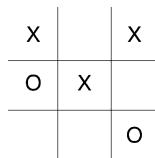
A move of a human player is represented by one of nine single strands of DNA, each corresponding to one of the nine positions on the board of tic-tac-toe. A logic circuit made of

Fig. 17

The schematic figure of the gate $A \wedge \neg B$. The DNAzyme is functional only if A is present and B is not.

**Fig. 18**

A move in tic-tac-toe. The next move of the DNA side is at the upper right position.



DNAzymes are placed at each position. For each move of the human player, the corresponding single strand is given to all the circuits. The programmed strategy shows the next move by making the circuit at the position of the move return output 1. More concretely, fluorescence emitted by the cleaved strand is increased at the position of the next move.

In [Fig. 18](#), two strands corresponding to the two moves of the human player (denoted by O) are put to all the nine positions. Then the circuit at the upper right position is triggered and gives output 1, or fluorescence at the position is increased.

In this work by Stojanovic and Stefanovic, all logic circuits at the nine positions are disjunctive normal forms and implemented by simply mixing the gates of all disjuncts. Each disjunct is a conjunction consisting of at most three literals. Each conjunction with three literals is of the form $A \wedge B \wedge \neg C$. This kind of conjunction can be implemented by a modified DNAzyme shown in [Fig. 19](#), which has three additional loops.

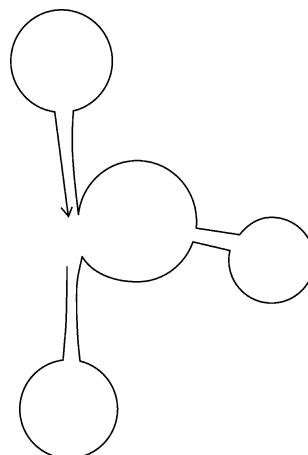
DNAzymes or ribozymes are used to implement logic circuits in many applications other than that of Stojanovic and Stefanovic as mentioned above. For example, Win and Smolke use a hammerhead ribozyme to control translation from mRNA to protein (Win and Smolke 2008).

They insert a hammerhead ribozyme with some additional structures into an untranslated region of a gene. As in the upper panel of [Fig. 20](#), the hammerhead ribozyme cleaves itself at the site shown by the arrow. The lower panel of the same figure shows additional structures attached at the loops of the ribozyme. These structures contain sensors and logic gates. Sensors are implemented by so-called aptamers, which are RNA molecules that change their structure if they bind to a certain small molecule. Logic gates are implemented by loop structures similar to those in Stojanovic's logic gates.

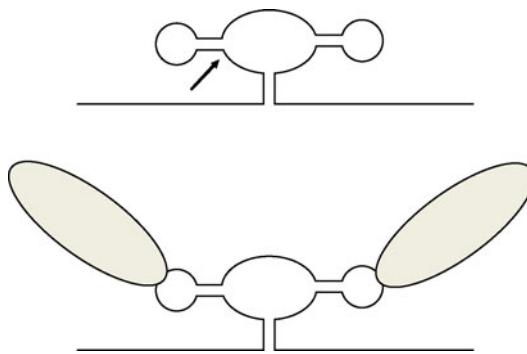
After the gene is transcribed, the hammerhead ribozyme in the mRNA transcript becomes active under some specific conditions depending on the additional structures, and cleaves the transcript itself. As a result, the rate of translation from the transcript is decreased. In this way,

Fig. 19

A schematic figure of a three-literal conjunction. A conjunction of the form $A \wedge B \wedge \neg C$ can be implemented in this way.

**Fig. 20**

A hammerhead ribozyme. The upper panel shows the original structure of the ribozyme, which cleaves itself at the site shown by the arrow. The lower panel shows the additional structures containing sensors and logic gates.



it is possible to control the expression of the gene by a combination of conditions that can be detected by the sensors.

4.2.2 Strand Displacements

As shown in the previous section, many bio-molecular machineries and logic gates have been proposed. Such molecular systems sophisticatedly utilized various restriction enzymes or DNAzymes (RNAzymes) to control their state transitions. However, enzyme-based logic

gates have a disadvantage in that it is difficult to cascade from one gate to another one because of design restrictions.

In this section, logic gates without enzymes (or enzyme-free logic gates) are introduced, which were developed by Seelig et al. (2006b). The driving force of Seelig's logic gates is a branch migration rather than enzymatic reactions. A branch migration, which is also called "strand displacement" or "oligonucleotide replacement," is the process in which a single strand hybridizes to a dangling end of duplex and then removes the resident strand from the duplex, resulting in the formation of a new duplex (see ▶ Fig. 21). The dangling end used to promote a branch migration is called a toehold, the length of which dominantly affects the kinetics of the reaction (Yurke and Mills 2003). The process of branch migration is regarded as a random walk (▶ Fig. 21a–c); the branching point moves left or right when two strands competitively hybridize to their complementary strand. If the resident strand wins the competition, the state comes back to ▶ Fig. 21a. In contrast, once the resident strand is removed from the duplex, the reaction stops because the new duplex does not have a toehold region (▶ Fig. 21d).

By using this branch migration reaction, various logic gates were successfully constructed (▶ Figs. 22 and ▶ 23). The reaction pathways of these gates are as follows:

- **AND gate** – First, by the branch migration, the input1 hybridizes to the toehold of the output complex (▶ Fig. 22b) and removes one single strand from the complex, resulting in

Fig. 21
Branch migration.

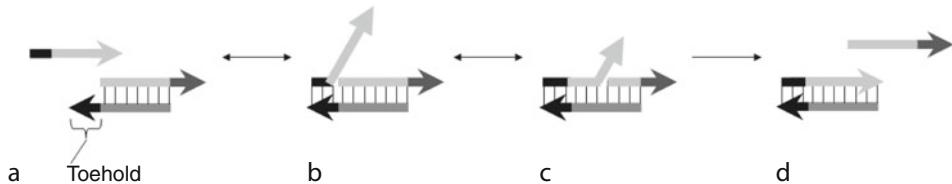


Fig. 22
AND gate.

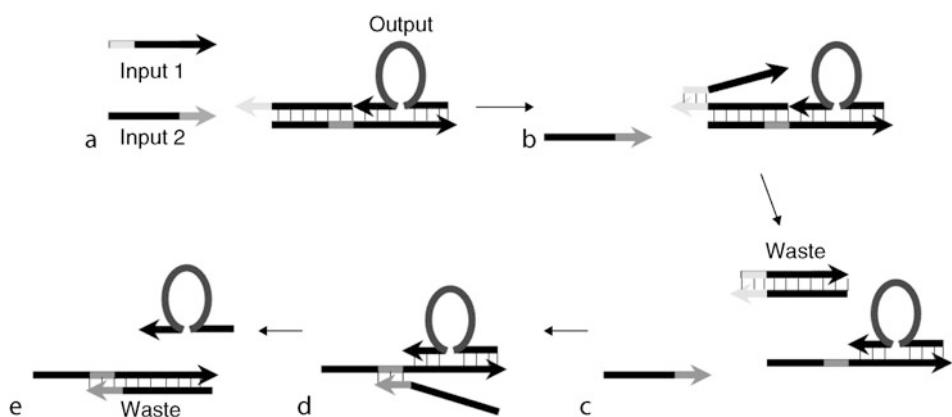
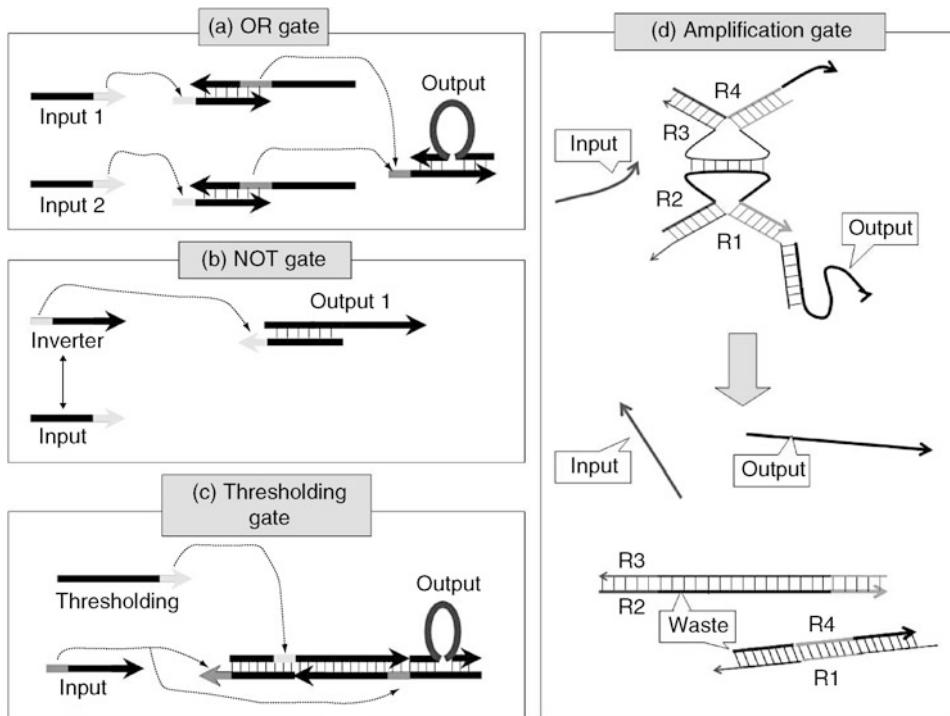


Fig. 23
Various gates.



the exposed toehold region for input2 (► Fig. 22c). Then, input2 removes the second single strand from the complex (► Fig. 22d, e), producing a single output strand. Thus, only if both input strands exist, is an output released as a single strand.

- **OR gate** – After either input1 or input2 hybridizes to the corresponding intermediate gate, either of their output strands produces the final output strand (► Fig. 23a). Therefore, the output strand is produced if either input1 or input2 exists.
- **NOT gate** – This gate consists of the output complex and the inverter strand that is complementary to the input strand (► Fig. 23b). The inverter strand hybridizes to the toehold of the complex and produces the single output strand. In contrast, if the input exists, it hybridizes to the inverter and prevents the branch migration between the inverter and output complex. Thus, this works as a NOT gate.
- **Thresholding gate** – This gate, which consists of a “thresholding” strand and output complex, can be used to remove the extra single strand that was produced unexpectedly (► Fig. 23c). The output complex has three toeholds. The first and third toeholds are complementary to the subsequence of input, while the second toehold corresponds to the thresholding strand. Thus, the output strand is released only if two inputs and one thresholding strand complete their branch migrations. Due to this stoichiometry, if input concentrations are twice as high as the thresholding strand, most gates will produce the output (see the supporting online material of Seelig et al. (2006b) for the theoretical consideration). Therefore, the threshold value can be controlled by the concentration of the thresholding strand.

- **Amplification gate** – Although the previous thresholding gate can remove the extra single strand, the concentration of the output does not exceed half of the input concentration. To restore the signal, this amplification gate is used (Fig. 23d). This gate produces input and output as a single strand after the multiple-stage branch migrations. Note that the input strand can react again with the other amplification gate. Thus, as long as amplification gates remain, input strands produce the output. Therefore, by combining the thresholding and amplification gates, one can exclude incorrect signals without deleting correct ones wrongly. For the detailed reaction pathway, please refer to Seelig et al. (2006a).

These enzyme-free logic gates have a great advantage in the cascading of gates. Because the output of each gate is a single strand, it can be easily used as the input of the other gates.

Here, readers may wonder if the above logic gates must be redesigned and reconstructed when one selects the other sequences as inputs. This problem can be solved with the following converter, which converts an input strand to another output strand (Fig. 24). By two-step branch migration, the input strand is converted into the output strand, whose sequence is not relevant to the input sequence. Thus, logic gates shown above can accept any input sequences by combining appropriate converters unless the sequences unfortunately cross-hybridize with each other.

With these logic gates shown above, one can construct the more complex logic circuits. In fact, Seelig et al. achieved the logic circuit including 11 gates and six inputs (Seelig et al. 2006b).

Based on Seelig's logic gates, Tanaka et al. developed a DNA comparator, which is a DNA machine for comparing DNA concentrations (Tanaka et al. 2009). The purpose of the DNA comparator is to determine whether the concentration of the target strand is higher than that of the reference strand. If the concentration of the target is higher, the DNA comparator outputs a single-strand, which may be an input strand to another molecular logic gate.

A DNA comparator consists of three double strands and two inputs (see Fig. 25). One of the two inputs is a target strand whose concentration one wants to compare, while the other is a reference strand whose concentration is the standard. Two of three double strands have bulge loops in the middle of the sequences and toehold structures, whose two inputs can hybridize to, at the 5'-end of the opposite sequences. The DNA comparator was designed to work as follows: First, two inputs *reference* and *target* hybridize to the toeholds of *cp_B1* and *cp_B2*, respectively. Then, *reference* and *cp_B1* form a double strand, while *target* and *cp_B2* also form a double strand by the branch migration reaction. After that, *B1* and *B2* are released as single strands and immediately form a double strand because the loop domains of *B1* and *B2* are complementary to each other. Thus, only the extra amount of *B2* can hybridize to the toehold of *cp_out*, resulting in branch migration. As a result, *output* is released as a single

Fig. 24
Converter based on branch migration.

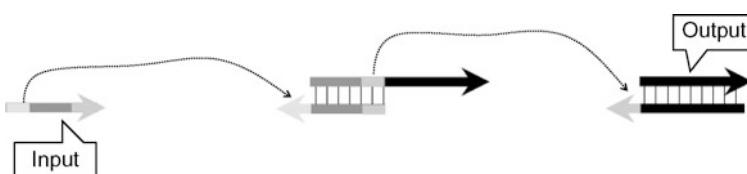
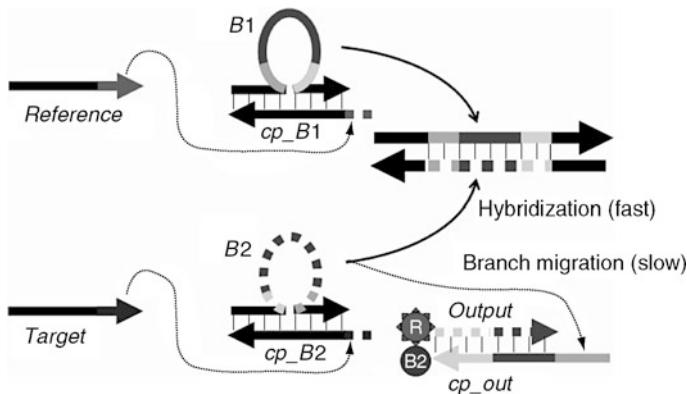


Fig. 25
DNA comparator.



strand, which can be monitored by the fluorescent intensity of a fluorophore. The key point is that the hybridization between $B1$ and $B2$ is much faster than the branch migration between $B2$ and $output \& cp_out$. Thus, if the concentration of $B2$ is lower than that of $B1$, almost all $B2$ will probably hybridize to $B1$ rather than the toehold of cp_out . Therefore, only if the concentration of $target$ is higher than that of $reference$ can $B2$ hybridize to the toehold of cp_out and remove $output$ from cp_out by branch migration.

4.2.3 DNA Tiles

Self-assembly of tiles has a powerful computational ability as shown by Wang (1961, 1962), and it even has universal computability, because it can simulate cellular automata and therefore Turing machines. More precisely, Wang showed that it is undecidable whether a given set of tiles can completely fill out the entire plane. Winfree reformulated this result in the context of DNA tiles (Winfree 1998). In this section, the focus is on the use of DNA tiles for implementing logic gates.

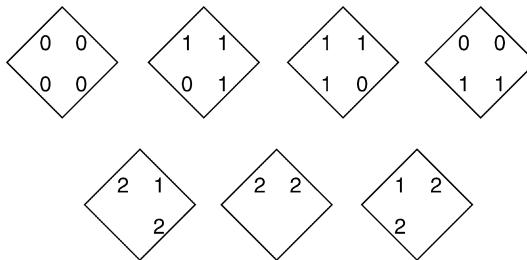
Figure 26 shows tiles for computing XOR and constructing Sierpinski's triangle. The four tiles in the top row implement the XOR function. The two edges facing down accept two inputs to XOR and the other two edges facing up give the (same) output of XOR. The three tiles in the bottom row form the boundary of Sierpinski's triangle.

In the mathematical model of tile assembly known as aTAM (abstract tile assembly model), proposed by Winfree, tiles self-assemble under the condition that only edges with the same label can bind together. Moreover, each label has its own binding strength. A tile can bind to a tile complex (aggregate) only if the sum of the strengths of the binding edges is greater than or equal to the predefined threshold called the temperature.

In this example, labels 0 and 1 have binding strength 1, while label 2 has binding strength 2. The temperature is set to 2. Therefore, as in Fig. 27, beginning with the corner tile (the middle tile in the bottom row of Fig. 26), the boundary of Sierpinski's triangle is formed. The tiles for XOR then bind to the boundary and construct Sierpinski's triangle as in Fig. 28.

Fig. 26

Tiles for Sierpinski's triangle. The upper tiles compute the XOR function. The lower tiles form the boundary of Sierpinski's triangle.

**Fig. 27**

The boundary of Sierpinski's triangle.

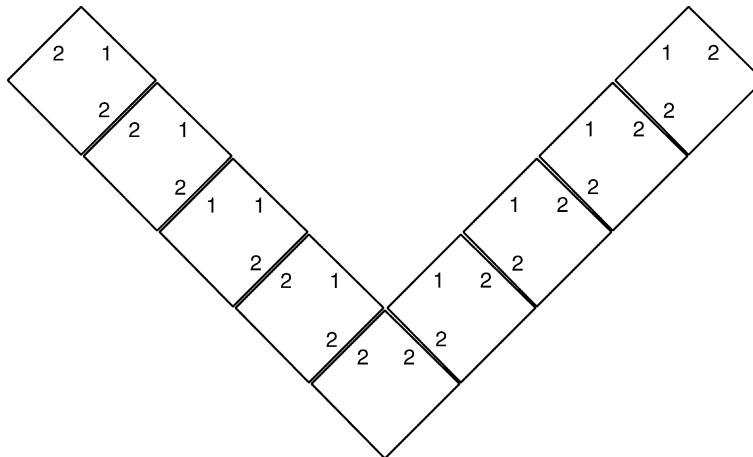


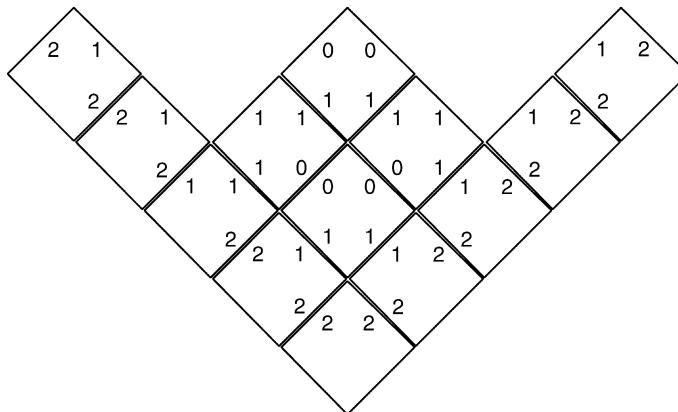
Figure 29 shows an actual implementation of tiles by DNA (Rothenmund et al. 2004). This structure, called a double crossover molecule, consists of four single strands of DNA. In other words, it consists of two double strands that exchange their single strands at two positions. This rectangular structure has four sticky ends, through which one instance of the structure can hybridize with four other instances.

Consequently, it works as a tile with four edges as mentioned above, and as in Figure 30, the self-assembly of DNA tiles forms a planar structure. Sticky ends of a DNA tile correspond to edges of an abstract tile, and different sequences of sticky ends correspond to different edge labels. Their length roughly corresponds to their binding strength.

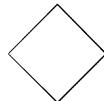
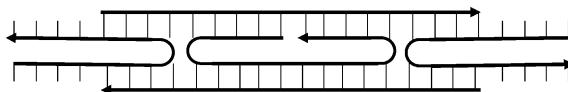
More complex computations can be performed by tiling (Mao 2000). For example, Figure 31 shows the addition of two binary numbers. The full-adder tile in the upper panel receives the corresponding bits of two numbers and the carry bit through its upper, lower, and right edges. It propagates its carry bit through its left edge. Addition of two three-bit numbers is performed as in the lower panel of the figure. The full-adder tile can be implemented by a triple crossover molecule, which has six sticky ends.

Fig. 28

Sierpinski's triangle.

**Fig. 29**

A double crossover molecule, also called a DNA tile. It consists of four single strands of DNA and forms a rectangular planar structure.



4.3 Reaction Graphs

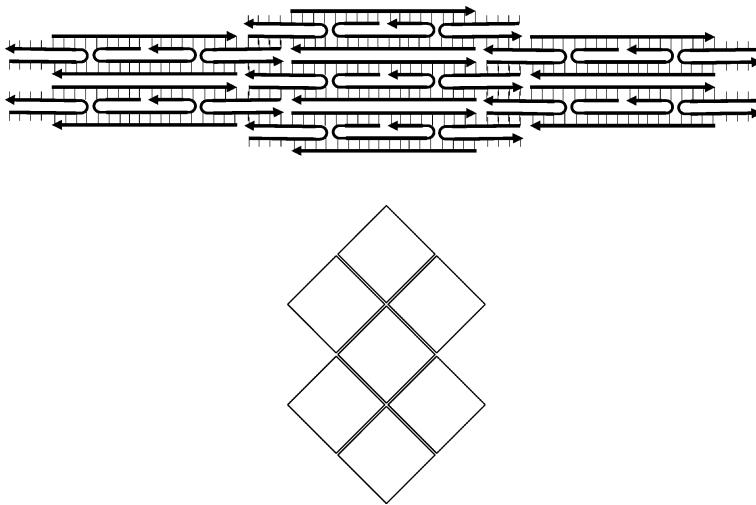
Many bio-molecular machineries and logic gates have been constructed successfully (Seelig et al. 2006a, b). As mentioned in [Sect. 4.2.2](#), DNA logic circuits without enzymes have attracted attention recently (Stojanovic et al. 2002; Stojanovic and Stefanovic 2003). However, such logic circuits are sometimes complicated because multiple sequences react with each other in various ways such as assembly, disassembly, and branch migration. Furthermore, sequences often consist of many domains, where some domains may be toeholds, and others may be loops of hairpin structures. This complexity leads to difficulties in understanding or designing the reaction dynamics of logic circuits.

To simply describe the self-assembly pathways, Peng Yin and his colleagues proposed the abstract model named “reaction graph,” which mainly focuses on circuits consisting of versatile DNA hairpin motifs (Yin et al. 2008a). With the reaction graph, the reaction dynamics can be simply described by nodes including input/output ports and arrows between nodes.

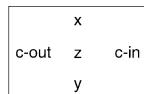
A simple example is shown in [Fig. 32](#). [Fig. 32a](#) shows the reaction dynamics. First, the sequence I hybridizes to the toehold region of hairpin A and then opens the hairpin by the branch-migration reaction ([Fig. 32a](#) (1)). Second, the complex structure $A \cdot I$ hybridizes to

Fig. 30

Self-assembly of DNA tiles. Through sticky ends, a DNA tile can hybridize with four other DNA tiles.

**Fig. 31**

Addition by tiles. The upper panel shows a full-adder tile, and the lower panel shows addition of two three-bit numbers.



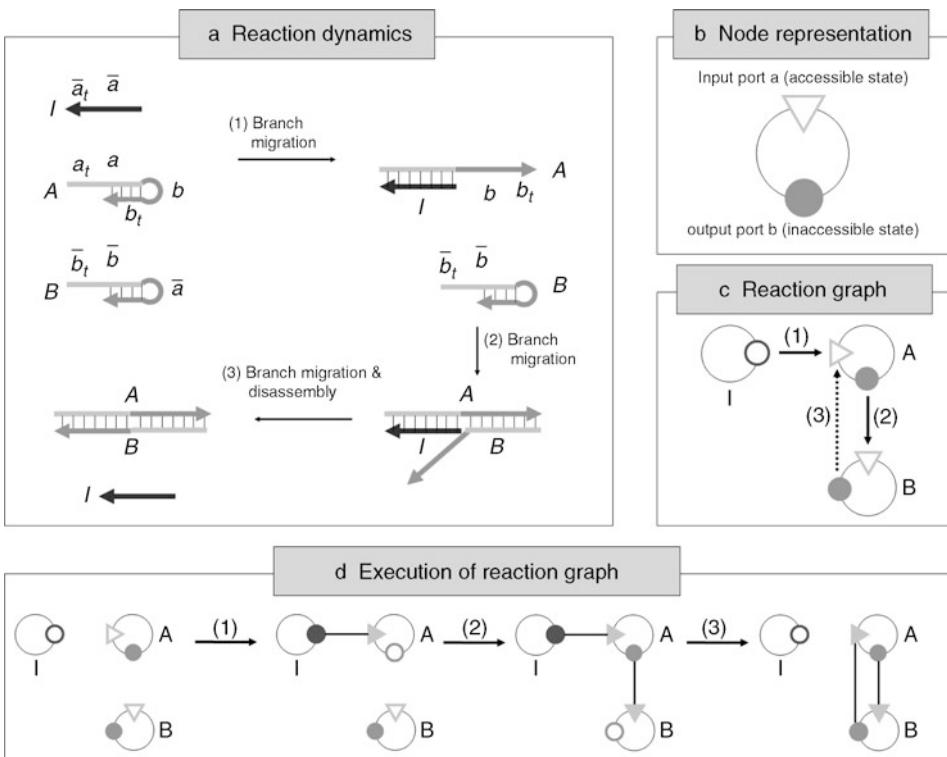
x2	x1	x0
x2	x1	x0
c-out	z2	z0 0
y2	y1	y0
y2	y1	y0

the toehold region of B and then opens the hairpin (Fig. 32a (2)). Third, the sequence B removes the sequence I from A by branch migration (Fig. 32a (3)). Because the removed I is a single strand, it can hybridize to the toehold of hairpin A again. Thus, the sequence I works as a catalyst to promote the hybridization between sequences A and B . (Note that without I , sequences A and B cannot hybridize to each other, because the complementary regions are blocked by their hairpin structures.)

This reaction explained above can be represented by the reaction graph shown in Fig. 32c. Each sequence is represented by a node (see Fig. 32b). Each node has some ports (input/output ports), which correspond to domains in each sequence. Input ports correspond to the domains which can receive the upstream outputs and are represented by triangles, while output ports are the domains which can be inputs for downstream hairpins and are

Fig. 32

Simple example from Yin et al. (2008a) (slightly changed).



represented by circles. Input and output ports have two states, accessible or inaccessible. If a domain is a single-stranded region, the corresponding port is accessible (represented by open triangles/circles) because the single-stranded region can hybridize with other sequences. In contrast, if a domain is a double-stranded region, the corresponding port is inaccessible (represented by filled triangles/circles). A solid arrow from an output port to an input port represents that the output port can hybridize to the input port if both ports are accessible. Furthermore, a dashed arrow from an output port to an input port represents that the output port can hybridize to the input port and remove the hybridized sequence from the input port. For example, the solid arrow denoted by (1) in Fig. 32c represents that the sequence I can hybridize to the toehold domain of A , while the dashed arrow denoted by (3) represents that B can hybridize to A and remove I from A .

Figure 32d is the transition of the reaction graph shown in Fig. 32c. The thin lines represent the hydrogen bonds between input and output ports. As readers can check easily, the reaction graph correctly represents the reaction dynamics: When I hybridizes to A , the output port of I and the input port of A are flipped to inaccessible states and connected by a thin line (Fig. 32d (1)). Then, the output port of A is flipped to accessible state and hybridizes to the input port of B . Both ports are flipped to inaccessible states and connected by a thin line, while the output port of B removes I from A and hybridizes to the input port of A , when both ports are flipped to inaccessible and

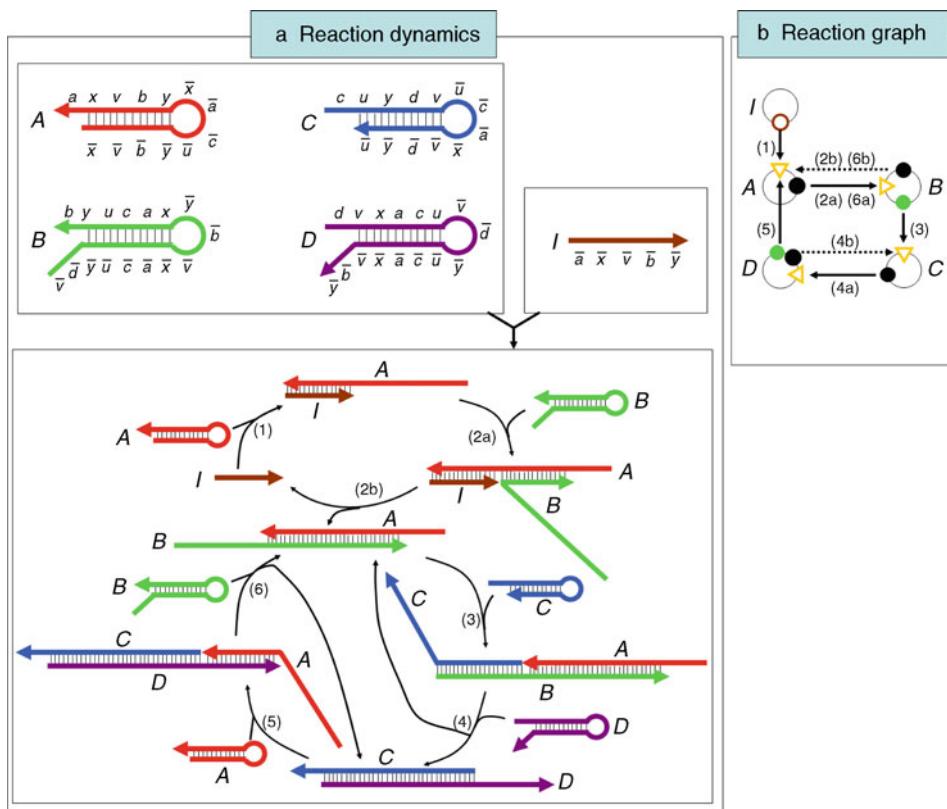
connected by a thin line. Finally, the system outputs I with the accessible output port and the $A \cdot B$ complex with inaccessible input/output ports (► Fig. 32d (3)).

► Figure 33 shows a more complicated example from Yin et al. (2008a). This example may be somewhat confusing. Complex $A \cdot B$ is produced by the hybridization between A and B promoted by I . Furthermore, $A \cdot B$ promotes the hybridization between C and D , while $C \cdot D$ does the hybridization between A and B simultaneously. Thus, the reaction dynamics are complicated and cannot be easily understood. With the reaction graph, however, the system can be simplified as shown in ► Fig. 33b:

- Sequence I hybridizes to the input port of A and opens the hairpin, resulting in the accessible output port of A (1).
- Then, the output port of A hybridizes to the input port of B (2a), when one output port of B , denoted by a black circle, is flipped to accessible, removes I from A , and hybridizes to the input port of A (2b).
- At the same time, the other output port of B , denoted by a gray circle, is flipped to accessible. This accessible output port can hybridize to the input port of C , resulting in the accessible output port of C (3).

► Fig. 33

More complicated example from Yin et al. (2008a) (slightly changed).



- The accessible output port of C hybridizes to the input port of D , when two output ports of D are flipped to accessible (4a).
- The accessible output port of D , denoted by a black circle, removes the complex $A \cdot B$ from C and hybridizes to the input port of C (4b), while the gray output port of D hybridizes to the input port of A (5).
- Then, the accessible output port of A hybridizes to the input port of B (6a), when the accessible black output port of B removes the complex $C \cdot D$ from A (6b).
- As a result, the system promotes the formation of complexes $A \cdot B$ and $C \cdot D$.

Readers can find some other examples of reaction graphs in Yin et al. (2008a), including three- or four-arm junctions, dendritic growth structures, and DNA walkers with two legs.

4.4 Whiplash Machines

4.4.1 Single-Molecule DNA State Machine

A distinctive DNA-based computing machine, called the *Whiplash machine*, has been continuously studied by a Japanese research group; it is a finite state machine implemented by encoding with a single DNA molecule. The parallel execution of multiple programs with different inputs in one reaction pot is realized by *Whiplash PCR* (WPCR), which is the recursive, self-directed polymerase extension of single-stranded DNA (ssDNA), proposed by Hagiya et al. (2000) and Sakamoto et al. (1999).

DNA Implementation of a Finite State Machine

The Whiplash machine utilizes the facts: a single-stranded DNA can form a *hairpin structure* by autonomously hybridizing the complementary sequences, and polymerase extension can generate a new DNA sequence as a reading out operation. Note that the rate of the computing process of the Whiplash machine is independent of the DNA concentration since hairpin formation is the intramolecular reaction.

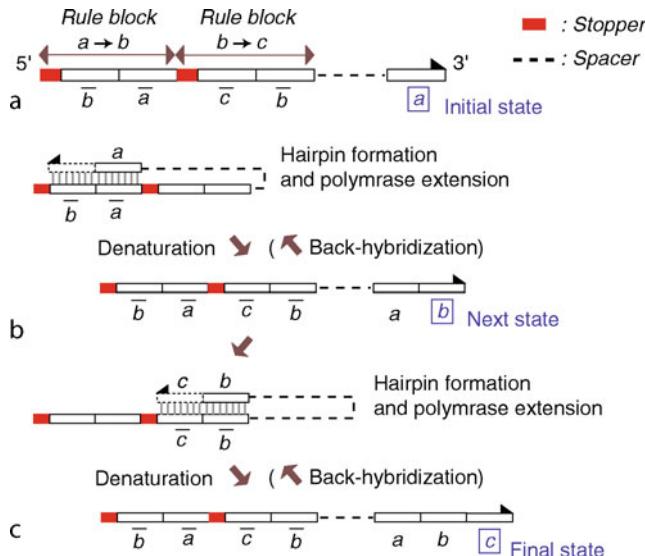
Suppose that a finite state machine M having transition rules for the three-state, two-step transitions, $a \rightarrow b \rightarrow c$. Then, the DNA implementation of M is carried out as follows.

(1) *Designing transition rules:* WPCR involves a sophisticated coding design of a ssDNA sequence (Fig. 34a). The set of transition rules are encoded in the form of catenated *rule blocks*, each encoding a state transition from state x to state y by an adjacent pair of sequences, $5' - \bar{y} \, \bar{x} - 3'$. Here, \bar{x} denotes the complementary sequence of x . A rule block also contains at its 5' side the *stopper*, which is commonly implemented by a short DNA sequence, AAA. Specifically, states a, b, c are encoded by sequences consisting of C, G, and T. The *transition rule region* contains rule blocks, $a \rightarrow b$ and $b \rightarrow c$. The 3' most sequence forms the *head*, which represents the Whiplash machine's current state.

(2) *Carrying out state transition:* By annealing, the complementary sequences a and \bar{a} come to hybridize, then a polymerase extension of the head takes place and the extension will be terminated automatically when the polymerase encounters the stopper, where the polymerization buffer is assumed to miss T (deoxythymidine triphosphate). The first transition, $a \rightarrow b$ is completed (Fig. 34b (the upper panel)). Then, by denaturation, the DNA again becomes the single-stranded form (Fig. 34b (the lower panel)). The 3' end of the ssDNA retains the current state b , renewed by reading out the complementary sequence \bar{b} in the rule block $a \rightarrow b$.

Fig. 34

Computing process of finite state machine by Whiplash PCR.



(3) *Successive transitions*: The process mentioned above can be repeated to perform further simulation of the state transition of M . After denaturation, the head representing the current state b is now ready for performing another simulation of the next transition, $b \rightarrow c$ (Fig. 34c).

Eight successive state transitions were so far achieved in the wet experiment (Komiya et al. 2006). As each DNA may be encoded with a distinct set of transition rules and a distinct initial state, parallel computation is achievable by the production of a combinatorial mixture of ssDNA. What is called the “multiple-instruction, multiple-data (MIMD)” computing paradigm is implemented by biomolecules. A solution to a 6-city instance of the directed Hamiltonian path problem using the Whiplash machine was demonstrated by Komiya et al. (2006). In addition, the WPCR model equipped with input/output interface was proposed in Komiya et al. (2001). Note that, in WPCR, a record of computation is encoded at the ssDNA’s 3’ end as a string of catenated state sequences. In Fig. 34c, this corresponds to the string, 5’-abc-3’. Implementation of evolutionary computation and its application to a novel protein evolution technique, taking advantage of the *computation record region*, were proposed (Rose et al. 2002b, 2003; Wood et al. 2002).

Back-Hybridization

Although the Whiplash machine is highly versatile in principle, it is known to suffer from an efficiency problem due to its systematic tendency towards self-inhibition, a problem called *back-hybridization*. Consider again the two-step transitions, shown in Fig. 34. The second polymerase extension requires the hybridization of the head sequence b with the sequence \bar{b} of rule block $b \rightarrow c$ (Fig. 34c). The efficiency of this process, however, is compromised by the ability of the ssDNA to form the alternative hairpin shown in Fig. 34b, which is no longer extendable and energetically more favorable than the planned, extendable hairpin. For multiple rounds of WPCR implementation, the number of distinct back-hybridized configurations

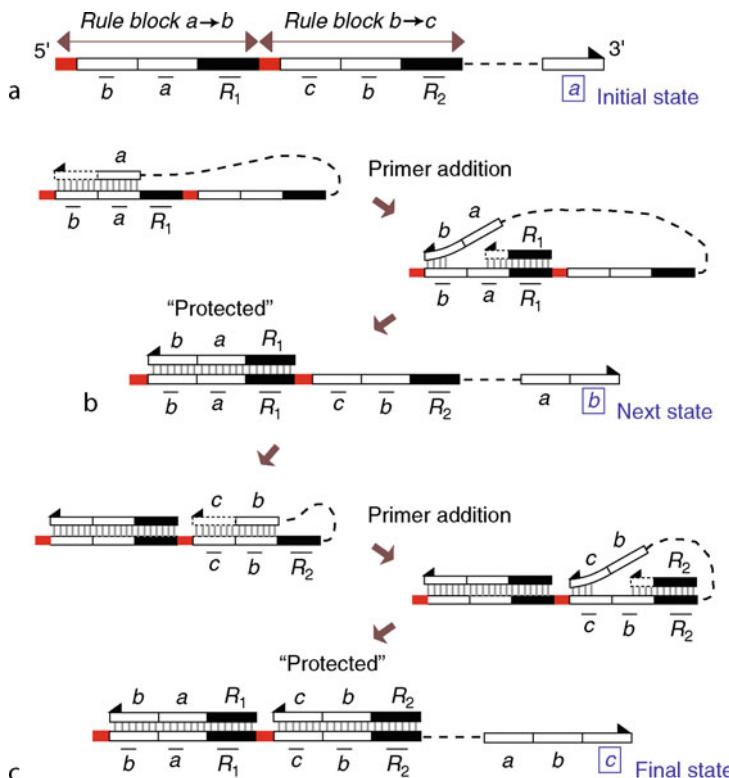
increases as the computation continues. Back-hybridization is a serious obstacle to practical application of the Whiplash machine (Rose et al. 2002a). In the next section, an optimally reengineered architecture of the Whiplash machine to overcome this efficiency problem is described.

4.4.2 Efficient and Controllable Whiplash Machine

An extended architecture of the WPCR, *displacement whiplash PCR* (DWPCR), was proposed by Rose et al. (2006). In DWPCR, the new *rule protect* operation, which is the primer-directed conversion of each implemented rule-block to double-stranded DNA, accompanied by opening the back-hybridized hairpins by strand displacement, is introduced. DWPCR implements the same series of operations applied in standard WPCR, adding only a protection step, where primer annealing, extension, and strand displacement occur at the targeted rule block, following each round of transition. A rule block contains an additional sequence at its 3' side for primer binding, represented by a black rectangle in Fig. 35.

DWPCR supports isothermal operation at physiological temperatures and is expected to attain a near-ideal efficiency by abolishing back-hybridized hairpins, and thus open the door for potential biological applications. Note that the use of rule-protect for controlling computation is being investigated (Komiya et al. 2009). Operation timing can be controlled

Fig. 35
Computing process of displacement Whiplash PCR.



by primer addition as an external signal for regulation. Furthermore, as transition rules may also be deactivated via rule-protect prior to their implementation, the transition rules to be implemented can be switched dependent on primer addition. Consequently, the extended Whiplash machine performs successive state transitions, following both a computational program encoded with an ssDNA molecule and a regulatory program implemented by a series of additions of primers, as a signal-dependent self-directed operation.

4.5 SAT Engines

The satisfiability problem (SAT, in short) is one of the well-known NP-complete problems where, given a Boolean formula, one has to decide whether or not the formula is satisfiable (i.e., there exists a truth value assignment under which the truth value of the formula is “true”).

As its name suggests, the *SAT Engine* (Sakamoto et al. 2000) has been developed for solving the SAT problem, in which the mechanism of hairpin formation is successfully employed to detect and remove many “inconsistent assignments” to a given Boolean formula.

Using an example, the idea of the SAT algorithm via DNA molecules is outlined. Consider a Boolean formula F with three variables, consisting of five clauses:

$$\begin{aligned} F &= C_1 \cdot C_2 \cdots C_5 \\ &= (a \vee b \vee c) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c) \end{aligned}$$

where a variable x and its negation $\neg x$ are called *literal*. A truth value assignment is a mapping, which for each variable assigns 1 or 0 (true or false). A formula F is *satisfiable* iff there exists a truth value assignment for which the value of F is true. In the running example, since there is a truth value assignment $(a, b, c) = (1, 1, 0)$ which leads to $F = 1$, F is satisfiable. If each clause C_i in F contains at most k literals, then F is called an instance of the k -SAT problem.

4.5.1 DNA SAT Algorithm

First, let one consider a string $u_F = t_1 t_2 \dots t_5$ over the alphabet $\Gamma = \{a, \neg a, b, \neg b, c, \neg c\}$, where each t_i is an arbitrary literal chosen from C_i , for each $i = 1, 2, \dots, 5$. A string u_F is called a *literal string* of F . For example, $aab \neg c \neg c$ and $acbb \neg c$ are literal strings of F . Each literal t_i chosen from C_i can be regarded as the value for which $C_i = 1$. (In the example, a literal string $aabb \neg c$ is interpreted as $(a, b, c) = (1, 1, 0)$ and it makes F consistently true, while a literal string $acbb \neg c$ is inconsistent with a variable c , that is, it contains both c and $\neg c$, failing to make F true.) A literal string of F is satisfiable if it can make F true, and it is unsatisfiable otherwise.

Now, the SAT algorithm comprises the following three steps;

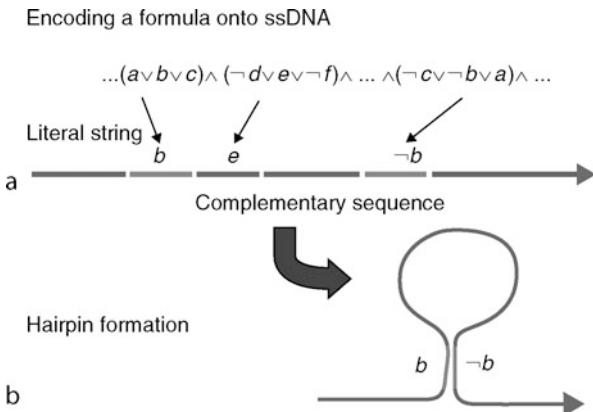
- (Step 1): For a formula F with n clauses, make the set $LS(F)$ of all literal strings of F whose length are all n . (In case of the 3-SAT problem, $LS(F)$ consists of at most 3^n literal strings.)
- (Step 2): Remove from $LS(F)$ any literal string that is unsatisfiable.
- (Step 3): After removing in Step 2, if there is any literal string remaining, then that is a satisfiable one, so that the formula F turns out to be satisfiable. Otherwise, F is not.

These can be implemented using biomolecular experimental techniques in the following manner:

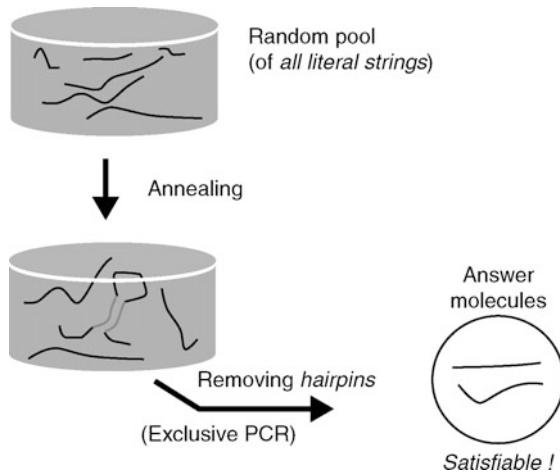
- (Step 1): By ligating each DNA molecule representing each literal, make all elements of the set $LS(F)$ in parallel.

Fig. 36

Encoding a formula onto ssDNA and hairpin formation.

**Fig. 37**

SAT algorithmic schema.



(Step 2): A variable x and its negation $\neg x$ are encoded so that these two are just complementary. Therefore, a literal string containing x and $\neg x$ is encoded into a molecule containing two subsequences which are complementary. (See [Fig. 36a](#).) Thus, an encoded molecule for an unsatisfiable literal string can form a *hairpin structure*. (See [Fig. 36b](#).)

(Step 3): Remove all DNA molecules that form hairpin structures. If there remains any molecule, then detect it and read the sequence for decoding the variable assignment. (See [Fig. 37](#))

4.5.2 Implementation and Experiments

A wet lab experiment was carried out by Sakamoto et al. (2000) where the following Boolean formula was considered

$$\begin{aligned} F = & (a \vee b \vee \neg c) \wedge (a \vee c \vee d) \wedge (a \vee \neg c \vee \neg d) \\ & \wedge (\neg a \vee \neg c \vee d) \wedge (a \vee \neg c \vee e) \wedge (a \vee d \vee \neg f) \\ & \wedge (\neg a \vee c \vee d) \wedge (a \vee c \vee \neg d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d) \end{aligned}$$

The total number of literal strings of F is 3^{10} ($= 59,049$) among which only 24 literal strings are satisfiable. (In fact, these 24 literal strings are for the identical truth value assignment.)

In their experiment, each literal was conceptually encoded onto a single-stranded DNA molecule (called *literal DNA*) of length 30 as follows:

a: TTGGTTGATAGACCCAGGATCGAGTGCAT
 b: TTGGCATAAGTTGCCAGGCAGGAACCTCCAT
 c: TTGGAACGTAGTACCAGGAGTCTCCTCCAT
 d: TTGGTGATACGGACCAGGCCTTCTTACCAT
 e: TTGGTTGCCCTCCAGGTGACTAATCCAT
 f: TTGGACTGCTCATCCAGGACTGAAGACCAT,

while the negative counterpart $\neg x$ of x (x in $\Gamma = \{a, b, c, d, e, f\}$) was encoded as the complementary sequence of x .

(Step 1) was actually carried out by concatenating with ligase all literal DNAs for x and $\neg x$ ($x \in \Gamma$), where in an actual implementation, each literal DNA was designed as a dsDNA with sticky ends at either side, playing the role of a linker for concatenation. (This is because ligase used for concatenation does not act on ssDNAs but on dsDNAs.) Another purpose of a linker is to ensure that the resultant of concatenating literal DNAs contains exactly one literal from each clause. A final pot of a random pool consisting of all molecules for $LS(F)$ was created autonomously. The experimental result of (Step 1) is reported to have been successful.

In (Step 2), each dsDNA in the pot was first heated to separate into two ssDNAs, one of which was washed out. Then, the remaining ssDNAs are now ready to form *hairpin structures* if they are DNAs encoding unsatisfiable assignments. In other words, if there exist ssDNAs that do not form hairpin structures, then it means that there exists truth value assignments for a given F (i.e., the formula F is satisfiable). Note that each literal DNA contains a recognition site of a restriction enzyme *Bst*NI, and this hidden sequence will play an important role, as seen below.

In (Step 3), by using a restriction enzyme *Bst*NI, the hairpin structure formed in (Step 2) is cut out, so that all DNAs containing hairpin structures lose a chance to survive until the next procedure in which the application of PCR will multiply the remaining DNAs for sequencing them.

In practice, 6 literal strings were finally obtained which include

$$bc\neg d\neg ae\neg f\neg ac\neg a\neg a \text{ and } bc\neg d\neg ae\neg f\neg a\neg d\neg a\neg a$$

It is reported that all those literal strings were for the unique assignment:

$$(a, b, c, d, e, f) = (0, 1, 1, 0, 1, 0)$$

which provides, in fact, a correct solution to the example formula.

4.6 Bibliographic Notes

Research on computing Boolean circuits using DNA is well established and dates back to Ogihara and Ray (1999) and Amos et al. (1998). Their methods, however, require that for each logic gate an external experimental operation is applied to a test tube. On the other hand, this section focused on molecular computing machineries that can compute a series of logic gates in a self-controlled way without human intervention in the sense of autonomous molecular computers advocated by Hagiya (1999).

As DNA automata are implemented by enzymes, logic gates can also be implemented by enzymes. In particular, an autonomous molecular computer Reverse-transcription-and-TTranscription-based Autonomous Computing System (RTRACS) proposed and implemented in Nitta and Suyama (2004) and Takinoue et al. (2008) has some unique features. Modeled after the retroviral replication mechanism, RTRACS utilizes not only DNA/RNA molecules but also reverse transcriptase and other enzymatic substrates, and the computation is autonomously carried out under an isothermal environment. An *in vitro* experiment of simulating an AND gate in Takinoue et al. (2008) demonstrates a potentially high computational capability of the integrated circuits of the modularized basic units of RTRACS.

More and more efforts are being made to implement *in vivo* logic circuits as proposed by Benenson et al. in [Sect. 4.1.2](#). Smolke's RNA computer mentioned in [Sect. 4.2.1](#) is one such effort. Benenson also proposed to use small interfering RNAs (siRNAs) for implementing logic circuits (Rinaudo et al. 2007). They both use RNA to regulate gene expression. Implementing logic gates is also a main research topic in the field of synthetic biology. Logic circuits are implemented by modifying genetic circuits made of genes and regulatory proteins that regulate gene expression (Endy 2005). Artificial genetic circuits are not covered in this section, as synthetic biology is considered out of the scope of this chapter.

As is also mentioned in the bibliographic notes of [Sect. 3](#), self-assembly of DNA tiles has been one of the main topics in DNA computing research (Jonoska and McColm 2009; Winfree 1998; Winfree et al. 1996, 2000), though logic gates implemented by DNA tiling is briefly touched on in [Sect. 4.2.3](#).

The machineries based on DNAzymes and those based on strand displacements, which have been extended to reaction graphs in [Sect. 4.3](#), are being applied to implement more typical molecular machines, including walkers and transporters (Bath and Turberfield 2007).

References

- Adleman L (1994) Molecular computation of solutions to combinatorial problems. *Science* 266:1021–1024
- Adleman L (1996) On constructing a molecular computer. DNA based computers. Series in mathematics and theoretical computer science, vol 27. American Mathematical Society, Providence, RI, pp 1–22
- Amos M, Dunne PE, Gibbons A (1998) DNA simulation of Boolean circuits. In: Koza et al. (eds) Proceedings of the third annual conference on genetic programming, University of Wisconsin, Madison, WI, July 1998. Morgan Kaufmann, San Francisco, CA, pp 679–683
- Arita M, Hagiya M, Suyama A (1997) Joining and rotating data with molecules. In: IEEE international conference on evolutionary computation, Indianapolis, IN, June 1996, IEEE Service Center, pp 243–248
- Bath J, Turberfield JA (2007) DNA nanomachines. *Nat Nanotechnol* 2:275–284
- Beaver D (1995) A universal molecular computer. In: Lipton J, Baum B (eds) DNA based computers. DIMACS series in discrete mathematics and theoretical computer science, vol 27. American Mathematics Society, Providence, RI, pp 29–36
- Benenson Y, Adar R, Paz-Elizur T, Livneh Z, Shapiro E (2003) DNA molecule provides a computing machine with both data and fuel. *Proc Natl Acad Sci USA* 100(5):2191–2196

- Benenson Y, Gil B, Ben-Dor U, Adar R, Shapiro E (2004) An autonomous molecular computer for logical control of gene expression. *Nature* 429: 423–429
- Benenson Y, Paz-Elizur T, Adar R, Keinan E, Livneh Z, Shapiro E (2001) Programmable and autonomous computing machine made of biomolecules. *Nature* 414:430–434
- Brauer W, Ehrig H, Karhumäki J, Salomaa A (eds) (2002) Formal and natural computing. Lecture notes in computer science, vol 2300. Springer, Berlin
- Brendel V, Busse HG (1984) Genome structure described by formal languages. *Nucl Acids Res* 12:2561–2568
- Cheptea D, Martin-Vide C, Mitrana V (2006) A new operation on words suggested by DNA biochemistry: hairpin completion. In: Proceedings of transgressive computing, Universidad de Granada, Spain, April 2006, 216–228
- Condon A (2004) Automata make antisense. *Nature* 429:351–352
- Conrad M (1985) On design principles for a molecular computer. *Comm ACM* 28(5):464–480
- Conrad M (1992) Molecular computing paradigms. *IEEE Comput* 25(11):6–9
- Csuhaj-Varju E, Kari L, Păun Gh (1996) Test tube distributed systems based on splicing. *Comput AI* 15:211–232
- Csuhaj-Varju E, Verlan S (2008) On length-separating test tube systems. *Nat Comput* 7:167–181
- Dassow J, Păun Gh (1989) Regulated rewriting in formal language theory. Springer, Berlin
- Doramatzki M (2006) Hairpin structures defined by DNA trajectories. In: DNA 12: Proceedings of 12th international meeting on DNA computing, Seoul, Korea, June 2006. Lecture notes in computer science, vol 4287. Springer, Berlin, pp 182–194
- Eberling W, Jimenez-Montano MA (1980) On grammars, complexity and information measures of biological macromolecules. *Math Biosci* 52:53–72
- Endy D (2005) Foundations for engineering biology. *Nature* 438:449–453; (vol 438 – 24 November 2005 – doi:10.1038/nature04342.)
- Eng T (1997) Linear DNA self-assembly with hairpins generates the equivalent of linear context-free grammars. DNA based computers III. DIMACS series in discrete mathematics and theoretical computer science, vol 48. American Mathematical Society, Providence, RI, pp 289–296
- Engelfriet J, Rozenberg G (1980) Fixed point languages, equality languages, and representation of recursively enumerable languages. *J ACM*, 27(3):499–518
- Freund R, Kari L, Păun Gh (1999a) DNA computing based on splicing: the existence of universal computers. Technical report, Fachgruppe Informatik, Tech. Univ. Wien, 1995, and Theory Comput Syst 32:69–112
- Freund R, Păun Gh, Rozenberg G, Salomaa A (1999b) Watson-Crick finite automata. DNA based computers III. DIMACS series in discrete mathematics and theoretical computer science, vol 48. American Mathematical Society, Providence, RI, pp 297–327
- Gao Y, Garzon M, Murphy RC, Rose JA, Deaton R, Franceschetti DR, Stevens SE Jr (1999) DNA implementation of nondeterminism. DNA based computers III. DIMACS series in discrete mathematics and theoretical computer science, vol 48. American Mathematical Society, Providence, RI, pp 137–148
- Geffert V (1991) Normal forms for phrase-structure grammars. *RAIRO Theor Inform Appl* 25:473–496
- Hagiya M (1999) Perspectives on molecular computing. *New Generation Comput* 17:131–151
- Hagiya M (2001) From molecular computing to molecular programming. In: DNA6: Proceedings of sixth international meeting on DNA based computers, Leiden, the Netherlands, June 2000. Lecture notes in computer science, vol 2054. Springer, Berlin, pp 89–102
- Hagiya M, Arita M, Kiga D, Sakamoto K, Yokoyama S (2000) Towards parallel evaluation and learning of Boolean μ -formulas with molecules. In: Rubin H, Wood D (eds). DNA based computers III, DIMACS series in discrete mathematics, vol 48. American Mathematical Society, Providence, RI, pp 57–72
- Hagiya M, Ohuchi A (2002) Preliminary proceedings of the eighth international meeting on DNA based computers, Hokkaido University, Hokkaido, Japan, June 10–13, 2002
- Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bull Math Biol* 49:737–759
- Head T, Păun Gh, Pixton D (1997) Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination In: Rozenberg G, Salomaa A (eds) Handbook of formal languages, vol 2. Springer, Berlin, pp 295–360
- Hopcroft JE, Motwani R, Ullman JD (2001) Introduction to automata theory, languages, and computation, 2nd edn. Addison-Wesley, Reading, MA
- Horn A (1951) On sentences which are true of direct unions of algebras. *J Symbolic Logic* 16:14–21
- Ignatov Z, Martinez-Perez I, Zimmermann K-H (2008) DNA Computing Models. Springer, New York
- Jimenez-Montano MA (1984) On the syntactic structure of protein sequences and the concept of grammar complexity, *Bull Math Biol* 46:641–659
- Jonoska N, Karl SA, Saito M (1998) Three dimensional DNA structures in computing. In: Kari L (ed) Proceedings of 4th DIMACS meeting on DNA based computers, University of Pennsylvania, Philadelphia, PA, June 16–19, 1998, American Mathematics Society, pp 189–200
- Jonoska N, McColm GL (2009) Complexity classes for self-assembling flexible tiles. *Theor Comput Sci* 410:332–346

- Joshi AK, Schabes Y (1997) Tree-adjoining grammars. In: Rozenberg G, Salomaa A (eds) *Handbook of formal languages*, vol. 3. Springer, Berlin, pp 69–123
- Kari L (1996) DNA computers: tomorrow's reality. *Tutorial Bull EATCS* 59:256–266
- Kari L, Konstantinidis S, Losseva E, Sosik P, Thierrin G (2005a) Hairpin structures in DNA words. In: *DNA11: Proceedings of the 11th international meeting on DNA computing*, London, ON, Canada, June 2005, Lecture notes in computer science, vol 3892. Springer, Berlin, pp 267–277
- Kari L, Konstantinidis S, Sosik P, Thierrin G (2005b) On hairpin-free words and languages. In: deFelice C, Restivo A (eds) *Proceedings of the 9th international conference on developments in language theory*, Palermo, Italy, July 2005. Lecture notes in computer science, vol 3572. Springer, Berlin, pp 296–307
- Kari L, Rozenberg G (2008) The many facets of natural computing. *C. ACM*: 51(10):72–83
- Kari L, Konstantinidis S, Sosik P (2005c) On properties of bond-free DNA languages. *Theor Comput Sci* 334:131–159
- Kobayashi S (1999) Horn clause computation with DNA molecules. *J Combinatorial Optimization* 3:277–299
- Kobayashi S, Mitrana V, Păun G, Rozenberg G (2001) Formal properties of PA-matching. *Theor Comput Sci* 262:117–131
- Kobayashi S, Sakakibara Y (1998) Multiple splicing systems and the universal computability. *Theor Comput Sci* 264:3–23
- Kobayashi S, Yokomori T, Sanpei G, Mizobuchi K (1997) DNA implementation of simple Horn clause computation. In: *IEEE international conference on evolutionary computation*, Indianapolis, IN, April 1997, IEEE Service Center, pp 213–217
- Komiya K, Rose JA (2009) Experimental validation of signal dependent operation in Whiplash PCR. In: Goel A, Simmel FC (eds) *DNA computing*. 14th international workshop on DNA-based computers, Prague, Czech Republic, June 2008. Lecture notes in computer science, vol 5347, pp 1–10
- Komiya K, Sakamoto K, Gouzu H, Yokohama S, Arita M, Nishikawa A, Hagiya M (2001) Successive state transitions with I/O interface by molecules. In: Condon A, Rozenberg G (eds) *DNA computing*. 6th international workshop on DNA-based computers, Leiden, the Netherlands, June 2000. Lecture notes in computer science, vol 2054, pp 17–26
- Komiya K, Sakamoto K, Kameda A, Yamamoto M, Ohuchi A, Kiga D, Yokoyama S, Hagiya M (2006) DNA polymerase programmed with a hairpin DNA incorporates a multiple-instruction architecture into molecular computing. *Biosystems* 83:18–25
- Kuramochi J, Sakakibara Y (2005) Intensive *in vitro* experiments of implementing and executing finite automata in test tube. In: *DNA11: Proceedings of 11th international workshop on DNA computers*, London, Canada, June 2005. Lecture notes in computer science, vol 3892, pp 193–202
- LaBean TH, Winfree E, Reif JH (2000) Experimental progress in computation by self-assembly of DNA tilings. In: Winfree E, Gifford DK (eds) *DNA based computers V*. DIMACS series in discrete mathematics and theoretical computer science, vol 54. American Mathematical Society, Providence, RI, pp 123–140
- Lagoudakis MG, LaBean TH (2000) 2D DNA self-assembly for satisfiability. In: Winfree E, Gifford DK (eds) *DNA based computers V*. DIMACS series in discrete mathematics and theoretical computer science, vol 54. American Mathematical Society, Providence, RI, pp 141–154
- Laun E, Reddy K (1997) Wet splicing systems. In: *Proceedings of 3rd DIMACS meeting on DNA based computers*, University of Pennsylvania, Philadelphia, PA, June 23–25, 1997, pp 115–126
- Lipton RJ (1995) DNA solution of hard computational problems. *Science* 268:542–545
- Lipton RJ, Baum EB (eds) (1996) *DNA based computers*. Series in mathematics and theoretical computer science, vol 27. American Mathematical Society, Providence, RI
- Manea F, Mitrana V (2007) Hairpin completion versus hairpin reduction. In: *Computation in Europe*, CiE 2007, Siena, Italy, June 2007. Lecture notes in computer science, vol 4497. Springer, Berlin, pp 532–541
- Manea F, Mitrana V, Yokomori T (2008) Some remarks on the hairpin completion. In: *Proceedings of 12th international conference on AFL*, Hungary, May 2008. (to appear in *Int J Found Comput Sci*)
- Manea F, Mitrana V, Yokomori T (2009a) Two complementary operations inspired by the DNA hairpin formation: completion and reduction. *Theor Comput Sci* 410:417–425
- Manea F, Martín-Vide C, Mitrana V (2009b) On some algorithmic problems regarding the hairpin completion. *Discrete Appl Math* 157:2143–2152
- Mao C, LaBean TH, Relf JH, Seeman NC (Sep 2000) Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature* 407 (6803):493–496, Sep 2000
- Marcus S (1969) Contextual grammars. *Revue Roum Math Pures Appl* 14(10):473–1482
- Nitta N, Suyama A (2004) Autonomous biomolecular computer modeled after retroviral replication. In: Chen J, Reif J (eds) *DNA9: Proceedings of 9th international meeting on DNA-based computers*, Madison, WI, June 2003. Lecture notes in computer science, vol 2943, pp 203–212
- Ogihara M, Ray A (1999) Simulating Boolean circuits on a DNA computer. *Algorithmica* 25:239–250
- Păun Gh (1996a) Five (plus two) universal DNA computing models based on the splicing operation.

- In: Proceedings of 2nd DIMACS workshop on DNA based computers, Princeton, NJ, June 1996, pp 67–86
- Păun Gh (1996b) Regular extended H systems are computationally universal. *J Automata, Lang Combinatorics* 1(1):27–36
- Păun Gh (1999) (DNA) Computing by carving. *Soft Comput* 3(1):30–36
- Păun Gh, Rozenberg G, Salomaa A (1998) DNA computing: new computing paradigms. Springer, Berlin
- Păun Gh, Rozenberg G, Yokomori T (2001) Hairpin languages. *Int J Found Comput Sci* 12(6):837–847
- Pixton D (1995) Linear and circular splicing systems. In: Proceedings of 1st international symposium on intelligence in neural and biological systems, Herndon, VA, May 1995. IEEE, Washington, DC, pp 38–45
- Post E (1943) Formal reductions of the general combinatorial decision problem. *Am J Math* 65:197–215
- Reif J (1995) Parallel molecular computation. In: SPA'95: 7th annual ACM symposium on parallel algorithms and architectures, Santa Barbara, CA, July 1995, pp 213–223
- Reif J (1999) Parallel biomolecular computation: models and simulations. *Algorithmica* 25:142–176
- Rinaudo K, Bleris L, Maddamsetti R, Subramanian S, Weiss R, Benenson Y (2007) A universal RNAi-based logic evaluator that operates in mammalian cells. *Nat Biotechnol* 25:795–801. (Published online: May 21, 2007 – doi:10.1038/nbt1307.)
- Rooß D, Wagner K (1996) On the power of DNA-computing. *Infor Comput* 131:95–109
- Rose JA, Deaton RJ, Hagiya M, Suyama A (2002a) Equilibrium analysis of the efficiency of an autonomous molecular computer. *Phys Rev E* 65:021910
- Rose JA, Hagiya M, Deaton RJ, Suyama A (2002b) A DNA-based in vitro genetic program. *J Biol Phys* 28:493–498
- Rose JA, Komiya K, Yaegashi S, Hagiya, M (2006) Displacement Whiplash PCR: optimized architecture and experimental validation. In: Mao C, Yokomori T (eds) DNA computing. 12th international workshop on DNA-based computers, Seoul, Korea, June 2006. Lecture notes in computer science, vol 4287, pp 393–403
- Rose JA, Takano M, Hagiya M, and Suyama A (2003) A DNA computing-based genetic program for in vitro protein evolution via constrained pseudomodule shuffling. *J Genet Programming Evolvable Mach* 4:139–152
- Rothenmund PWK (1995) A DNA and restriction enzyme implementation of Turing machines. In: Lipton J, Baum B (eds) DNA based computers. DIMACS series in discrete mathematics and theoretical computer science, vol 27. American Mathematical Society, Providence, RI, pp 75–119
- Rothenmund PWK, Papadakis N, Winfree E (Dec 2004) Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol* 2(12):e424
- Rozenberg G, Salomaa A (eds) (1997) Handbook of formal languages, 3 volumes. Springer, Berlin
- Salomaa A (1985) Computation and automata. Cambridge University Press, Cambridge
- Sakakibara Y, Ferretti C (1997) Splicing on tree-like structures. In: Proceedings of 3rd DIMACS meeting on DNA based computers, University of Pennsylvania, Philadelphia, PA, June 23–25, 1997, pp 348–358. Also, in *Theor Comput Sci* 185:15–45, 1999
- Sakakibara Y, Kobayashi S (2001) Sticker systems with complex structures. *Soft Comput* 5:114–120
- Sakakibara Y, Suyama A (2000) Intelligent DNA chips: logical operation of gene expression profiles on DNA computers. In: Genome Informatics 2000: Proceedings of 11th workshop on genome informatics, Tokyo, Japan, December 2000. Universal Academy Press, Tokyo, Japan, pp 33–42
- Sakamoto K, Gouzu H, Komiya K, Kiga D, Yokoyama S, Yokomori T, Hagiya M (2000) Molecular computation by DNA hairpin formation. *Science* 288:1223–1226
- Sakamoto K, Kiga D, Komiya K, Gouzu H, Yokoyama S, Ikeda S, Sugiyama H, Hagiya M (1999) State transitions by molecules. *BioSystems* 52(1–3):81–91
- Seelig G, Yurke B, Winfree E (2006a) Catalyzed relaxation of a metastable DNA fuel. *J Am Chem Soc* 128 (37):12211–12220
- Seelig G, Soloveichik D, Zhang DY, Winfree E (Dec 2006b) Enzyme-free nucleic acid logic circuits. *Science* 314(5805):1585–1588
- Shih W (Feb 2008) Biomolecular assembly: dynamic DNA. *Nat Mater* 7(2):98–100
- Siromoney R, Subramanian KB, Rajkumar Dare V (1992) Circular DNA and splicing systems. In: Proceedings of Parallel Image Analysis, Ube, Japan, December 1992. Lecture notes in computer science, vol 654. Springer, Berlin, pp 260–273
- Smith WD (1995) DNA computers in vitro and vivo. In: Lipton J, Baum B (eds) DNA based computers. DIMACS series in discrete mathematics and theoretical computer science, vol 27. American Mathematics Society, Providence, RI, pp 121–185
- Stojanovic MN, Mitchell TE, Stefanovic D (Apr 2002) Deoxyribozyme-based logic gates. *J Am Chem Soc* 124(14):3555–3561
- Stojanovic MN, Semova S, Kolpashchikov D, Macdonald J, Morgan C, Stefanovic D (May 2005) Deoxyribozyme-based ligase logic gates and their initial circuits. *J Am Chem Soc* 127(19):6914–6915
- Stojanovic MN, Stefanovic D (Sep 2003) A deoxyribozyme-based molecular automaton. *Nat Biotechnol* 21(9):1069–1074

- Takinoue M, Kiga D, Shohda K, Suyama A (2008) Experiments and simulation models of a basic computation element of an autonomous molecular computing system. *Phys Rev E* 78:041921
- Tanaka F, Tsuda T, Hagiya M (2009) Towards DNA comparator: the machine that compares DNA concentrations. In: Goel A, Simmel FC (eds) DNA computing, 14th international workshop on DNA-based computers, Prague, Czech Republic, June 2008. Lecture notes in computer science, vol 5347, pp 11–20
- Tarnlund S (1977) Horn clause computability. *BIT* 17:215–226
- Uejima H, Hagiya M, Kobayashi S (2001) Horn clause computation by self-assembly of DNA molecules. In: Proceedings of 7th international workshop on DNA-based computers, Tampa, FL, June 2001. Lecture notes in computer science, vol 2340, pp 308–320
- Vaintsava MN, Liberman EA (1973) Formal description of cell molecular computer. *Biofizika* 18:939–942
- Venkataraman S, Dirks RM, Rothemund PWK, Winfree E, Pierce NA (2007) An autonomous polymerization motor powered by DNA hybridization. *Nat Nanotechnol* 2:490–494
- Wang H (1961) Proving theorems by pattern recognition-II. *Bell Syst Tech J* 40:1–41
- Wang H (1962) Dominoes and the AEA case of the decision problem. In: Proceedings of the symposium on mathematical theory of automata. New York, April 1962. Polytechnic Institute of Brooklyn, Brooklyn, New York, pp 23–55
- Win MN, Smolke CD (Oct 2008) Higher-order cellular information processing with synthetic RNA devices. *Science* 322(5900):456–460
- Winfree E (1998) Algorithmic self-assembly of DNA. Ph.D. thesis, California Institute of Technology
- Winfree E, Eng T, Rozenberg G (2000) String tile models for DNA computing by self-assembly. In: Proceedings of the 6th international meeting on DNA based computers. Leiden University, Leiden, the Netherlands, June 13–17, 2000, pp 65–84
- Winfree E, Yang X, Seeman NC (1996) Universal computation via self-assembly of DNA: some theory and experiments DNA based computers II. DIMACS series in discrete mathematics and theoretical computer science, vol 44. American Mathematical Society, Providence, RI, pp 191–213
- Wood D, Bi H, Kimbrough S, Wu D-J, Chen J (2002) DNA starts to learn poker. In: Jonoska N, Seeman N (eds) DNA computing, 7th international workshop on DNA-based computers Tampa, FL, June 2001. Lecture notes in computer science, vol 2340, pp 22–32
- Yin P, Choi HMT, Calvert CR, Pierce NA (Jan 2008a) Programming biomolecular self-assembly pathways. *Nature* 451(7176):318–322
- Yin P, Hariadi RF, Sahu S, Choi HMT, Park SH, LaBean TH, Reif JH (2008b) Programming DNA tube circumferences. *Science* 321:824–826
- Yokomori T (1999) Computation = self-assembly+ conformational change: toward new computing paradigms. In: DLT'99: Proceedings of 4th international conference on developments in language theory, Aachen, Germany, July 1999, pp 21–30
- Yokomori T (2000) YAC: yet another computation model of self-assembly. In: Winfree E, Gifford DK (eds) DNA based computers V. DIMACS series in discrete mathematics and theoretical computer science, vol 54. American Mathematical Society, Providence, RI, pp 155–169
- Yokomori T, Kobayashi S (1999) DNA-EC: a model of DNA computing based on equality checking. DNA based computers III. DIMACS series in discrete mathematics and theoretical computer science, vol 48. American Mathematical Society, Providence, RI, pp 347–360
- Yokomori T, Kobayashi S, Ferretti C (1997) On the power of circular splicing systems and DNA computability. In: Proceedings of IEEE international conference on evolutionary computation, Indianapolis, IN, April 1997, IEEE Service Center, pp 219–224
- Yokomori T, Sakakibara Y, Kobayashi S (2002) A magic pot: self-assembly computation revisited. In: Brauer W, Ehrig H, Karhumaki J, Salomaa A (eds) Formal and natural computing. Lecture notes in computer science, vol 2300. Springer, Berlin, pp 418–429
- Yurke B, Mills AP Jr (2003) Using DNA to power nanostructures. *Genet Programming Evolvable Mach* 4:111–122
- Zhang DY, Turberfield AJ, Yurke B, Winfree E (Nov 2007) Engineering entropy-driven reactions and networks catalyzed by DNA. *Science* 318(5853):1121–1125

35 DNA Computing by Splicing and by Insertion–Deletion

Gheorghe Păun

Institute of Mathematics of the Romanian Academy, Bucharest, Romania

Department of Computer Science and Artificial Intelligence,

University of Seville, Spain

gpaun@us.es

george.paun@imar.ro

1	<i>Introduction</i>	1186
2	<i>Some Prerequisites</i>	1186
3	<i>Computing by Splicing</i>	1187
4	<i>Computing by Insertion–Deletion</i>	1195
5	<i>Concluding Remarks</i>	1200

Abstract

This chapter is devoted to two of the most developed theoretical computing models inspired by DNA biochemistry, computing by splicing (a formal operation with strings that models the recombination of DNA molecules under the influence of restriction enzymes and ligase) and by insertion–deletion. Only basic ideas and results are presented, as well as a comprehensive – although not complete – list of titles where further information can be found.

1 Introduction

DNA computing was mainly developed after the 1994 Adleman experiment of solving a small instance of the Hamiltonian path problem in a test tube, making use of standard lab operations, but from a theoretical point of view, one may say that everything started much earlier. For instance, as early as in 1987, Head introduced the so-called splicing operation (Head 1987), an operation with strings that models the recombination of DNA molecules cut by restriction enzymes and pasted together by means of ligase. This initiated a powerful research direction in computability (mainly formal language and automata theory, see Păun et al. (1998), but recently also in terms of complexity, see, e.g., Loos et al. (2008) and Loos and Ogihsara (2007)), *DNA computing by splicing*. Actually, the “prehistory” of DNA computing can be pushed further, for instance, considering the operations of inserting and of deleting strings, considered in linguistics and formal languages well before – only Galiukschov (1981) and Marcus (1969) are mentioned – and whose study was continued in the new framework of handling DNA molecules.

In what follows, both of these directions of research in theoretical DNA computing, computing by splicing and computing by insertion–deletion, will be briefly overviewed. Only basic notions are given; a few typical proofs (mainly proof ideas) and a series of results will be recalled – without attempting to be complete from this last point of view, because, on the one hand, there exists a large number of results and, on the other hand, the research in these areas is still active, so that the existing open problems are under continuous examination and there is continuous progress. In particular, only a few basic references are provided and the reader can refer to Head et al. (1997), Jonoska et al. (2004), and Păun et al. (1998, 2009) for further details and references; many PhD theses devoted to (theoretical) DNA computing can also be useful in this respect – some of them are mentioned in the bibliography of this chapter.

2 Some Prerequisites

It is assumed that the reader is familiar with elementary facts from language and automata theory, so that only a few notions and notations are recalled here. If necessary, any of the many monographs in this area may be consulted. Only Harrison (1978) and Salomaa (1973) and the handbook by Rozenberg and Salomaa (1997) are mentioned as sources of comprehensive information.

For an alphabet V , V^* is the set of all strings over V , the empty string, denoted by λ , included. The length of $x \in V^*$ is denoted by $|x|$. A mapping $h : V \rightarrow U^*$ extended to $h : V^* \rightarrow U^*$ by $h(\lambda) = \lambda$ and $h(uv) = h(u)h(v)$ for all $u, v \in V^*$ is called a *morphism*. A morphism $h : V^* \rightarrow U^*$ is called a *coding* if $h(a) \in U$ for all $a \in V$ and a *weak coding* if

$h(a) \in U \cup \{\lambda\}$. A weak coding $h: V \rightarrow U$ with $U \subseteq V$ such that $h(a) = a$ for $a \in U$, and $h(a) = \lambda$ for $a \in V - U$ is called a *projection* (on U).

A Chomsky grammar (also called type-0 grammar) is a quadruple $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, and P is the set of rewriting rules. If these rules are of the following forms,

- $AB \rightarrow CD$, where $A, B, C, D \in N$ (type 1: context-sensitive rules)
- $A \rightarrow BC$, where $A, B, C \in N$ (type 2: context-free rules)
- $A \rightarrow a$, where $A \in N$ and $a \in T \cup \{\lambda\}$ (type 3: terminal rules; they may be erasing rules)

then the grammar is in the *Kuroda normal form*.

A type-0 grammar $G = (\{S, A, B, C, D\}, T, S, P)$ with P containing rules of the forms $S \rightarrow uSv$, $S \rightarrow x$, with $u, v, x \in (T \cup \{A, B, C, D\})^*$, as well as the non-context-free rules $AB \rightarrow \lambda$, $CD \rightarrow \lambda$, is said to be in the *Geffert normal form*.

The families of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages are denoted by *FIN*, *REG*, *LIN*, *CF*, *CS*, and *RE*, respectively.

A family of languages closed under union, concatenation, λ -free morphisms, inverse morphisms, intersection with regular languages, and Kleene closure is called an AFL (abstract family of languages); an AFL closed under arbitrary morphisms is called a full-AFL.

The Dyck language over an alphabet $V_n = \{a_i, b_i \mid 1 \leq i \leq n\}$ is the language generated by the context-free grammar $G_n = (\{S\}, V_n, S, P_n)$ with the following rules

$$P_n = \{S \rightarrow SS, S \rightarrow \lambda\} \cup \{S \rightarrow a_i S b_i \mid 1 \leq i \leq n\}$$

Every context-free language L can be written in the form $L = h(D_n \cap R)$, where h is a projection, D_n , $n \geq 1$, is a Dyck language, and R is a regular language. This result is known as the *Chomsky–Schützenberger characterization of context-free languages*.

A deterministic finite automaton is a construct $M = (K, V, q_0, F, \delta)$, where K is the finite set of states, V is the alphabet of the automaton, q_0 is the initial state, $F \subseteq K$ is the set of terminal states, and $\delta: K \times V \rightarrow K$ is the next-state mapping. Finite automata characterize the family, *REG*, of languages generated by Chomsky regular grammars.

3 Computing by Splicing

As mentioned in [Sect. 1](#), the splicing operation was introduced in Head ([1987](#)), as an abstraction of a biochemical operation (actually, a sequence of cut- and paste-operations) with DNA molecules. Here this abstraction process is not recalled, but we refer to the seminal paper and to Păun et al. ([1998](#)) for details. The operation proposed in Head ([1987](#)) was further generalized in Păun ([1996a](#)), formulated as a “pure” operation with strings and formal languages, and it was then mainly considered in this form in the subsequent mathematical developments. In particular, it was this version of the splicing operation that was used for defining a computing model, called an *H system* in Păun et al. ([1996](#)), with H coming from the name of the originator of this approach, Tom Head. Later, further variations were introduced, for example, in Pixton ([2000](#)) and Loos ([2006](#)). In what follows, the DNA computing is presented by splicing based on the operation as defined in Păun ([1996](#)), hence in terms of strings and languages, abstracting away such important features of the DNA molecules, always present in the background of the theoretical developments, such as the use of only four letters (representing nucleotides), the double stranded structure, the Watson–Crick

complementarity, the bidirectionality, the palindromicity of the restriction sites of many enzymes, and so on. The use of the other variants of splicing, in particular, comparisons between them, can be found in a series of papers; some of them are also included in the bibliography of this chapter.

3.1 The Splicing Operation

One can consider an arbitrary finite alphabet V . A splicing rule over V is a string $u_1\#u_2\$u_3\#u_4$, where u_1, u_2, u_3, u_4 are strings over V and $\#, \$$ are special symbols not in V . (The idea is that $(u_1, u_2), (u_3, u_4)$ represent the restriction sites of two enzymes, which cut DNA molecules in such a way that they produce sticky ends which match, hence the fragments produced by these enzymes can be recombined; the first enzyme cuts molecules – strings here – in between u_1 and u_2 , and the second enzyme cuts in between u_3 and u_4 .)

For a splicing rule $r = u_1\#u_2\$u_3\#u_4$ and four strings x, y, z, w over V , we write

$$\begin{aligned} (x, y) \models_r (z, w) &\text{ iff } x = x_1 u_1 u_2 x_2, \quad y = y_1 u_3 u_4 y_2 \\ &z = x_1 u_1 u_4 y_2, \quad w = y_1 u_3 u_2 x_2 \\ &\text{for some } x_1, x_2, y_1, y_2 \in V^* \end{aligned}$$

We say that we *splice* x, y at the sites u_1u_2, u_3u_4 , respectively, and the result consists of the strings z, w . This is called the 2-splicing, because both strings obtained by recombination are taken as the result of the operation. From a mathematical point of view, in many contexts it is sufficient/more elegant to consider only the first string, z , as the result of the splicing – and this is called the 1-splicing. The idea is that in most cases, we work with a set of splicing rules and then, together with the rule $u_1\#u_2\$u_3\#u_4$ we can also consider the symmetric rule, $u_3\#u_4\$u_1\#u_2$, and, by splicing the same strings x, y , the first rule produces the first result, z , and the second one produces the second result, w . In what follows it will be specified which of the two operations is used. Most of the results (for instance, the regularity and the universality results) are similar for the two versions of the splicing, but there are cases where a difference can be found – see, for example, Verlan and Zizza (2003).

In particular, we use \vdash to denote the 1-splicing and \models for the 2-splicing operation. When only “splicing” is referred to, it will be clear from the context which type of splicing is meant.

These operations can be extended to languages in a natural way. One can first introduce the notion of an *H scheme*, as a pair $\sigma = (V, R)$, where V is an alphabet and $R \subseteq V^* \# V^* \$ V^* \# V^*$ is a set of splicing rules. Note that in this formulation R is a language, hence we can consider its complexity with respect to various language hierarchies. For instance, if $R \in FL$, for a given family of languages, FL , then we say that the H scheme σ is of *FL type*.

Now, for a given H scheme $\sigma = (V, R)$ and a language $L \subseteq V^*$, we define

$$\sigma_1(L) = \{z \in V^* \mid (x, y) \models_r z, \text{ for some } x, y \in L, r \in R\}$$

and then, for two families FL_1, FL_2 of languages, we denote

$$S_1(FL_1, FL_2) = \{\sigma_1(L) \mid L \in FL_1 \text{ and } \sigma = (V, R) \text{ with } R \in FL_2\}$$

This is the one-step splicing operation, extended to languages and performed with respect to sets of splicing rules. The operation can be iterated, in the natural way: for an H scheme $\sigma = (V, R)$ and a language $L \subseteq V^*$ we define

$$\begin{aligned}\sigma_1^0(L) &= L \\ \sigma_1^{i+1}(L) &= \sigma_1^i(L) \cup \sigma_1(\sigma_1^i(L)), \quad i \geq 0 \\ \sigma_1^*(L) &= \bigcup_{i \geq 0} \sigma_1^i(L)\end{aligned}$$

Consequently, $\sigma_1^*(L)$ is the closure of L under the splicing with respect to σ , that is, the smallest language L' that contains L , and is closed under the splicing with respect to σ , that is to say, $\sigma_1(L') \subseteq L'$.

For two families of languages, FL_1, FL_2 , we define

$$H_1(FL_1, FL_2) = \{\sigma_1^*(L) \mid L \in FL_1 \text{ and } \sigma = (V, R) \text{ with } R \in FL_2\}$$

Of course, the same definitions can be considered for the 2-splicing and then we obtain the families $S_2(FL_1, FL_2), H_2(FL_1, FL_2)$, respectively.

3.2 The Power of the Splicing Operation

We examine now the size of families $S_1(FL_1, FL_2), H_1(FL_1, FL_2)$, for FL_1, FL_2 one of the following families: *FIN, REG, LIN, CF, CS, RE*.

For the non-iterated splicing, the following theorem synthesizes the best results known in the framework specified above; proofs and references can be found in Păun et al. (1998) and Head et al. (1997).

Theorem 1 *The relations in Table 1 hold, where at the intersection of the row marked with FL_1 with the column marked with FL_2 there appear either the family $S_1(FL_1, FL_2)$, or two families FL_3, FL_4 such that $FL_3 \subset S_1(FL_1, FL_2) \subset FL_4$. These families FL_3, FL_4 are the best possible estimations among the six families considered here.*

In the case of iterated splicing, things are more difficult, especially concerning the characterization of the families $H_1(FL, FIN)$, $FL \in \{FIN, REG\}$. Several proofs of the regularity of the languages in these families were given, some of them rather complex and some of them extended to abstract families of languages.

Table 1

The size of families $S_1(FL_1, FL_2)$

	<i>FIN</i>	<i>REG</i>	<i>LIN</i>	<i>CF</i>	<i>CS</i>	<i>RE</i>
<i>FIN</i>	<i>FIN</i>	<i>FIN</i>	<i>FIN</i>	<i>FIN</i>	<i>FIN</i>	<i>FIN</i>
<i>REG</i>	<i>REG</i>	<i>REG</i>	<i>REG, LIN</i>	<i>REG, CF</i>	<i>REG, RE</i>	<i>REG, RE</i>
<i>LIN</i>	<i>LIN, CF</i>	<i>LIN, CF</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>CF</i>	<i>CF</i>	<i>CF</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>CS</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>

Proofs of the following result, called in Păun et al. (1998) *the Regularity Preserving Lemma*, can be found in Culik II and Harju (1991), Pixton (1996) (recalled in Păun et al. (1998)), and Manca (2000):

Lemma 1 $H_1(\text{REG}, \text{FIN}) \subseteq \text{REG}$.

A stronger result has been presented in Pixton (2000); the proof is recalled in Head et al. (1997).

Lemma 2 *If FL is a full AFL, then $H_1(\text{FL}, \text{FIN}) \subseteq \text{FL}$.*

Thus, when using a finite set of rules (remember: this corresponds to a finite set of restriction enzymes), we cannot compute too much, we remain inside the computing competence of finite automata. Much more can be computed if we use a set of rules of the next level in the hierarchy of language families considered above, that is, a regular set. This result is presented, called in Păun et al. (1998) *the Basic Universality Lemma*, with a proof, mainly because the proof idea, the so-called rotate-and-simulate technique, is useful in many related contexts.

Lemma 3 *Every language $L \in \text{RE}$, $L \subseteq T^*$, can be written in the form $L = L' \cap T^*$ for some $L' \in H_1(\text{FIN}, \text{REG})$.*

Proof Consider a Chomsky type-0 grammar $G = (N, T, S, P)$, denote $U = N \cup T \cup \{B\}$, where B is a new symbol, and construct the H scheme $\sigma = (V, R)$, where

$$V = N \cup T \cup \{X, X', B, Y, Z\} \cup \{Y_\alpha \mid \alpha \in U\}$$

and R contains the following groups of rules:

- Simulate* : 1. $Xw\#uY\$Z\#vY$, for $u \rightarrow v \in P, w \in U^*$
- Rotate* : 2. $Xw\#\alpha Y\$Z\#Y_\alpha$, for $\alpha \in U, w \in U^*$
- 3. $X'\alpha\#Z\$X\#wY_\alpha$, for $\alpha \in U, w \in U^*$
- 4. $X'w\#Y_\alpha\$Z\#Y$, for $\alpha \in U, w \in U^*$
- 5. $X\#Z\$X'\#wY$, for $w \in U^*$
- Terminate* : 6. $\#ZY\$XB\#wY$, for $w \in T^*$
- 7. $\#Y\$XZ\#$

Consider also the language

$$\begin{aligned} L_0 = & \{XBSY, ZY, XZ\} \cup \{ZvY \mid u \rightarrow v \in P\} \\ & \cup \{ZY_\alpha, X'\alpha Z \mid \alpha \in U\} \end{aligned}$$

We obtain $L = \sigma_1^*(L_0) \cap T^*$.

Indeed, one can examine the work of σ , namely, the possibilities to obtain a string in T^* .

No string in L_0 is in T^* . All rules in R involve a string containing the symbol Z , but this symbol will not appear in the string produced by splicing. Therefore, at each step, we have to use a string in L_0 and, excepting the case of using the string $XBSY$ in L_0 , a string produced at a previous step.

The symbol B is a marker for the beginning of the sentential forms of G simulated by σ .

By rules in group 1 we can simulate the rules in P . Rules in groups 2–5 move symbols from the right-hand end of the current string to the left-hand end, thus making possible the simulation of rules in P at the right-hand end of the string produced by σ . However, because B is always present and marks the place where the string of G begins, it is known in each moment which is that string. Namely, if the current string in σ is of the form $\beta_1 w_1 B w_2 \beta_2$, for some β_1, β_2 markers of types X, X', Y, Y_α with $\alpha \in U$, and $w_1, w_2 \in (N \cup T)^*$, then $w_2 w_1$ is a sentential form of G .

We start from $XBSY$, hence from the axiom of G , marked to the left hand with B and bracketed by X, Y .

Let us see how the rules 2–5 work. Take a string $Xw\alpha Y$, for some $\alpha \in U, w \in U^*$. By a rule of type 2 we get

$$(Xw|\alpha Y, Z|Y_\alpha) \vdash XwY_\alpha$$

The symbol Y_α memorizes the fact that α has been erased from the right-hand end of $w\alpha$. No rule in R can be applied to XwY_α excepting the rules of type 3:

$$(X'\alpha|Z, X|wY_\alpha) \vdash X'\alpha wY_\alpha$$

Note that the same symbol α removed at the previous step is now added in the front of w . Again, we have only one way to continue, namely, by using a rule of type 4. We get

$$(X'\alpha w|Y_\alpha, Z|Y) \vdash X'\alpha wY$$

If we now use a rule of type 7, removing Y , then X' (and B) can never be removed, and hence the string cannot be turned to a terminal one. A rule of type 5 must be used:

$$(X|Z, X'|\alpha wY) \vdash X\alpha wY$$

We have started from $Xw\alpha Y$ and obtained $X\alpha wY$, a string with the same end markers. These steps can be iterated as long as we want, so any circular permutation of the string between X and Y can be produced. Moreover, what we obtain are exactly the circular permutations and nothing more (for instance, at every step, there will still be one and only one occurrence of B).

To every string XwY one can also apply a rule of type 1, providing w ends with the left-hand member of a rule in P . Any rule of P can be simulated in this way, at any place we want in the corresponding sentential form of G , by preparing the string as above, using rules in groups 2–5.

Consequently, for every sentential form w of G there is a string $XBwY$, produced by σ , and, conversely, if $Xw_1 B w_2 Y$ is produced by σ , then $w_2 w_1$ is a sentential form of G .

The only way to remove the symbols not in T from the strings produced by σ is by using rules in groups 6, 7. More precisely, the symbols XB can only be removed in the following conditions: (1) Y is present (hence the work is blocked if one uses first rule 7, removing Y : the string cannot participate in any further splicing, and it is not terminal), (2) besides one occurrence of B , the current string bracketed by X, Y consists of terminal symbols only, and (3) the symbol B is adjacent to X , in the right hand of it. After removing X and B , one can remove Y , too, and what one obtains is a string in T^* . From the previous discussion, it is clear that such a string is in $L(G)$, hence $\sigma_1^*(L_0) \cap T^* \subseteq L(G)$. Conversely, each string in $L(G)$ can be produced in this way, hence $L(G) \subseteq \sigma_1^*(L_0) \cap T^*$. We have the equality $L(G) = \sigma_1^*(L_0) \cap T^*$, which completes the proof.

Table 2The size of families $H_1(FL_1, FL_2)$

	<i>FIN</i>	<i>REG</i>	<i>LIN</i>	<i>CF</i>	<i>CS</i>	<i>RE</i>
<i>FIN</i>	<i>FIN, REG</i>	<i>FIN, RE</i>				
<i>REG</i>	<i>REG</i>	<i>REG, RE</i>				
<i>LIN</i>	<i>LIN, CF</i>	<i>LIN, RE</i>				
<i>CF</i>	<i>CF</i>	<i>CF, RE</i>				
<i>CS</i>	<i>CS, RE</i>	<i>CS, RE</i>	<i>CS, RE</i>	<i>CS, RE</i>	<i>CS, RE</i>	<i>CS, RE</i>
<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>

We also recall, without a proof, a result concerning the limitations of families $H_1(FL_1, FL_2)$.

Lemma 4 *Let FL be a family of languages closed under intersection with regular languages and restricted morphisms. For every $L \subseteq V^*$, $L \notin FL$, and $c, d \notin V$, the language $L' = (dc)^*L(dc)^* \cup c(dc)^*L(dc)^*d$ does not belong to the family $H_1(FL, RE)$.*

Based on the previous results, we get the following synthesis theorem.

Theorem 2 *The relations in [Table 2](#) hold, where at the intersection of the row marked with FL_1 with the column marked with FL_2 there appear either the family $H_1(FL_1, FL_2)$, or two families FL_3, FL_4 such that $FL_3 \subset H_1(FL_1, FL_2) \subset FL_4$. These families FL_3, FL_4 are the best possible estimations among the six families considered here.*

3.3 Extended Splicing Systems

In the sections above the power of splicing operations is considered without explicitly introducing a computing model. This can be done now, by defining the main device investigated in DNA computing by splicing, the H systems; these devices are presented directly in the extended version, with a terminal alphabet used for selecting the result of computations out of all results of splicing operations.

An *extended H system* is a quadruple $\gamma = (V, T, A, R)$, where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$, and $R \subseteq V^* \# V^* \$ V^* \# V^*$, where $\#, \$$ are special symbols not in V .

We call V the alphabet of γ , T is the *terminal alphabet*, A is the set of *axioms*, and R is the set of splicing rules. Therefore, we have an *underlying H scheme*, $\sigma = (V, R)$, augmented with a given subset of V and a set of axioms. When $T = V$ we say that γ is a *non-extended H system*.

The *language generated* by γ is defined by $L(\gamma) = \sigma_1^*(A) \cap T^*$, where σ is the underlying H scheme of γ .

For two families of languages, FL_1, FL_2 , we denote by $EH_1(FL_1, FL_2)$ the family of languages $L(\gamma)$ generated by extended H systems $\gamma = (V, T, A, R)$, with $A \in FL_1, R \in FL_2$.

A number of the results from the previous section can be reformulated in terms of extended H systems. Moreover, we have the following inclusion (we recall its simple proof from Păun et al. (1998) in order to have an example of an extended H system).

Lemma 5 $REG \subseteq EH_1(FIN, FIN)$.

Proof Consider a regular grammar $G = (N, T, S, P)$ (hence with rules in P of the forms $A \rightarrow aB, A \rightarrow a$, for $A, B \in N$ and $a \in T$). We construct the H system

$$\begin{aligned}\gamma &= (N \cup T \cup \{Z\}, T, A_1 \cup A_2 \cup A_3, R_1 \cup R_2) \\ A_1 &= \{S\} \\ A_2 &= \{ZaY \mid X \rightarrow aY \in P, X \in N, a \in T\} \\ A_3 &= \{ZZa \mid X \rightarrow a \in P, X \in N, a \in T\} \\ R_1 &= \{\#X\$Z\#aY \mid X \rightarrow aY \in P, X \in N, a \in T\} \\ R_2 &= \{\#X\$ZZ\#a \mid X \rightarrow a \in P, X \in N, a \in T\}\end{aligned}$$

The work of γ can be now examined. If a string ZxX is spliced, possibly one from A_2 (this is the case if $x = c \in T$ and $U \rightarrow cX \in P$) using a rule in R_1 , then we get a string of the form $ZxaY$. The symbol Z cannot be eliminated, hence no terminal string can be obtained if one continues to use the resulting string as the first term of a splicing. On the other hand, a string ZxX with $|x| \geq 2$ cannot be used as the second term of a splicing. Thus, the only way to obtain a terminal string is to start from S , to use splicings with respect to rules in R_1 an arbitrary number of times, and to end with a rule in R_2 . Always the first term of a splicing is that obtained by a previous splicing and the second one is from A_2 or from A_3 (at the last step). This corresponds to a derivation in G , hence we have $L(\gamma) = L(G)$.

Combining now the previous results, we get the following theorem.

Theorem 3 *The relations in \bullet Table 3 hold, where at the intersection of the row marked with FL_1 with the column marked with FL_2 there appear either the family $EH_1(FL_1, FL_2)$, or two families FL_3, FL_4 such that $FL_3 \subset EH_1(FL_1, FL_2) \subseteq FL_4$. These families FL_3, FL_4 are the best possible estimations among the six families considered here.*

It is worth noting that the only family that is not equal to a family in the Chomsky hierarchy is $EH_1(LIN, FIN)$.

Two of the relations summarized in \bullet Table 3 are central for the DNA computing based on splicing, $EH_1(FIN, FIN) = REG$ and $EH_1(FIN, REG) = RE$.

Therefore, using a finite extended H system, that is, a system with a finite set of axioms and a finite set of splicing rules, we only obtain the computing power of finite automata.

Table 3

The generative power of extended H systems

	<i>FIN</i>	<i>REG</i>	<i>LIN</i>	<i>CF</i>	<i>CS</i>	<i>RE</i>
<i>FIN</i>	<i>REG</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>REG</i>	<i>REG</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>LIN</i>	<i>LIN, CF</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>CF</i>	<i>CF</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>CS</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>
<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>	<i>RE</i>

However, working with a regular set of rules that is not finite is not of much interest from a practical point of view since we cannot physically realize an infinite “computer” (in this case, an infinite number of restriction enzymes).

3.4 Universal H Systems

The results in [Table 3](#) hold also for 2-splicing – which is important from a “practical” point of view, as in a test tube both results of a splicing operation are equally possible. From a “practical” point of view, the difficulty suggested by the equalities $EH_2(FIN, FIN) = REG$, $EH_2(FIN, REG) = RE$ can also be avoided, and the idea is to use a finite set of rules, but not to apply them freely, as in a usual extended H system, but impose some control on the strings that can be handled by a specific rule. In this way, characterizations of the computing power of Turing machines are obtained by means of finite H systems. The proofs are constructive, in most cases based on the rotate-and-simulate technique from the proof of Basic Universality Lemma, hence, starting from a universal type-0 grammar, we get a universal H system (a system that simulates any particular H system after introducing a coding of this particular system in the axioms of the universal one). That is why *universality results* are spoken about even when explicitly dealing with results which, mathematically speaking, provide characterizations of Turing computability (of RE languages).

There are many controls that can be imposed to a finite extended H system in order to increase its computing power from the power of finite automata to the power of Turing machines. (This shows that the splicing operation needs very little additional power – more technically: context-sensitivity – in order to reach the power of Turing machines.)

Several of these controls are mentioned here without entering into technical details: (1) permitting contexts (each rule has associated to it some symbols and the rule can be applied to splice two strings only if these strings contain the respective symbols), (2) forbidding contexts (as above, but the associated symbols should not be present in the spliced strings), (3) target languages, with two possibilities, local targets and a global target (in the first case, a language is associated with each rule and the results of a splicing operation must be a string in the language associated with the used rule; in the global case, the target languages associated with different rules are identical), (4) programmed H systems (a next rule mapping is considered, which controls the sequence of rules used), (5) double splicing (similar to the previous restriction, but only couples of consecutive rules are specified; the strings resulting after applying the first rule in a couple are spliced by the second rule), (6) using multisets (counting the copies of strings present in any step of the computation). Also considered were other computing devices based on splicing, using suggestions from the regulated rewriting area (Dassow and Păun 1989), from grammar systems area (Csuhaj-Varju et al. 1994), or from membrane computing (Păun et al. 2009) (the handbook (Păun et al. 2009) contains a comprehensive chapter devoted to membrane systems using string objects processed by splicing). Details and source references for all these ways to get universality by means of controlled finite H systems can be found in Păun et al. (1998).

Therefore, universal computing devices based on the splicing operation can be constructed – at the theoretical level. Implementing such “computers” is, of course, another story. From the computing theory point of view, important insights have been obtained; the universality itself, in any of the cases mentioned above, shows that the computability can be

“reconstructed” on the basis of an operation, splicing, which is much different from the usual operation of rewriting (replacing a short substring of a longer string) that is used in most, if not all, classic computing models: Turing machines, Chomsky grammars, Thue and Post systems, and Markov algorithms. This is significant, for instance, in view of the question “what does it mean to compute in a natural way?”: at the level of DNA, nature mainly “computes” by means of splicing. (In the next section, it will also be seen that the insertion–deletion operation can lead to universality, but such operations are not so far from rewriting.)

3.5 Further Developments

There are many other directions of research and many results related to the splicing operation. One idea is to have variations of the basic operations, starting with those considered in Head (1987) and Pixton (2000) and continuing with many others, either closer to the biological reality or motivated mathematically. A well-investigated class of H systems is that of simple H systems, whose rules are of the form $r_a = a\$a\#$ or $r'_a = \#a\$a$, where a is a symbol. Then, we can consider multiple splicing (crossover) taking place in each step, or we can pass to circular strings; this latter case is both well motivated and interesting and still raises a series of problems that were not solved. Indeed, as we mentioned above, there are open problems also related to the relationship between regular languages and languages in $H_1(FIN, FIN)$ (characterization, decidability, etc.). The splicing was also considered for other data structures than strings, such as trees, arbitrary graphs, arrays of various forms, and even for multisets of symbols.

Furthermore, the architecture of the computing device based on splicing can be of different forms, not only H systems, which directly correspond to formal grammars. Grammar systems and membrane systems (P systems) based on splicing are mentioned above; for instance, test-tube systems, time-varying H systems, two-level systems, and more recently, networks of language processors using splicing were also considered.

On the other hand, many other problems than the computing power were investigated. Descriptive complexity was considered from the very beginning: number of axioms and of rules, length of axioms and size of rules (the maximal length of the strings u_1, u_2, u_3, u_4 in splicing rules, or the vector of lengths of the four strings). Recently, also complexity classes based on H systems were introduced. Universality was proved since the very beginning, but also concrete universal H systems were produced. Splicing was related to Schützenberger recombination of strings by means of *constants* and to other combinatorics on word notions.

Of course, splicing was tested also in laboratory, but this is beyond the scope of this chapter.

For all these topics and for further ones the reader can find information in various titles listed in the bibliography.

4 Computing by Insertion–Deletion

As in the case of splicing, details are not recalled about the way insertion and deletion operations are realized in terms of DNA biochemistry, but a formal language presentation is followed, directly introducing the basic notions related to this approach and then recalling some of the results known in this area. As above, only a sample proof will be sketched.

4.1 Insertion and Deletion Operations

An *ins-del system* is a construct $\gamma = (V, T, A, I, D)$, where V is an alphabet, $T \subseteq V$ (terminal symbols), A is a finite language over V (axioms), and I, D are finite sets of triples of the form (u, α, v) , where u, v, α are strings over V ; the triples in I are called insertion rules, those in D are called deletion rules, and they are used as follows.

For $x, y \in V^*$ we define

$$\begin{aligned} x \Rightarrow_{ins} y &\text{ iff } x = x_1 u v x_2, y = x_1 u w v x_2 \text{ for some } (u, w, v) \in I, x_1, x_2 \in V^* \\ x \Rightarrow_{del} y &\text{ iff } x = x_1 u w v x_2, y = x_1 u v x_2 \text{ for some } (u, w, v) \in D, x_1, x_2 \in V^* \end{aligned}$$

The reflexive and transitive closure of \Rightarrow_α , $\alpha \in \{\text{ins}, \text{del}\}$ is denoted by \Rightarrow_α^* . When this is clear from the context, the subscript *ins* or *del* is omitted.

For a system γ as above we define the language

$$L(\gamma) = \{w \in T^* \mid z \Rightarrow^* w, \quad z \in A\}$$

Thus, we start from an axiom of γ , proceed through a finite number of insertion and deletion operations, and we select the strings consisting of terminal symbols – very similar to the way the language generated by a grammar is defined. Actually, there are papers where an ins-del system has the rules presented in a rewriting style: $(u, \lambda/\alpha, v)$ or even $uv \rightarrow u\alpha v$ for insertion rules and $(u, \alpha/\lambda, v)$ or even $u\alpha v \rightarrow uv$ for deletion rules.

The complexity of an ins-del system is evaluated in terms of the complexity of its rules, that is, the length of strings used as contexts and the length of inserted or deleted strings. More precisely, an ins-del system $\gamma = (V, T, A, I, D)$ is of weight $(n, m; p, q)$ if

$$\begin{aligned} n &= \max\{|\alpha| \mid (u, \alpha, v) \in I\} \\ m &= \max\{|u| \mid (u, \alpha, v) \in I \text{ or } (v, \alpha, u) \in I\} \\ p &= \max\{|\alpha| \mid (u, \alpha, v) \in D\} \\ q &= \max\{|u| \mid (u, \alpha, v) \in D \text{ or } (v, \alpha, u) \in D\} \end{aligned}$$

We denote by $INS_n^m DEL_p^q$, for $n, m, p, q \geq 0$, the family of languages $L(\gamma)$ generated by ins-del systems of weight $(n', m'; p', q')$ such that $n' \leq n$, $m' \leq m$, $p' \leq p$, $q' \leq q$. When one of the parameters n, m, p, q is not bounded, we replace it by $*$. Thus, the family of all ins-del languages is $INS^* DEL^*$. Because the insertion-deletion of the empty string changes nothing, when $n = 0$ we also suppose that $m = 0$, and when $p = 0$ we also suppose that $q = 0$. The meaning of INS_0^0 is that no insertion rule is used, and the meaning of DEL_0^0 is that no deletion rule is used. When $m = 0$ or $q = 0$, we have a context-free insertion or deletion, respectively; if both these parameters are zero, then we get a context-free ins-del system.

This section will be concluded with a few simple examples.

The ins-del system

$$\gamma = (\{a, b\}, \{a, b\}, \{ab\}, \{(a, ab, b)\}, \emptyset)$$

clearly generates the (linear nonregular) language $L(\gamma) = \{a^n b^n \mid n \geq 1\}$. If the set of insertion rules is replaced with $I = \{(\lambda, ab, \lambda)\}$, then the generated language is the Dyck language over the alphabet $\{a, b\}$, excepting the empty string.

As a more complex ins-del system, the following one from Margenstern et al. (2005) can be recalled:

$$\gamma = (\{S, S', a, b\}, \{a, b\}, \{S\}, \{(\lambda, S' a S b, \lambda), (\lambda, S' a b, \lambda), ((\lambda, S S', \lambda)\})$$

Because of the selection of terminal strings, symbols S, S' should not be present in the strings that form the language $L(\gamma)$; such symbols can be removed only by the deletion rule (λ, SS', λ) , which means that always S' is introduced in the right hand of a symbol S . Initially, S is alone; assume that we have a string w_1Sw_2 (initially, $w_1 = w_2 = \lambda$). We have to use the insertion rule $(\lambda, S'aSb, \lambda)$ such that we get $w_1SS'aSbw_2$, otherwise SS' cannot be removed. In this way, we pass to the string w_1aSbw_2 . This process can be repeated. When using the insertion rule $(\lambda, S'ab, \lambda)$, the obtained string, $w_1SS'abw_2$, can be turned to a terminal one. Consequently, we get $L(\gamma) = \{a^n b^n \mid n \geq 1\}$.

4.2 The Power of Ins-Del Systems

Continuing the previous examples, we recall now a counterpart of [Lemma 5](#) for ins-del systems.

Lemma 6 $REG \subset INS^*_\ast DEL_0^0$.

Proof Let L be a regular language and let $M = (K, V, q_0, F, \delta)$ be the minimal deterministic finite automaton recognizing L .

For each $w \in V^*$, we define the mapping $\rho_w : K \rightarrow K$ by

$$\rho_w(q) = q' \text{ iff } (q, w) \vdash^*(q', \lambda), \quad q, q' \in K$$

Obviously, if $x_1, x_2 \in V^*$ are such that $\rho_{x_1} = \rho_{x_2}$, then for every $u, v \in V^*$, ux_1v is in L if and only if ux_2v is in L .

The set of mappings from K to K is finite. Hence the set of mappings ρ_w as above is finite. Let n_0 be their number. We construct the ins-del system $\gamma = (V, V, A, I, \emptyset)$ with

$$A = \{w \in L \mid |w| \leq n_0 - 1\}$$

$$I = \{(w, v, \lambda) \mid |w| \leq n_0 - 1, 1 \leq |v| \leq n_0, |wv| \leq n_0, \text{ and } \rho_w = \rho_{wv}\}$$

From the definition of mappings ρ_w and the definitions of A, I , it follows immediately that $L(\gamma) \subseteq L$.

Assume that the converse inclusion is not true and let $x \in L - L(\gamma)$ be a string of minimal length with this property. Thus $x \notin A$. Hence $|x| \geq n_0$. Let $x = zz'$ with $|z| = n_0$ and $z' \in V^*$. If $z = a_1a_2 \dots a_{n_0}$, then it has $n_0 + 1$ prefixes, namely, $\lambda, a_1, a_1a_2, \dots, a_1 \dots a_{n_0}$. There are only n_0 different mappings ρ_w . Therefore there are two prefixes u_1, u_2 of z such that $u_1 \neq u_2$ and $\rho_{u_1} = \rho_{u_2}$. With no loss in generality it may be assumed that $|u_1| < |u_2|$. By substituting u_2 by u_1 we obtain a string x' that is also in L . As $|x'| < |x|$ and x was of minimal length in $L - L(\gamma)$, we obtain $x' \in L(\gamma)$. However, $|u_2| - |u_1| \leq |u_2| \leq n_0$, so if $u_2 = u_1u_3$, then (u_1, u_3, λ) is an insertion rule in I . This implies that $x' \Rightarrow_{ins} x$, that is, $x \in L(\gamma)$, a contradiction. In conclusion, $L \subseteq L(\gamma)$.

The strictness of the inclusion is obvious (see, for instance, the first example considered in the end of the previous section).

We recall now a few non-universality results, that is, pointing out classes of ins-del systems, which generate families of languages strictly included in RE (references are omitted for those results that also appear in Păun et al. (1998); the proofs are in general based on simulating type-0 grammars in Kuroda normal form or in Geffert normal form).

Theorem 4 *The following relations hold:*

1. $\text{INS}_*^*\text{DEL}_0^0 \subset \text{CS}$ and $\text{LIN} - \text{INS}_*^*\text{DEL}_0^0 \neq \emptyset$, but $\text{INS}_2^2\text{DEL}_0^0$ contains non-semilinear languages.
2. $\text{INS}_*^1\text{DEL}_0^0 \subset \text{CF}$.
3. $\text{INS}_*^0\text{DEL}_1^0 \subset \text{CF}$, $\text{INS}_2^0\text{DEL}_2^0 \subset \text{CF}$, $\text{INS}_1^0\text{DEL}_*^0 \subset \text{REG}$ (Verlan 2007).
4. $\text{REG} \subset \text{INS}_*^*\text{DEL}_0^0$ and each regular language is the coding of a language in the family $\text{INS}_*^1\text{DEL}_0^0$.

There are however a lot of universality results in this area (characterizations of RE languages). Some of them are recalled here.

Theorem 5 *We have $\text{INS}_n^m\text{DEL}_p^q = \text{RE}$ for all the following quadruples (n,m,p,q) (for each of these quadruples we also mention the place where a proof can be found, omitting again the cases whose proof can be found in Păun et al. (1998)): (i) $(1,2;1,1)$, (ii) $(1,2;2,0)$, (iii) $(2,1;2,0)$, (iv) $(1,1;1,2)$ – Tanaka and Yokomori (2003), (v) $(2,1;1,1)$ – Tanaka and Yokomori (2003), (vi) $(2,0;3,0)$ – Margenstern et al. (2005), (vii) $(3,0;2,0)$ – Margenstern et al. (2005), (viii) $(1,1;2,0)$, (ix) $(1,1;1,1)$ – Tanaka and Yokomori (2003).*

There are also a series of characterizations of RE languages by means of operations with languages starting from languages generated by ins-del systems. Only one of them is recalled here, together with some details of its proof, namely, the central result from Păun et al. (2008), which gives a Chomsky–Schützenberger-like characterization for RE languages (note that only insertion rules are used).

Theorem 6 *Each language $L \in \text{RE}$ can be represented in the form $L = h(L' \cap D)$, where $L' \in \text{INS}_3^0\text{DEL}_0^0$, h is a projection, and D is a Dyck language.*

Proof (Sketch) Consider a language $L \subseteq T^*$, generated by a type-0 grammar $G = (N, T, S, P)$ in Kuroda normal form. Assume that the rules of P are labeled in a one-to-one manner with elements of a set $\text{Lab}(P)$.

We construct an ins-del system $\gamma = (V \cup \overline{V}, V \cup \overline{V}, \{S\}, I, \emptyset)$, of weight $(3, 0; 0, 0)$, with

$$V = N \cup T \cup \text{Lab}(P)$$

and with I containing the following insertion rules.

- Group 1: For each rule $r : AB \rightarrow CD$ of type 1 in P we construct the following two insertion rules: (λ, CDr, λ) and $(\lambda, \overline{B}\overline{A}\overline{r}, \lambda)$
- Group 2: For each rule $r : A \rightarrow BC$ of type 2 in P we construct the following two insertion rules: (λ, BCr, λ) and $(\lambda, \overline{A}\overline{r}, \lambda)$
- Group 3: For each rule $r : A \rightarrow a$ of type 3 in P we construct the following two insertion rules: $(\lambda, a\overline{a}r, \lambda)$ and $(\lambda, \overline{A}\overline{r}, \lambda)$, where $\overline{\lambda} = \lambda$

For a rule $r : u \rightarrow v$ in P we say that two rules (λ, vr, λ) and $(\lambda, \overline{u}^R\overline{r}, \lambda)$ in P' are r -complementary, and denote their labels by r_+ and r_- , respectively, (x^R is the reversal of the string x).

We define the projection $h : (V \cup \overline{V})^* \rightarrow T^*$ by $h(a) = a$ for all $a \in T$, and $h(a) = \lambda$ otherwise. Let D be the Dyck language over V , taking as pairs (a, \overline{a}) for $a \in V$.

We have $L(G) = h(L(\gamma) \cap D)$. To begin with some useful notions are introduced.

For any rule $r : u \rightarrow v \in P$, let $U_r(u) = ru\overline{u}^R\overline{r}$; we call this an r -block. Then, this notion is extended to define *U-structures* as follows:

1. An r -block $U_r(u)$ is a U-structure.
2. If U_1 and U_2 are U-structures, then $U_1 U_2$ is a U-structure.
3. Let α_i , $i = 1, 2, 3$, be U-structures or empty, with at least one α_i being non-empty; consider a string of the form $r\alpha_1 u_1 \alpha_2 u_2 \alpha_3 \overline{u}^R \overline{r}$, where $u = u_1 u_2$ is such that $r : u \rightarrow v \in P$. Then, this string, denoted by $U_r(\alpha_1 u_1 \alpha_2 u_2 \alpha_3)$, is a U-structure.
4. Nothing else is a U-structure.

Let us now define a mapping ϕ over $(V \cup \overline{V})^*$ as follows: For any $a \in V - T$, let $a\overline{a} \sim \lambda$, and for any $a \in T$, let $a\overline{a} \sim a$. Then, one can consider a reduction operation over $(V \cup \overline{V})^*$ by iteratively using the binary relation \sim . We define $\phi(w)$ as the string finally obtained as the *irreducible* string in terms of this reduction operation. (Because the symbols from T and from $V - T$ are subject of different “reduction rules,” the irreducible string reached when starting from a given string is unique, hence the mapping ϕ is correctly defined.)

The following lemma holds:

Lemma 7 *Let $S \Rightarrow^{n-1} z_{n-1} (= \alpha u \beta) \Rightarrow z_n (= \alpha v \beta)$ in G , where $r : u \rightarrow v$ is used in the last step. Then, there exists a derivation of γ such that $S \Rightarrow^{2n} \tilde{z}_n$ and $\phi(\tilde{z}_n) = z_n$.*

The following observations are useful for the proof of the inclusion $h(L(\gamma) \cap D) \subseteq L(G)$.

Observation 1 For a rule $r : u \rightarrow v$ in P , let $r_+ : (\lambda, vr, \lambda)$ and $r_- : (\lambda, \overline{u}^R\overline{r}, \lambda)$ be the two r -complementary rules.

1. Any successful derivation of γ requires the use of both r -complementary rules r_+ and r_- .
2. Let \tilde{z} be any sentential form in a derivation of γ that eventually leads to a string in D (we say that such a derivation is successful). Then, it must hold that for any prefix α of \tilde{z} , $\#_{vr}(\alpha) \geq \#_{\overline{u}^R\overline{r}}(\alpha)$, where $\#_x(\alpha)$ denotes the number of occurrences of a string x in α .
3. Applying insertion rules within a U-structure leads to only invalid strings. Indeed, suppose that r_+ and r_- are applied on some occurrence of u appearing in a U-structure $U_r(\delta_1 u \delta_2) = r'\delta_1 u \delta_2 \overline{u}^R \overline{r}'$, where $r' : u \rightarrow v'$. This derives a string $r'\delta_1 v U_r(u) \delta_2 \overline{u}^R \overline{r}'$ that leads to an invalid string (i.e., *not* in D) unless $u = v$. This also occurs in the case where u appears in separate locations in the U-structure.
4. A location in \tilde{z} is called *valid* for two r -complementary rules if it is either immediately before u_1 for r_+ or immediately after u_2 for r_- , by ignoring U-structures in \tilde{z} , where $u = u_1 u_2$. Then, applying insertion rules at valid locations only leads to valid strings. This is seen as follows: From 1, 2, 3 above, the locations for r_+ and r_- to be used are restricted to somewhere in the left and right, respectively, of u . In order to derive a valid string from \tilde{z} , it is necessary to apply r_+ and r_- to u so that these two rules together with u may eventually lead to forming a U-structure.

Now, the following result holds:

Lemma 8 Let $S \Rightarrow^{2n} \tilde{z}$ in γ and $\phi(\tilde{z}) = z (\in (N \cup T)^*)$. Then, one has $S \Rightarrow^n z$ in G .

The proof of the theorem can be completed now.

For any $w \in L(G)$, consider a derivation $S \Rightarrow^* w$. Then, by [Lemma 7](#) there exists a derivation $S \Rightarrow^* \tilde{w}$ in γ such that $\phi(\tilde{w}) = w$. Since ϕ deletes only U-structures and elements of \overline{T} , this implies that $\tilde{w} \in D$ and $h(\tilde{w}) = w \in T^*$. Thus, $w \in h(L(\gamma) \cap D)$. Hence, we have $L(G) \subseteq h(L(\gamma) \cap D)$.

Conversely, suppose that $S \Rightarrow^*$ in γ and $\phi(\tilde{w}) = w (\in T^*)$. Then, by [Lemma 8](#) we have $S \Rightarrow^* w$ in G . Again, $\phi(\tilde{w}) \in T^*$ implies that $\tilde{w} \in D$ and $h(\tilde{w}) = w$. Thus, we have $h(L(\gamma) \cap D) \subseteq L(G)$.

4.3 Further Remarks

As in the case of computing by splicing, there also are several other developments – and a series of open problems – in the case of computing by insertion–deletion, starting with the linguistically motivated models and results (for instance, related to Marcus contextual grammars), and ending with insertion–deletion operations for data structures other than strings. The bibliography below provides several titles that might be useful to the interested reader.

5 Concluding Remarks

Computing by splicing and by insertion–deletion provides convincing evidence that computer science – more specifically computability theory – has much to learn from the biochemistry of DNA. Whether or not this will prove to be useful to practical computer science is still a matter for further research, but from a theoretical point of view, we stand to gain much insight into computability from reasoning in this framework; in particular, it enables everyone to ask the question: “what does it mean to compute in a *natural way*?“

References

- Bonizzoni P, Mauri G (2005) Regular splicing languages and subclasses. *Theor Comput Sci* 340:349–363
- Bonizzoni P, Mauri G (2006) A decision procedure for reflexive regular splicing languages. In: Proceedings of the developments in language theory '06, Santa Barbara, Lecture notes in computer science, vol. 4036. Springer, Berlin, pp 315–326
- Bonizzoni P, De Felice C, Mauri G, Zizza R (2003) Regular languages generated by reflexive finite splicing systems. In: Proceedings of the developments in language theory '03, Szeged, Hungary, Lecture notes in computer science, vol. 2710. Springer, Berlin, pp 134–145
- Bonizzoni P, De Felice C, Mauri G (2005a) Recombinant DNA, gene splicing as generative devices of formal languages. In: Proceedings of the Computability in Europe '05, Amsterdam, The Netherlands, Lecture notes in computer science, vol. 3536. Springer, Berlin, pp 65–67
- Bonizzoni P, De Felice C, Zizza R (2005b) The structure of reflexive regular splicing languages via Schützenberger constants. *Theor Comput Sci* 334:71–98
- Cavaliere M, Jonoska N, Leupold P (2006) Computing by observing DNA splicing. Technical Report 11/2006, Microsoft Center for Computational Biology, Trento
- Ceterchi R, Subramanian KG (2003) Simple circular splicing systems. *Romanian J Inform Sci Technol* 6:121–134
- Csuha-Jarju E, Dassow J, Kelemen J, Păun Gh (1994) Grammar systems. A grammatical approach to distribution and cooperation. Gordon & Breach, London
- Culik II K, Harju T (1991) Splicing semigroups of dominoes and DNA. *Discrete Appl Math* 31:261–277

- Dassen R, Hoogeboom HJ, van Vugt N (2001) A characterization of non-iterated splicing with regular rules. In: Martin-Vide C, Mitrana V (eds) Where mathematics, computer science, linguistics and biology meet. Kluwer, Dordrecht, pp 319–327
- Dassow J, Păun Gh (1989) Regulated rewriting in formal language theory. Springer, Berlin
- Dassow J, Vasile G (2004) Multiset splicing systems. *BioSystems* 74:1–7
- De Felice C, Fici G, Zizza R (2007) Marked systems and circular splicing. In: Proceedings of the Fundamentals of Computation theory, Budapest, Hungary, Lecture notes in computer science, vol. 4639. Springer, Berlin, pp 238–249
- Frisco P (2004) Theory of molecular computing. Splicing and membrane systems. Ph.D. thesis, Leiden University, The Netherlands
- Galiukschov BS (1981) Semicontextual grammars (in Russian). Mat logika i mat ling, Tallinn Univ 38–50
- Goode E, Pixton D (2001) Semi-simple splicing systems. In: Martin-Vide C, Mitrana V (eds) Where mathematics, computer science, linguistics and biology meet. Kluwer, Dordrecht, pp 343–352
- Goode E, Pixton D (2007) Recognizing splicing languages: syntactic monoids and simultaneous pumping. *Discrete Appl Math* 155:989–1006
- Harju T, Margenstern M (2005) Splicing systems for universal Turing machines. In: Proceedings of the DNA Computing '04, Milano, Italy, Lecture notes in computer science, vol. 3384. Springer, Berlin, pp 149–158
- Harrison M (1978) Introduction to formal language theory. Addison-Wesley, Reading, MA
- Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bull Math Biol* 49:737–759
- Head T, Păun Gh, Pixton D (1997) Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination. In: Rozenberg G, Salomaa A (eds) Handbook of formal languages, vol. 2. Springer, Berlin, pp 295–360
- Hemalatha S (2007) A study on rewriting P systems, splicing grammar systems and picture array languages. Ph.D. thesis, Anna University, Chennai, India
- Jonoska N, Păun Gh, Rozenberg G (eds) (2004) Aspects of molecular computing. Essays dedicated to Tom Head on the occasion of his 70th birthday, Lecture notes in computer science, vol. 2950. Springer, Berlin
- Kari L (1991) On insertion and deletion in formal languages. Ph.D. thesis, University of Turku
- Kari L, Sosik P (2008) On the weight of universal insertion grammars. *Theor Comput Sci* 396:264–270
- Krassovitskiy A, Rogozhin Y, Verlan S (2007) Further results on insertion-deletion systems with one-sided contexts. Proceedings of the LATA 2007, Tarragona, Spain, Technical Rep. RGML, 36/2008, pp 347–358
- Krassovitskiy A, Rogozhin Y, Verlan S (2008) One-sided insertion and deletion: traditional and P systems case. In: Csuha-Jarai et al. (eds) International workshop on computing with biomolecules. Vienna, Austria, pp 51–63
- Loos R (2006) An alternative definition of splicing. *Theor Comput Sci* 358:75–87
- Loos R, Ogihara M (2007) Complexity theory of splicing systems. *Theor Comput Sci* 386:132–150
- Loos R, Malcher A, Wotschke D (2008) Descriptive complexity of splicing systems. *Int J Found Comput Sci* 19:813–826
- Manca V (2000) Splicing normalization and regularity. In: Calude CS, Păun Gh (eds) Finite versus infinite. Contributions to an eternal dilemma, Springer, Berlin, pp 199–215
- Marcus S (1969) Contextual grammars. *Rev Roum Math Pures Appl* 14:1525–1534
- Margenstern M, Rogozhin Y, Verlan S (2002) Time-varying distributed H systems of degree 2 can carry out parallel computations. In: Proceedings of the DNA computing '02, Sapporo, Japan, Lecture notes in computer science, vol. 2568. Springer, Berlin, pp 326–336
- Margenstern M, Rogozhin Y, Verlan S (2004) Time-varying distributed H systems with parallel computations: the problem is solved. In: Proceedings of the DNA Computing '04, Madison, Wisconsin, Lecture notes in computer science, vol. 2943. Springer, Berlin, pp 48–53
- Margenstern M, Păun Gh, Rogozhin Y, Verlan S (2005) Context-free insertion-deletion systems. *Theor Comput Sci* 330:339–348
- Mateescu A, Păun Gh, Rozenberg G, Salomaa A (1998) Simple splicing systems. *Discrete Appl Math* 84:145–163
- Matveevici A, Rogozhin Y, Verlan S (2007) Insertion-deletion systems with one-sided contexts. In: Proceedings of the machines, computations, and universality '07, Orleans, France, LNCS 4664, Springer, 2007, 205–217
- Păun Gh (1996a) On the splicing operation. *Discrete Appl Math* 70:57–79
- Păun Gh (1996b) Regular extended H systems are computationally universal. *J Auto Lang Comb* 1:27–36
- Păun Gh (1997) Marcus contextual grammars. Kluwer, Boston, MA
- Păun A (2003) Unconventional models of computation: DNA and membrane computing. Ph.D. thesis, University of Western Ontario, Canada
- Păun Gh, Rozenberg G, Salomaa A (1996) Computing by splicing. *Theor Comput Sci* 168:321–336
- Păun Gh, Rozenberg G, Salomaa A (1998) DNA computing. New computing paradigms. Springer, Berlin

- Păun Gh, Pérez-Jiménez MJ, Yokomori T (2008) Representations and characterizations of languages in Chomsky hierarchy by means of insertion-deletion systems. *Int J Found Comput Sci* 19:859–871
- Păun Gh, Rozenberg G, Salomaa A (eds) (2009) *Handbook of membrane computing*. Oxford University Press, Oxford, UK
- Penttonen M (1974) One-sided and two-sided contexts in phrase structure grammars. *Inform Control* 25:371–392
- Pixton D (1996) Regularity of splicing languages. *Discrete Appl Math* 69:101–124
- Pixton D (2000) Splicing in abstract families of languages. *Theor Comput Sci* 234:135–166
- Rozenberg G, Salomaa A (eds) (1997) *Handbook of formal languages*, 3 vol. Springer, Berlin
- Salomaa A (1973) *Formal languages*. Academic, New York
- Tanaka A, Yokomori T (2003) On the computational power of insertion-deletion systems. In: *Proceedings of the DNA Computing '02*, Sapporo, Japan, Lecture notes in computer science, vol. 2568. Springer, Berlin, pp 269–280
- Thomas DG, Begam MH, David NG (2007) Hexagonal array splicing systems. *Ramanujan Math Soc Lect Notes Ser* 3:197–207
- Verlan S (2007) On minimal context-free insertion-deletion systems. *J Auto Lang Comb* 12:317–328
- Verlan S, Zizza R (2003) 1-splicing vs. 2-splicing: separating results. In: Harju T, Karhumaki J (eds) *Proceedings of the WORDS'03*, TUCS General Publisher, 27, pp 320–331
- Zizza R (2002) On the power of classes of splicing systems. Ph.D. thesis, University of Milano-Bicocca

36 Bacterial Computing and Molecular Communication

Yasubumi Sakakibara¹ · Satoshi Hiyama²

¹Department of Biosciences and Informatics, Keio University, Yokohama, Japan

yasu@bio.keio.ac.jp

²Research Laboratories, NTT DOCOMO, Inc., Yokosuka, Japan

hiyama@nttdocomo.co.jp

1	<i>Introduction</i>	1204
2	<i>Bacterial Computing</i>	1207
3	<i>Molecular Communication</i>	1219

Abstract

Emerging technologies that enable the engineering of nano- or cell-scale systems using biological and/or artificially synthesized molecules as computing and communication devices have been receiving increased attention. This chapter focuses on “bacterial computing,” which attempts to create an autonomous cell-based Turing machine, and “molecular communication,” which attempts to create non-electromagnetic-wave-based communication paradigms by using molecules as an information medium. ➤ [Section 2](#) introduces seminal works for constructing *in vivo* logic circuits, and focuses on research into implementing *in vitro* and *in vivo* finite automata in the framework of DNA-based computing. Furthermore, the first experimental development of a programmable *in vivo* computer that executes a finite-state automaton in bacteria is highlighted. ➤ [Section 3](#) reports on the system design, experimental results, and research trends of molecular communication components (senders, molecular communication interfaces, molecular propagation systems, and receivers) that use bacteria, lipids, proteins, and DNA as communication devices.

1 Introduction

1.1 Brief Introduction to Bacterial Computing

Biological molecules such as DNA, RNA, and proteins are natural devices that store information, activate (chemical) functions, and communicate between systems (such as cells). Research in DNA computing aims to utilize these biological devices to make a computer. One of the ultimate goals is to make an autonomous cell-based Turing machine for genetic and life engineering.

Finite-state automata (machines) is the most basic computational model in the Chomsky hierarchy and is the starting point for building universal DNA computers. Many works have proposed theoretical models of DNA-based computer to develop finite-state automata (Păun et al. 1998), and several experimental research works have attempted to implement finite automata *in vitro* and *in vivo*. For example, Benenson et al. (2001, 2003) successfully implemented the two-state finite automata *in vitro* by sophisticated use of the restriction enzyme (actually, *FokI*), which is cut outside of its recognition site in a double-stranded DNA. However, their method has some limitations for extending to more than two states. Yokomori et al. (2002) proposed a theoretical framework using a length-encoding technique to implement finite automata on DNA molecules. Theoretically, the length-encoding technique has no limitations to implement finite automata of any larger states. Based on the length-encoding technique, Sakakibara and his group (Kuramochi and Sakakibara 2005) attempted to implement and execute finite automata of a larger number of states *in vitro*, and carried out intensive laboratory experiments on various finite automata from two states to six states for several input strings.

On the other hand, Sakakibara and his group (Nakagawa et al. 2005) proposed a method using a protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata *in vivo* (a codon is normally a triplet of bases, and different base triplets encode different amino acids in protein). The proposed method is quite promising, has several advanced features such as the protein-synthesis process,

is very accurate, overcomes the mis-hybridization problem in the self-assembly computation, and further offers an autonomous computation.

Attempts to implement a finite number of states *in vivo* have also been actively studied in the area of synthetic biology. One of the most significant goals of synthetic biology is the rewiring of genetic regulatory networks to realize controllable systems in a cell. Recently, this technology has been applied to the development of biomolecular computations *in vivo*. Weiss et al. (2003, 2004) explored *in vivo* technology to develop genetic circuits and perform logical computations in cells. They developed cellular units that implement the logical functions NOT, IMPLIES, and AND. These units are combined into genetic circuits for cellular logical computation.

The first half of this chapter overviews seminal works of Weiss et al. (2003, 2004) on the construction of *in vivo* logic circuits, and research by Sakakibara and his group (Kuramochi and Sakakibara 2005; Nakagawa et al. 2005) on implementing finite automata *in vitro* and *in vivo* in the framework of DNA-based computing.

First, the length-encoding technique proposed in Yokomori et al. (2002) and Sakakibara and Hohsaka (2003) is introduced to implement finite automata in test tubes. In the length-encoding method, the states and state transition functions of a target finite automaton are effectively encoded into DNA sequences, a computation (accepting) process of finite automata is accomplished by self-assembly of encoded complementary DNA strands, and the acceptance of an input string is determined by the detection of a completely hybridized double-strand DNA. Second, intensive *in vitro* experiments are shown in which several finite-state automata have been implemented and executed in test tubes. Practical laboratory protocols that combine several *in vitro* operations such as annealing, ligation, PCR, and streptavidin–biotin bonding are designed and developed to execute *in vitro* finite automata based on the length-encoding technique. Laboratory experiments are carried out on various finite automata from two states to six states for several input strings. Third, a novel framework to develop a programmable and autonomous *in vivo* computer using *Escherichia coli* (*E. coli*) is presented, and *in vivo* finite-state automata are implemented based on the framework by employing the protein-synthesis mechanism of *E. coli*. The fundamental idea to develop a programmable and autonomous finite-state automata in *E. coli* is that first an input string is encoded into one plasmid, state-transition functions are encoded into the other plasmid, and those two plasmids are introduced into an *E. coli* cell by electroporation. Fourth, a protein-synthesis process in *E. coli* combined with four-base codon techniques is executed to simulate a computation (accepting) process of finite automata, as proposed for *in vitro* translation-based computations in Sakakibara and Hohsaka (2003). This approach enables one to develop a programmable *in vivo* computer by simply replacing a plasmid encoding a state-transition function with others. Furthermore, the *in vivo* finite automata are autonomous because the protein-synthesis process is autonomously executed in the living *E. coli* cell.

1.2 Brief Introduction to Molecular Communication

The second half of this chapter overviews molecular communication that uses molecules (chemical substances such as proteins and deoxyribonucleic acid (DNA)) as information media and offers a new communication paradigm based on biochemical reactions caused by received molecules (Hiyama et al. 2005a, 2008b; Moritani et al. 2007a; Suda et al. 2005).

Molecular communication has received increasing attention as an interdisciplinary research area that spans the fields of nanotechnology, biotechnology, and communication engineering.

Molecular communication is inspired by the observation of specific features of biological molecular systems that living organisms have acquired through the evolutionary process over billions of years. Communication in biological systems is typically done through molecules (i.e., chemical signals). For instance, multicellular organisms including human beings perform maintenance of homeostasis, growth regulation, kinematic control, memory, and learning through intercellular and intracellular communication using signal-transducing molecules such as hormones (Alberts et al. 1997; Pollard and Earnshaw 2004). Molecular communication aims to develop artificially designed and controllable systems that could transmit biochemical information such as phenomena and the status of living organisms that cannot be feasibly carried with traditional communication using electromagnetic waves (i.e., electronic and optical signals).

In molecular communication, a sender generates molecules, encodes information into the molecules (called information molecules), and emits the information molecules to the propagation environment. A propagation system transports the emitted information molecules to a receiver. The receiver, upon receiving the transported information molecules, reacts biochemically according to the information molecules (this biochemical reaction represents decoding of the information).

Molecular communication is a new communication paradigm and is different from the traditional communication paradigm (● *Table 1*). Unlike traditional communication which utilizes electromagnetic waves as an information medium, molecular communication utilizes molecules as an information medium. In addition, unlike traditional communication in which encoded information such as voice, text, and video is decoded and regenerated at a receiver, in molecular communication information molecules activate some biochemical reactions at a receiver and recreate phenomena and/or chemical status that a sender transmits. Other features of molecular communication include aqueous environmental communication, the stochastic nature of communication, low energy-consumption communication, and high compatibility with biological systems.

● **Table 1**

Comparisons of key features between traditional communication and molecular communication

Key features	Traditional communication	Molecular communication
Information medium	Electromagnetic waves	Molecules
Signal type	Electronic and optical signals	Chemical signals
Propagation speed	Light speed (3×10^5 km/s)	Slow speed (a few $\mu\text{m}/\text{s}$)
Propagation distance	Long (ranging from m to km)	Short (ranging from nm to m)
Propagation environment	Airborne and cable medium	Aqueous medium
Encoded information	Voice, text, and video	Phenomena and chemical status
Behavior of receivers	Decoding of digital info	Biochemical reaction
Communication model	Deterministic communication	Stochastic communication
Energy consumption	High	Extremely low

Although the communication speed/distance of molecular communication is slower/shorter than that of traditional communication, molecular communication may carry information that is not feasible to carry with traditional communication (such as the biochemical status of a living organism) between entities to which traditional communication does not apply (such as biological entities). Molecular communication has unique features that are not seen in traditional communication and is not competitive but complementary to traditional communication.

Molecular communication is an emerging research area and its feasibility is being verified through biochemical experiments. Detailed system design, experimental results, and research trends in molecular communication are described in [Sect. 3](#).

2 Bacterial Computing

2.1 In Vivo Genetic Circuit

Seminal work to develop genetic circuits and perform logical computations in cells was done by Weiss et al. ([2003, 2004](#)). They developed cellular units that implement the logical functions NOT, IMPLIES, and AND. These units were combined into genetic circuits for cellular logical computation. These genetic circuits employed messenger RNA (mRNA), DNA-binding proteins, small molecules that interacted with the related proteins, and DNA motifs that regulated the transcription and expression of the proteins. They further attempted to construct a circuit component library so that genetic circuit design is the process of assembling library components into a logical circuit.

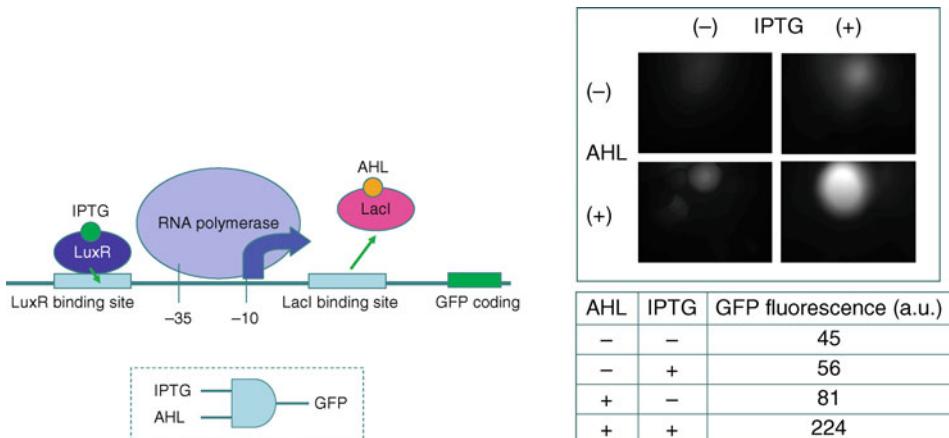
2.1.1 AND Gate

Given a library of circuit components, various genetic circuits are designed by combining and assembling a set of library components. For example, the AND logical function was implemented in cells by using RNA polymerase, activator proteins, inducer molecules, and DNA that encodes the RNA polymerase binding domain, the activator binding domain, and an output gene (Weiss et al. [2003](#)). In this genetic circuit, the input and output signals are represented by the concentrations of some mRNAs or small molecules.

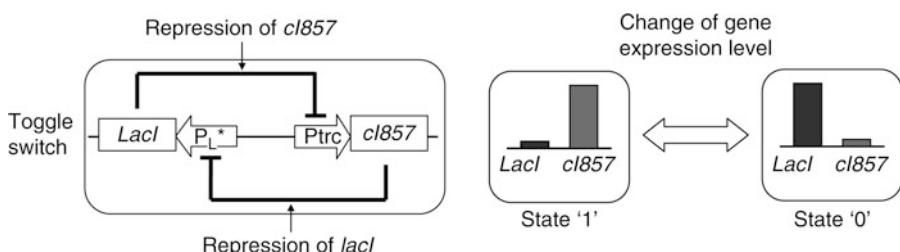
There can be several designs to implement the same logical function AND. Sakakibara et al. ([2006](#)) developed another circuit design for implementing the AND function and performed an *in vivo* experiment to run the AND circuit. As basic components, they used RNA polymerase, activator protein LuxR, repressor protein LacI, two inducer molecules, GFP protein, and DNA that encodes the RNA polymerase binding domain, the activator binding domain, the repressor binding domain, and the GFP gene. In this circuit design, the input signals are two inducers AHL and IPTG, and the output signal is GFP protein. This genetic circuit executes the AND logical function as follows. If only the IPTG inducer is present, the activator LuxR is active with IPTG, binds to the DNA promoter and activates RNA polymerase transcription, but the repressor LacI is still active, it keeps binding to the DNA promoter and hence prevents transcription. If only the AHL inducer is present, the inducer binds to the repressor LacI and changes the conformation of the repressor. The conformation change prevents the repressor from binding to the DNA promoter. However, the activator LuxR is inactive without IPTG and hence does not activate the RNA polymerase transcription. Finally, if both inducers AHL and IPTG are present, the inactive repressor LacI does not bind to the

Fig. 1

(Left) A genetic circuit design for AND function and (right) its experimental results.

**Fig. 2**

(Left) Toggle switch circuit and (right) two states using gene expression levels.



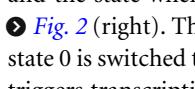
DNA promoter and the activator LuxR activates the RNA polymerase transcription, and therefore RNA polymerase transcribes the GFP gene, yielding the GFP output. The genetic circuit design is illustrated in **Fig. 1** (left) and the experimental results for executing the AND logical function are shown in **Fig. 1** (right).

2.1.2 Toggle Switch

In order to realize a finite number of states *in vivo*, Gardner et al. (2000) proposed the toggle switch mechanism. The toggle switch is composed of two repressors and two constitutive promoters (see **Fig. 2** (left)). Each promoter is inhibited by the repressor that is transcribed by the opposing promoter. The behavior of the toggle switch and the conditions for bistability to represent two states can be formalized using the following equations:

$$\frac{dV_{cl857}}{dt} = \frac{\alpha_1}{1 + V_{lacI}^\beta} - V_{cl857}$$

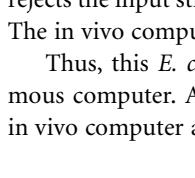
$$\frac{dV_{lacI}}{dt} = \frac{\alpha_2}{1 + V_{cl857}^\gamma} - V_{lacI}$$

where V_{cl857} is the concentration of repressor gene *cl857*, V_{lacI} is the concentration of repressor gene *lacI*, α_1 is the effective rate of synthesis of repressor *cl857*, α_2 is the effective rate of synthesis of repressor *lacI*, β is the cooperativity of repression of promoter *P_{trc}*, and γ is the cooperativity of repression of promoter *P_{L*}*. The bistability of the toggle switch depends on these parameters. The state where *cl857* is expressed and *lacI* is repressed is termed state “1” and the state where *cl857* is repressed and *lacI* is expressed is termed state “0,” as shown in  Fig. 2 (right). The state 1 is switched to state 0 by a thermal pulse of 42°C induction and the state 0 is switched to state 1 by a pulse of IPTG induction, which is a molecular compound that triggers transcription of the lac operon.

2.2 Bacterial Finite-State Machine

2.2.1 Preliminaries

The aim of bacteria-based computing is to develop a computing machine in a living system. Sakakibara and his group (Nakagawa et al. 2005) have attempted to simulate a computation process of finite automata in *Escherichia coli* (*E. coli*) by employing the *in vivo* protein-synthesis mechanism of *E. coli*. (*Escherichia coli* is a typical bacterium living inside our body, in the large intestine.) This *in vivo* computation possesses the following two novel features, not found in any previous biomolecular computer. First, an *in vivo* finite automaton is implemented in a living *E. coli* cell. Second, this automaton increases in number very rapidly according to bacterial growth; one bacterial cell can multiply to over a million cells overnight. This chapter explores the feasibility of *in vivo* computation.

The main feature of the *in vivo* computer based on *E. coli* is that first an input string is encoded into one plasmid, state-transition functions are encoded into the other plasmid, and *E. coli* cells are transformed with these two plasmids by electroporation. Second, a protein-synthesis process in *E. coli* combined with four-base codon techniques is executed to simulate a computation (accepting) process of finite automata, which has been proposed for *in vitro* translation-based computations in Sakakibara and Hohsaka (2003). The successful computations are detected by observing the expressions of a reporter gene linked to mRNA encoding input data. Therefore, when an encoded finite automaton accepts an encoded input string, the reporter gene, *lacZ*, is expressed and hence a blue color is observed. When the automaton rejects the input string, the reporter gene is not expressed and hence no blue color is observed. The *in vivo* computer system based on *E. coli* is illustrated in  Fig. 3.

Thus, this *E. coli*-based computer enables one to develop a programmable and autonomous computer. As known to the authors, this is the first experimental development of an *in vivo* computer and it has succeeded in executing a finite-state automaton using *E. coli*.

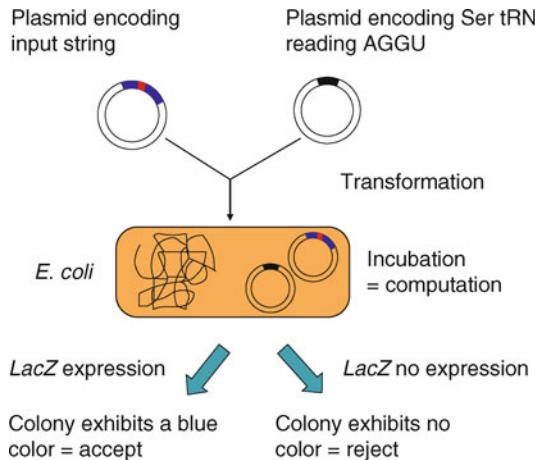
2.2.2 A Framework of a Programmable and Autonomous In Vivo Computer using *E. coli*

Length-Encoding Method to Implement Finite-State Automata

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a (deterministic) finite automaton, where Q is a finite set of states numbered from 0 to k , Σ is an alphabet of input symbols, δ is a state-transition function such that $\delta : Q \times \Sigma \rightarrow Q$, q_0 is the initial state, and F is a set of final states. The length-encoding

Fig. 3

The framework of an *in vivo* computer system based on *E. coli*.



technique (Yokomori et al. 2002) is adopted to encode each state in Q by the length of DNA subsequences.

For the alphabet Σ , each symbol a in Σ is encoded into a single-strand DNA subsequence, denoted $e(a)$, of fixed length. For an input string w on Σ , $w = x_1x_2 \cdots x_m$ is encoded into the following single-strand DNA subsequence, denoted $e(w)$:

$$5' - e(x_1) \underbrace{x_1x_2 \cdots x_k}_{k \text{ times}} e(x_2) \underbrace{x_1x_2 \cdots x_k}_{k \text{ times}} \cdots e(x_m) \underbrace{x_1x_2 \cdots x_k}_{k \text{ times}} - 3'$$

where x_i is one of four nucleotides A,C,G,T, and the subsequences $x_1x_2 \cdots x_k$ are used to encode $k+1$ states of the finite automaton M . For example, when a symbol “1” is encoded into a ssDNA subsequence GCGC and a symbol “0” into GGCC, and we encode three states into TT, a string “1101” is encoded into the following ssDNA sequence:

$$5' - \overbrace{\text{GCGC}}^1 \text{TT} \overbrace{\text{GCGC}}^1 \text{TT} \overbrace{\text{GGCC}}^0 \text{TT} \overbrace{\text{GCGC}}^1 \text{TT} - 3'$$

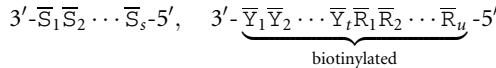
In addition, two supplementary subsequences are appended at both ends for PCR primers and probes for affinity purifications with magnetic beads that will be used in the laboratory protocol:

$$5' - \underbrace{S_1 S_2 \cdots S_s}_{\text{PCR primer}} e(x_1) x_1 x_2 \cdots x_k \cdots e(x_m) x_1 x_2 \cdots x_k \underbrace{Y_1 Y_2 \cdots Y_t}_{\text{probe}} \underbrace{R_1 R_2 \cdots R_u}_{\text{PCR primer}} - 3'$$

For a state-transition function from state q_i to state q_j with input symbol $a \in \Sigma$, the state-transition function $\delta(q_i, a) = q_j$ is encoded into the following complementary single-strand DNA subsequence:

$$3' - \overbrace{\overline{x}_{i+1} \overline{x}_{i+2} \cdots \overline{x}_k}^{k-i \text{ times}} \overbrace{\overline{e(a)}}^j \overbrace{\overline{x}_1 \overline{x}_2 \cdots \overline{x}_j}^j - 5'$$

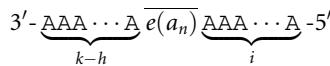
where \bar{X}_i denotes the complementary nucleotide of x_i , and \bar{y} denotes the complementary sequence of y . Further, two more complementary ssDNA sequences are added for the supplementary subsequences at both ends:



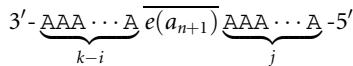
where the second ssDNA is biotinylated for streptavidin–biotin bonding.

Now, all those ssDNAs encoding an input string w and encoding state-transition functions and the supplementary subsequences of probes and PCR primers are put together into a test tube. Then, a computation (accepting) process of the finite automata M is accomplished by self-assembly among those complementary ssDNAs, and the acceptance of an input string w is determined by the detection of a completely hybridized double-strand DNA.

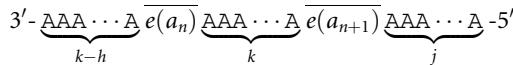
The main idea of the length-encoding technique is explained as follows. Two consecutive valid transitions $\delta(h, a_n) = i$ and $\delta(i, a_{n+1}) = j$ are implemented by concatenating two corresponding encoded ssDNAs, that is,



and



together make



Thus, the subsequence $\underbrace{\text{AAA} \cdots \text{A}}_k$ plays the role of “joint” between two consecutive state-transitions and it guarantees that the two transitions are valid in M .

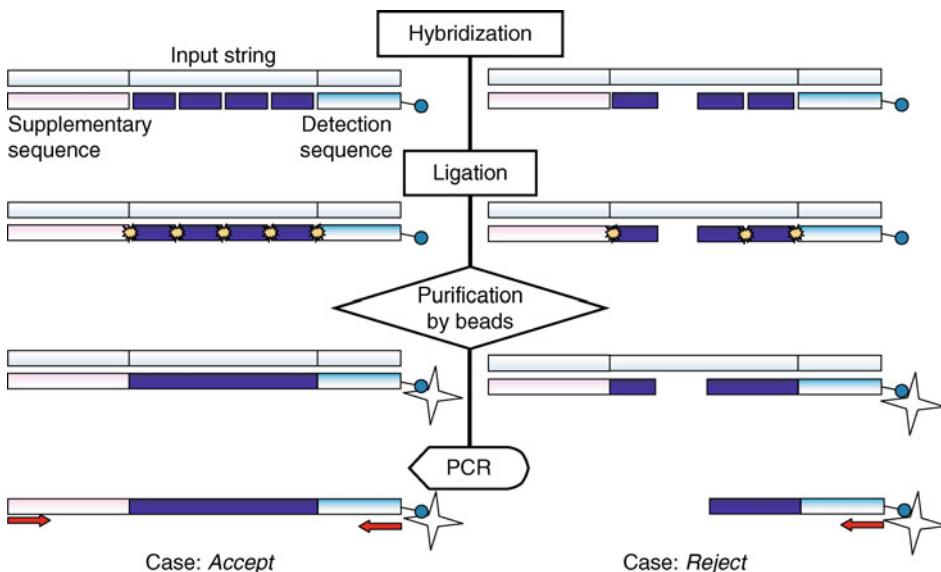
Designing Laboratory Protocols to Execute Finite Automata in Test Tubes

First, the length-encoding method was applied to in vitro experiments to see the effectiveness of the length-encoding techniques for implementing finite-state automata in vitro and in vivo. In order to practically execute the laboratory in vitro experiments for the length-encoding method, the following experimental laboratory protocol was designed, which is also illustrated in Fig. 4:

0. Encoding: Encode an input string into a long ssDNA, and state-transition functions and supplementary sequences into short pieces of complementary ssDNAs.
1. Hybridization: Put all those encoded ssDNAs together into one test tube, and anneal those complementary ssDNAs to be hybridized.
2. Ligation: Put DNA “ligase” into the test tube and invoke ligations at a temperature of 37 °C. When two ssDNAs encoding two consecutive valid state-transitions $\delta(h, a_n) = i$ and $\delta(i, a_{n+1}) = j$ are hybridized at adjacent positions on the ssDNA of the input string, these two ssDNAs are ligated and concatenated.

Fig. 4

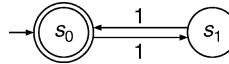
The flowchart of the laboratory protocol used to execute in vitro finite automata consists of five steps: hybridization, ligation, denature and extraction by affinity purification, amplification by PCR, and detection by gel-electrophoresis. The acceptance of the input string by the automata is the left case, and the rejection is the right case.



3. Denature and extraction by affinity purification: Denature double-stranded DNAs into ssDNAs and extract concatenated ssDNAs containing the biotinylated probe subsequence by streptavidin–biotin bonding with magnetic beads.
 4. Amplification by PCR: Amplify the extracted ssDNAs with PCR primers.
 5. Detection by gel-electrophoresis: Separate the PCR products by length using gel-electrophoresis and detect a particular band of the full-length. If the full-length band is detected, that means a completely hybridized double-strand DNA has been formed, and hence the finite automaton “accepts” the input string. Otherwise, it “rejects” the input string. In the laboratory experiments, a “capillary” electrophoresis microchip-based system, called Bioanalyser 2100 (Agilent Technologies), was used in place of conventional gel-electrophoresis. Capillary electrophoresis is of higher resolution and is more accurate than gel electrophoresis using agarose gel.
- #### Simulating Computation Process of Finite Automata Using Four-Base Codons and the Protein-Synthesis Mechanism
- Sakakibara and Hohsaka (2003) have proposed a method using the protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata. An approach to make an *in vivo* computer is to execute the proposed method in *E. coli* in order to improve the efficiency of the method and further develop a programmable *in vivo* computer. The proposed method is described using the example of the simple finite automaton, illustrated in [Fig. 5](#), which is of two states $\{s_0, s_1\}$,

Fig. 5

A simple finite automaton of two states $\{s_0, s_1\}$, defined on one symbol “1”, and accepting input strings with even numbers of symbol 1 and rejecting input strings with odd numbers of 1s.



defined as one symbol “1”, and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

The input symbol “1” is encoded into the four-base subsequence AGGU and an input string is encoded into an mRNA by concatenating AGGU and A alternately, and adding a special sequence AAUAAC that contains a stop codon at the 3'-end. This one nucleotide A in between AGGU is used to encode two states $\{s_0, s_1\}$, which is the same technique presented in Yokomori et al. (2002). The stop codon UAA is used for the decision of acceptance or rejection together with a downstream reporter gene. For example, a string “111” is encoded into an mRNA:



The four-base anticodon (3')UCCA(5') of tRNA encodes the transition rule $s_0 \xrightarrow{1} s_1$, that is a transition from state s_0 to state s_1 with input symbol 1, and the combination of two three-base anticodons (3')UUC(5') and (3')CAU(5') encodes the rule $s_1 \xrightarrow{1} s_0$. Furthermore, the encoding mRNA is linked to the *lacZ*-coding RNA subsequence as a reporter gene for the detection of successful computations. Together with these encodings and the tRNAs containing a four-base anticodon (3')UCCA(5'), if a given mRNA encodes an input string with odd numbers of symbol 1, an execution of the *in vivo* protein-synthesis system stops at the stop codon, which implies that the finite automaton does not accept the input string, and if a given mRNA encodes even numbers of 1s, the translation goes through the entire mRNA and the detection of acceptance is found by the *blue* signal of *lacZ*. Examples of accepting processes are shown in **Fig. 6** (upper): For an mRNA encoding a string “1111,” the translation successfully goes through the entire mRNA and translates the reporter gene of *lacZ* that emits the blue signal (lower). For an mRNA encoding a string “111,” the translation stops at the stop codon UAA, does not reach the *lacZ* region, and produces no blue signal.

If the competitive three-base anticodon (3')UCC(5') comes faster than the four-base anticodon (3')UCCA(5'), the incorrect translation (computation) immediately stops at the following stop codon UAA.

In order to practically execute the laboratory experiments for the *in vivo* finite automata described in this section, the details of the laboratory protocol are described in Nakagawa et al. (2005).

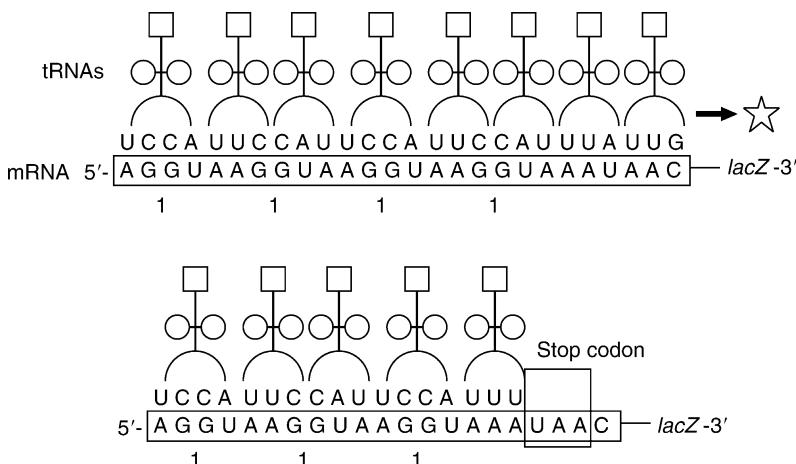
2.2.3 Programmability and Autonomy

Two important issues to consider when developing DNA-based computers are whether they are *programmable* and *autonomous*.

Programmability means that a program is stored as data (i.e., stored program computer) and any computation can be accomplished by just choosing a stored program. In DNA-based computers, it is required that a program is encoded into a molecule different from the main and

Fig. 6

Examples of accepting processes: (Upper) For an mRNA encoding a string “1111,” the translation successfully goes through the mRNA and translates the reporter gene of *lacZ* emitting the blue signal. (Lower) For an mRNA encoding a string “111,” the translation stops at the stop codon UAG, does not reach the *lacZ* region and produces no blue signal.



fixed units of the DNA computer, that molecule encoding program can be stored and changed, and that a change of the molecule encoding program accomplishes any computations.

The main features of the *in vivo* computer enable one to develop a programmable *in vivo* computer. A plasmid encoding a state-transition function can be simply replaced with another plasmid encoding a different state-transition function, and the *E. coli* cell transformed a new plasmid computes a different finite automaton.

With autonomous DNA computers, once one sets a program and input data and starts a computation, the entire computational process is carried out without any operations from the outside. The *in vivo* finite automata are autonomous in the sense that the protein-synthesis process that corresponds to a computation (accepting) process of an encoded finite automata is autonomously executed in a living *E. coli* cell and requires no laboratory operations from the outside (❸ Fig. 7).

2.3 In Vitro and In Vivo Experiments

2.3.1 In Vitro Experiments

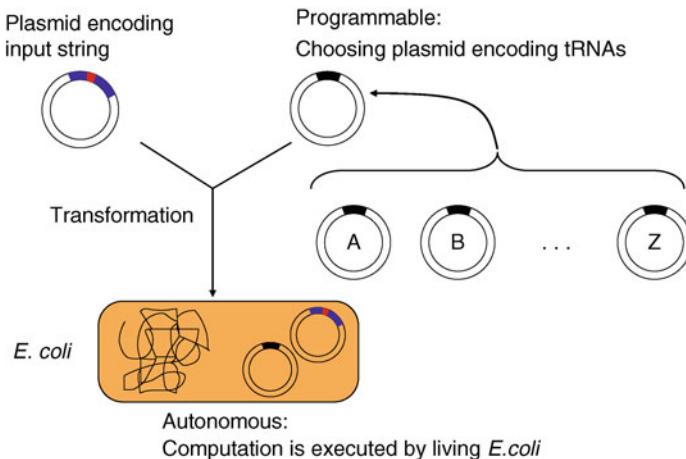
Laboratory *in vitro* experiments have been carried out on various finite automata for several input strings.

Two-States Automaton with Two Input Strings

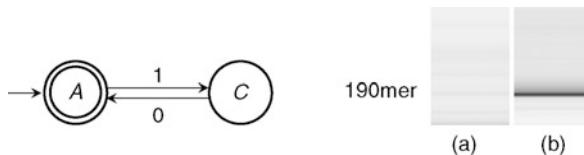
The experiments begin with a simple two-states automaton shown in ❸ Fig. 8 (left) with two input strings, (a) “1101” and (b) “1010.” The language accepted by this automaton is $(10)^+$, and hence the automaton accepts the string 1010 and rejects the other string 1101.

Fig. 7

A programmable and autonomous *in vivo* computer system based on *E. coli*.

**Fig. 8**

(Left) The simple two-states automaton used for this experiment. (Right) The results of electrophoresis are displayed in the gel-like image. Lane (a) is for the input string 1101, and lane (b) for 1010. Since the full-length band (190 mer) is detected only in lane (b), the automaton accepts only the input string (b) 1010.



The results of electrophoresis using the Bioanalyser are displayed in the form of the gel-like image (as shown in Fig. 8 (right)). For these two input strings, the full-length DNA is of 190 bps (mer). Hence, if a band at a position of 190 mer is detected in the results of the electrophoresis, that means a completely hybridized double-strand DNA is formed, and hence the finite automaton “accepts” the input string.

Figure 8 (right) clearly shows that the *in vitro* experiments have successfully identified the correct acceptance of this automaton for two input strings, and hence correctly executed the computation process of the automaton *in vitro*.

Four-States Automaton with Three Input Strings

The second experiment attempts a four-states automaton, shown in Fig. 9 (left) for the three input strings (a) 1101, (b) 1110, and (c) 1010. This four-states automaton accepts the language $(1(0 \cup 1)1)^* \cup (1(0 \cup 1)1)^*0$, and hence it accepts 1110 and 1010 and rejects 1101.

The results are shown in Fig. 9 (right) in the gel-like image. As in the first experiment, the full-length DNA is of 190 bps (mer). Bands at the position of 190 mer are detected in lane

Fig. 9

(Left) The four-states automaton used for this experiment. (Right) The results of electrophoresis are displayed in the gel-like image. Lane (a) is for the input string 1101, lane (b) for 1110, and lane (c) for 1010. Since the full-length band (190 mer) is detected in lanes (b) and (c), the automaton accepts two input strings (b) 1110 and (c) 1010.



(b) and lane (c). Hence, the in vitro experiments have successfully detected that the automaton accepts the two input strings (b) 1110 and (c) 1010.

From Two-States to Six-States Automata with One Input String “111111” of Length 6

The second experiments are five different automata from two states to six states shown in [Fig. 10](#) (upper) for one input string “111111” of length 6. The automaton (2) accepts the language $(11)^*$, that is, strings with even numbers of symbol “1”, (3) accepts the language $(111)^*$, strings repeating three times 1s, (4) accepts the language $(1111)^*$, strings repeating four times 1s, (5) accepts the language $(11111)^*$, strings repeating five times 1s, (6) accepts the language $(111111)^*$, strings repeating six times 1s. Since 6 is a multiple of 2, 3 and 6, the automata (2), (3) and (6) accept the input string 111111 of length 6.

The results are shown in [Fig. 10](#) (lower) in the gel-like image. For the input string 111111, the full-length DNA is of 240 bps (mer). Bands at position 240 mer are detected in lanes (2), (3), and (6) in [Fig. 10](#). Hence, in these in vitro experiments, the automaton (2), (3), and (6) have correctly accepted the input string 111111 and the automaton (4) and (5) have correctly rejected 111111.

2.3.2 In Vivo Experiments

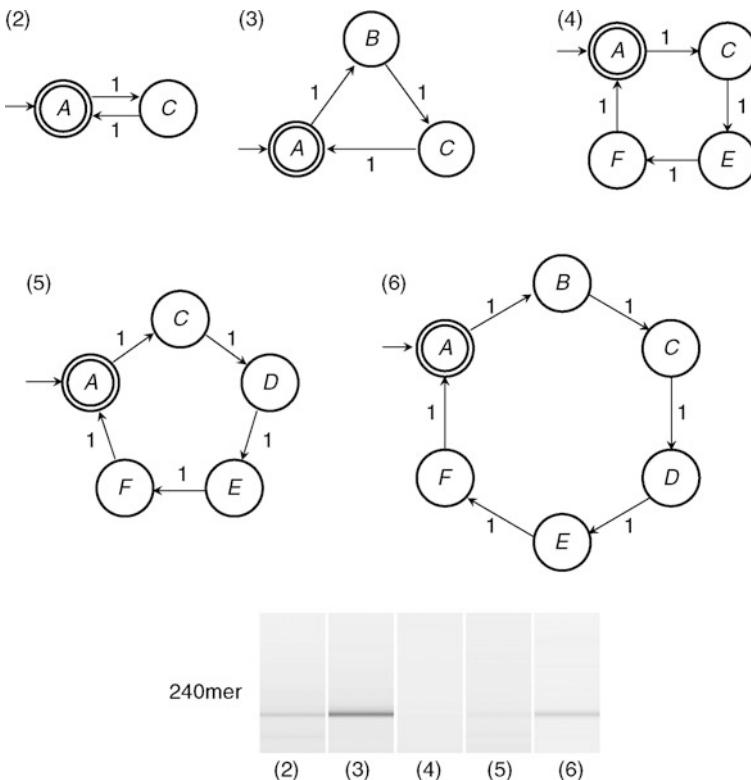
In vivo experiments that followed the laboratory protocols presented in Nakagawa et al. (2005) were done to execute the finite automaton shown in [Fig. 5](#), which is of two states $\{s_0, s_1\}$, defined on one symbol “1,” and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

Six input strings, “1,” “11,” “111,” “1111,” “11111,” and “111111,” were tested to see whether the method correctly accepts the input strings “11,” “1111,” “111111,” and rejects the strings “1,” “111,” “11111.”

The results are shown in [Fig. 11](#). Blue-colored colonies that indicate the expression of *lacZ* reporter gene were observed only in the plates for the input strings 11, 1111, and 111111. Therefore, the in vivo finite automaton succeeded in correctly computing the six input strings, that is, it correctly accepted the input strings 11, 1111, 111111 of even numbers of symbol “1” and correctly rejected 1, 111, 11111 of odd number of 1s. To the best of our knowledge, this is the first experimental implementation and in vivo computer that succeeded in executing an finite-state automaton in *E. coli*.

Fig. 10

(Upper) Five different automata from two states to six states used for this experiment.
 (Lower) The results of the electrophoresis are displayed in the gel-like image. Lane (2) is for the automaton (2), (3) for (3), (4) for (4), (5) for (5), and (6) for (6). Since the full-length bands (240 mer) are detected in lanes (2), (3), and (6), the automata (2), (3), and (6) accept the input string 111111.

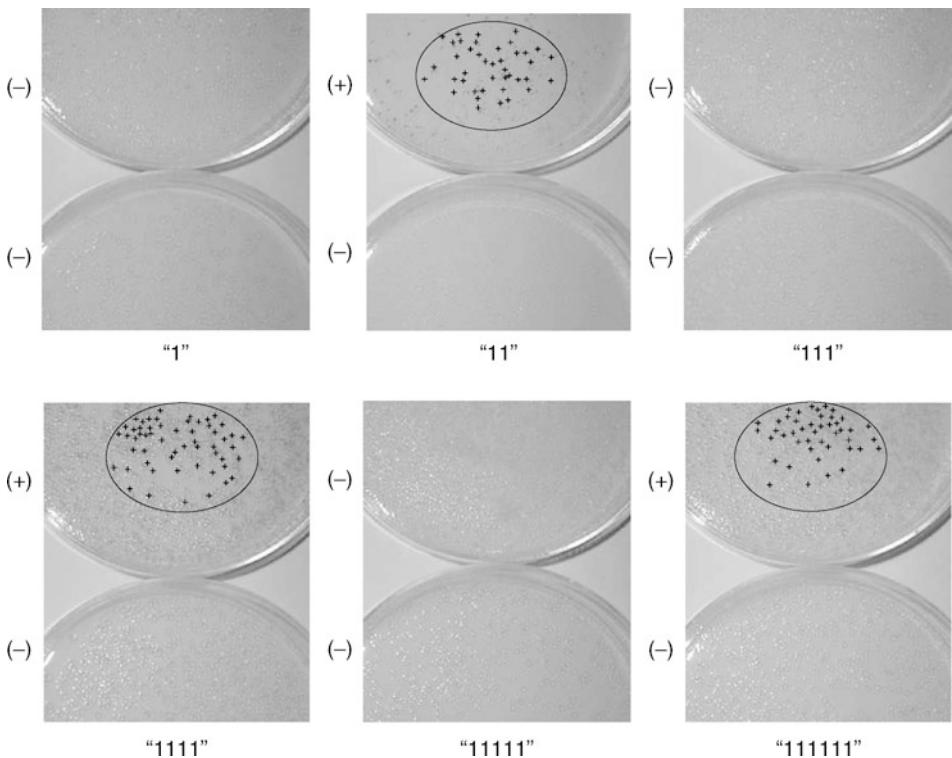


2.4 Further Topics

While a single *E. coli* bacteria cell contains approximately 10^{10} order of active molecules (Weiss et al. 2004), the previous works only utilized a small set of molecules to implement genetic circuits and finite-state automata. The main difficulty is to control such a huge number of molecules and complex wired networks for our purposes to implement bacterial computers. The idea of a “genomic computer” relating to developmental biology and evolution proposed by Istrail et al. (2007) presents some interesting direction to handle hundreds of thousands of regulatory modules that genomes contain for information processing. An animal genome contains many thousands of *cis*-regulatory control modules (called CRMs) that are wired in large networks and constitute genomic computers of information processing capability. Each CRM receives multiple inputs in the form of transcription factors that bind to CRM and execute information processing, such as AND, OR and NOT functions, for their multiple inputs. The idea is very similar to genetic circuits. The interesting features of genomic

Fig. 11

Computation by *E. coli* cells with plasmids of the input strings 1, 11, 111, 1111, 11111, 111111. In each panel, the upper plate (part of a LB plate) shows the result in the presence of the suppressor tRNA with UCCU anticodon in the cell, while the lower plate shows the result of the control experiment with no suppressor tRNA expressed. The signs (+) and (−) indicate the theoretical values about the expressions of the *lacZ* reporter gene: (+) means that the cultured *E. coli* cells must express *lacZ* theoretically, and (−) means it must not express. Circles indicate the blue-colored colonies expressing *lacZ*. Therefore, the *in vivo* finite automaton has correctly computed the six input strings, that is, it correctly accepts the input strings 11, 1111, 111111 of even numbers of symbol “1” and correctly rejects 1, 111, 11111 of odd number of 1s.



computers are: (1) multiplicity of processors and synchronicity, (2) robustness against system failure, and (3) evolvability especially relating to developmental biology.

Another recent interesting approach to utilizing genomes is the genetic memory proposed by Ham et al. (2008) that employs genome rearrangement operations to develop biological memory. They designed and synthesized an inversion switch using the DNA inversion recombination systems to create a heritable sequential memory switch.

One of the important directions of *in vivo* computation is multicellular bacterial computation. One necessary and fundamental mechanism for implementing multicellular computation is the communication between cells for sending and receiving the signals. Such intercellular communication was realized by means of the quorum-sensing mechanism,

which in nature allows the living bacteria to detect cell density. By using such a quorum sensing mechanism, Sakakibara and his group (Sakakibara et al. 2007) constructed a genetic circuit in *E.coli* that provides a one-dimensional cellular automaton. The main purpose was to extend previous works of implementing finite-state automata and develop *in vivo* computational systems higher than finite-state machine in terms of the Chomsky hierarchy. The computational capacity of the cellular automaton is equivalent to the universal Turing Machine. CA are multicellular systems in which uniform cells are allocated on a lattice grid. Each cell has a finite number of states. The state transition is determined by the current state and the states of the neighbor cells. Three mechanisms are required to implement CA in bacteria: (1) sending and receiving signals between cells, (2) sustaining the state, and (3) sensing input signals and changing the state following the state transition rules. To encode signals for mechanism (1), small molecules inducing transcription were used. For mechanism (2), the toggle switch circuit was employed to represent a finite number of states using gene expression. If one gene is expressed and the other is not, the state is 1, and, vice versa, 0. For mechanism (3), the state transition functions were implemented as logic gates using transcriptional regulatory proteins that bind to specific signal molecules (chemicals).

3 Molecular Communication

Molecular communication is a new and interdisciplinary research area; as such, it requires research into a number of key areas. Key research challenges in molecular communication include (1) design of a sender that generates molecules, encodes information onto the molecules (called information molecules), and emits the information molecules, (2) design of a molecular propagation system that transports the emitted information molecules from a sender to a receiver, (3) design of a receiver that receives the transported information molecules and biochemically reacts to the received information molecules resulting in decoding of the information, (4) design of a molecular communication interface between a sender and a molecular propagation system and also between the propagation system and a receiver to allow a generic transport of information molecules independent of their biochemical and physical characteristics, (5) mathematical modeling of molecular communication components and systems.

This section describes system design in detail, experimental results, and research trends in molecular communication.

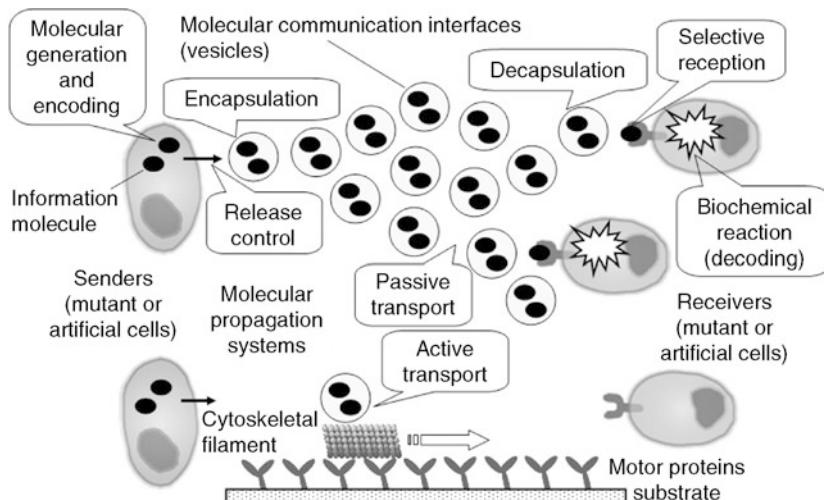
3.1 Overview of the Molecular Communication System

► *Figure 12* depicts an overview of a molecular communication system that includes senders, molecular communication interfaces, molecular propagation systems, and receivers.

A sender generates molecules, encodes information into the molecules (called information molecules), and emits the information molecules into a propagation environment. The sender may encode information into the type of the molecules used or the concentration of the molecules used. Possible approaches to creating a sender include genetically modifying eukaryotic cells and artificially constructing biological devices to control the encoding.

Fig. 12

An overview of a molecular communication system that includes senders, molecular communication interfaces, molecular propagation systems, and receivers.



A molecular communication interface acts as a molecular container that encapsulates information molecules to hide the characteristics of the information molecules during the propagation from the sender to a receiver to allow a generic transport of information molecules independent of their biochemical and physical characteristics. Using a lipid bilayer vesicle (Luisi and Walde 2000) is a promising approach to encapsulate the information molecules. Encapsulated information molecules are decapsulated at a receiver.

A molecular propagation system passively or actively transports information molecules (or vesicles that encapsulate information molecules) from a sender to an appropriate receiver through the propagation environment. The propagation environment is an aqueous solution that is typically found within and between cells. Using biological motor systems (motor proteins and cytoskeletal filaments) (Vale 2003) is a promising approach to actively and directionally transport information molecules.

A receiver selectively receives transported and decapsulated information molecules, and biochemically reacts with the received information molecules (this biochemical reaction represents decoding of the information). Possible approaches to creating a receiver are to genetically modify eukaryotic cells and to artificially construct biological devices to control the biochemical reaction.

3.2 Molecular Communication Interface

A vesicle-based communication interface provides a mechanism for transporting different types of information molecules in diverse propagation environments (Moritani et al. 2006a). The vesicle structure (i.e., a lipid bilayer membrane) compartmentalizes information molecules from the propagation environments and provides a generic architecture for transporting

diverse types of information molecules independent of their biochemical and physical characteristics. The vesicle structure also protects information molecules from denaturation (e.g., molecular deformation caused by enzymatic attacks or changes in pH) in the propagation environment. Key research issues in implementing the vesicle-based communication interface include how vesicles encapsulate information molecules at a sender and how vesicles de-encapsulate the information molecules at a receiver.

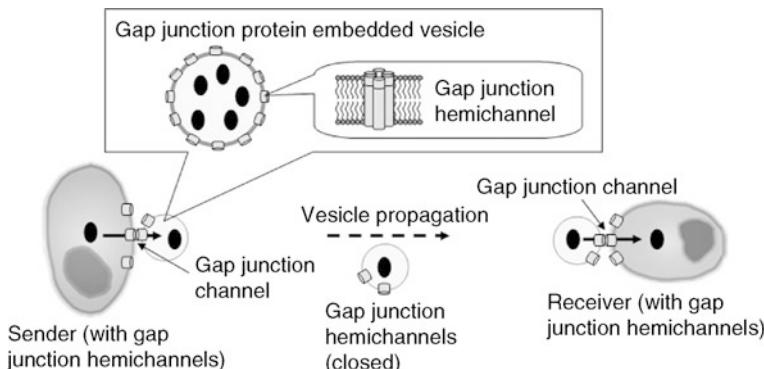
On these grounds, a molecular communication interface that uses a vesicle embedded with gap junction proteins has been proposed (Moritani et al. 2006b, 2007b). A gap junction is an intercellular communication channel formed between neighboring two cells, and it consists of two docked hemichannels (connexons) constructed from self-assembled six gap junction proteins (connexins) (Kumar and Gilula 1996). When a gap junction is open, molecules whose molecular masses are less than 1.5 kDa can directly propagate through the gap junction channel connecting two cells according to the molecular concentration gradient. A gap junction hemichannel is closed unless two hemichannels are docked.

In the molecular communication interface shown in Fig. 13 a sender stores information molecules inside itself and has gap junction hemichannels. When a vesicle with gap junction hemichannels physically contacts the sender, gap junction channels are formed between the sender and the vesicle, and the information molecules are transferred from the sender to the vesicle according to the molecular concentration gradient. When the vesicle detaches from the sender spontaneously, the gap junction hemichannels at the sender and at the vesicle close, and the information molecules transferred from the sender to the vesicle are encapsulated in the vesicle. Encapsulation of information molecules in a vesicle allows a molecular propagation system to transport the information molecules from the sender to a receiver independent of their biochemical and physical characteristics. The receiver also has gap junction hemichannels, and when the transported vesicle physically contacts the receiver, a gap junction channel is formed between the vesicle and the receiver, and the information molecules in the vesicle are transferred into the receiver according to the molecular concentration gradient.

The feasibility of the designed communication interface has been investigated by creating connexin-43 (one of the gap junction proteins) embedded vesicles (Moritani et al. 2006b). Microscopic observations confirmed that calceins (hydrophilic dyes used as model

Fig. 13

A schematic diagram of a molecular communication interface using a vesicle embedded with gap junction proteins.



information molecules) were transferred between connexin-43 embedded vesicles and the transferred calceins were encapsulated into the vesicles (Moritani et al. 2007b). This result indicates that the created connexin-43 embedded vesicle (a molecular communication interface) may encapsulate information molecules and receive/transfer information molecules from/into a sender/receiver through gap junctions.

3.3 Molecular Propagation System

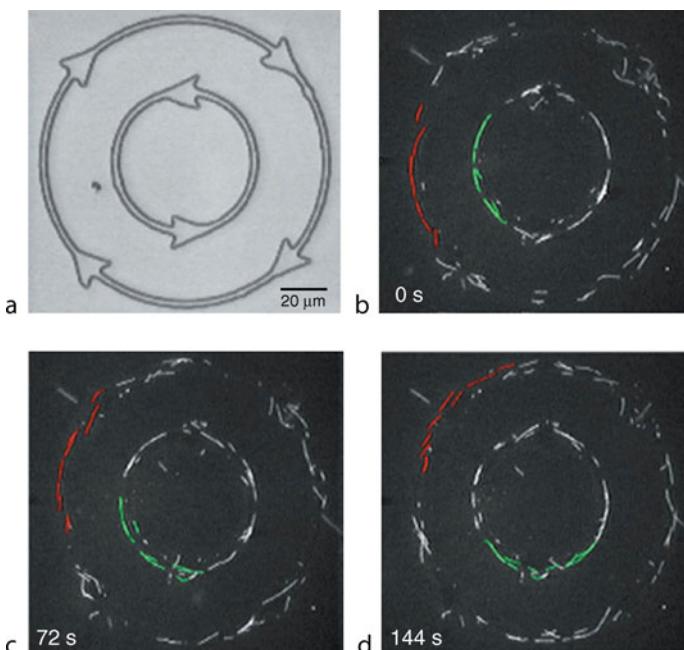
The simplest and easiest approach to transport information molecules (or vesicles that encapsulate information molecules) from a sender to a receiver is to use free diffusion. In biological systems, for instance, neurotransmitters such as acetylcholines diffuse at a synapse (around 100 nm gap between a nerve cell and a target cell), cellular slime molds such as amebas exhibit chemotaxis by detecting the molecular concentration gradient of cyclic adenosine monophosphates (cAMPs) diffused from hundreds of micrometers away, and pheromones secreted from insects can affect receiver insects even if they diffuse over a distance of several meters. Generally, the achievable rates of transmitted information molecules at a receiver are low due to Brownian motion and the dilution effect in the propagation environment; however, these observed facts indicate that diffusion-based passive molecular communication is feasible as long as the receiver sensitivity is high.

On the other hand, biological systems have a directional molecular propagation mechanism as well as nondirectional free diffusion. Motor proteins such as kinesins directionally transport cargo such as subcellular organelles and vesicles almost exactly to their destinations in spite of frequent collisions with disturbance molecules (e.g., water molecules, inorganic ions, organic small molecules, and organic macromolecules) within a biological cell. This mechanism is known as active transport and is realized by motor proteins' enzymatic actions that convert chemical energy called adenosine triphosphate (ATP) into mechanical work during their directional walk along filamentous proteins such as microtubules (MTs) (Alberts et al. 1997; Pollard and Earnshaw 2004). Motor proteins are nanometer-scaled actuators and have received increasing attention as engineering materials because they also work in an artificial environment outside of biological cells where aqueous conditions such as temperature and pH are reasonable (Goel and Vogel 2008; Van den Heuvel and Dekker 2007). For instance, as shown in Fig. 14, directional gliding of MTs was demonstrated by immobilizing kinesins onto a glass substrate etched by lithography (Hiratsuka et al. 2001). This mechanism may be applied to a molecular propagation system that uses gliding MTs as cargo transporters of information molecules. The idea is to load information molecules onto gliding MTs at a sender, to transport (glide) the information molecule-loaded MTs from a sender to a receiver, and to unload the transported information molecules from the gliding MTs at a receiver. Major issues in demonstrating this idea are how to load specified information molecules onto gliding MTs at a sender and how to unload the information molecules from the gliding MTs at a designated receiver in a reversible manner. Note that cargo transporters that utilize avidin–biotin (Bachand et al. 2004; Hess et al. 2001) or antigen–antibody (Bachand et al. 2006; Ramachandran et al. 2006) bindings for cargo loading may not be suitable for cargo unloading because of their tight bindings.

On these grounds, a molecular propagation system that uses the reverse geometry of MT motility on kinesins and also uses DNA hybridization and strand exchange has been proposed (Hiyama et al. 2005b, 2008a, c). The propagation system uses DNA hybridization and strand

Fig. 14

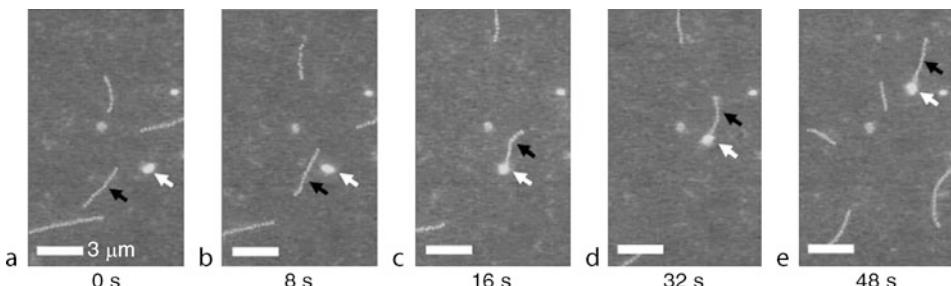
Unidirectional movement of MTs along circular tracks. (a) Transmission electron microscopy (TEM) image of the micro-patterned track equipped with arrow-shaped rectifiers. (b–d) Snapshots of the gliding MTs labeled with fluorescent dyes. MTs in the outer and inner circle are gliding clockwise and counterclockwise, respectively. MTs entering the arrowheads from the wrong direction often make 180° turns and move out in the correct direction. Each panel is reproduced with permission from Hiratsuka et al. (2001); copyright 2001, Biophysical Society.



exchange to achieve autonomous and reversible loading/unloading of specified cargos (e.g., vesicles encapsulating information molecules) at a sender/receiver and MT motility to transport the loaded cargo from a sender to a receiver. In order to use DNA hybridization and strand exchange, each gliding MT, cargo, and unloading site (receiver) is labeled with different single-stranded DNAs (ssDNAs). Note that the length of an ssDNA attached to an MT is designed to be shorter than that of the cargo, and the length of an ssDNA attached to a cargo is designed to be as long as that of the unloading site. Cargo are pooled at a given loading site (a given sender) and the ssDNA for the cargo is designed to be either complementary or noncomplementary to that of the MT. When an MT labeled with an ssDNA passes through a given loading site (● Fig. 15a–b), a cargo labeled with an ssDNA complementary to that of the MT is selectively loaded onto the gliding MT through DNA hybridization (● Fig. 15c), while cargo labeled with a noncomplementary ssDNA remain at the loading site. The cargo loaded onto the MT (i.e., an MT-cargo complex) is transported by MT motility on kinesins toward a given unloading site (● Fig. 15d–e). To achieve autonomous unloading at a given unloading site, the ssDNA attached to each unloading site is designed to be either complementary or noncomplementary to that attached to the cargo. When the MT-cargo complex passes

Fig. 15

Microscopic images of cargo loading and transport. Black and white arrows indicate an MT and a cargo, respectively. Each panel is reproduced with permission from Hiyama et al. (2008a); copyright 2008, Wiley-VCH Verlag GmbH & Co. KGaA.



through an unloading site, the cargo labeled with an ssDNA complementary to the one attached to the unloading site is selectively unloaded from the gliding MT through DNA strand exchange (Hiyama et al. 2008a, c). These results indicate that gliding MTs may load/unload cargo-vesicles at a sender/receiver through DNA hybridization and strand exchange.

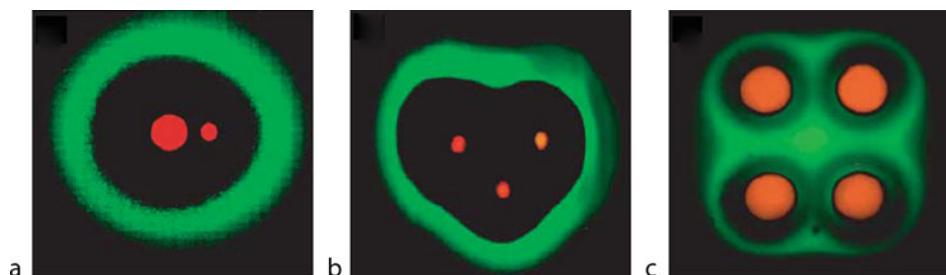
3.4 Senders and Receivers

The most straightforward approach to create senders and receivers in molecular communication is to use living biological cells. In eukaryotic cells, organelles called ribosomes synthesize some proteins that are transported to cell membranes through the vesicle transport between the endoplasmic reticulum (ER) and the Golgi apparatus. Then the transported molecules are secreted into the outside of the cell by exocytosis and diffused away (Alberts et al. 1997; Pollard and Earnshaw 2004). The diffused molecules are selectively captured by cell surface receptors. The received molecules are transduced into intracellular signals that induce cell behavior or are taken inside the cell by endocytosis, resulting in biochemical reactions such as activation of enzymes and expression of genes (Alberts et al. 1997; Pollard and Earnshaw 2004). These facts indicate that it is the fastest way to use living biological cells as senders/receivers in molecular communication, rather than creating senders/receivers artificially, because biological cells inherently have most of the required functionalities as senders/receivers in molecular communication. In that case, the issues to be solved are how to control these inherent functionalities and how to encode information onto molecules.

For instance, *Escherichia coli* (*E. coli*) bacteria produce intercellular messengers such as acyl-homoserine lactones (AHLs) and respond according to the sensed population of surrounding bacteria. It was demonstrated that genetic manipulations for these mechanisms can enable one to artificially control the fluctuation of bacterial population and bioluminescence (Basu et al. 2005; You et al. 2004). This uses a synthetic multicellular system in which genetically engineered “receiver” cells that produce fluorescent proteins were programmed to form ring-like patterns of differentiation based on chemical gradients of AHL signals that were synthesized by “sender” cells. Other patterns, such as ellipses and clovers, were achieved by

Fig. 16

Bacterial pattern formation and molecular communication. Red and green colored portions in each panel represent colonies of “sender” cells and “receiver” cells that received AHL signals sent by sender cells, respectively. Genetically engineered receiver cells formed ring-like patterns such as ellipses (a), hearts (b), and clovers (c) by placing sender cells in different configurations. Each panel is reproduced with permission from Basu et al. (2005); copyright 2005, Macmillan Publishers Ltd.



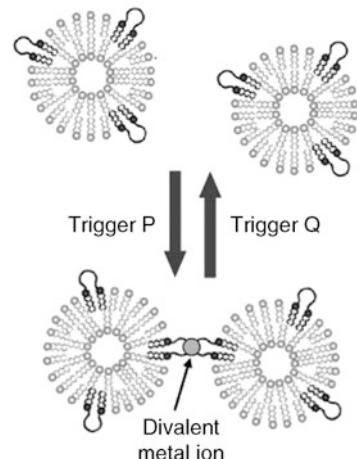
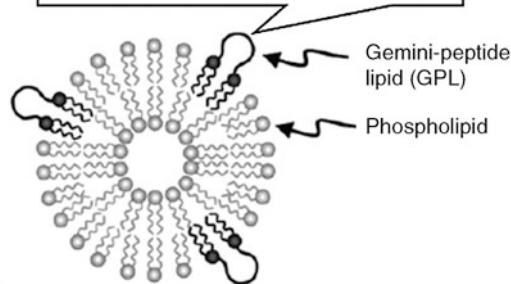
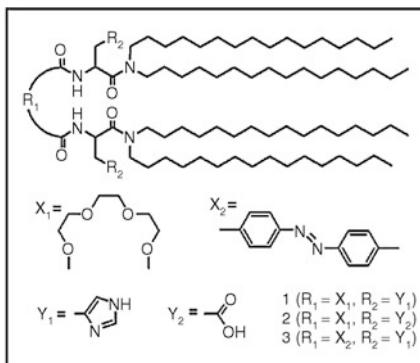
placing senders in different configurations (► Fig. 16). Such a synthetic multicellular system is an example of a molecular communication system that uses bacteria as molecular senders/receivers.

Genetically altered mutant eukaryotic cells can be also used as molecular senders/receivers. In the process of vesicle transport within eukaryotic cells such as yeasts, export molecules are selected and sorted strictly at the ER and recycled between the ER and the Golgi apparatus (Sato and Nakano 2003). If information is encoded onto the type of molecules or the concentration of molecules, the genetically altered cells, which change the sorting and secreting molecules in response to external stimuli such as temperature and light, may act as a sender in molecular communication.

In contrast, another possible approach to creating senders and receivers in molecular communication is to assemble functional molecules that are designed and synthesized artificially. For instance, a synthetic sender/receiver that uses a giant liposome embedded with gemini-peptide lipids has been proposed (Mukai et al. 2009; Sasaki et al. 2006a). A liposome is an artificially created vesicle that has a lipid bilayer membrane structure similar to cell membranes. The gemini-peptide lipids are composed of two amino acid residues, each having a hydrophobic double-tail and a functional spacer unit connecting to the polar heads of the lipid (► Fig. 17a). The liposomes embedded with the same type of gemini-peptide lipids in their lipid bilayer membranes assemble in response to specific triggers such as ions and light (► Fig. 17b) (Iwamoto et al. 2004; Sasaki et al. 2006b). Note that assembled liposomes with gemini-peptide lipids can be dissociated reversibly by applying a complementary trigger (e.g., applying UV-light for liposomal assembly and applying visible light for liposomal dissociation). The reversible liposomal dissociation and assembly mechanisms may be applied to the selective transmission and reception mechanisms of information molecules (or small liposomes that encapsulate information molecules) at a sender and a receiver (Mukai et al. 2009; Sasaki et al. 2006a), respectively (► Fig. 18). The gemini-peptide lipids are used as a molecular tag. A small liposome embedded with molecular tags acts as a container of information molecules (a molecular container) and a giant liposome embedded with a

Fig. 17

Example structures and features of gemini-peptide lipids. (a) Chemical structures of ion-responsive gemini-peptide lipids (GPL type 1 and GPL type 2) and a photo-responsive gemini-peptide lipid (GPL type 3). (b) A schematic diagram of reversible liposomal assembly and dissociation. For instance, liposomes embedded with GPL type 3 assemble/dissociate by exposure to UV-light/visible light in the presence of divalent metal ions due to the photoisomerization of azobenzene-spacer units in the GPL type 3. This figure is reproduced with permission from Iwamoto et al. (2004); copyright 2004, Elsevier Ltd.



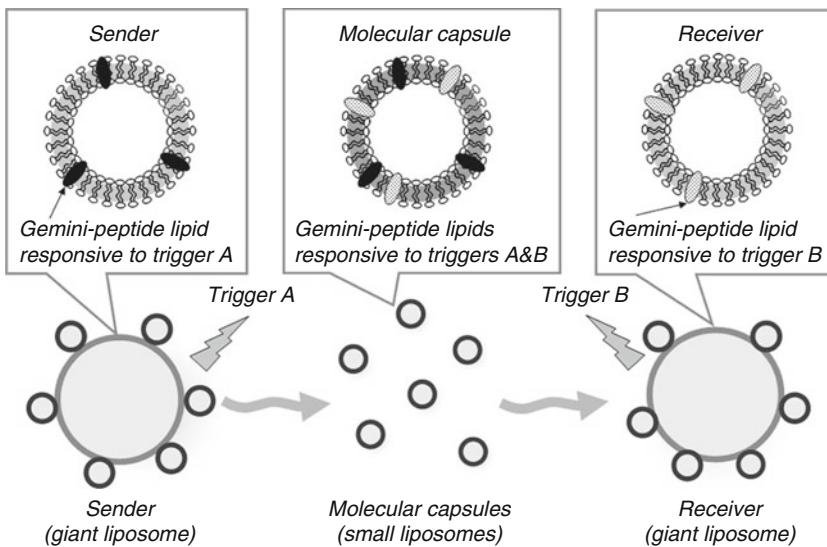
GPL	Trigger P	Trigger Q
1	Zn ²⁺	K ⁺
2	Ca ²⁺	K ⁺
3	Zn ²⁺ & UV	Zn ²⁺ & VIS

molecular tag acts as a sender/receiver. A sender/receiver is embedded with a specific molecular tag, and a molecular container whose destination is the receiver is also embedded with the same type of molecular tag. When trigger A/B is applied to the sender/receiver and the molecular container, a sender/receiver embedded with a molecular tag that is responsive to the applied trigger transmits/receives the molecular container embedded with the same type of molecular tag (► Fig. 18). This selective transmission/reception mechanism controlled by specific triggers may lead to the creation of not only unicast-type but also multicast- and broadcast-type molecular communication.

As for the biochemical reaction at a receiver, artificial signal transduction systems have been fabricated on a liposomal membrane (Fukuda et al. 2001; Mukai et al. 2009). The artificial signal transduction system is inspired by biological signal transduction that involves ligand–receptor interaction and G-protein-linked pathways. The system is composed of three molecular components: an artificial receptor, an enzyme, and bilayer-forming lipids (► Fig. 19) (Fukuda et al. 2001). In this figure, an azobenzene-containing artificial receptor is embedded in the bilayer-forming lipids (dimyristoylphosphatidylcholine: DMPC) and an enzyme (lactate dehydrogenase: LDH) is immobilized on the bilayer-forming lipids. In the

Fig. 18

A schematic diagram of a sender/receiver using a giant liposome embedded with gemini-peptide lipids and molecular capsules using small liposomes embedded with gemini-peptide lipids. By applying a complementary trigger (e.g., visible light/UV-light exposure) to the system, selective transmission/reception of molecular capsules can be performed at a sender/receiver.

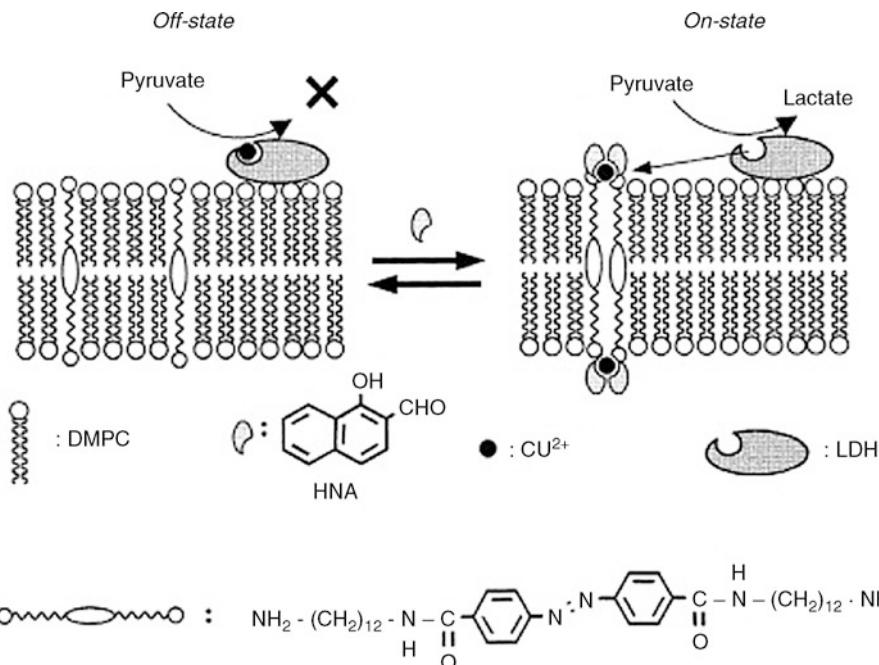


absence of an external signal (1-hydroxy-2-naphthoaldehyde: HNA), Cu^{2+} ions bind to LDH and inhibit LDH activity (corresponding to the off-state). In contrast, in the presence of HNA, Cu^{2+} ions bind to the HNA-receptor complex and activate LDH activity (corresponding to the on-state), resulting in the switching of enzymatic activity through the molecular recognition of the receptor toward the signal. Similar systems can be constructed by using gemini-peptide lipids as photo-responsive molecular tags for liposomal dissociation and assembly. In this case, the photo-responsive molecular tag embedded in the receiver can also act as an artificial receptor in an artificial signal transduction system fabricated on a liposomal membrane (Mukai et al. 2009). The photo-responsive molecular tag drastically changes Cu^{2+} ion-binding affinity through the photoisomerization of azobenzene-spacer units in the GPL type 3. Thus input of a photonic signal (UV-light/visible light exposure) to the molecular tag would be converted to amplified chemical signal output at the enzyme through the translocation of Cu^{2+} ions as a mediator between them. These results indicate that a receiver may biochemically react to the received information molecules by applying a specific trigger.

On the other hand, in the long run it may be possible to create a “white box”-type artificial cell that is only composed of known biological materials and functions. In the field of synthetic biology, some innovative challenges are going on: creating artificial organisms that do not exist or could exist in nature, and creating standardized biological “parts” to turn artisanal genetic engineering into “real” engineering (Bio Fab Group 2006). By combining these technologies, it may be possible to create an artificial cell acting as a sender/receiver in molecular communication from scratch.

Fig. 19

A schematic diagram of an artificial signal transduction system fabricated on a liposomal membrane. The system is composed of an azobenzene-containing artificial receptor, an enzyme (LDH), and bilayer-forming lipids (DMPC), achieving the switching of enzymatic activity through the molecular recognition of the receptor toward an external signal (HNA). This figure is reproduced with permission from Fukuda et al. (2001); copyright 2001, Elsevier Ltd.



3.5 Research Trends and Future Prospects

Although molecular communication is a new research area, it has received increasing attention in the areas of biophysics, biochemistry, information science, and communication engineering. As summarized in [Table 2](#), various workshops and symposia have been organized worldwide in recent years. The National Science Foundation (NSF) has recognized the importance and impact of molecular communication research, and has already started investigation into funding (NSF 2008). In the middle of 2008, the Institute of Electrical and Electronics Engineers (IEEE) Emerging Technologies Committee launched a new subcommittee to advance the concept of nanoscale networking and molecular communication (IEEE 2008).

Molecular communication research started with experimental approaches to first verify the feasibility, and now it has been extended to theoretical approaches based on information theory. For instance, achievable information rates have been calculated assuming that distinct information molecules are freely diffused (Eckford 2007). Other examples include an information-theoretic model of molecular communication based on cellular signaling (Liu and Nakano 2007), calculating the maximum capacity for the molecular communication channel between a sender and a receiver (Atakan and Akan 2007), and

Table 2**Selected major activities in molecular communication**

Date	Venue	Event
Mar. 2005	Miami (FL, USA)	Panel at IEEE INFOCOM (2005)
Oct. 2005	Huntington Beach (CA, USA)	Technical session at IEEE CCW'05
Jan. 2006	Tokyo (Japan)	International symposium
Nov. 2006	Okinawa (Japan)	Symposium at EABS & BSJ'06
Dec. 2006	Cavalese (Italy)	Panel at BIONETICS (2006)
Sep. 2007	Okazaki (Japan)	Domestic workshop
Dec. 2007	Budapest (Hungary)	Workshop at BIONETICS (2007)
Dec. 2007	Yokohama (Japan)	Symposium at biophysics conference
Feb. 2008	Arlington (VA, USA)	NSF workshop (2008)
Sep. 2008	Kawasaki (Japan)	Tutorial at IEICE society conference
Nov. 2008	Hyogo (Japan)	Workshop at BIONETICS (2008)
Aug. 2009	San Francisco (CA, USA)	Workshop at IEEE ICCN (2009)

mathematical modeling of molecular communication among floating nanomachines (acting as senders/receivers) in an aqueous environment based on probabilistic timed finite-state automata (Wiedermann and Petru 2008). These mathematical models are also now in the initial phase of research under many constraints and partly nonrealistic assumptions, but they will become sophisticated soon.

In conclusion, molecular communication is an emerging interdisciplinary research area, and, as such, it requires a lot of research effort and collaboration among biophysicists, biochemists, and information scientists. The authors of this chapter hope that many researchers will participate in and contribute to the development of molecular communication.

Acknowledgments

The first author of this book chapter would like to thank many people who executed and supported the research shown in this chapter, especially Junna Kuramochi, Hirotaka Nakagawa, Kensaku Sakamoto, Takahiro Hohsaka, Takashi Yokomori, and Satoshi Kobayashi. This work was supported in part by special coordination funds for Promoting Science and Technology from the Ministry of Education, Culture, Sports, Science and Technology of the Japanese Government, and through a grant from the Bioinformatics Research and Development of the Japanese Science and Technology Agency.

The second author of this chapter would like to thank Prof. Kazuo Sutoh and Associate Prof. Shoji Takeuchi (The University of Tokyo), Prof. Jun-ichi Kikuchi (Nara Institute of Science and Technology), Prof. Kazunari Akiyoshi and Associate Prof. Yoshihiro Sasaki (Tokyo Medical and Dental University), Prof. Tatsuya Suda (University of California, Irvine), and Mr. Yuki Moritani (NTT DOCOMO, Inc.) for their considerable help with the promotion of molecular communication research.

References

- Alberts B, Johnson A, Raff M, Walter P, Bray D, Roberts K (1997) Essential cell biology – an introduction to the molecular biology of the cell. Garland, New York
- Atakan B, Akan ÖB (2007) An information theoretical approach for molecular communication. In: Proceedings of the bio inspired models of network, information and computing systems, December 2007. Budapest, Hungary
- Bachand GD, Rivera SB, Boal AK, Gaudioso J, Liu J, Bunker BC (2004) Assembly and transport of nanocrystal CdSe quantum dot nanocomposites using microtubules and kinesin motor proteins. *Nano Lett* 4:817–821
- Bachand GD, Rivera SB, Carroll-Portillo A, Hess H, Bachand M (2006) Active capture and transport of virus particles using a biomolecular motor-driven, nanoscale antibody sandwich assay. *Small* 2:381–385
- Basu S, Gerchman Y, Collins CH, Arnold FH, Weiss R (2005) A synthetic multicellular system for programmed pattern formation. *Nature* 434:1130–1134
- Benenson Y, Paz-Ellzur T, Adar R, Keinan E, Livneh Z, Shapiro E (2001) Programmable and autonomous computing machine made of biomolecules. *Nature* 414:430–434
- Benenson Y, Adar R, Paz-Ellzur T, Livneh Z, Shapiro E (2003) DNA molecule provides a computing machine with both data and fuel. *Proc Natl Acad Sci* 100:2191–2196
- Bio Fab Group (2006) Engineering life: building a FAB for biology. *Sci Am* June:44–51
- Eckford A (2007) Achievable information rates for molecular communication with distinct molecules. In: Proceedings of the workshop on computing and communications from biological systems: theory and applications, December 2007. Budapest, Hungary
- Fukuda K, Sasaki Y, Ariga K, Kikuchi J (2001) Dynamic behavior of a transmembrane molecular switch as an artificial cell-surface receptor. *J Mol Catal B Enzym* 11:971–976
- Gardner TS, Cantor CR, Collins JJ (2000) Construction of a genetic toggle switch in *Escherichia coli*. *Nature* 403:339–342
- Goel A, Vogel V (2008) Harnessing biological motors to engineer systems for nanoscale transport and assembly. *Nat Nanotechnol* 3:465–475
- Ham TS, Lee SK, Keasling JD, Arkin AP (2008) Design and construction of a double inversion recombination switch for heritable sequential genetic memory. *PLoS ONE* 3(7):e2815
- Hess H, Clemmens J, Qin D, Howard J, Vogel V (2001) Light-controlled molecular shuttles made from motor proteins carrying cargo on engineered surfaces. *Nano Lett* 1:235–239
- van den Heuvel MGL, Dekker C (2007) Motor proteins at work for nanotechnology. *Science* 317:333–336
- Hiratsuka Y, Tada T, Oiwa K, Kanayama T, Uyeda TQP (2001) Controlling the direction of kinesin-driven microtubule movements along microlithographic tracks. *Biophys J* 81(3):1555–1561
- Hiyama S, Moritani Y, Suda T, Egashira R, Enomoto A, Moore M, Nakano T (2005a) Molecular communication. In: Proceedings of the NSTI nanotechnology conference and trade show, vol 3. May 2005. Anaheim, CA, pp 391–394
- Hiyama S, Isogawa Y, Suda T, Moritani Y, Sutoh K (2005b) A design of an autonomous molecule loading/transporting/unloading system using DNA hybridization and biomolecular linear motors. In: Proceedings of the European nano systems workshop, December 2005. Paris, France, pp 75–80
- Hiyama S, Inoue T, Shima T, Moritani Y, Suda T, Sutoh K (2008a) Autonomous loading, transport, and unloading of specified cargoes by using DNA hybridization and biological motor-based motility. *Small* 4:410–415
- Hiyama S, Moritani Y, Suda T (2008b) A biochemically-engineered molecular communication system (invited paper). In: Proceedings of the international conference on nano-networks, September 2008. Boston, MA
- Hiyama S, Takeuchi S, Gojo R, Shima T, Sutoh K (2008c) Biomolecular motor-based cargo transporters with loading/unloading mechanisms on a micro-patterned DNA array. In: Proceedings of the IEEE international conference on micro electro mechanical systems, January 2008. Tucson, AZ, pp 144–147
- Hohsaka T, Ashizuka Y, Murakami H, Sisido M (2001a) Five-base codons for incorporation of nonnatural amino acids into proteins. *Nucleic Acids Res* 29:3646–3651
- Hohsaka T, Ashizuka Y, Taira H, Murakami H, Sisido M (2001b) Incorporation of nonnatural amino acids into proteins by using various four-base codons in an *Escherichia coli* in vitro translation system. *Biochemistry* 40:11060–11064
- IEEE Emerging Technical Subcommittee on Nano-Scale, Molecular, and Quantum Networking. <http://www.comsoc.org/socsTR/org/operation/comm/subengineering.html>. Accessed Sep 2008
- Istrail S, De-Leon S, Davidson E (2007) The regulatory genome and the computer. *Dev Biol* 310:187–195
- Iwamoto S, Otsuki M, Sasaki Y, Ikeda A, Kikuchi J (2004) Gemini peptide lipids with ditopic ion-recognition

- site. Preparation and functions as an inducer for assembling of liposomal membranes. *Tetrahedron* 60:9841–9847
- Kumar NM, Gilula NB (1996) The gap junction communication channel. *Cell* 84:381–388
- Kuramochi J, Sakakibara Y (2005) Intensive in vitro experiments of implementing and executing finite automata in test tube. In: Proceedings of 11th international meeting on DNA based computers. London, Canada, pp 59–67
- Liu J-Q, Nakano T (2007) An information theoretic model of molecular communication based on cellular signaling. In: Proceedings of the workshop on computing and communications from biological systems: theory and applications, December 2007, Budapest, Hungary
- Luisi PL, Walde P (2000) Giant vesicles. Wiley, New York
- Moritani Y, Hiyama S, Suda T (2006a) Molecular communication among nanomachines using vesicles. In: Proceedings of the NSTI nanotechnology conference and trade show, vol 2, May 2006, Boston, MA, pp 705–708
- Moritani Y, Nomura S-M, Hiyama S, Akiyoshi K, Suda T (2006b) A molecular communication interface using liposomes with gap junction proteins. In: Proceedings of the bio inspired models of network, information and computing systems, December 2006, Cavalese, Italy
- Moritani Y, Hiyama S, Suda T (2007a) Molecular communication – a biochemically-engineered communication system. In: Proceedings of the frontiers in the convergence of bioscience and information technologies, October 2007, Jeju Island, Korea, pp 839–844
- Moritani Y, Nomura S-M, Hiyama S, Suda T, Akiyoshi K (2007b) A communication interface using vesicles embedded with channel forming proteins in molecular communication. In: Proceedings of the bio inspired models of network, information and computing systems, December 2007, Budapest, Hungary
- Mukai M, Maruo K, Kikuchi J, Sasaki Y, Hiyama S, Moritani Y, Suda T (2009) Propagation and amplification of molecular information using a photo-responsive molecular switch. *Supramolecular Chem* 21:284–291
- Nakagawa H, Sakamoto K, Sakakibara Y (2005) Development of an in vivo computer based on *Escherichia coli*. In: Proceedings of 11th international meeting on DNA based computers. London, Ontario, pp 68–77
- NSF Workshop on Molecular Communication (2008) Biological communications technology. <http://netre.search.ics.uci.edu/mc/nsfws08/index.html>. Accessed Feb 2008
- Panel on Nanoscale Communications. (2005) IEEE INFOCOM'05. <http://www.ieee-infocom.org/2005/parts.htm>. Accessed Mar 2005
- Panel on Nano Scale Communications and Computing. (2006) IEEE/ACM BIONETICS'06. <http://www.bionetics.org/2006/>. Accessed Dec 2006
- Păun Gh, Rozenberg G, Salomaa A (1998) DNA computing. Springer, Heidelberg
- Pollard TD, Earnshaw WC (2004) Cell biology (Updated Edition). Saunders, Philadelphia, PA
- Ramachandran S, Ernst K-H, Bachand GD, Vogel V, Hess H (2006) Selective loading of kinesin-powered molecular shuttles with protein cargo and its application to biosensing. *Small* 2:330–334
- Sakakibara Y, Hohsaka T (2003) In vitro translation-based computations. In: Proceedings of 9th international meeting on DNA based computers. Madison, WI, pp 175–179
- Sakakibara Y, Yugi K, Takagi H (2006) Implementation of AND genetic circuit in *Escherichia coli*. unpublished manuscript
- Sakakibara Y, Nakagawa H, Nakashima Y, Yugi K (2007) Implementing in vivo cellular automata using toggle switch and inter-bacteria communication mechanism. In: Proceedings of workshop on computing and communications from biological systems (in conjunction with IEEE/ACM BIONETICS'07). Budapest, Hungary. <http://www.bionetics.org/2007/cbcs.shtml>
- Sasaki Y, Hashizume M, Maruo K, Yamasaki N, Kikuchi J, Moritani Y, Hiyama S, Suda T (2006a) Controlled propagation in molecular communication using tagged liposome containers. In: Proceedings of the bio inspired models of network, information and computing systems, December 2006, Cavalese, Italy
- Sasaki Y, Iwamoto S, Mukai M, Kikuchi J (2006b) Photo- and thermo-responsive assembly of liposomal membranes triggered by a gemini peptide lipid as a molecular switch. *J Photoch Photobiol A* 183:309–314
- Sato K, Nakano A (2003) Oligomerization of a cargo receptor directs protein sorting into COPII-coated transport vesicles. *Mol Biol Cell* 14:3055–3063
- Suda T, Moore M, Nakano T, Egashira R, Enomoto A (2005) Exploratory research on molecular communication between nanomachines. In: Proceedings of the genetic and evolutionary computation conference, June 2005, Washington, DC
- Symposium on Molecular Computing and Molecular Communication (2007) New computing and communication systems using biological functions. In: 45th annual meeting of the Biophysical Society of Japan. http://www.tuat.ac.jp/~biophys07/symposium_e.html#sinpo5. Accessed Dec 2007
- Vale RD (2003) The molecular motor toolbox for intracellular transport. *Cell* 112:467–480

- Weiss R, Basu S, Hooshangi S, Kalmbach A, Karig D, Mehreja R, Netravali I (2003) Genetic circuit building blocks for cellular computation, communications, and signal processing. *Nat Comput* 2:47–84
- Weiss R, Knight T, Sussman G (2004) Genetic process engineering. In: Amos M (ed) *Cellular computing*. Oxford University Press, Oxford, UK, pp 42–72
- Wiedermann J, Petřík L (2008) Communicating mobile nano-machines and their computational power. In: Proceedings of the international conference on nano-networks, September 2008. Boston, MA
- Workshop on Computing and Communications from Biological Systems (2007) Theory and applications (in conjunction with IEEE/ACM BIONETICS'07). <http://www.bionetics.org/2007/ccbs.shtml>. Accessed Dec 2007
- Workshop on Computing and Communications from Biological Systems (2008) Theory and applications (in conjunction with IEEE/ACM BIONETICS'08). <http://www.bionetics.org/ccbs.html>. Accessed Nov 2008
- Workshop on Nano, Molecular, and Quantum Communications (in conjunction with IEEE ICCCN'09) (2009) http://cms.comsoc.org/eprise/main/SiteGen/Nano/Content/Home/NanoCom_369.html. Accessed Aug 2009
- Yokomori T, Sakakibara Y, Kobayashi S (2002) A magic pot: self-assembly computation revisited. In: Formal and natural computing, Lecture notes in computer science, vol 2300. Springer, Berlin, pp 418–429
- You L, Cox RS III, Weiss R, Arnold FH (2004) Programmed population control by cell-cell communication and regulated killing. *Nature* 428:868–871

37 Computational Nature of Gene Assembly in Ciliates

Robert Brijder¹ · Mark Daley² · Tero Harju³ · Nataša Jonoska⁴ · Ion Petre⁵ ·
Grzegorz Rozenberg^{1,6}

¹Leiden Institute of Advanced Computer Science, Universiteit Leiden,
The Netherlands

robert.brijder@uhasselt.be

²Departments of Computer Science and Biology, University of Western
Ontario, London, Ontario, Canada

daley@csd.uwo.ca

³Department of Mathematics, University of Turku, Finland
harju@utu.fi

⁴Department of Mathematics, University of South Florida, Tampa, FL,
USA

jonoska@math.usf.edu

⁵Department of Information Technologies, Åbo Akademi University,
Turku, Finland

ipetre@abo.fi

⁶Department of Computer Science, University of Colorado, Boulder, CO,
USA

rozenber@liacs.nl

1	<i>Introduction</i>	1234
2	<i>The Basic Biology of Gene Arrangement in Ciliated Protozoa</i>	1235
3	<i>Mathematical Preliminaries</i>	1239
4	<i>The Intermolecular Model for Gene Assembly</i>	1240
5	<i>The Intramolecular Model for Gene Assembly</i>	1242
6	<i>Invariant Properties of Gene Assembly</i>	1261
7	<i>Template-Guided Recombination</i>	1265
8	<i>Discussion</i>	1276

Abstract

Ciliates are a very diverse and ancient group of unicellular eukaryotic organisms. A feature that is essentially unique to ciliates is the nuclear dualism, meaning that they have two functionally different types of nuclei, the macronucleus and the micronucleus. During sexual reproduction a micronucleus is transformed into a macronucleus – this process is called *gene assembly*, and it is the most involved naturally occurring example of DNA processing that is known to us. Gene assembly is a fascinating research topic from both the biological and the computational points of view.

In this chapter, several approaches to the computational study of gene assembly are considered. This chapter is self-contained in the sense that the basic biology of gene assembly as well as mathematical preliminaries are introduced. Two of the most studied molecular models for gene assembly, *intermolecular* and *intramolecular*, are presented and the main mathematical approaches used in studying these models are discussed. The topics discussed in more detail include the string and graph rewriting models, invariant properties, template-based DNA recombination, and topology-based models. This chapter concludes with a brief discussion of a number of research topics that, because of the space restrictions, could not be covered in this chapter.

1 Introduction

The mathematical study of gene assembly in ciliates was initiated in Landweber and Kari (1999, 2002), where it was noted that the DNA rearrangements performed by ciliates have a strong computational appeal. The first results dealt with the computational capabilities of suitably defined models for gene assembly, using classical approaches from theoretical computer science, especially based on formal languages and computability theory. Shortly afterward, a parallel line of research was initiated in Ehrenfeucht et al. (2001b) and in Prescott et al. (2001a), where the focus was to study various properties of the gene assembly process itself, understood as an information processing process that transforms one genetic structure into another.

This research area has witnessed an explosive development, with a large number of results and approaches currently available. Some of them belong to computer science: models based on rewriting systems, permutations, strings, graphs, and formal languages, invariants results, computability results, etc.; see, for example, Ehrenfeucht et al. (2003a). While others, such as template-based DNA recombination, belong to theoretical and experimental biology; see, for example, Ehrenfeucht et al. (2007) and Angeleska et al. (2007).

In this chapter, several approaches and results in the computational study of gene assembly are reviewed. In [Sect. 2](#), the basic biological details of the gene assembly process as currently understood and experimentally observed are introduced. After mathematical preliminaries in [Sect. 3](#), two of the most studied molecular models for gene assembly (intermolecular and intramolecular) are introduced in [Sects. 4](#) and [5](#). Several mathematical approaches used in studying these models are also discussed. In [Sect. 6](#), some properties of the gene assembly process, called invariants, that hold independently of molecular model and assembly strategy, are discussed. In [Sect. 7](#), models for template-based DNA recombination are presented as a possible molecular implementation of the gene assembly process. The chapter is concluded with a brief discussion in [Sect. 8](#).

2 The Basic Biology of Gene Arrangement in Ciliated Protozoa

All living cells can be classified as belonging to one of two high-level groups: the *prokaryotes* or the *eukaryotes*. The prokaryotes are defined chiefly by their simple cellular organization: within a prokaryotic cell, there are no compartments or subdivisions, all of the intracellular materials (e.g., enzymes, DNA, food, waste) are contained within a single cellular membrane and are free to intermix. By contrast, eukaryotic cells contain nested membranes with many functionally and morphologically distinct organelles, the most well known being the nucleus that contains the DNA, and DNA processing machinery. All bacteria and archaea are prokaryotes while all higher multicellular organisms are eukaryotes.

The *protozoa* are single-celled eukaryotes of striking complexity; each cell functions as a complete, individual organism capable of advanced behaviors typically associated with multicellular organisms. Protozoa are equipped with extensive sensory capabilities and various species can sense temperature, light, and motion, as well as chemical and magnetic gradients. In addition to locomotion via swimming, some types of ciliated protozoa are able to coordinate legs made from fused cilia to walk along substrates in search of food. Many protozoan species are active hunters and possess sensory apparatus enabling them to identify, hunt, and consume prey.

Protozoa are found in nearly every habitat on Earth, and form a critical portion of the microbial food web. Beyond such ecological significance, the study of protozoa is fundamental to evolutionary inquiry as the protozoa represent a significant portion of eukaryotic evolutionary diversity on Earth – through their evolution they have produced unique features, one of which we study further in this chapter.

The *ciliated protozoa* (phylum Ciliophora) are a particularly interesting group due to their unique, and complex, nuclear morphology and genetics. Where most eukaryotic cells contain only one type of nucleus (sometimes in many copies), ciliate cells contain two functionally different types of nucleus: *Macronucleus* (abbreviated as *MAC*) and *Micronucleus* (*MIC*), each of which may be present in various multiplicities. For details on the many other aspects of ciliate biology which are not touched upon here, the reader can refer to Hausmann and Bradbury (1997) and Prescott (2000).

The diversity within the phylum Ciliophora is staggering and even relatively closely related species have extreme evolutionary distances, for example, the ciliates *Tetrahymena thermophila* and *Paramecium caudatum* have an evolutionary distance roughly equivalent to that between rat and corn (Prescott 1994). In this chapter, the discussion is restricted to ciliates of the subclass Stichotrichia for which the unique features that are discussed in this chapter are especially pronounced.

In Stichotrichia there is a profound difference in genome organization between the *MIC* and *MAC* on both a global level (where one considers the organization of chromosomes) and a local level (considering the organization of DNA sequence within individual genes).

On the global level, the *MIC* is diploid (there are exactly two copies of each chromosome) and composed of long chromosomes containing millions of base pairs of DNA. Each of these chromosomes contains a large number of genes, although these genes account for only a very small portion of the total sequence of the chromosomes (approximately 2–5% in stichotrichs). This long, but genetically sparse, chromosome structure is similar to that found in the nuclei of most other eukaryotes.

The global organization of the *MAC* provides a stark contrast to that of the *MIC*: most obviously, the number of copies of chromosomes in *MACs* is very large. For example, typically

more than 1,000 copies in stichotrichs, more than 10,000 in *Stylonichia*, and at the extreme end up to millions of copies of the rDNA-containing chromosome in the stichotrich *Oxytricha trifallax* (also called *Sterkiella histriomuscorum*). Along with much higher number of copies, MAC chromosomes are much shorter than those of the MIC and they contain only 1–3 genes each. Although they are small, these chromosomes are genetically dense – indeed, approximately 85% of a typical chromosome sequence contain genes. Hence, although we have a huge multiplicity of these chromosomes, the total DNA content is still much lower than in the MIC. Thus, the MIC chromosomes are long and “sparse” while the MAC chromosomes are short and “dense.”

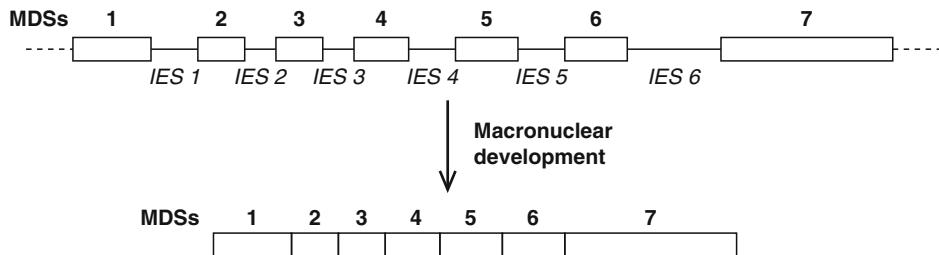
The difference in global genetic organization between the MAC and the MIC is impressive, but turns out to be much less startling than the difference in the local organization of the genes. On the local level, the MAC is a functional nucleus and, like most eukaryotic nuclei, it carries out the day-to-day genetic “housekeeping” tasks of the cell including the production of proteins from the functional MAC genes. In contrast, the genes of the MIC are not functional (not expressible as proteins) due to the presence of many noncoding sequences that break up the genes. Indeed, the macronuclear forms of ciliate genes are heavily modified from the original micronuclear configurations. MIC genes can be divided into two interleaving types of regions: *macronuclear destined segments* or MDSs and *internal eliminated sequences* or IESs. The MDSs are the regions of the MIC gene that end up in the functional MAC version of the gene, while the IESs are interspersed nongenetic regions of sequence that do not appear in the MAC version of the gene, see  Fig. 1. The MDSs are assembled, via overlapping regions called *pointers* to form the macronuclear genes. Each MDS, with the exception of the first and last, is flanked on either side by one of these pointer regions. The outgoing pointer region on one side, which we depict at the right side of MDS n has identical DNA sequence to the incoming pointer region on the left side of MDS $n + 1$. An illustration of this can be seen in  Fig. 2; note that the outgoing pointer of MDS n is identical to the incoming pointer of MDS $(n + 1)$.

In some ciliates, in addition to the appearance of IESs, the MIC genes have the further complication that the MDSs do not appear in the same order as they do in the functional MAC gene. That is, the order of MDSs in the MIC gene is *scrambled* relative to their “orthodox” order in the MAC gene and may even contain segments that are inverted (i.e., rotated 180°). A schematic representation of the gene actin I from *O. trifallax* is shown in  Fig. 3 (see Prescott and Greslin (1992)). Note that the numbers enumerating MDSs refer to the orthodox order of the MDSs in the macronuclear version of the gene, and a bar is used to denote MDSs that are inverted.

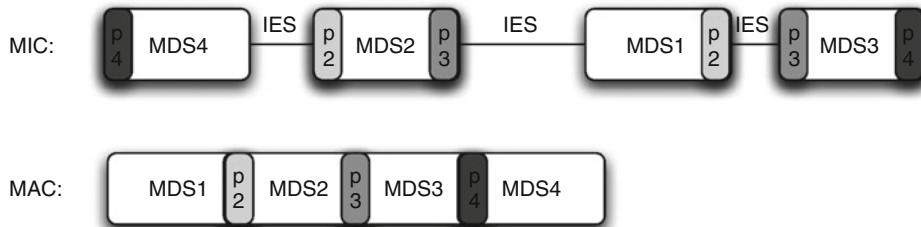
Ciliates reproduce asexually but, during times of environmental stress (e.g., starving due to lack of nutrients) can also undergo the (nonreproductive) sexual activity of *conjugation*. Rather than producing offspring, the purpose of conjugation in ciliates is to give each cell a “genetic facelift” by incorporating new DNA from the conjugating partner cell. The specifics of conjugation are highly species dependent, but almost all ciliate species follow the same basic outline, the generic form of which is illustrated in  Fig. 4. Two cells form a cytoplasmic bridge while meiotically dividing their diploid MICs into four haploid MICs. Each cell sends one haploid MIC across the bridge to the conjugating partner. The cell then combines one of its own haploid MICs with the newly arrived haploid MIC from the partner to form a new fused diploid MIC. Also, each cell destroys its old MAC and remaining haploid MICs, the fused MIC divides into two parts, one of which will be the new MIC and the other one will develop into (will be transformed into) the new MAC.

Fig. 1

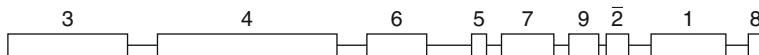
A diagram of the arrangement of six IESs and seven MDSs in the micronuclear (top) and the macronuclear (bottom) gene encoding β TP protein. During macronuclear development the IESs are excised and the MDSs are ligated (by overlapping of the ends) to yield a macronuclear gene. MDSs are rectangles. IESs are line segments between rectangles (from Prescott and DuBois (1996)).

**Fig. 2**

Schematic representation of MIC gene (top) and associated MAC gene (bottom) with pointers indicated on the MDSs.

**Fig. 3**

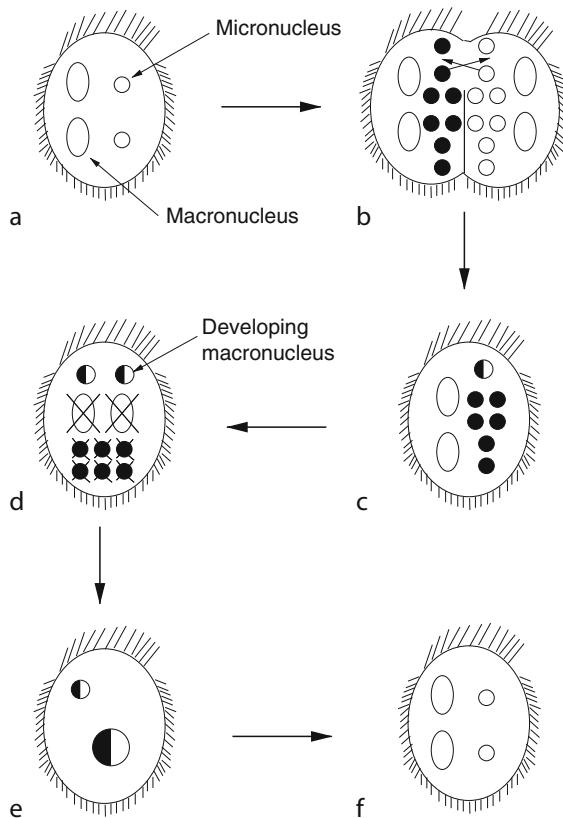
Schematic representation of the structure of the micronuclear gene-encoding actin protein in the stichotrich *Sterkiella nova*. The nine MDSs are in a scrambled disorder (from Prescott and Greslin (1992)).



The topic of interest in this chapter is this transformation of a new MIC into a new MAC. We focus in particular on the transformations of the individual genes – a process called *gene assembly*. This process is fascinating both from the biological and information processing points of view. Indeed, gene assembly is the most complex example of DNA processing known in nature, underscored by the dramatic difference in the structure, and composition, of the MIC and MAC genomes explained above. To form a functional macronuclear gene, all IESs must be excised, all MDSs must be put in their original (orthodox) order with inverted segments switched to their proper orientation.

Fig. 4

Conjugation in stichotrichs: (a) A stichotrich with two macronuclei and two micronuclei. (b) Two stichotrichs have joined and formed a cytoplasmic channel. The two diploid micronuclei have each formed four haploid micronuclei. (c) The two cells exchanged haploid micronuclei, and the two organisms have separated. The exchanged haploid micronucleus has fused with a resident haploid micronucleus forming a new diploid micronucleus (half white and half black). (d) The new diploid micronucleus has divided by mitosis. The unused haploid micronuclei (six) and the two macronuclei are degenerating. (e) One of the new daughter micronuclei has developed into a new macronucleus. The old macronucleus and the unused haploid micronuclei have disappeared. (f) Conjugation has been completed. The micronucleus and macronucleus have divided, yielding the appropriate nuclear numbers (in this case, two MICs and two MACs) (from Ehrenfeucht et al. (2003a)).



Understanding and investigating the computational nature of the gene assembly process, and its biological implications, becomes the central focus of this chapter. For further details on the biology of gene assembly we refer to Jahn and Klobutcher (2000), Prescott (1999, 2000), Möllenbeck et al. (2008), and Ehrenfeucht et al. (2007) and, in particular to Ehrenfeucht et al. (2003a), which contains chapters explaining basic biology, basic cell biology, and the basic biology of ciliates written specifically for motivated computer scientists.

3 Mathematical Preliminaries

In this section we fix basic mathematical notions and terminology used in this chapter.

3.1 Strings

For an alphabet Σ , let Σ^* denote the set of all strings over Σ . Let Λ denote the *empty string*. For a string $u \in \Sigma^*$, we denote by $|u|$ its length, that is, the number of letters u consists of.

A string u is a *substring* of a string v , if $v = xuy$, for some $x, y \in \Sigma^*$. In this case, we denote $u \leq v$. We say that u is a *conjugate* of v if $v = w_1w_2$ and $u = w_2w_1$, for some $w_1, w_2 \in \Sigma^*$.

For an alphabet Σ , let $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ be a signed disjoint copy of Σ . The set of all strings over $\Sigma \cup \bar{\Sigma}$ is denoted by $\Sigma^{\pm} = (\Sigma \cup \bar{\Sigma})^*$. A string $v \in \Sigma^{\pm}$ is called a *signed string over Σ* . We adopt the convention that $\bar{\bar{a}} = a$ for each letter $a \in \Sigma$.

Let $v \in \Sigma^{\pm}$ be a signed string over Σ . We say that a letter $a \in \Sigma \cup \bar{\Sigma}$ occurs in v , if a or \bar{a} is a substring of v . Let $\text{dom}(v) \subseteq \Sigma$, called the *domain* of v , be the set of the (unsigned) letters that occur in v .

Example 1 For the alphabet $\Sigma = \{2, 3\}$ of pointers, $\bar{\Sigma} = \{\bar{2}, \bar{3}\}$. Here $u = 233 \in \Sigma^* \subseteq \Sigma^{\pm}$, while $v = 2\bar{3}\bar{2} \in \Sigma^{\pm}$ is a signed string over Σ for which $\text{dom}(v) = \{2, 3\}$ although 3 is not a substring of v .

The signing $a \mapsto \bar{a}$ of letters extends to strings in a natural way: for a signed string $u = a_1a_2 \dots a_n \in \Sigma^{\pm}$, with $a_i \in \Sigma \cup \bar{\Sigma}$ for each i , let the *inversion* of u be

$$\bar{u} = \bar{a}_n\bar{a}_{n-1} \dots \bar{a}_1 \in \Sigma^{\pm}$$

For any set of strings $S \subseteq \Sigma^{\pm}$, we denote $\bar{S} = \{\bar{u} \mid u \in S\}$. For two strings $u, v \in \Sigma^{\pm}$, we say that u and v are *equivalent*, denoted $u \approx v$, if u is a conjugate of either v or \bar{v} .

For an alphabet Σ , let $\|\cdot\|$ be the substitution that unsigns the letters: $\|a\| = a = \|\bar{a}\|$. Mapping $\|\cdot\|$ extends to Σ^{\pm} in the natural way. A signed string v over Σ is a *signing* of a string $u \in \Sigma^*$, if $\|v\| = u$. A signed string u , where each letter from Σ occurs exactly once in u , is called a *signed permutation*.

For two alphabets Σ and Δ , a mapping $f : \Sigma^{\pm} \rightarrow \Delta^{\pm}$ is called a *morphism* if $f(uv) = f(u)f(v)$ and $f(\bar{u}) = \bar{f(u)}$. If $\Delta \subseteq \Sigma$, a morphism $f : \Sigma^* \rightarrow \Delta^*$ is called a *projection* if $f(x) = x$ for $x \in \Delta$ and $f(x) = \lambda$, for $x \in \Sigma \setminus \Delta$.

Example 2 For the alphabet $\Sigma = \{2, 3, 4, 5\}$, there are $2^4 \cdot 4! = 384$ signed permutations. The signed strings 2 3 4 5 and $\bar{4}\bar{2}\bar{5}3$ are among them.

Throughout this chapter we $\{2, 3, \dots\}$ the set of *pointers*.

3.2 Graphs

For a finite set V , let $E(V) = \{\{x, y\} \mid x, y \in V, x \neq y\}$ be the set of all unordered pairs of different elements of V . A (*simple*) *graph* is an ordered pair $G = (V, E)$, where V and E are finite sets of

vertices, and *edges*, respectively. If $e = \{x, y\} \in E$ is an edge, then the vertices x and y are the *ends* of e . In this case, x and y are *adjacent* in G . If $V = \emptyset$, then we denote $G = \emptyset$ and call it the *empty graph*.

For a vertex x in G , let $N_G(x) = \{y \mid \{x, y\} \in E\}$ be the *neighborhood* of x in G . A vertex x is *isolated* in G , if $N_G(x) = \emptyset$.

A *signed graph* $G = (V, E, \sigma)$ consists of a simple graph (V, E) together with a labeling $\sigma : V \rightarrow \{-, +\}$ of the vertices. A vertex x is said to be *positive*, if $\sigma(x) = +$; otherwise x is *negative*. We write $x^{\sigma(x)}$ to indicate that the vertex x has sign $\sigma(x)$.

Let $G = (V, E, \sigma)$ be a signed graph. For a subset $A \subseteq V$, its *induced subgraph* is the signed graph $(A, E \cap E(A), \sigma)$, where σ is restricted to A . The *complement* of G is the signed graph $G^c = (V, E(V) \setminus E, \sigma^c)$, where $\sigma^c(x) = +$ if and only if $\sigma(x) = -$. Also, let $\text{loc}_x(G)$ be the signed graph obtained from G by replacing the subgraph induced by $N_G(x)$ by its complement. For a subset $A \subseteq V$, we denote by $G - A$ the subgraph induced by $V \setminus A$.

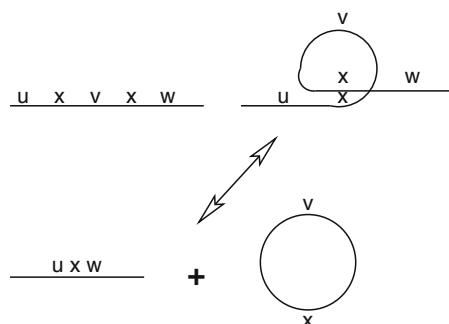
A *multigraph* is a (undirected) graph $G = (V, E, \varepsilon)$, where parallel edges are possible. Therefore, E is a finite set of edges and $\varepsilon : E \rightarrow \{\{x, y\} \mid x, y \in V\}$ is the *endpoint mapping*. We allow $x = y$, and therefore edges can be of the form $\{x, x\} = \{x\}$ – an edge of this form should be seen as a “loop” for x .

4 The Intermolecular Model for Gene Assembly

The very first formal model of the ciliate gene assembly process was the *intermolecular model* proposed in Landweber and Kari (1999, 2002). In this section we introduce a string-based formalization of the model and we refer to Landweber and Kari (1999) for the biological details of the model. Given a string $uxvxw$, where u, x, v, w are nonempty substrings, the authors defined the following intramolecular operation: $uxvxw \rightarrow \{uxw, \cdot vx\}$ where \cdot denotes that the string vx is circular. Intuitively, this models the action of a strand of DNA, $uxvxw$, looping over onto itself and aligning the regions containing the subsequence x . With the x 's aligned, the DNA strand can undergo recombination, yielding the two products uxw and $\cdot vx$. Likewise, the inverse, intermolecular, operation was also defined: $\{uxw, \cdot vx\} \rightarrow uxvxw$. These operations are illustrated in Fig. 5

Fig. 5

The (reversible) recombination of $uxvxw \leftrightarrow \{uxw, \cdot vx\}$ in the intermolecular model for gene assembly of Landweber and Kari (1999, 2002).



Example 3 Consider a hypothetical micronuclear gene with 4 MDSs where the MDSs come in the order $M_1M_2M_4M_3$. If we denote each MDS by its pair of incoming/outgoing pointers and/or markers, we can then denote the whole micronuclear gene as $\delta = (b, p_2)(p_2, p_3)(p_4, e)(p_3, p_4)$. (For more details on formal representation of ciliate genes, see [Sect. 5.](#)) An assembly strategy for this gene in the intermolecular model is the following (we indicate for each operation the pointer on which the molecule is aligned):

$$\begin{aligned}\delta &\xrightarrow{p_3} (b, p_2)(p_2, p_3, p_4) + \cdot(p_4, e)p_3 \xrightarrow{p_4} (b, p_2)(p_2, p_3, p_4, e)p_3p_4 \\ &\xrightarrow{p_2} (b, p_2, p_3, p_4, e)p_3, p_4 + \cdot p_2\end{aligned}$$

As indicated by the formal notation above, the gene gets assembled with copies of pointers p_3 and p_4 following the assembled gene and a copy of pointer p_2 placed on a separate circular molecule. Note that the notation above ignores all IESs of the micronuclear and of the assembled gene.

The obvious difficulty with the intermolecular model is that it cannot deal with DNA molecules in which a pointer is inverted – this is the case, for example, for the actin I gene in *S. nova*. Nevertheless, we can show that inverted pointers can be handled in this model, provided the input molecule (or its MDS–IES descriptor) is available in two copies. Moreover, we consider all linear descriptors modulo inversion. The first assumption is essentially used in research on the intermolecular model, see Kari et al. ([1999](#)), Kari and Landweber ([1999](#)), and Landweber and Kari ([2002](#)). The second assumption is quite natural whenever double-stranded DNA molecules are modeled.

Example 4 Consider the micronuclear actin gene in *Sterkiella nova*, see [Fig. 3](#). Denoting each of its MDSs by the pair of incoming/outgoing pointers and/or markers similarly to our previous example, the gene may be written as

$$\delta = (p_3, p_4)(p_4, p_5)(p_6, p_7)(p_5, p_6)(p_7, p_8)(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)(p_8, p_9)$$

Then δ can be assembled in the intermolecular model as follows:

$$\begin{aligned}\delta &\xrightarrow{p_5} (p_3, p_4)(p_4, p_5, p_6)(p_7, p_8)(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)(p_8, p_9) + \cdot p_5(p_6, p_7) \\ &\xrightarrow{p_8} (p_3, p_4)(p_4, p_5, p_6)(p_7, p_8, p_9) + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) + \cdot p_5(p_6, p_7) \\ &\xrightarrow{p_4} (p_3, p_4, p_5, p_6)(p_7, p_8, p_9) + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) + \cdot p_5(p_6, p_7) \\ &\xrightarrow{p_7} (p_3, p_4, p_5, p_6)p_7p_5(p_6, p_7, p_8, p_9) + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) \\ &\xrightarrow{p_6} (p_3, p_4, p_5, p_6, p_7, p_8, p_9) + \cdot p_6p_7p_5 + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) \\ &\xrightarrow{p_9} (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)p_8p_9 + \cdot p_6p_7p_5 + \cdot p_4\end{aligned}$$

Assuming that $\overline{\delta}$ is also available, the assembly continues as follows. Here, for a (circular) string τ , we use 2τ to denote $\tau + \tau$:

$$\begin{aligned}
& \delta + \bar{\delta} \longrightarrow \dots \longrightarrow (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\bar{p}_3, \bar{p}_2)(b, p_2)p_8p_9 \\
& \quad + \bar{p}_9 \bar{p}_8(\bar{p}_2, \bar{b})(p_2, p_3)(\bar{e}, \bar{p}_9, \bar{p}_8, \bar{p}_7, \bar{p}_6, \bar{p}_5, \bar{p}_4, \bar{p}_3) \\
& \quad + 2 \cdot p_6p_7p_5 + 2 \cdot p_4 \\
& \xrightarrow{p_2} (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\bar{p}_3, \bar{p}_2)(b, p_2, p_3)(\bar{e}, \bar{p}_9, \bar{p}_8, \bar{p}_7, \bar{p}_6, \bar{p}_5, \bar{p}_4, \bar{p}_3) \\
& \quad + \bar{p}_9 \bar{p}_8(\bar{p}_2, \bar{b})p_2p_8p_9 + 2 \cdot p_6p_7p_5 + 2 \cdot p_4 \\
& \xrightarrow{\bar{p}_2} (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\bar{p}_3, \bar{p}_2, \bar{b})p_2p_8p_9 \\
& \quad + \bar{p}_9 \bar{p}_8\bar{p}_2(b, p_2, p_3)(\bar{e}, \bar{p}_9, \bar{p}_8, \bar{p}_7, \bar{p}_6, \bar{p}_5, \bar{p}_4, \bar{p}_3) \\
& \quad + 2 \cdot p_6p_7p_5 + 2 \cdot p_4 \\
& \xrightarrow{\bar{p}_3} (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)\bar{p}_3 \\
& \quad + \bar{p}_9 \bar{p}_8\bar{p}_2(b, p_2, p_3)(\bar{e}, \bar{p}_9, \bar{p}_8, \bar{p}_7, \bar{p}_6, \bar{p}_5, \bar{p}_4, \bar{p}_3, \bar{p}_2, \bar{b})p_2p_8p_9 \\
& \quad + 2 \cdot p_6p_7p_5 + 2 \cdot p_4 \\
& \xrightarrow{p_3} p_3(\bar{e}, \bar{p}_9, \bar{p}_8, \bar{p}_7, \bar{p}_6, \bar{p}_5, \bar{p}_4, \bar{p}_3, \bar{p}_2, \bar{b})p_2p_8p_9 \\
& \quad + \bar{p}_9 \bar{p}_8\bar{p}_2(b, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)\bar{p}_3 + 2 \cdot p_6p_7p_5 + 2 \cdot p_4 \\
& = 2\bar{p}_9 \bar{p}_8\bar{p}_2(b, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)\bar{p}_3 + 2 \cdot p_6p_7p_5 + 2 \cdot p_4
\end{aligned}$$

This calculation shows that the gene is eventually assembled, with p_3 preceding the assembled gene and $p_2p_8p_9$ following it, and that two circular molecules are excised during the assembly.

We refer to [Sect. 5](#) for intramolecular assembly strategies of the actin I gene in *S. nova* and to [Sect. 6](#) for a set of properties that are independent of particular assembly strategies, called invariants. The DNA sequences of all molecules produced by gene assembly are particular invariants of the process. We refer to Harju et al. (2004a) for a detailed discussion on intermolecular assembly strategies and a comparison to intramolecular assemblies.

Consider now contextual versions of the operations. We define rules of the form (p, x, q) (p', x, q') with the intended semantics that p (p' , resp.) and q (q' , resp.) represent contexts flanking x (x' , resp.). These contexts are not directly involved in the recombination process, but instead regulate it: recombination may only take place if x (x' , resp.) is flanked by p (p' , resp.) and q (q' , resp.). Our intramolecular operation now becomes $uxvxw \rightarrow \{uxw, \cdot vx\}$, where $u = u'p, v = qv' = v''p', w = q'w'$, with similar constraints for the intermolecular operation.

It was shown in Landweber and Kari (2002) that iterated nondeterministic application of these contextual rules to an initial axiom string yields a computing system with the generative power of a Turing machine.

5 The Intramolecular Model for Gene Assembly

5.1 Molecular Model

The *intramolecular model* was introduced in Ehrenfeucht et al. (2001b) and Prescott et al. (2001a). It consists of a set of three irreversible molecular operations explaining the excision of IESs and the unscrambling and ligation of MDSs during the MIC–MAC development. All

three operations postulate the folding of a DNA molecule into a specific pattern that allows the alignment of some pointers. Subsequent DNA recombination on those pointers leads to the MDSs (and IESs) being rearranged. Each of the three operations is described in the following.

5.1.1 Loop, Direct-Repeat Excision (in Short, **ld**)

The **ld** operation is applicable to a DNA molecule having two occurrences of a pointer, say p , on the same strand. We say in this case that pointer p has a *direct repeat* along the molecule. The molecule is then folded into a *loop* (► Fig. 6a) so that the two occurrences of p are aligned. Recombination on p is thus facilitated (► Fig. 6b) and as a result, a linear molecule and a circular molecule are obtained, (► Fig. 6c). Each of the two molecules has an occurrence of p .

5.1.2 Hairpin, Inverted-Repeat Excision/Reinsertion (in Short, **hi**)

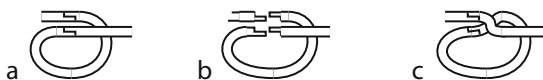
The **hi** operation is applicable to a DNA molecule having two occurrences of a pointer, say p , on different strands of the molecule. We say in this case that pointer p has an *inverted repeat* along the molecule. Then the molecule is folded into a *hairpin* (► Fig. 7a) so that the two occurrences of p have a direct alignment. Recombination on p is thus facilitated (► Fig. 7b), and as a result, a new linear molecule is obtained, where one block of nucleotides has been inverted (► Fig. 7c).

5.1.3 Double Loop, Alternating Direct-Repeat Excision (in Short, **dlad**)

The **dlad** operation is applicable to a DNA molecule having two pointers, say p and q , each with two occurrences. All four pointer occurrences should be on the same strand. Moreover, pointer p has one occurrence in-between the two occurrences of q and one occurrence outside

► Fig. 6

Illustration of the ld-rule. (a) The molecule is first folded into a loop, aligning the two direct repeats of a pointer. (b) Recombination is facilitated on the two occurrences of the pointer. (c) One linear and one circular molecule are produced as a result of the operation (from Harju et al. (2004b)).



► Fig. 7

Illustration of the hi-rule. (a) The molecule is first folded into a hairpin, aligning the two inverted repeats of a pointer. (b) Recombination is facilitated on the two occurrences of the pointer. (c) A new linear molecule is produced as a result of the operation (from Harju et al. (2004b)).

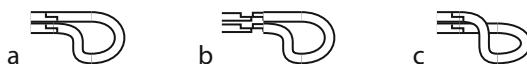
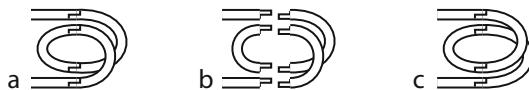


Fig. 8

Illustration of the dlad-rule. (a) The molecule is first folded into a double loop, simultaneously aligning two pairs of pointer occurrences. (b) Recombination is facilitated on both pointer alignments. (c) A new linear molecule is produced as a result of the operation (from Harju et al. (2004b)).



them. (By consequence, the same holds true for pointer q with respect to the two occurrences of pointer p .) The molecule is then folded into a double loop (► Fig. 8a), so that the two occurrences of p and the two occurrences of q are simultaneously aligned. Recombination events on p and on q are facilitated (► Fig. 8b), and as a result, a new linear molecule is obtained, see ► Fig. 8c.

5.1.4 Discussion

The {Id, hi, dlad} model is often referred to as the *intramolecular model*, to stress that in this model, the input to which operations are applied is always a single molecule. (In contrast, the model presented in ► Sect. 4 is referred to as the *intermolecular model*.) It is important to note however that Id yields as an output two molecules. As such, for the intramolecular assembly to succeed, that is, to assemble all MDSs, it is essential that all coding blocks remain within one of the molecules produced by Id. This gives two restrictions for applying Id on a pointer p in a successful gene assembly:

- (i) No MDSs exist in-between the two occurrences of p . (Equivalently, only one, possibly composite, IES exists in-between the two occurrences of p .) We say in this case that we have a *simple* application of Id. As a result, a (possibly composite) IES is excised as a circular molecule and all MDSs remain on the resulting linear molecule.
- (ii) All MDSs are placed in-between the two occurrences of p . We say in this case that we have a *boundary* application of Id. As a result, all MDSs are placed on the resulting circular molecule. The final assembled gene will be a circular molecule. It has been shown in Ehrenfeucht et al. (2001a, 2003a) that using a boundary application of Id can be postponed to the last step of any successful assembly. In this way, in-between the two occurrences of p there is only one (possibly composite) IES, similarly as in the case of simple Id.

The mechanistic details of the alignment and recombination events postulated by the three molecular operations Id, hi, and dlad are not indicated in the original proposal for the intramolecular model. Two mechanisms were later proposed in Prescott et al. (2001b) and in Angeleska et al. (2007). The details of both are discussed in ► Sect. 7.

5.2 String and Graph Representations for Ciliate Genes

We present three different formalizations for the gene assembly process: signed permutations, legal strings, and overlap graphs. The first two of these are linear in the sense that the order of the MDSs can be readily seen from the presentation. On the other hand, from the overlap

graphs the order of the components is more difficult to capture. The gene assembly process will, however, be different in nature for signed permutations as for legal strings as well as for overlap graphs. Permutations need to be sorted while strings and graphs need to be reduced to empty string and graph, respectively. Another formalization in terms of *descriptors* is given in [Sect. 6](#).

5.2.1 Signed Permutations and Legal Strings

Each micronuclear arrangement of MDSs and IESs can be represented as a signed permutation over the alphabet $\{M_1, M_2, \dots, M_\kappa\}$, for an integer $\kappa \geq 1$ corresponding to the number of macronuclear destined sequences that assemble to a functional gene. We identify such a string with a signed permutation over the index set $\{1, 2, \dots, \kappa\}$.

Example 5 Consider the micronuclear arrangement of the actin I gene of *Sterkiella nova*: $M_3 M_4 M_6 M_5 M_7 M_9 \bar{M}_2 M_1 M_8$. This is represented as the signed permutation $\alpha = 3 4 6 5 7 9 \bar{2} 1 8$.

The gene assembly process is equivalent to sorting a signed permutation in proper order, that is, in the order $p(p+1) \dots \kappa 1 \dots (p-1)$ or its inverse for suitable p (and κ , the number of MDSs in the micronuclear gene). If $p = 1$ here, then the MDSs are linearly ordered; otherwise they are cyclically ordered.

Representation by strings will preserve the order of the MDSs that are coded as “pairs of pointers.”

A string $v \in \Sigma^*$ over an alphabet Σ is said to be a *double occurrence string*, if every letter $a \in \text{dom}(v)$ occurs exactly twice in v . A signing of a nonempty double occurrence string is a *legal string*. A letter $a \in \Sigma \cup \bar{\Sigma}$ is *positive* in a legal string $v \in \Sigma^*$, if v contains both a and \bar{a} ; otherwise, a is *negative* in v .

Example 6 Consider the legal string $u = 2 4 3 \bar{2} \bar{5} 3 4 5$ of pointers. Pointers 2 and 5 are positive in u , while 3 and 4 are negative in u . On the other hand, the string $w = 2 4 3 \bar{2} \bar{5} 3 5$ is not legal, since 4 has only one occurrence in w .

Let $u = a_1 a_2 \dots a_n \in \Sigma^*$ be a legal string over Σ with $a_i \in \Sigma \cup \bar{\Sigma}$ for each i . For $a \in \text{dom}(u)$, let $1 \leq i < j \leq n$ be such that $\|a_i\| = a = \|a_j\|$. Then the substring

$$u_{(a)} = a_i a_{i+1} \dots a_j$$

is called the *a-interval* of u . Two different letters $a, b \in \Sigma$ are said to *overlap* in u , if the *a*-interval and the *b*-interval of u overlap: if $u_{(a)} = a_{i_1} \dots a_{j_1}$ and $u_{(b)} = a_{i_2} \dots a_{j_2}$, then either $i_1 < i_2 < j_1 < j_2$ or $i_2 < i_1 < j_2 < j_1$.

Example 7 Let $u = 2 4 3 5 3 \bar{2} \bar{6} \bar{5} 4 6$ be a signed string of pointers. The 2-interval of u is the substring $u_{(2)} = 2 4 3 5 3 \bar{2}$, and hence pointer 2 overlaps with 4 and 5 but not with 3 or 6. Similarly, for example, $u_{(4)} = 4 3 5 3 \bar{2} \bar{6} \bar{5} 4$ and hence 4 overlaps with 2 and 6.

A signed permutation will be represented by a legal string using the following substitution $\varrho_\kappa : \{1, 2, \dots, \kappa\}^* \rightarrow \{2, 3, \dots, \kappa\}^*$

$$\varrho_\kappa(1) = 2, \quad \varrho_\kappa(\kappa) = \kappa, \quad \varrho_\kappa(p) = p(p+1) \text{ for } 2 \leq p < \kappa$$

and $\varrho_\kappa(\overline{p}) = \overline{\varrho_\kappa(p)}$ for each p with $1 \leq p \leq \kappa$.

Example 8 Consider the signed permutation α from Example 5. We have

$$\varrho_9(\alpha) = 34\ 45\ 67\ 56\ 78\ 9\ \overline{3}\ \overline{2}\ 2\ 89.$$

5.2.2 Overlap Graphs

The focus now shifts to representations by graphs. We use signed graphs to represent the structure of overlaps of letters in legal strings as follows: let $v \in \Sigma^*$ be a legal string. The *overlap graph* of v is the signed graph $G_v = (\text{dom}(v), E, \sigma)$ such that

$$\sigma(x) = \begin{cases} +, & \text{if } x \text{ is positive in } v \\ -, & \text{if } x \text{ is negative in } v \end{cases}$$

and

$$\{x, y\} \in E \iff x \text{ and } y \text{ overlap in } v$$

Example 9 Consider the legal string $v = 3\ 4\ \overline{5}\ 2\ \overline{3}\ \overline{2}\ 4\ 5$ of pointers. Then its overlap graph G_v is given in [Fig. 9](#).

Overlap graphs of double occurrence strings are also known as *circle graphs*.

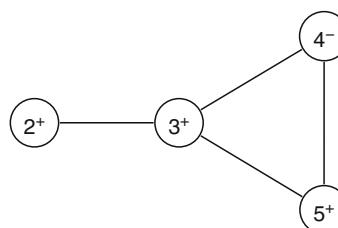
Example 10 Notice that the mapping $w \mapsto G_w$ of legal strings to overlap graphs is not injective. The following eight legal strings of pointers have the same overlap graph (of one edge): $2\overline{3}23, \overline{3}232, 23\overline{2}3, 32\overline{2}3, \overline{2}323, 32\overline{3}2, \overline{2}3\overline{2}3, 32\overline{3}2$. For a more complicated example, we mention that the strings $v_1 = 23342554$ and $v_2 = 35242453$ define the same overlap graph.

5.2.3 Reduction Graph

Recall that legal strings represent the MIC form of genes. We now introduce a graph, called the *reduction graph*, that represents the MAC form of a gene and the other molecules obtained as results of the assembly, given a legal string (the MIC form of that gene). In this way, the

[Fig. 9](#)

The overlap graph of the signed string $v = 3\ 4\ \overline{5}\ 2\ \overline{3}\ \overline{2}\ 4\ 5$.



reduction graph represents the end result after recombination on all pointers. First, we define a *2-edge colored graph* as a tuple (V, E_1, E_2, f, s, t) , where V are the vertices, $s, t \in V$ are called *source* and *target*, and $f: V \setminus \{s, t\} \rightarrow \Gamma$ is a vertex labeling function with Γ a finite set of vertex labels. There are two (not necessarily disjoint) sets of undirected edges E_1 and E_2 . We let $\text{dom}(G)$ be the range of f , and say that 2-edge colored graphs G and G' are *isomorphic*, denoted $G \approx G'$, when they are equal up to a renaming of the vertices. However, we require that the labels of the identified vertices be equal, and that the sources and targets of G and G' be identified.

A reduction graph (Brijder et al. 2006) is a 2-edge colored graph where the two types of edges E_1 and E_2 are called *reality edges* and *desire edges* respectively. Moreover, each vertex, except s and t , is labeled by an element of $\Delta_\kappa = \{2, 3, \dots, \kappa\}$. As an example, consider the representation of legal string $u = \bar{2}\bar{7}47353\bar{4}2656$ over $\Pi_\kappa = \Delta_\kappa \cup \bar{\Delta}_\kappa$ given in [Fig. 10](#). This legal string will be used as our running example.

A reduction graph \mathcal{R}_u for a legal string u is defined in such a way that (1) each occurrence of a pointer of u appears twice (in unbarred form) as a label of a vertex in the graph to represent both sides of the pointer in the representation of u , (2) the reality edges (depicted as “double edges” to distinguish them from the desire edges) represent the segments between the pointers, (3) the desire edges represent which segments should be glued to each other when recombination operations are applied on the corresponding pointers. To enforce this last requirement, positive pointers are connected by crossing desire edges (cf. pointers 4 and 7 in [Fig. 11](#)), while negative pointers are connected by parallel desire edges. The vertices s and t represent the left end and the right end, respectively. Note that, since the reduction graph is fixed for a given u , the end product after recombination is fixed as well. The notion of reduction graph is similar to the breakpoint graph (or reality-and-desire diagram) known from the theory of sorting by reversal, see, for example, Setubal and Meidanis (1997) and Pevzner (2000). A formal definition of reduction graph is found, for example, in Brijder et al. (2006). The reduction graph for string $u = \bar{2}\bar{7}47353\bar{4}2656$ is in [Fig. 11](#).

In depictions of reduction graphs, the vertices (except for s and t) will be represented by their labels, because the exact identities of the vertices are not essential here – we consider reduction graphs up to isomorphism. Note that the reduction graph is defined for the general concept of legal strings. Therefore, the reduction graph represents the end product after recombination of arbitrary sequences of pointers (which by definition come in pairs) – not only those that correspond to sequences of MDSs.

Fig. 10

The representation of $u = \bar{2}\bar{7}47353\bar{4}2656$.

2	$\bar{7}$	4	7	3	5	3	$\bar{4}$	2	6	5	6
---	-----------	---	---	---	---	---	-----------	---	---	---	---

Fig. 11

The reduction graph for $u = \bar{2}\bar{7}47353\bar{4}2656$.

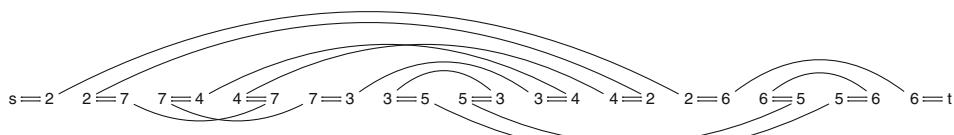
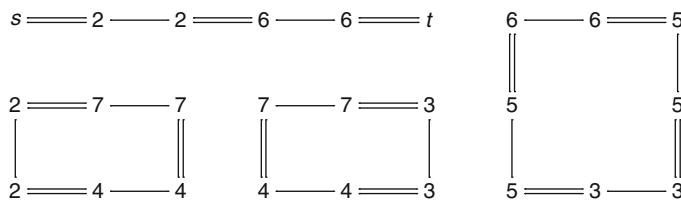


Fig. 12

The reduction graph \mathcal{R}_u of u from the running example.



In the running example, the reduction graph \mathcal{R}_u of u is again depicted in Fig. 12 – we have only rearranged the vertices.

Each reduction graph has a connected component, called the *linear component*, containing both vertices s and t . The other connected components are called *cyclic components*.

5.3 Mathematical Formalizations of the Intramolecular Model

In this section we formalize the gene assembly process involving the three molecular operations **Id**, **hi**, and **dld** in the framework of legal strings and overlap graphs.

5.3.1 Assembly Operations on Strings

Recall that each micronuclear MDS structure can be faithfully represented as a signed permutation α , which in turn has a presentation as a legal string $v = \varrho_\kappa(\alpha)$, where κ is the number of the MDSs in the micronuclear gene, and as an overlap graph G_v with κ vertices.

The assembly operations for strings are described first. The rules are (snr) the *string negative rule*, (spr) the *string positive rule*, and (sdr) the *string double rule*. For simplicity we consider only strings of pointers. Recall that we denote $\Delta_\kappa = \{2, 3, \dots, \kappa\}$ and $\Pi_\kappa = \Delta_\kappa \cup \overline{\Delta}_\kappa$. Below, we assume that $p, q \in \Pi_\kappa$.

- snr_p applies to a legal string of the form $u = u_1 p p u_2$ resulting in

$$\text{snr}_p(u_1 p p u_2) = u_1 u_2 \quad (1)$$

- spr_p applies to a legal string of the form $u = u_1 p u_2 \bar{p} u_3$ resulting in

$$\text{spr}_p(u_1 p u_2 \bar{p} u_3) = u_1 \bar{u}_2 u_3 \quad (2)$$

- $\text{sdr}_{p,q}$ applies to a legal string of the form $u = u_1 p u_2 q u_3 p u_4 q u_5$ resulting in

$$\text{sdr}_{p,q}(u_1 p u_2 q u_3 p u_4 q u_5) = u_1 u_4 u_3 u_2 u_5 \quad (3)$$

We define $\text{dom}(\rho)$ for a string reduction rule ρ by $\text{dom}(\text{snr}_p) = \text{dom}(\text{spr}_p) = \{\|p\|\}$ and $\text{dom}(\text{sdr}_{p,q}) = \{\|p\|, \|q\|\}$ for $p, q \in \Pi_\kappa$.

We adopt the following graphical notations for the applications of these operations:

$$u \xrightarrow{\text{snr}_p} \text{snr}_p(u), \quad u \xrightarrow{\text{spr}_p} \text{spr}_p(u), \quad u \xrightarrow{\text{sdr}_{p,q}} \text{sdr}_{p,q}(u)$$

A composition $\varphi = \varphi_n \dots \varphi_1$ of the above operations φ_i is a *string reduction* of u , if φ is applicable to u . Also, φ is *successful* for u , if $\varphi(u) = \Lambda$, the empty string. Moreover, we define $\text{dom}(\varphi) = \text{dom}(\varphi_1) \cup \text{dom}(\varphi_2) \cup \dots \cup \text{dom}(\varphi_n)$.

Example 11 The rule snr_2 is applicable to the legal string $u = 5\ 2\ 2\ 3\ \bar{5}\ \bar{4}\ 3\ 4$: $\text{snr}_2(u) = 5\ 3\ \bar{5}\ \bar{4}\ 3\ 4$. Moreover, we have

$$5\ 2\ 2\ 3\ \bar{5}\ \bar{4}\ 3\ 4 \xrightarrow{\text{snr}_2} 5\ 3\ \bar{5}\ \bar{4}\ 3\ 4 \xrightarrow{\text{spr}_4} 5\ 3\ \bar{5}\ \bar{3} \xrightarrow{\text{spr}_3} 5\ 5 \xrightarrow{\text{snr}_5} \Lambda,$$

and hence φ is successful for u .

The following is the basic universality result for legal strings.

Theorem 1 (Ehrenfeucht et al. 2000; Brijder et al. 2006) *Each legal string has a successful string reduction.*

5.3.2 Assembly operations on graphs

The assembly operations for graphs will now be described. The rules are (gnr) the *graph negative rule*, (gpr) the *graph positive rule*, (gdr) the *graph double rule*.

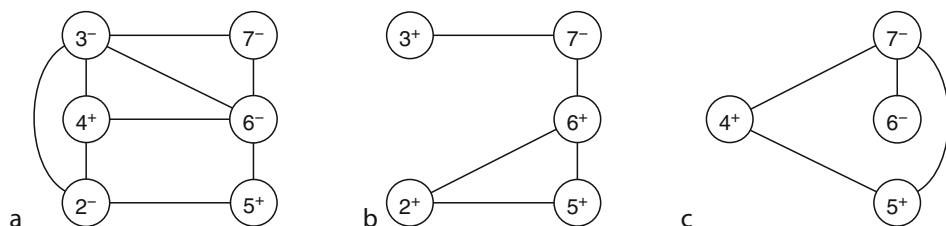
Let x and y be vertices of a signed graph G .

- gnr_x is applicable to G , if x is isolated and negative. The result is $\text{gnr}_x(G) = G - x$.
- gpr_x is applicable to G , if x is positive. The result is $\text{gpr}_x(G) = \text{loc}_x(G) - x$.
- $\text{gdr}_{x,y}$ is applicable to G , if x and y are adjacent and negative. The result is $\text{gdr}_{x,y}(G) = \text{loc}_x \text{loc}_y \text{loc}_x(G) - \{x, y\}$ obtained by complementing the edges between the sets $N_G(x) \setminus N_G(y)$, $N_G(y) \setminus N_G(x)$, and $N_G(x) \cap N_G(y)$.

Example 12 Consider the overlap graph $G = G_w$ for $w = 3\ \bar{5}\ 2\ 6\ 5\ 4\ 7\ 3\ 6\ 7\ 2\ \bar{4}$; see Fig. 13a. The graph $\text{gpr}_4(G)$ is given in Fig. 13b, and the graph $\text{gdr}_{2,3}(G)$ is given in Fig. 13c.

Fig. 13

The graphs (a) $G = G_w$, (b) $\text{gpr}_4(G)$, (c) $\text{gdr}_{2,3}(G)$, where $w = 3\ \bar{5}\ 2\ 6\ 5\ 4\ 7\ 3\ 6\ 7\ 2\ \bar{4}$.



A composition $\varphi = \varphi_n \dots \varphi_1$ of the above graph operations is called a *graph reduction* for a signed graph G , if φ is applicable to G . Also, φ is *successful*, if $\varphi(G)$ is the empty graph.

Example 13 The overlap graph $G = G_w$ given in [Fig. 13a](#) is reduced to the empty graph by the composition $\text{gpr}_5\text{gpr}_6\text{gpr}_7\text{gpr}_4\text{gdr}_{2,3}$.

The above operations are universal for signed graphs.

Theorem 2 (Harju et al. 2004b) *Each signed graph G has a successful graph reduction.*

5.3.3 Equivalence of the Systems

The relation between the systems for strings and graphs is now studied, and we show that there is a correspondence between these operations.

Theorem 3 (Ehrenfeucht et al. 2003b) *Let w be a legal string. Each string reduction $\varphi = \varphi_n \dots \varphi_1$ for w translates into a graph reduction $\varphi' = \varphi'_n \dots \varphi'_1$ for the overlap graph G_w by the translation:*

$$\text{snr}_p \mapsto \text{gnr}_p, \quad \text{spr}_p \mapsto \text{gpr}_p, \quad \text{sdr}_{p,q} \mapsto \text{gdr}_{p,q}.$$

Consequently, if φ is successful for w , then φ' is successful for G_w .

The reverse implication, from graphs to strings, is not as straightforward. Recall first that the mapping from the legal strings to the overlap graphs is not injective.

Denote by $p(w)$ the *first occurrence* of p or \bar{p} in w .

Theorem 4 (Ehrenfeucht et al. 2003b) *Let w be a legal string, and let φ be a successful reduction for G_w . Then there exists a permutation $\varphi'' = \varphi_n \dots \varphi_1$ of φ , which is successful for G_w , and which can be translated to a successful string reduction $\varphi' = \varphi'_n \dots \varphi'_1$ for w by the following translations:*

$$\text{gnr}_p \mapsto \text{snr}_{p(w)}, \quad \text{gpr}_p \mapsto \text{spr}_{p(w)}, \quad \text{gdr}_{p,q} \mapsto \text{sdr}_{p(w),q(w)}.$$

5.4 Properties of Intramolecular Assemblies

Some properties of intramolecular gene assemblies are discussed in this section. Many of these properties hold in all the mathematical models described in [Sect. 5.3](#). In each case however, we choose to describe the results only on a level that allows for the simplest or the most elegant formulation. For more results we refer to Ehrenfeucht (2003a) and Brijder (2008).

5.4.1 Nondeterminism and Confluence

It is easy to show on all model levels, from the molecular level to that of graphs, that for a given gene (permutation, string, graph, resp.), there can be more than one strategy to assemble it.

Different assembly reduction strategies have recently also been confirmed experimentally (Möllenbeck et al. 2008). We say that the intramolecular model is *nondeterministic*. Consider, for example, the signed string associated to the actin I gene in *S. nova*: $u = 3\ 4\ 4\ 5\ 6\ 7\ 5\ 6\ 7\ 8\ 9\ \bar{3}\ \bar{2}\ 2\ 8\ 9$. There are 3060 different sequential strategies to reduce this string (assemble the gene), see Ehrenfeucht et al. (2003a), of which only two are shown in [Table 1](#). Note in particular that these two strategies differ in the number and type of operations used. On the other hand, both strategies are successful, reducing the input string to the empty string. As shown in Theorems 1 and 2, this is true in general: although several operations may be applicable to a given input, successful strategies for that input exist starting with any of those operations. The biological interpretation is that all (potentially many) assembly strategies of a given micronuclear gene, have the same result: the assembled corresponding macronuclear gene. We call such a model *confluent*. Consequently, ciliates “need not remember” a particular sequential strategy which in turn contributes to the robustness of the gene assembly process.

In [Sect. 6](#) it will be proved that the assembly strategies of a given gene share a number of other properties beyond yielding the same assembled gene: the number of molecules (linear and circular) produced throughout the assembly, their nucleotide sequence, whether the assembled gene is linear or circular.

5.4.2 The Structure of MAC Genes with By-products

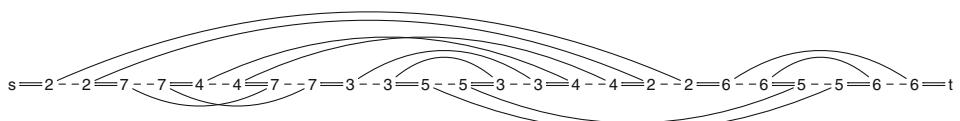
Since legal strings represent the initial configuration (gene in MIC form) and the corresponding reduction graph the end result (the same gene in MAC form and its excised products), it is natural to study the possible forms of reduction graphs. Formally, we now characterize the graphs that are (isomorphic to) reduction graphs.

A graph G isomorphic to a reduction graph must be a 2-edge colored graph (V, E_1, E_2, f, s, t) such that for each p in the range of f , $p \in \mathcal{A}$ and there must be exactly four vertices labeled by p . Each vertex must be connected to exactly one (reality) edge from E_1 , and each vertex, except s and t , must be connected to exactly one (desire) edge from E_2 . Finally, edges from E_2 must connect vertices with a common label. One can call these graphs *abstract reduction graphs*, and let the set of abstract reduction graphs be ARG. It turns out that there are graphs in ARG that are not (isomorphic to) reduction graphs.

To obtain a characterization, one more property of reduction graphs is needed: the ability to linearly order the vertices to resemble its (in general not unique) underlying legal string, as done in [Fig. 11](#). To make this linear order of vertices explicit, we introduce a third set of edges, called *merge edges*, to the reduction graph as done in [Fig. 14](#).

[**Fig. 14**](#)

Merge edges are added to the reduction graph of [Fig. 11](#).



Now, when is a set of edges M for $G \in \text{ARG}$ a set of merge edges? Like desire edges, they have the properties that (1) the edges connect vertices with a common label and (2) each vertex except s and t is connected to exactly one merge edge. Moreover, M and the set E_2 are disjoint – no desire edge is parallel to a merge edge. Finally, the reality edges and merge edges must allow for a path from s to t passing each vertex once. This last requirement is equivalent to the fact that the reality and merge edges induce a connected graph.

If it is possible to add a set of merge edges to the graph, then it is not difficult to see that the graph is isomorphic to a reduction graph \mathcal{R}_u . Indeed, we can identify such a u for this reduction graph by simply considering the alternating path from s to t over the reality and merge edges. The orientation (positiveness or negativeness) of each pointer is determined by the crossing or noncrossing of the desire edges (exactly as we defined the notion of reduction graph).

To characterize reduction graphs we need the notion of a pointer-component graph. Given an abstract reduction graph, a pointer-component graph describes how the labels of that abstract reduction graph are distributed among its connected components.

Definition 1 Let $G \in \text{ARG}$. The pointer-component graph of G , denoted by \mathcal{PC}_G , is a multigraph (ζ, E, ε) , where ζ is the set of connected components of G , $E = \text{dom}(G)$ and ε is, for $e \in E$, defined by $\varepsilon(e) = \{C \in \zeta \mid C \text{ contains vertices labeled by } e\}$.

The pointer-component graph of $G = \mathcal{R}_u$ of Fig. 12 is given in Fig. 15. We have $\zeta = \{C_1, C_2, C_3, R\}$ where R is the linear component and the other elements are cyclic components of \mathcal{R}_u .

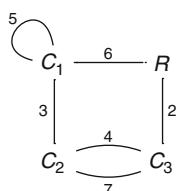
It is shown in Brijder and Hoogeboom (2008a) that, surprisingly, $G \in \text{ARG}$ has a set of merge edges precisely when the pointer-component graph \mathcal{PC}_G is a connected graph. Therefore, we have the following result.

Theorem 5 (Brijder and Hoogeboom 2008a) An abstract reduction graph G is isomorphic to a reduction graph iff \mathbf{PC}_G is a connected graph.

Consider the legal string $u = 2\bar{7}47353\bar{4}2656$ as before and $v = 2\bar{7}426\bar{5}3\bar{4}7356$. It turns out that they have the same reduction graph (up to isomorphism): $\mathcal{R}_u \approx \mathcal{R}_v$ (see Fig. 11). The reason for this is that a reduction graph may have more than one set of merge edges – each one corresponding to a different legal string. Thus, there can be many legal strings giving the same reduction graph. In Brijder and Hoogeboom (2008a) it is shown how for a given legal string u we can obtain precisely the set of all legal strings having the same reduction graph

Fig. 15

The graph $\mathcal{PC}_{\mathcal{R}_u}$ of the graph \mathcal{R}_u in [Fig. 12](#).



(up to isomorphism). In fact, it turns out that this set is exactly the set of all legal strings obtained by applying compositions of the following string rewriting rules.

For all $p, q \in \Pi_\kappa$ with $\|p\| \neq \|q\|$ we define

- The *dual string positive rule* for p is defined by $\text{dspr}_p(u_1 p u_2 p u_3) = u_1 p \bar{u}_2 p u_3$
- The *dual string double rule* for p, q is defined by $\text{dsdr}_{p,q}(u_1 p u_2 q u_3 \bar{p} u_4 \bar{q} u_5) = u_1 p u_4 q u_3 \bar{p} u_2 \bar{q} u_5$

where u_1, u_2, \dots, u_5 are arbitrary (possibly empty) strings over Π_κ . Notice the strong similarities of these rules with the string positive rule and the string double rule. As an example, if we take $u = 2\bar{7}4735342656$ and $v = 2\bar{7}426\bar{5}3\bar{4}7356$ given earlier, then $\text{dsdr}_{4,5} \circ \text{dspr}_3(u) = v$ and hence both legal strings indeed have a common reduction graph.

5.4.3 Intermediate Legal Strings

We now show that we can generalize the notion of reduction graph to allow for representations of any intermediate product during the reduction process. In such an intermediate product, some pointers, represented as a subset D of $\text{dom}(u)$, where u is a legal string, have not yet been used in recombination operations, while the other pointers, in $\text{dom}(u) \setminus D$, have already been used in recombination operations. A reduction graph of u with respect to this set D , denoted by $\mathcal{R}_{u,D}$, represents such intermediate product. As before, we simply ignore the pointers in D – they are put as strings on the reality edges that are now directed edges.

Figure 16 gives an example of $\mathcal{R}_{u,D}$ with $u = 2\bar{7}47353\bar{4}2656$ and $D = \{2, 4\}$ (recall that Λ represents the empty string).

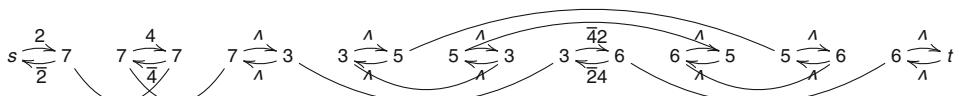
We denote the legal string obtained from a legal string u by removing the pointers from $D \subseteq \text{dom}(u)$ and its barred variants by $\text{rem}_D(u)$. In our example, $\text{rem}_D(u) = \bar{7}7353656$. We define $\text{red}(u, D)$ as the label of the alternating path from s to t . Thus, in our example $\text{red}(u, D) = 2\bar{4}\bar{4}2$. Assuming that gene assembly is intramolecular (all recombination takes place on a single DNA molecule), then the cyclic connected components must have only empty strings as edge labels. In our example, it is easy to obtain an “invalid” intermediate product: take, for example, $D = \{3, 4, 5, 6, 7\}$. Hence, it is not possible to first recombine pointer 2, followed by recombination of the remaining pointers.

Theorem 6 (Brijder et al. 2006) *Let u be a legal string, let φ be a composition of reduction rules with $\text{dom}(\varphi) \subseteq \text{dom}(u)$, and let $D = \text{dom}(u) \setminus \text{dom}(\varphi)$. Then φ is applicable to u iff φ is applicable to $\text{rem}_D(u)$ and $\text{red}(u, D)$ is a legal string with domain D . Moreover, if this is the case, then $\varphi(u) = \text{red}(u, D)$.*

As a consequence of Theorem 6, reductions φ_1 and φ_2 with the same domain have the same effect: $\varphi_1(u) = \varphi_2(u)$ for all legal strings u . Note that in general there are $D \subseteq \text{dom}(u)$ for

Fig. 16

Graph $\mathcal{R}_{u,D}$ with $u = 2\bar{7}47353\bar{4}2656$ and $D = \{2, 4\}$.



which there is no reduction φ of u with $D = \text{dom}(\varphi(u))$. In our example, $\text{red}(u, D)$ is a legal string with domain D and we have, for example, $(\text{snr}_6 \text{sdr}_{3,5} \text{spr}_7)(u) = \overline{2}\overline{4}\overline{4}2 = \text{red}(u, D)$.

5.4.4 Cyclic Components

Since the reduction graph is a representation of the end result after recombination, the cyclic components of a reduction graph represent circular molecules. If we now consider again the intramolecular model of gene assembly, we notice that each such molecule is obtained by loop recombination. Hence, although there can be many different sequences of operations that obtain the fixed end product, the *number* of loop recombination operations (string negative rules in the model) in each such sequence is the same.

Theorem 7 (Brijder et al. 2006) *Let N be the number of cyclic components in the reduction graph of legal string u . Then every successful reduction of u has exactly N string negative rules.*

Example 14 Since \mathcal{R}_u in Fig. 12 has three cyclic components, by Theorem 7, every successful reduction φ of u has exactly three string negative rules. For example, $\varphi = \text{snr}_2 \text{snr}_4 \text{spr}_7 \text{snr}_6 \text{sdr}_{3,5}$ is a successful reduction of u . Indeed, φ has exactly three string negative rules. Alternatively, $\text{snr}_6 \text{snr}_3 \text{snr}_7 \text{spr}_2 \text{spr}_5 \text{spr}_4$ is also a successful reduction of u , with a different number of (spr and sdr) operations.

It turns out that the reduction graph also allows for determining *on which pointers* the string negative rules can be applied using the pointer-component graph (Brijder et al. 2008). For convenience, one can denote $\mathcal{PC}_{\mathcal{R}_u}$ by \mathcal{PC}_u . Also, one can denote $\mathcal{PC}_u|_D$ as the graph obtained from \mathcal{PC}_u by removing the edges outside D . Finally, for a reduction φ , let $\text{snrdom}(\varphi) \subseteq \text{dom}(\varphi)$ be the (unbarred) pointers used in snr rules in φ .

Theorem 8 (Brijder et al. 2008) *Let u be a legal string, and let $D \subseteq \text{dom}(u)$. There is a successful reduction φ of u with $\text{snrdom}(\varphi) = D$ iff $\mathcal{PC}_u|_D$ is a tree.*

In our running example, we see that $D = \{2, 3, 6\}$ induces a (spanning) tree of \mathcal{PC}_u . Therefore, there is a successful reduction φ of u with $\text{snrdom}(\varphi) = D$. Indeed, we have $\text{sdr}_{4,5} \text{spr}_7(u) = 226336$. It is clear that we can extend $\text{sdr}_{4,5} \text{spr}_7$ to a successful reduction that applies string negative rules on 2, 3, and 6. Notice that here snr_3 must be applied *before* snr_6 . In fact in Brijder et al. (2009) it is shown that the possible orders in which the string negative rules can be applied is also deducible from \mathcal{PC}_u by considering rooted trees.

The results above can be carried over to intermediate products; for example, the number of string negative rules from u to $\varphi(u)$ is fixed and is equal to the number of cyclic components of $\mathcal{R}_{u,D}$.

5.5 Simple and Parallel Gene Assemblies

The general formulation of the intramolecular operations allows for the aligned pointers to be arbitrarily far from each other. We discuss in this section a *simple* variant of the model, where

all alignments and folds involved in the operations are *local*. In the simple versions of *ld*, *hi*, and *dlad*, the pointers involved in the recombination are at a minimal distance from each other. It turns out that the simple model is able to explain the successful assemblies of all currently known micronuclear gene sequences, see Cavalcanti et al. (2005), Prescott et al. (2001a), and Langille et al. (2010). In this section we discuss the molecular and the mathematical formulation of the simple model and indicate some interesting properties of the model.

In the second part of this section, a notion of *parallel gene assembly* is discussed. In each (parallel) step of the assembly, we apply a number of well-selected operations simultaneously in such a way that the total number of steps is minimal. In each step, the operations are selected in such a way that their application is independent of the others applied in the same step: All sequential compositions of those operations are applicable to the current graph. Several difficult computational problems arise in this context, including deciding whether a given graph has a parallel assembly of a given length, or deciding whether there are graphs (or even trees) of arbitrarily high parallel complexity.

5.5.1 Simple Gene Assembly

The three intramolecular operations allow in their general formulation that the MDSs participating in an operation may be located anywhere along the molecule. Arguing on the principle of parsimony, a simplified model was discussed already in Prescott et al. (2001a) and then formalized in Harju et al. (2006c), asking that all operations are applied “locally”. In the simple model, the restriction is that there is at most one coding block involved in each of the three operations. This idea was then further developed into two separate models. In one of them, which is referred to as the *simple model* (Langille and Petre 2006), both micronuclear, as well as composite MDSs (obtained by splicing of several micronuclear MDSs) may be manipulated in each of the three molecular operations. In the other, called the *elementary model* and introduced in Harju et al. (2006b, 2008c), the model was further restricted so that only *micronuclear*, but not *composite*, MDSs could be manipulated by the molecular operations. Consequently, once two or more micronuclear MDSs are combined into a larger composite MDS, they can no longer be moved along the sequence. In this section, only the simple model is discussed and for details of the elementary model one can refer to Harju et al. (2006b, 2008c), Langille et al. (2010), and Petre and Rogojin (2008).

We already discussed in [Sect. 5.1](#) that *ld* must always be *simple* in a successful assembly. As such, the effect of *ld* is that it will combine two consecutive MDSs into a bigger composite MDS. For example, consider that M_3M_4 is a part of the molecule, that is, MDS M_4 succeeds M_3 being separated by one IES I . Thus, pointer 4 has two occurrences that flank I : one in the end of MDS M_3 and the other one in the beginning of MDS M_4 . Then *ld* makes a fold as in [Fig. 6a](#) aligned by pointer 4, IES I is excised as a circular molecule and M_3 and M_4 are combined into a longer coding block as shown in [Fig. 6c](#).

In the case of *hi* and *dlad*, the pointers involved can be separated by arbitrarily large sequences; for example, in the actin I gene in *S. nova*, pointer 3 has two occurrences: one in the beginning of M_3 and one, inverted, in the end of M_2 . Thus, *hi* is applicable to this sequence with the hairpin aligned on pointer 3, even though five MDSs separate the two occurrences of pointer 3. Similarly, *dlad* is applicable to the MDS sequence $M_2M_8M_6M_5M_1M_7M_3M_{10}M_9M_4$,

with the double loops aligned on pointers 3 and 5. Here the first two occurrences of pointers 3 and 5 are separated by two MDSs (M_8 and M_6) and their second occurrences are separated by four MDSs (M_3 , M_{10} , M_9 , M_4).

An application of the *hi* operation on pointer p is *simple* if the part of the molecule that separates the two copies of p in an inverted repeat contains only one MDS and one IES. We have here two cases, depending on whether the first occurrence of p is incoming or outgoing, see [Fig. 17a](#).

An application of *dlad* on pointers p , q is *simple* if the sequence between the first occurrences of p and q , as well as the sequence between the second occurrences of p and q consist of either one MDS or one IES. We have again two cases, depending on whether the first occurrence of p is incoming or outgoing, see [Fig. 17b](#).

The simple operations can be formalized as operations on signed permutations, signed strings, and signed graphs. We only give here the definitions for the string-based operations, where the mathematical formulation is more concise. For the other formulations, including the relationships among these models, one can refer to Harju et al. (2006c), Langille et al. (2010), and Brijder and Hoogeboom (2008b).

The *simple hi* operation for pointer p , denoted sspr_p is applicable to strings of the form $u = u_1 p u_2 \bar{p} u_3$, where $|u_2| \leq 1$, resulting in $\text{sspr}_p(u_1 p u_2 \bar{p} u_3) = u_1 \bar{u}_2 u_3$.

The *simple dlad* operation for pointers p , q , denoted $\text{ssdr}_{p,q}$ is applicable to strings of the form $u = u_1 p q u_2 p q u_3$, resulting in $\text{ssdr}_{p,q}(u_1 p q u_2 p q u_3) = u_1 u_2 u_3$.

Let ϕ be a composition of *snr*, *sspr*, and *ssdr* operations such that ϕ is applicable to string u . We say that ϕ is a *simple reduction* for u if either $\phi(u) = \Lambda$ (in which case we say that φ is *successful*), or $\phi(u) \neq \Lambda$ and no simple operation is applicable to $\phi(u)$ (in which case we say that ϕ is *unsuccessful*). For example, the string reduction in [Table 1b](#) is simple, unlike the one in [Table 1a](#).

The simple model has a number of properties that do not hold for the general model. One of them concerns the length of reduction strategies for a given string. While in the general model a string may have reduction strategies of different lengths, see the example in [Table 1](#), the same is not true in the simple model, see the next result of Langille and Petre (2007). Moreover, if we consider *parallel applications of simple operations* (a notion of Langille and Petre (2007) that is not defined in this chapter), we get a new twist: for any given n there exists a string having maximal parallel reductions of any length between n and $2n$.

[Fig. 17](#)

The MDS/IES structures where (a) *simple hi*-rules and (b) *simple dlad*-rules are applicable. The MDSs are indicated by rectangles and their flanking pointers are shown. Between the two MDSs there is only one IES represented by a straight line (From Harju et al. (2006c) and Prescott et al. (2001a)).

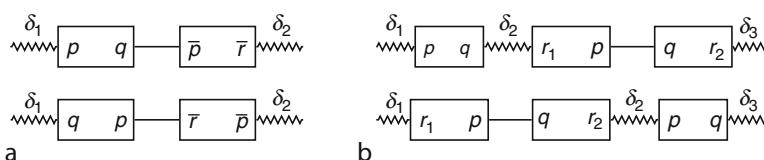


Table 1

Two reduction strategies for the signed string corresponding to the actin I micronuclear gene in *S.nova*

(a)	(b)
$u_1 = \text{spr}_3(u) = \bar{9}\bar{8}\bar{7}\bar{6}\bar{5}\bar{4}\bar{4}\bar{2}289$	$u'_1 = \text{snr}_4(u) = 356756789\bar{3}\bar{2}89$
$u_2 = \text{snr}_4(u_1) = \bar{9}\bar{8}\bar{7}\bar{6}5\bar{2}289$	$u'_2 = \text{sdr}_{5,6}(u'_1) = 37789\bar{3}\bar{2}89$
$u_3 = \text{spr}_8(u_2) = \bar{9}\bar{2}25675679$	$u'_3 = \text{snr}_7(u'_2) = 389\bar{3}\bar{2}89$
$u_4 = \text{spr}_2(u_3) = \bar{9}5675679$	$u'_4 = \text{sdr}_{8,9}(u'_3) = 3\bar{3}\bar{2}2$
$u_5 = \text{sdr}_{5,7}(u_4) = \bar{9}669$	$u'_5 = \text{spr}_2(u'_4) = 3\bar{3}$
$u_6 = \text{snr}_6(u_5) = \bar{9}9$	$u'_6 = \text{spr}_3(u'_5) = \Lambda$
$u_7 = \text{spr}_9(u_6) = \Lambda$	

Theorem 9 (Langille and Petre 2007) *Let u be a signed double occurrence string and ϕ, ψ two reduction strategies for u . Then ϕ and ψ have the same number of operations.*

Regarding the outcome of reduction strategies, the simple model is different from the general model in several respects; for example, there are strings that cannot be reduced in the simple model, unlike in the general model where all strings have reduction strategies. Indeed, no simple operation is applicable to the string $\bar{2}43423$. The following is a result of Langille and Petre (2006).

Theorem 10 (Langille and Petre 2006) *No signed string has both successful and unsuccessful reductions in the simple model.*

On the other hand, the outcome of various strategies for a given string can differ; for example, for $u = 23467856782345$, $u_1 = \text{ssdr}_{2,3} \circ \text{ssdr}_{7,8}(u) = 465645$, whereas $u_2 = \text{ssdr}_{3,4} \circ \text{ssdr}_{6,7}(u) = 285825$. The strings u_1 and u_2 are, however, identical modulo a relabeling of their letters. This observation can be extended to define a notion of *structure* that can be used to prove that the results of various reductions, although different, always have the same structure. For details, one can refer to Langille and Petre (2006), where the discussion is in terms of signed permutations, rather than signed strings.

5.5.2 Parallel Gene Assembly

The notion of parallelism is usually defined in concurrency theory for processes whose application is independent of each other. In other words, a number of processes can be applied in parallel to a signed graph if they can be (sequentially) applied in any order. Adopting this approach, the following gives the definition of parallel application of the three molecular operations on a signed graph. For a similar discussion, albeit technically more tedious, on the level of signed strings, one can refer to Harju et al. (2006a).

Definition 2 (Harju et al. 2006a) *Let S be a set of *gnr*, *gpr*, and *gdr* operations and let G be a signed graph. We say that the rules in S are applicable in parallel to G if for any ordering $\varphi_1, \varphi_2, \dots, \varphi_k$ of S , the composition $\varphi_k \circ \dots \circ \varphi_1$ is applicable to G .*

Based on the definition of parallelism, which presumes that the rules are applicable in any possible order, the following theorem shows that the result is always the same regardless of the order in which they are applied.

Theorem 11 (Harju et al. 2006a) *Let G be a signed graph and let S be a set of operations applicable in parallel to G . Then for any two compositions φ and ψ of the operations of S , $\varphi(G) = \psi(G)$.*

Based on Theorem 11, we can write $S(G) = \varphi(G)$ for any set S of operations applicable in parallel to G and any composition φ of these operations. The notion of parallel complexity is defined as follows.

Definition 3 (Harju et al. 2007) *Let G be a signed graph, and let S_1, \dots, S_k be sets of gnr, gpr, gdr operations. If $(S_k \circ \dots \circ S_1)(G) = \emptyset$, then we say that $S = S_k \circ \dots \circ S_1$ is a parallel reduction for G . In this case the parallel complexity of S is $\mathcal{C}(S) = k$. The parallel complexity of the signed graph G is:*

$$\mathcal{C}(G) = \min\{\mathcal{C}(S) \mid S \text{ is a parallel reduction strategy for } G\}.$$

Deciding whether a given set of graph operations is applicable in parallel to a given graph turns out to be a difficult problem if gdr operations are involved. When at most two gdr operations are involved, then simple characterizations were given in Harju et al. (2008a). The computational complexity of the general problem was upper bounded in the co-NP class, see Alhazov et al. (2009).

It can be easily verified that the parallel complexity of the graphs corresponding to the currently known micronuclear gene sequences is at most two, see Harju et al. (2008b). However, examples of graphs of higher complexity can be given. For example, the graph with the highest known complexity has 24 vertices and can be reduced in six parallel steps. The tree with the highest known parallel complexity has 12 vertices and can be reduced in five parallel steps. One can refer to Harju et al. (2008b) for more examples. Although the parallel complexity of certain types of graphs (e.g., for uniformly signed trees) is known to be finitely bounded, see Harju et al. (2008a), the general problem is currently open and seems to be very difficult. In particular, it seems to require a characterization for the parallel applicability of arbitrary sets of operations, another open problem. The problem is open even in seemingly simpler cases: for trees, or for negative graphs. The computational complexity of deciding whether the parallel complexity of a given graph is upper bounded by a given contact was placed in the NP^{NP} class in Alhazov et al. (2009). Two algorithms for computing the parallel complexity of signed graphs were given in Alhazov et al. (2009, 2010), both with exponential computational complexity. A visual graph editor including support for computing the parallel complexity of signed graphs can be found in Petre and Skogman (2006).

5.6 Gene Assembly by Folding and Unfolding

In this section, gene assembly is considered from a somewhat more general viewpoint. The section is based on Ehrenfeucht et al. (2002). The molecular operations provided by the gene assembly process each involve one molecule. This observation underlies the model of gene assembly as fold-and-recombine computing paradigm. For convenience, we consider

circular graphs to be representations of DNA molecules. Our initial situation is a set of circular DNA molecules represented by bicolored and labeled circular graphs. The fold-and-recombine process is reflected by a two-stage processing of the graphs: (1) fold on vertices representing pointers; (2) unfold using a pairing function. In this setup, gene assembly becomes a dynamic process for recombination graphs.

Graphs with multiple edges and loops are allowed. The vertices denote all pointers of the gene. We consider bicolored graphs, where color 1 is used to indicate an IES and color 2 is used to indicate an MDS. To represent the sequence of nucleotides comprising various (IES or MDS) segments we use a labeling of the edges.

Each edge will be oriented in both directions: $a = (x, y)$ and $\bar{a} = (y, x)$ are *reverse pairs*. Let V be a set of vertices, and consider E as a set of edge symbols such that $E = \{e_1, \dots, e_n, \bar{e}_1, \dots, \bar{e}_n\}$ with $\bar{e}_i = e_i$. A (*general*) *bicolored graph* G consists of an *end point map* $\varepsilon_G = \varepsilon: E \rightarrow V \times V$ such that $\varepsilon(\bar{e}) = \varepsilon(e)$ for all $e \in E$; a *labeling* $f_G = f: E \rightarrow \Sigma^*$ for an alphabet Σ , with $f(\bar{e}) = \bar{f}(e)$ for all $e \in E$; a *coloring* $h_G = h: E \rightarrow \{1, 2\}$ such that $h(e) = h(\bar{e})$ for all $e \in E$.

For simplicity, we write $e = (x, y)$ for $\varepsilon(e) = (x, y)$. An edge $e = (x, y) \in E$ is a *loop*, if $x = y$. The *valency* $\text{val}_G(x)$ of a vertex x is the number of edges leaving x . A bicolored graph is *even* if its valencies are all even. A bicolored graph G is a *recombination graph*, if $\text{val}_G(x) \in \{2, 4\}$ for all x , and every vertex of valency 4 is balanced: two incident edges have color 1 and the other two have color 2; see Fig. 18a.

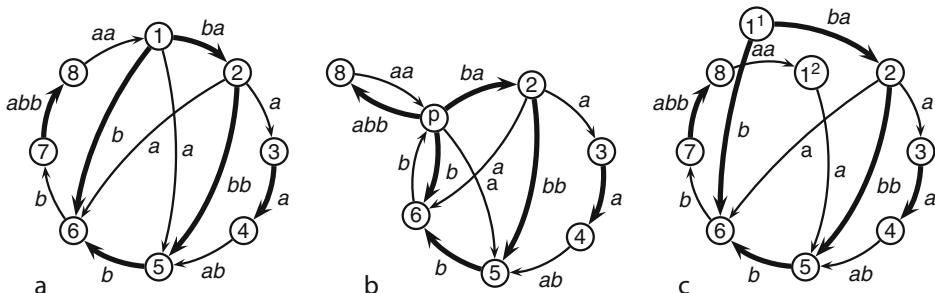
For each vertex x in an even bicolored graph G let ψ_x be a bijection that maps incoming edges to outgoing edges respecting inversions, such that $\psi_x(\overline{\psi_x(e)}) = \bar{e}$ and $\psi_x(e) = \bar{e}$ if and only if e is a loop. Then the map $\psi: x \mapsto \psi_x$ is a *pairing*. Each recombination graph G has the *natural pairing* ψ where e and $\psi_x(e)$ have the same color whenever $\text{val}_G(x) = 4$.

5.6.1 Folding and Unfolding

A pair $p = \{x, y\} \in E(V)$ will be called a *pointer* with *ends* x and y . A set P of mutually disjoint pointers is a *pointer set*. The p -*folded graph* $G * p$ is obtained by identifying the ends of p ; see Fig. 18a, b. For a pointer set $\{p, q\}$, $G * p * q = G * q * p$. This allows one to define, for a pointer set $P = \{p_1, \dots, p_m\}$, the P -*folded graph* $G * P$ as $G * p_1 * \dots * p_m$.

Fig. 18

(a) A recombination graph, where color 1 is represented by thick edge. (b) The p -folded graph $G * p$ for $p = \{1, 7\}$. (c) The ψ -unfolded graph $G \diamond_{\psi} 1$ with the natural pairing (from Ehrenfeucht et al. (2002)).



Let G be an even bicolored graph with a pairing ψ . For a vertex x , let $e_{11}, e_{12}, \dots, e_{m1}, e_{m2}$ be the incoming edges with $\psi_x(e_{il}) = \bar{e}_{l2}$. In the ψ -unfolded graph, the vertex x is replaced by the new vertices x^1, \dots, x^m and the edges are redirected according to the pairing ψ_x ; see [Fig. 18c](#), where the redirection is determined by the colors.

Notice that if G is a recombination graph, so is $G \diamond_\psi x$. Also, if $x \neq y$, then $G \diamond_\psi x \diamond_\psi y = G \diamond_\psi y \diamond_\psi x$. Therefore, we can write $G \diamond_\psi A = G \diamond_\psi x_1 \diamond_\psi \dots \diamond_\psi x_m$ for a subset $A = \{x_1, \dots, x_m\}$.

For an even bicolored graph G with a pairing ψ , let $F(G) = \{x \in V_G \mid \text{val}_G(x) \geq 4\}$. Then the graph $G \diamond_\psi F(G)$ is called the ψ -unfolded graph of G .

Lemma 1 (Ehrenfeucht et al. 2002) *If G is an even bicolored graph with a pairing ψ , then its ψ -unfolded graph is a disjoint union of cycles.*

Let G be a bicolored graph with a pointer set P , and let ψ be a pairing of the P -folded graph $G * P$. We denote $G \circledast_\psi P = (G * P) \diamond_\psi P$. We shall write $G \circledast P$ for $G \circledast_\psi P$, if $G * P$ is a recombination graph and ψ is its natural pairing.

Lemma 2 (Ehrenfeucht et al. 2002) *Let G be a disjoint union of bicolored cyclic graphs. Let P be a pointer set of G , and let ψ be a pairing of $G * P$. Then $G \circledast_\psi P$ is a disjoint union of bicolored cyclic graphs.*

5.6.2 Assembled Graphs of Genomes

Let G be a bicolored cyclic graph with $V_G = \{x_1, \dots, x_n\}$ and the edge set $E_G = \{e_1, \dots, e_n, \bar{e}_1, \dots, \bar{e}_n\}$, where $e_i = (x_i, x_{i+1})$ and $x_{n+1} = x_1$. A vertex x_i is a *boundary vertex* of G , if $h_G(e_{i-1}) \neq h_G(e_i)$, where $i - 1$ is modulo n . A path π is a *segment*, if its edges have color 1 and the ends of π are boundary vertices. For a disjoint union $G = \sum_{i=1}^m G_i$ of bicolored cyclic graphs G_i , we let its boundary vertex set be the union of corresponding sets of the components.

A pair $\mathcal{G} = (G, P)$ is called a *genome*, if G is a disjoint union of bicolored cyclic graphs and P is a pointer set of G containing boundary vertices only; see [Fig. 19](#).

The *assembled genome* of a genome $\mathcal{G} = (G, P)$ is $A(\mathcal{G}) = (G \circledast P, \emptyset)$, and it is a genome. Each segment of the unfolded graph $\gamma \circledast P$ is a *gene*.

Fig. 19

The genome (G, P) with $P = \{p, q\}$ for $p = \{2, 9\}$ and $q = \{5, 8\}$. The labels correspond to the MDSs M_1, M_2, N_1 and P_1, P_2 . The labels corresponding to IESs are omitted (from Ehrenfeucht et al. (2002)).

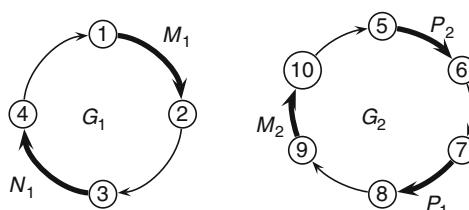
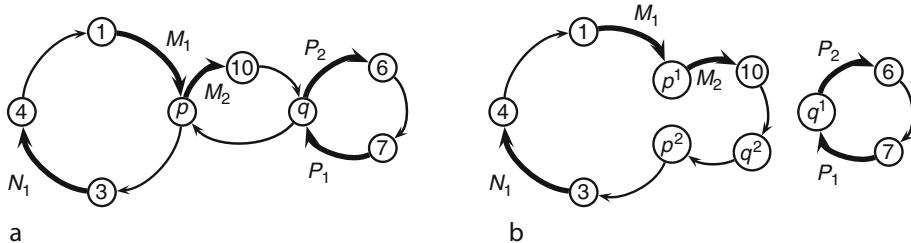


Fig. 20

Let \mathcal{G} be the genome of Fig. 19. Then $\mathcal{G} * P$ is given in (a). Unfolding gives $\mathcal{G} \circledast P$ in (b). The genes of \mathcal{G} are $g_1 : 1 \rightarrow p^1 \rightarrow 10$ (with the value $M_1 M_2$), $g_2 : 7 \rightarrow q^1 \rightarrow 6$ (with the value $P_1 P_2$), and $g_3 : 3 \rightarrow 4$ (with the value N_1) (from Ehrenfeucht et al. (2002)).



For $\mathcal{G} = (G, P)$, a sequence $\mathcal{S} = (P_1, P_2, \dots, P_m)$ of subsets of P is an *assembly strategy* of \mathcal{G} , if $\{P_1, \dots, P_m\}$ is a partition of P . One can show that in a genome $\mathcal{G} = (G, P)$, if $P_1, P_2 \subseteq P$ are disjoint, then $(G \circledast P_1) \circledast P_2 = G \circledast (P_1 \cup P_2) = (G \circledast P_2) \circledast P_1$. This gives the following general invariance property.

Theorem 12 (Ehrenfeucht et al. 2002) *Every assembly strategy $\mathcal{S} = (P_1, P_2, \dots, P_m)$ of a genome produces the same assembled genome $\mathcal{G} = (G, P)$: $G \circledast P = G \circledast P_1 \circledast P_2 \circledast \dots \circledast P_m$.*

A pointer set $R \subseteq P$ of $\mathcal{G} = (G, P)$ is *intracyclic*, if any two parts g' and g'' of each gene g that lie in the same connected component of G , lie in the same connected component of $G \circledast R$. The unfolding of a genome is illustrated in Fig. 20.

Theorem 13 (Ehrenfeucht et al. 2002) *For each genome \mathcal{G} , there exists an intracyclic assembly strategy $\mathcal{S} = (P_1, P_2, \dots, P_m)$ such that $1 \leq |P_i| \leq 2$ for all i .*

We can also show that the following stronger result holds.

Theorem 14 (Ehrenfeucht et al. 2002) *Let $\mathcal{G} = (G, P)$ be a genome for a connected $G \circledast P$. Then there is a genome $\mathcal{G}' = (G', P')$, where G' is connected, such that $A(\mathcal{G}) = A(\mathcal{G}')$, and \mathcal{G}' has an assembly strategy $\mathcal{S} = (P_1, P_2, \dots, P_m)$ of \mathcal{G}' for which $1 \leq |P_i| \leq 2$ for all i , and each $G \circledast \bigcup_{i=1}^j P_i$ is a cyclic graph for each j .*

6 Invariant Properties of Gene Assembly

As discussed already in this chapter, both an intermolecular model and an intramolecular model exist for gene assembly. Moreover, both models are nondeterministic: For a given gene, there may be several assembly strategies and also, a gene may be assembled either on a linear, or on a circular molecule. As such, a natural question is that of *invariants*: What properties of the assembled gene and of the assembly process hold for all assembly strategies of both models? For example, for a given gene, is the set of molecules excised during the assembly an invariant of the process? The same question for whether or not the assembled gene is linear or cyclic, and for whether or not the obtained structure of the IESs is fixed for given gene is also open.

An affirmative answer to these questions was given already in Ehrenfeucht et al. (2001a) for the intramolecular model, showing that these properties are invariants of the intramolecular model. We follow here a presentation of Petre (2006), where the result is given in a stronger form, showing that the properties above are invariants of *any model based on the paradigm of pointer-directed assembly*. This result may also be deduced based on the graph-theoretical framework of Ehrenfeucht et al. (2002). The presentation given in this section is in terms of strings and permutations. One can refer to Petre (2006) for more details, examples, and full proofs.

6.1 Gene Structure

In this section, we introduce a novel formal representation for the gene structures of ciliates, able to track the transformations witnessed by a gene from its micronuclear form to its assembled form. A gene is first represented as a signed permutation over the alphabet of MDSs by denoting their sequence and orientation. Our notation is then extended to denote also the IESs and all the pointers.

It can be recalled that $u, v \in \Sigma^*$ are called *equivalent*, denoted $u \approx v$, if u is a conjugate of either v or \bar{v} . Two finite sets $X_1, X_2 \subseteq \Sigma^*$ are called *equivalent*, denoted $X_1 \approx X_2$, if they have the same number of elements and for any $x_i \in X_i$ there is $x_j \in X_j$ such that $x_i \approx x_j$, with $i, j = 1, 2, i \neq j$. Intuitively, if u denotes a circular DNA molecule and $u \approx v$, then v denotes the same molecule, potentially starting from a different nucleotide and/or in the reverse direction. Similarly, if X_1 denotes a set of circular molecules and $X_1 \approx X_2$, then X_2 denotes the same set of molecules.

We denote the MDSs of the given gene by letters from the alphabet $\mathcal{M}_n = \{M_1, M_2, \dots, M_n\}$ in the order they occur in the macronuclear gene, where $n \geq 1$. Thus, the sequence of MDSs in the macronuclear gene is $M_1 M_2 \dots M_n$. On the other hand, the sequence of MDSs in the micronuclear gene is in general a signed permutation over \mathcal{M} .

Example 15 The MDSs of the micronuclear gene actin I in *S. nova* may be represented as the signed permutation $M_3 M_4 M_6 M_5 M_7 M_9 \bar{M}_2 M_1 M_8$, see Prescott and DuBois (1996). In this gene, MDS M_2 is inverted.

We call \mathcal{M}_n -descriptor any signed permutation over \mathcal{M}_n . We say that μ is an *assembled \mathcal{M}_n -descriptor* if μ or $\bar{\mu}$ is of the form $M_i \dots M_n M_1 \dots M_{i-1}$, for some $1 \leq i \leq n$.

Consider now the alphabet of IESs $\mathcal{I}_n = \{I_0, I_1, \dots, I_n\}$. For any $J \subseteq \mathcal{M}_n \cup \mathcal{I}_n$, a signed permutation over J will be called an \mathcal{MIJ}_n -descriptor.

Let π_n be the projection $\pi_n : (\mathcal{M}_n \cup \mathcal{I}_n)^* \rightarrow \mathcal{M}_n^*$. We say that $\delta \in (\mathcal{M}_n \cup \mathcal{I}_n)^*$ is an *assembled \mathcal{MIJ}_n -descriptor* if $\pi_n(\delta)$ is an assembled \mathcal{M}_n -descriptor.

We can associate an \mathcal{MIJ}_n -descriptor to any \mathcal{M}_n -descriptor as follows. Let $\mu = \tilde{M}_{i_1} \tilde{M}_{i_2} \dots \tilde{M}_{i_n}$ be an \mathcal{M}_n -descriptor, where $\tilde{M}_{i_k} \in \{M_{i_k}, \bar{M}_{i_k}\}$ and i_1, i_2, \dots, i_n is a permutation over $\{1, 2, \dots, n\}$. Then the \mathcal{MIJ}_n -descriptor associated to μ is $\tau_\mu = I_0 \tilde{M}_{i_1} I_1 \tilde{M}_{i_2} I_2 \dots \tilde{M}_{i_n} I_n$. We denote by I_0, I_1, \dots, I_n the noncoding blocks separating the MDSs. We say in this case that τ_μ is a *micronuclear \mathcal{MIJ}_n -descriptor*.

Example 16 (i) The \mathcal{MIJ}_9 -descriptor associated to the actin I gene in *S. nova*, see Example 15, is $I_0 M_3 I_1 M_4 I_2 M_6 I_3 M_5 I_4 M_7 I_5 M_9 I_6 \bar{M}_2 I_7 M_1 I_8 M_8 I_9$.

(ii) $\delta = I_0 \bar{I}_3 \overline{M}_3 \overline{M}_2 \overline{M}_1 I_2 I_1$ is an assembled \mathcal{M}_3 -descriptor.

The \mathcal{MI}_n -descriptors can now be extended to include also the information about the position of pointers in the gene. Consider then the alphabet $\mathcal{P}_n = \{2, 3, \dots, n\}$ and denote the markers by b and e . Denote $\Sigma_n = \mathcal{M}_n \cup \mathcal{I}_n \cup \mathcal{P}_n \cup \{b, e\}$ and let π_n be the projection $\pi_n : \Sigma_n^* \rightarrow (\mathcal{M}_n \cup \mathcal{I}_n)^*$. We say that $\sigma \in \Sigma_n^*$ is a Σ_n -descriptor if it has one of the following forms:

- (i) $\sigma = \alpha_0 p_1 p_1 \alpha_1 p_2 p_2 \dots p_k p_k \alpha_k \alpha_0 \alpha_k \neq \Lambda$ or
- (ii) $\sigma = p_1 \alpha_1 p_2 p_2 \dots p_{k-1} p_{k-1} \alpha_k p_1$

where $k \geq 0$, $p_i \in \mathcal{P}_n \cup \overline{\mathcal{P}}_n$, $\alpha_i \in (\Sigma_n \setminus \mathcal{P}_n)^*$, for all $0 \leq i \leq k$ and moreover, $\pi_n(\sigma)$ is a \mathcal{MI}_n -descriptor. In case (i), we call σ *linear*, and in case (ii) we call it *circular*. We say that σ is *assembled* if $\pi_n(\sigma)$ is an assembled \mathcal{MI}_n -descriptor.

Example 17 (i) $\sigma_1 = 22M_233M_3 eI_2\bar{I}_3 bM_122$ is an assembled circular Σ_3 -descriptor.

- (ii) $\sigma_2 = I_022M_233I_1\bar{2}\bar{2}\bar{M}_1\bar{b}I_233 M_3eI_3$ is a linear micronuclear Σ_3 -descriptor.
- (iii) $\sigma_3 = 22M_23M_3eI_2\bar{I}_3 bM_122$ is not a Σ_3 -descriptor.

Every MDS of micronuclear ciliate genes is flanked at its both ends by a pointer or a marker. We denote the pointers flanking MDS M_i by writing $iiM_i(i+1)(i+1)$. For M_1 and M_n we write bM_122 and nnM_ne , respectively. We use a double letter notation for pointers in order to deal with splicing in a simple way in [Sect. 6.2](#). Formally, to associate a Σ_n -descriptor to an \mathcal{MI}_n -descriptor, consider the morphism $\phi_n : (\mathcal{M}_n \cup \mathcal{I}_n)^* \rightarrow \Sigma_n^*$ defined as follows: $\phi_n(I) = I$, for all $I \in \mathcal{I}_n$; $\phi_n(M_i) = iiM_i(i+1)(i+1)$, for all $2 \leq i \leq n-1$; $\phi_n(M_1) = bM_122$ and $f(M_n) = nnM_ne$. For any \mathcal{MI}_n -descriptor δ , we say that $\phi_n(\delta)$ is the Σ_n -descriptor *associated* to δ . It can be said that $\phi_n(\delta)$ is a *micronuclear* Σ_n -descriptor if δ is a micronuclear \mathcal{MI}_n -descriptor. Note that all micronuclear Σ_n -descriptors are linear.

Example 18 The micronuclear Σ_9 -descriptor of the actin I gene in *S.nova*, see Example 16, is $I_033M_344I_144M_455I_266M_677I_355M_566I_477M_788I_599M_9eI_6\overline{33M}_2\bar{2}\bar{2}I_7bM_122I_888M_899I_9$.

For any micronuclear Σ_n -descriptor σ and any $p \in \mathcal{P}_n$, σ contains two occurrences from the set $\{pp, \bar{p}\bar{p}\}$: pp represents the pointer in the beginning of MDS M_p and at the end of M_{p-1} , while $\bar{p}\bar{p}$ is its inversion.

6.2 Invariants

In this section, we give a number of invariants of the gene assembly process: the circularity of the assembled gene (whether or not the gene is assembled on a circular molecule), with the IES-context of the gene (the sequence of IESs preceding and succeeding the assembled gene), but also with the set of molecules excised during assembly. It is worth emphasizing that we establish all these properties based solely on the generic paradigm of pointer-directed assembly, independently of the specificities of either the intra-, or the inter-molecular model.

During the pointer-directed assembly, ciliates allegedly align their DNA molecules along their pointers, and through recombination they sort the MDSs in the orthodox order. It is essential to observe that in this process, the two strands of any pointer p will be separated: One strand will remain with the block preceding the pointer, while the other strand will remain

with the block succeeding the pointer. The single strands will then recombine with the complementary strands obtained by separating in a similar way the second occurrence of p in the gene. This splicing on pointers can be formalized by a word-cutting operation defined in the following.

Let σ be a Σ_n -descriptor. If σ is linear, $\sigma = \alpha_0 p_1 p_1 \alpha_1 p_2 p_2 \alpha_2 \dots \alpha_{k-1} p_k p_k \alpha_k$, with $p_i \in \mathcal{P}_n \cup \overline{\mathcal{P}}_n$, $\alpha_i \in (\Sigma_n \setminus \mathcal{P}_n)^\pm$, then $W_\sigma = \{\alpha_0 p_1, p_k \alpha_k, p_i \alpha_i p_{i+1} \mid 1 \leq i < k\}$. If σ is circular, $\sigma = p_1 \alpha_1 p_2 p_2 \alpha_2 \dots \alpha_{k-1} p_k p_k \alpha_k p_1$, then $W_\sigma = \{p_i \alpha_i p_{i+1}, p_k \alpha_k p_1 \mid 1 \leq i < k\}$. For any set $S \subseteq \Sigma_n^\pm$, we denote $W_S = \cup_{\sigma \in S} W_\sigma$. Note that the set W_σ is equivalent to the set of edges of genome graphs (☞ Sect. 5.6) and to the reality edges of reduction graphs (☞ Sect. 5.2).

It is important to note that we do not conjecture that ciliates split their genes by cutting simultaneously on each pointer, to yield on the scale of 10^5 MDSs and IESs, followed then by a precise reassembly of all these blocks. Indeed, it is difficult to imagine that such a mechanism would lead to the precise effective assembly that we see in ciliates. Here we merely represent those pointer-delimited coding and noncoding blocks that will be eventually reshuffled to assemble the gene. Our main result states that, given the fixed order in which MDSs must be assembled, the pointer-directed assembly of all the other blocks (IESs) is uniquely determined by the micronuclear structure of the gene.

Example 19 For the Σ_9 -descriptor σ in Example 18, we have

$$W_\sigma = \{I_03, 3M_34, 4I_14, 4M_45, 5I_26, 6M_67, 7I_35, 5M_56, 6I_47, 7M_78, 8I_59, \\ 9M_9eI_6\bar{3}, \bar{3}\overline{M_22}, \bar{2}I_7bM_12, 2I_88, 8M_89, 9I_9\}.$$

Our invariant theorem may be stated now as follows.

Theorem 15 (Petre 2006) *Let σ be a micronuclear Σ_n -descriptor.*

- (i) *There exists a set \mathcal{A}_σ of Σ_n -descriptors such that*
 - (a) $W_\sigma \cup \overline{W_\sigma} = W_{\mathcal{A}_\sigma} \cup \overline{W_{\mathcal{A}_\sigma}}$
 - (b) *there exists an assembled Σ_n -descriptor in \mathcal{A}_σ*
- (ii) *For any other set S of Σ_n -descriptors, if S satisfies conditions (a)-(b) above, then $S \approx \mathcal{A}_\sigma$*

Moreover, \mathcal{A}_σ consists of exactly one linear Σ_n -descriptor and possibly several circular ones.

Theorem 15 may be stated informally by saying that the final result of gene assembly, including the molecule where the assembled gene is placed, as well as all the other noncoding molecules excised in the process, is unique.

Example 20 Consider the Σ_9 -descriptor σ associated to gene actin I in *S.nova* in Example 18, with W_σ given in Example 19. It follows from Theorem 15 that the results of assembling the gene are

$$\{I_03\bar{3}\bar{I}_6\bar{e}\overline{M_9}\overline{99M_8}\overline{88M_7}\overline{77M_6}\overline{66M_5}\overline{55M_4}\overline{44M_3}\overline{33M_2}\overline{22M_1}\bar{b}\bar{I}_722I_888I_599I_9, 4I_14\}.$$

Thus, the noncoding block $4I_14$ is excised as a circular molecule and the gene is assembled linearly in the inverse order from $\overline{M_9}$ to $\overline{M_1}$ with the noncoding block $I_03\bar{3}\bar{I}_6$ preceding it and $\bar{I}_722I_888I_599I_9$ succeeding it.

Note that Theorem 15 holds both for the intra-molecular and the inter-molecular models for gene assembly. Consequently, the set of molecules generated by the assembly cannot be

used to distinguish between different assembly strategies, either intramolecular, or intermolecular. Instead, to (in)validate either model, one could experimentally identify the sets of molecules generated at various stages of the assembly and verify it against the predictions made by the two models.

Our results hold also in a more general way. We have proved that the final result \mathcal{A}_σ of assembling a micronuclear Σ_n -descriptor σ is unique modulo conjugation and inversion. As a matter of fact, our proofs apply unchanged also to the following variant proved in Brijder et al. (2006) for the intramolecular model. Let $P \subseteq \mathcal{P}_n$. There exists a unique (modulo conjugation and inversion) set $\mathcal{A}_{P,\sigma}$ of Σ_n -descriptors with the following property: $M_{p-1}ppM_p \leq \alpha$ for some $\alpha \in \mathcal{A}_{P,\sigma} \cup \overline{\mathcal{A}_{P,\sigma}}$ if and only if $p \in P$. In other words, if the assembly is to be done only on a given set P (that may be different from the total set \mathcal{P}_n), then the result is unique. To prove the result, it is enough to replace the morphism ϕ_n in [Sect. 6.1](#) with a morphism $\phi_{P,n}$ that only inserts pp in case $p \in P$. This extension of Theorem 15 does not contradict the non-determinism of gene assembly: The ciliate may choose to reduce the pointers in any order. The result above only says that after assembling on a *fixed* set of pointers, the result, including the excised molecules, is unique.

7 Template-Guided Recombination

In the previous sections, we have considered models that take an abstract view of the gene assembly process in ciliates. A lower, implementation-oriented, level of abstraction is considered next in which we attempt to begin addressing the question of how the assembly process takes place *in vivo*. Our quest for the discovery of the “biological hardware” responsible for implementing assembly begins with a simple examination of how MDSs might overlap to fit together, in the correct order, while also removing IESs. As has been noted above, each MDS is flanked by pointer sequences – that is, looking at the level of DNA sequence, there will be a proper suffix of MDS n that is equal to a proper prefix of MDS $n + 1$. Considering this type of structure for an entire gene of several MDSs, a computer scientist will immediately recognize the *linked list data structure*: the core of the MDS is the data while the pointer sequence indicates the “address” of the next data item (MDS). The initial state of MDSs, distributed throughout the MIC, is reminiscent of the nonlinear distribution of linked list data in heap memory. It is worth noting that the ciliate data structure is, in fact, significantly more sophisticated than a classic linked list; whereas each element of a linked list contains two separate components, an area for a data payload and an area for a pointer to the next element, the ciliate version of the linked list actually combines these two elements. Since the pointers always lie within the MDSs, the pointer to the next MDS is also part of the “data” contained in the MDS and is fully integrated into the assembled gene.

Completing the process of gene assembly, starting with the MIC and ending with the MAC, can be seen as implementing a linked-list specification. However, it does not appear that the pointers alone are guiding this implementation process as some of them may be too short to serve as unique pointers to the following MDSs. Yet the pointer sequences are still always present. A natural question to be answered by any suggested implementation of gene assembly is thus: “what role do the pointers play and why are they present?”

A realistic, biologically implementable, model must incorporate a number of principal features. It must be *irreversible*, it must be *self-propagating* or *reusable*, it cannot be sequence specific, that is, it cannot rely on the presence of certain fixed sequences, as a huge variety of pointer sequences are known (instead it must be *configuration specific*), and it must have some

mechanism for identifying the MDS/IES boundary and hence the pointers. The basic DNA template model of *template-guided recombination* was introduced in Prescott et al. (2001b) to address exactly these requirements that will be clarified in more detail after describing the model.

7.1 DNA Template-Guided Recombination

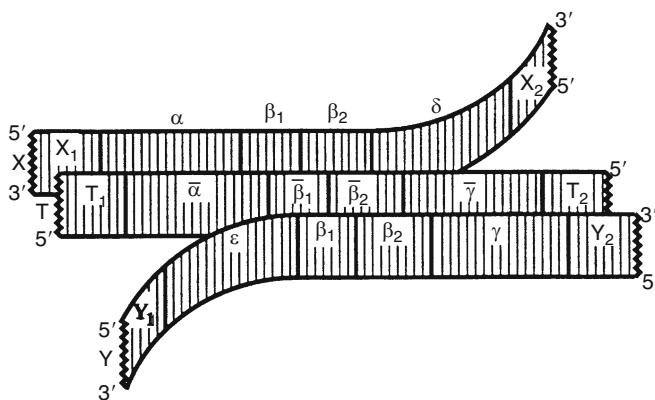
We consider here a schematic view of DNA as a picket fence with the sugar-phosphate backbones running horizontally along the top and bottom of the strand and the hydrogen bonds running vertically between the backbones. Suppose now that one wishes to assemble two strands of DNA, X and Y, having the sequences $X_1\alpha\beta\delta X_2$ and $Y_1\epsilon\beta\gamma Y_2$, respectively, where $\beta = \beta_1\beta_2$. In order to guide this assembly, we assume the existence of a DNA template T of the form $T_1\alpha\beta\gamma T_2$, which is placed in-between the two target strands X and Y, as seen in [Fig. 21](#). Note that we use the notation $\bar{\alpha}$ to denote the Watson–Crick complement of the DNA sequence α and that, in [Fig. 21](#), the $\alpha\beta$ region of X is now aligned with $\bar{\alpha}\beta$ on the template; similarly, $\beta\gamma$ of Y is aligned with its complement on the template. At the same time, the sequence of δ , beginning from the first nucleotide, must not be complementary to γ and, in a similar way, ϵ must not be complementary to α . It is important to note that X and Y need not be physically disconnected independent strands but may instead be different regions of a single, connected, strand of DNA.

The template-guided recombination now takes place in these principal steps:

- The hydrogen bonds in the $\alpha\beta_1$ region of X and the $\bar{\alpha}\beta_1$ in the template are broken and switch from binding the backbones within X and Y vertically to binding complementary sequence horizontally between X and Y; likewise for the second half of the template. In our fence analogy, the vertical pickets within DNA double-strands are replaced by floors and roofs across DNA strands, as pictured in [Fig. 22](#).
- Cuts are now made in the backbones of the roof/floor assembly (at the ends of the roof/floor structures) to yield the free-standing structure of [Fig. 23](#).
- The same cuts also yield a new copy of the original template strand, shown in [Fig. 24](#), and the strands $Y_1\epsilon\beta_1$ and $\beta_2\delta X_2$ (not pictured) that are left free to float away.

[Fig. 21](#)

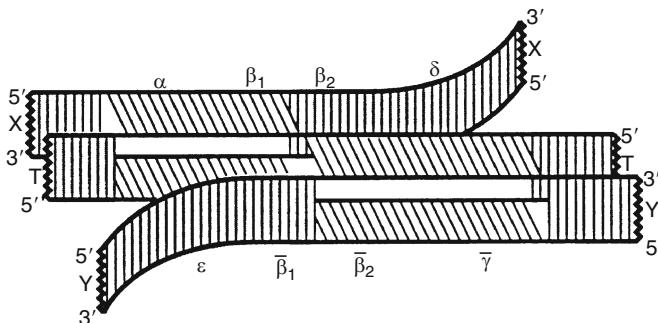
Sequences $X_1\alpha\beta\delta X_2$ and $Y_1\epsilon\beta\gamma Y_2$ stacked with DNA template $T_1\alpha\beta\gamma T_2$ in-between (from Prescott et al. (2001b)).



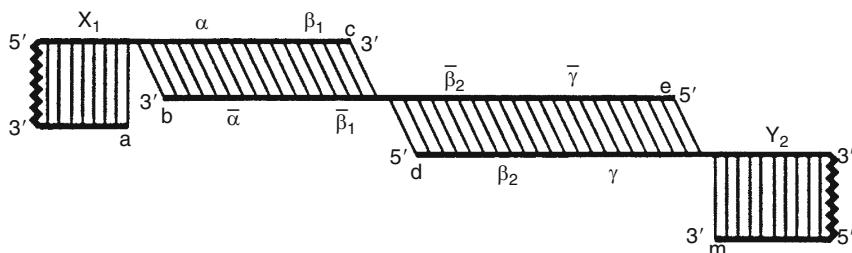
- The roof and floor structures of [Fig. 23](#) rotate to align the cut backbones that are then healed (via ligation) yielding the complete double-stranded DNA $X_1\alpha\beta\gamma Y_2$, which is the recombination of the prefix $X_1\alpha\beta_1$ of X and the suffix $\beta_2\gamma Y_2$ of Y – thus X and Y have been recombined.
- Likewise, the roof and floor structures of [Fig. 24](#) rotate, align, and ligate to yield $T_1\bar{\alpha}\bar{\beta}\bar{\gamma}T_2$ – thus the template is reconstituted.

Fig. 22

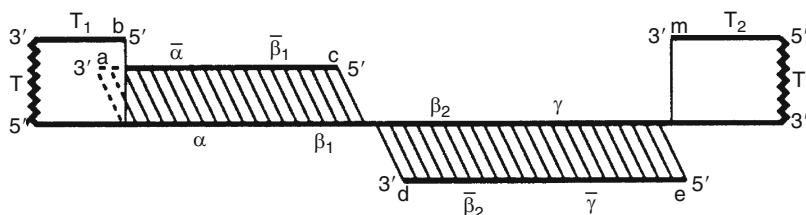
Hydrogen bonds switch from being vertical pickets holding individual strands together to forming floors and roofs across strands (from Prescott et al. (2001b)).

**Fig. 23**

One of the products resulting from cuts made in the backbones of the configuration of [Fig. 22](#) (from Prescott et al. (2001b)).

**Fig. 24**

The reconstituted template, resulting from cuts made in the backbones of the configuration of [Fig. 22](#) (from Prescott et al. (2001b)).



It can be seen clearly that the model, when considered in symbolic terms (as often done in computer science), meets our requirements for a biologically implementable system. The process of template-guided recombination may be described by the following implication:

$$\alpha\beta\delta + \alpha\beta\gamma + \epsilon\beta\gamma \Rightarrow \alpha\beta\gamma + \alpha\beta\gamma + \epsilon\beta_1 + \beta_2\delta$$

where $\beta_1\beta_2 = \beta$. On the left side, we have the first term consisting of the required subsequences for X , the second term is the template, and the third term is the required subsequence of Y . On the right side, the first term expresses the form of X recombined with Y , the form of the reconstituted template, and the last two terms are the forms of the “loose ends.” Note that we must have the components on the left side present in order for the reaction to take place. If they are present, this equation describes *what* happens, though not how it happens, and what we obtain is the four products on the right side. Note carefully that from this moment on, no three components of the right-hand side have the form required of the three components of the left-hand side – thus the process is irreversible (one-way).

Examining the right side, we see $\alpha\beta\gamma$ twice, demonstrating that the template in this model is self-propagating: we begin with one component $\alpha\beta\gamma$ on the left-hand side and after a single iteration we get two such components. Hence as the process progresses iteratively, we have an explanation of the growth of the number of available templates. Thus even if we begin with only a single copy of the template, we very quickly end up with an “abundance” of templates. This is necessary as we recall from [Sect. 2](#) on ciliate biology that many copies of the MIC chromosomes are present during the polytene chromosome stage so that multiple copies of each template are required to successfully assemble a full MAC genome.

Further, it is clear that the whole three-step process does not depend on *specific* sequences α, β, γ ; indeed, all that matters is the relationship between the sequences that causes the formation of a particular configuration.

Considering this process carefully, the true nature of pointers becomes apparent: Pointers are sequence segments within which the transfer of roofs, and dually the transfer of floors, takes place. Consequently, the most essential backbone cuts of the recombination process will take place in the pointer region. Pointers are records of transfers. Hence, in our scheme, pointers are the regions denoted $\beta = \beta_1\beta_2$ while $\alpha\beta$ and $\beta\gamma$ correspond to MDSs M_i and M_{i+1} .

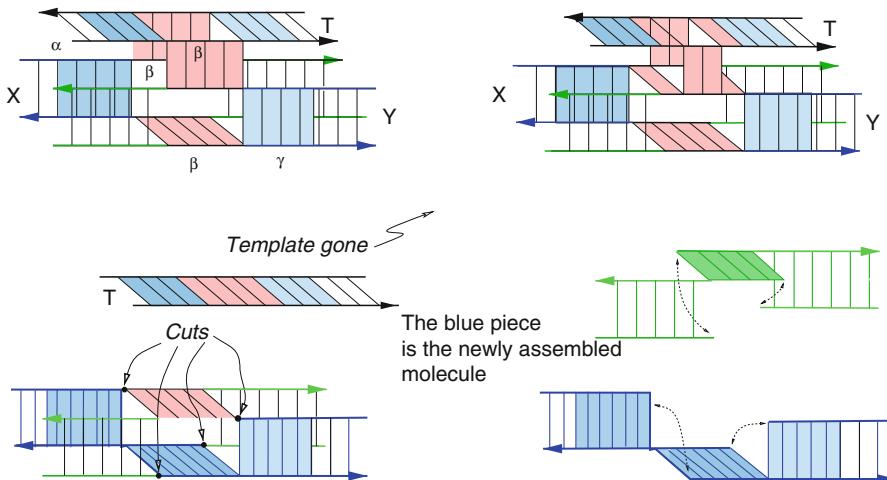
7.2 RNA Template-Guided Recombination

A variant of DNA template-guided recombination has been proposed in Angeleska et al. ([2007](#)), which considers *RNA templates*. Following through the steps of the DNA template model, one can see that a portion of the template strand ends up being incorporated into the final assembled product; this presents no problem for DNA templates but is infeasible with RNA templates since DNA and RNA backbones are incompatible. Rather than proposing a template which sits “in-between” the strands to be assembled, the RNA template model suggests a template that “hangs above” the strands to be recombined, guiding the recombination but never directly participating in it.

Consider DNA strands $\alpha\beta\delta$ and $\epsilon\beta\gamma$ again containing the MDSs $\alpha\beta$ and $\beta\gamma$ and a double-stranded RNA template molecule $\alpha\beta\gamma$. We stack the two substrate strands in tandem, as in the DNA model, but place the template horizontally above, rather than between, the substrates as in [Fig. 25](#). The $\bar{\beta}$ sequence of the RNA template begins to form hydrogen bonds with the sequence β in one of the substrates; note that while the backbones are incompatible,

Fig. 25

Reading left to right, top to bottom: The RNA template forms hydrogen bonds with β and $\bar{\beta}$ in the substrate strands, causing the formation of a floor between strands. When the RNA template is removed, a roof structure is now formed. Four cuts are made, followed by backbones swinging back into position and being healed to form the assembled strand (from Angeleska et al. (2007)).



it is certainly possible for RNA and DNA to share hydrogen bonds across two strands. The complementary strand of the template RNA similarly binds to the other substrate strand. With the internal “picket fence” hydrogen bonds of the substrate strands broken, the complementary portions at the bottom of each strand now form a floor of hydrogen bonds. If the RNA template is now removed, the complementary strands at the top of the substrates will form a roof of hydrogen bonds. Cutting the DNA backbones in the four places indicated in **Fig. 25** and rotating, followed by healing (ligation), the broken backbones similarly to the DNA model yields a correctly assembled DNA strand.

Note carefully that the RNA template is not integrated into the resulting structure; rather, the RNA template served only to break up the hydrogen bonds in the β region of the substrate strands, inducing the formation of a floor of hydrogen bonds between the two substrates, which then induced the formation of a complementary roof structure following the removal of the RNA template.

Yet another similar approach requires only a single-stranded RNA template, which is placed “diagonally” between the two DNA picket fence strands (details are given in Angeleska et al. (2007)).

The development of theoretical models of template-guided recombination has helped to direct biological inquiry into core questions surrounding gene assembly. It is natural, from a biological perspective, to ask where templates might originate. Both the RNA and DNA template-guided models suggest that templates are composed of sequence that is highly similar to that in the old MAC. Recent experimental results in the ciliate *Oxytricha trifallax* (*Sterkiella histriomuscorum*), guided by the insight provided in the theoretical models, support the hypothesis that short MAC-specific RNA templates are involved in gene assembly (Nowacki

et al. 2008). We also point out that the appendix of Angeleska et al. (2007) contains a straightforward RNA-template-combination explanation for the molecular operations Id , hi , and dlad introduced in [Sect. 5](#), which are the basis of the intramolecular models presented in this chapter.

7.3 Template-Guided Recombination on Words and Languages

We move now to consider theoretical research on the template model. Together with combinatorial models discussed in preceding sections, the theoretical work concerning the template model provided both some new insights into the nature of gene assembly and a whole spectrum of novel and interesting notions, models, and results for theoretical computer science.

A formal language theoretic version of the basic *DNA template-guided recombination* (abbreviated TGR) operation of Prescott et al. (2001b) was first studied in Daley and McQuillan (2005b). For words $x, y, z, t \in \Sigma^*$ and natural numbers $n_1, n_2 \geq 1$, we denote by z , the product of the recombination of x and y , guided by template t , by $(x, y) \vdash_{t, n_1, n_2} z$. More specifically, if $x = u_1\alpha\beta v_1$, $y = v_2\beta\gamma u_2$, $t = \alpha\beta\gamma$ with $\alpha, \beta, \gamma, u_1, u_2, v_1, v_2 \in \Sigma^*$, $|\alpha|, |\gamma| \geq n_1$ and $|\beta| = n_2$, then we may write the TGR product $z = u_1\alpha\beta\gamma u_2$. If $T, L \subseteq \Sigma^*$ are languages, then $\pitchfork_{T, n_1, n_2}(L)$ is defined by

$$\pitchfork_{T, n_1, n_2}(L) = \{z : \exists x, y \in L, t \in T \text{ such that } (x, y) \vdash_{t, n_1, n_2} z\}$$

The shorthand notation $\pitchfork_T(L)$ is used whenever n_1, n_2 are understood. We note that the restriction on pointer length, $|\beta| = n_2$, is not as strict as it appears since it has been proven equivalent to the restriction $|\beta| \geq n_2$ in Daley and McQuillan (2005b).

Given the nature of biochemical reactions, it is natural to consider an iterated version of TGR as well: let $\pitchfork_{T, n_1, n_2}^0(L) = L$ and for all $i \geq 1$, let

$$\pitchfork_{T, n_1, n_2}^i(L) = \pitchfork_{T, n_1, n_2}^{i-1}(L) \cup \pitchfork_{T, n_1, n_2}(\pitchfork_{T, n_1, n_2}^{i-1}(L))$$

Then we also define $\pitchfork_{T, n_1, n_2}^*(L)$ as

$$\pitchfork_{T, n_1, n_2}^*(L) = \bigcup_{i \geq 0} \pitchfork_{T, n_1, n_2}^i(L)$$

Finally, let \mathcal{L}, \mathcal{T} be classes of languages and $n_1, n_2 \geq 1$. We define the following closure classes:

$$\begin{aligned}\pitchfork_{\mathcal{T}, n_1, n_2}(\mathcal{L}) &= \{\pitchfork_{T, n_1, n_2}(L) : T \in \mathcal{T}, L \in \mathcal{L}\} \\ \pitchfork_{\mathcal{T}, n_1, n_2}^*(\mathcal{L}) &= \{\pitchfork_{T, n_1, n_2}^*(L) : T \in \mathcal{T}, L \in \mathcal{L}\}\end{aligned}$$

The families of finite languages are denoted by `FIN` and regular languages by `REG`, and we recall that a family of languages is said to be a full AFL if it is closed under homomorphism, inverse homomorphism, intersection with regular languages, union, concatenation, and Kleene plus.

On initial inspection of TGR, there appears to be a similarity with the well-known model of splicing systems, but this relationship has been shown to be superficial in Daley and McQuillan (2005b). Before the relevant formal result is stated, we recall the basic operational scheme of splicing systems.

A *splicing scheme* or *H scheme* is a pair $\sigma = (\Sigma, R)$ where Σ is an alphabet and $R \subseteq \Sigma^* \# \Sigma^* \$ \Sigma^* \# \Sigma^*$ is a set of splicing rules where $\$, \#$ are not elements of Σ . For a rule $r \in R$, we

define the relation $(x, y) \models_r z$ if $r = u_1 \# u_2 \$ u_3 \# u_4$, $x = x_1 u_1 u_2 x_2$, $y = y_1 u_3 u_4 y_2$, $z = x_1 u_1 u_4 y_2$, for some $u_1, u_2, u_3, u_4, y_1, y_2, x_1, x_2 \in \Sigma^*$.

For a language $L \subseteq \Sigma^*$ and an H scheme $\sigma = (\Sigma, R)$, we define $\sigma(L) = \{z \in \Sigma^* : \exists x, y \in L, r \in R \text{ such that } (x, y) \models_r z\}$ and extend to iterated splicing as follows: let $\sigma^0(L) = L$ and $\sigma^i(L)$ be defined by $\sigma^i(L) = \sigma^{i-1}(L) \cup \sigma(\sigma^{i-1}(L))$ for all $i \geq 1$. Finally, as expected,

$$\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L)$$

For classes of languages \mathcal{L}, \mathcal{R} , let $H(\mathcal{L}, \mathcal{R}) = \{\sigma^*(L) : L \in \mathcal{L}, \sigma = (\Sigma, R), R \in \mathcal{R}\}$.

Lemma 3 (Daley and McQuillan 2006) *For all $n_1, n_2 \geq 1$, for all full AFLs \mathcal{L} :*

$$\mathcal{L} = \dot{\cap}_{\text{FIN}, n_1, n_2}^*(\mathcal{L}) = H(\mathcal{L}, \text{FIN}),$$

while for all finite languages and templates:

$$\text{FIN} \subset \dot{\cap}_{\text{FIN}, n_1, n_2}^*(\text{FIN}) \subset H(\text{FIN}, \text{FIN}) \subset \text{REG}$$

Despite this result, it has been shown in Daley and McQuillan (2005b) that every regular language is the coding of a language in $\dot{\cap}_{\text{FIN}}^*(\text{FIN})$ demonstrating a relatively modest computational power for single-application TGR.

To investigate the computational power of the iterated case, it is necessary to define the notion of a “useful” template; we say that a template $t \in T$ is *useful* on L, n_1, n_2 if there exists $u_1 \alpha \beta v_1, v_2 \beta \gamma u_2 \in \dot{\cap}_{T, n_1, n_2}^*(L)$ with $|\alpha|, |\gamma| \geq n_1$, $|\beta| = n_2$, $u_1, u_2, v_1, v_2 \in \Sigma^*$ and $t = \alpha \beta \gamma$. If every template $t \in T$ is useful on L, n_1, n_2 , then we say that T is useful on L, n_1, n_2 . The following results, demonstrating the surprisingly limited power of iterated TGR, were shown in Daley and McQuillan (2006):

Theorem 16 (Daley and McQuillan 2006) *Let $n_1, n_2 \geq 1$, \mathcal{L} be a full AFL and $L, T \in \mathcal{L}$. If T is useful on L, n_1, n_2 , then $\dot{\cap}_{T, n_1, n_2}^*(L) \in \mathcal{L}$.*

Corollary 1 (Daley and McQuillan 2006) *Let $n_1, n_2 \geq 1$. For all full AFLs \mathcal{L} , $\dot{\cap}_{\text{REG}, n_1, n_2}^*(\mathcal{L}) = \mathcal{L}$.*

The problem of template equivalence, viz. “Given sets of templates T_1, T_2 , are $\dot{\cap}_{T_1}$ and $\dot{\cap}_{T_2}$ identical operations?” has been considered in Domaratzki (2007), which gives a characterization of when two sets of templates define the same TGR operation in formal language theoretic terms, leading to the following decidability result.

Theorem 17 (Domaratzki 2007) *Let $n_1, n_2 \geq 1$ and $T_1, T_2 \subseteq \Sigma^*$ ($|\Sigma| \geq 3$) be regular sets of templates. Then it is decidable whether or not $\dot{\cap}_{T_1, n_1, n_2}(L) = \dot{\cap}_{T_2, n_1, n_2}(L)$ for all $L \subseteq \Sigma^*$.*

Several variants of TGR have been studied as well, including a computationally universal version with added deletion contexts (Daley and McQuillan 2005a) and a purely intramolecular version that resembles a templated version of the `ld` operation (Daley et al. 2007) discussed above.

In addition to the very literal formalization of TGR considered in this section, the underlying theoretical model has also inspired work at a more abstract level.

7.4 Covers from Templates

The process of gene descrambling may be abstractly formulated in simple terms as a procedure that takes MDSs from the MIC and connects them, via overlap, to form the MAC. The template-guided recombination model discussed above provides a concrete suggestion of how this process might be implemented; one can view a template, T , as a sort of magnet, which glues together regions of MIC chromosomes containing MDSs to form orthodox MAC genes. Returning to a more abstract level, we now consider the set of all MDSs as our primary object of study. In this view, a template is now a request: “with this set of segments(MDSs), please cover me”: and therefore our core question is now “given a set of segments, how can a particular word be covered with these segments?” This leads naturally to the study of various properties of coverings, and the notion of uniqueness of coverings formalized as *scaffolds* presented in Ehrenfeucht and Rozenberg (2006).

An *interval* is a set of integers of the form $\{n, n+1, \dots, n+m\}$, where $n \in \mathbb{Z}$, and $m \in \mathbb{N}$. For a given alphabet Σ , we define a *segment* as a function $f:A \rightarrow \Sigma$, where A is an interval, and denote the set of all segments over Σ by \mathcal{S}_Σ . Since f is a function, we alternatively view segments as sets of ordered pairs of the form $(n, f(n))$, called *elements off*, with n called the *location* of $(n, f(n))$; thus, elements of f are ordered through their locations. This point of view is very convenient as it provides a set-theoretical calculus of segments: we can consider inclusions, union, intersections, differences, . . . of segments. Also, in this way, a set of segments is a family of sets.

For a set $C \subseteq \mathcal{S}_\Sigma$ and a segment $f \in \mathcal{S}_\Sigma$, we say that C *covers* f if $f = \bigcup C$. Intuitively, f is covered by C if each element of f is present in at least one segment of C , and all elements of all segments of C are present in f . Note that in general an element of f may be present in several segments of C , that is, the segments of C may overlap. One may also have redundant segments in C , that is, segments which cover only elements of f that are already covered by other segments of C . We thus say that a cover C of f is *tight* if for every $z \in C$, $C - \{z\}$ is not a cover of z .

Often covers are chosen from a subset of \mathcal{S}_Σ . Given such an $F \subseteq \mathcal{S}_\Sigma$, and a cover C of f with $C \subseteq F$ we say that C is a *small cover* of f (with respect to F) if $|C| \leq |Z|$ for all $Z \subseteq F$ covering f . Note that the property of being a small cover is a global property: C is a small cover of f with respect to F if any other cover Z of f , $Z \subseteq F$, has at least as many segments as C . The set of all small covers of f with respect to F is denoted $SC_F(f)$, and the *small index* of f (with respect to F) is the cardinality of the small covers of f (with respect to F). For any small cover C of f with respect to F , we get a natural order $C(1), \dots, C(m)$ of C , where m is the small index of f , and the order is determined by increasing locations of first elements of the segments of C .

Example 21 Let $\Sigma = \{a, b, c\}$ and $f \in \mathcal{S}_\Sigma$ be defined by

$$f = \{(3, a), (4, b), (5, b), (6, a), (7, c)\}$$

which may be abbreviated as $f = (3, abbac)$ since f begins at location 3.

Consider the sets $F = \{(3, ab), (4, bb), (4, bba), (5, bac), (6, ac)\}$ and $C = \{(3, ab), (4, bb), (6, ac)\}$. It is clear that C covers f and is tight, since no element of C can be removed while still covering f ; however, with respect to F , C is not small since $|C| = 3$ and the set $\{(3, ab), (5, bac)\} \subseteq F$ also covers f and has cardinality 2.

From the point of view of the original biological motivation, the segment f represents a scrambled MAC gene while the set F is the collection of MIC gene fragments available for assembly. We now proceed to investigate the structure of f by considering the family of all small covers of f with respect to some fixed F .

Let $f \in \mathcal{S}_\Sigma$, $F \subseteq \mathcal{S}_\Sigma$ be such that it contains a cover of f , and let m be the small index of f with respect to F . Let $1 \leq i \leq m$. The i th kernel of f with respect to F (denoted $\text{ker}_{i,F}(f)$) is defined by

$$\text{ker}_{i,F}(f) = \left(\bigcap_{C \in SC_F(f)} C(i) \right) - \bigcup_{\substack{C \in SC_F(f) \\ j \neq i}} C(j)$$

We now define the *scaffold* of f (with respect to F) as the set $\{\text{ker}_{1,F}(f), \dots, \text{ker}_{m,F}(f)\}$ of all kernels of f .

Theorem 18 (Ehrenfeucht and Rozenberg 2006) *For each $1 \leq i \leq m$, $\text{ker}_{i,F}(f)$ is a nonempty segment.*

For any segment f , the choice of F determines a certain natural class of “maximal” subsegments of f ; namely, those subsegments that are not strict subsegments of other subsegments. Let $P_F(f)$ be the set of all subsegments of f that belong to F . We say that a segment $g \in P_F(f)$ is *long* (with respect to F) if it is not properly included in any other segment in $P_F(f)$. The set of all long segments of f (with respect to F) is denoted by $LP_F(f)$. Additionally, we call a cover long if it consists solely of long segments.

Attention can now be turned to the study of the canonical class of covers that have both the “long” and “small” property. For each $1 \leq i \leq m$, let $LP_F(f,i) = \{y \in LP_F(f): \text{ker}_{i,F} \subseteq y\}$. That is, we categorize the long segments of f with respect to F according to containment of kernels. Since $LP_F(f,i)$ is an ordered set, we let $rt_F(f,i)$ (resp., $lt_F(f,i)$) be the maximal, or rightmost (resp., minimal, or leftmost) element of $LP_F(f,i)$. The following result provides a method for constructing “canonical” small covers of f .

Theorem 19 (Ehrenfeucht and Rozenberg 2006) *The sets $\{rt_F(f,1), \dots, rt_F(f,m)\}$ and $\{lt_F(f,1), \dots, lt_F(f,m)\}$ are long small covers of f with respect to F .*

More detailed analysis of the structure of scaffolds is given in Ehrenfeucht and Rozenberg (2006).

7.5 Topology-Based Models

An important question that arises from considering template-guided recombination concerns the three-dimensional structure of DNA undergoing multiple recombination events. Recently, two new approaches to this question have been undertaken. The physical structure of the DNA

strand undergoing recombination is directly considered in Angeleska et al. (2007) through the use of virtual knot diagrams. Micronuclear genes are represented in a schematic form that explicitly denotes only the relative locations of the pointer sequences. Consider, for example, the Uroleptus gene *USGI* (Chang et al. 2006) that has the following MDS descriptor: $M_1M_3M_4M_5M_7M_{10}M_{11}M_6M_8M_2M_9$. The corresponding legal string is the following: 2 3 4 5 6 7 8 10 11 11 6 7 8 9 2 3 9 10. It is now possible to interpret this sequence as a Gauss code. Each symbol in a Gauss code must occur exactly twice and to each code we associate a virtual knot diagram in a similar manner to the construction of the recombination graphs exposed above:

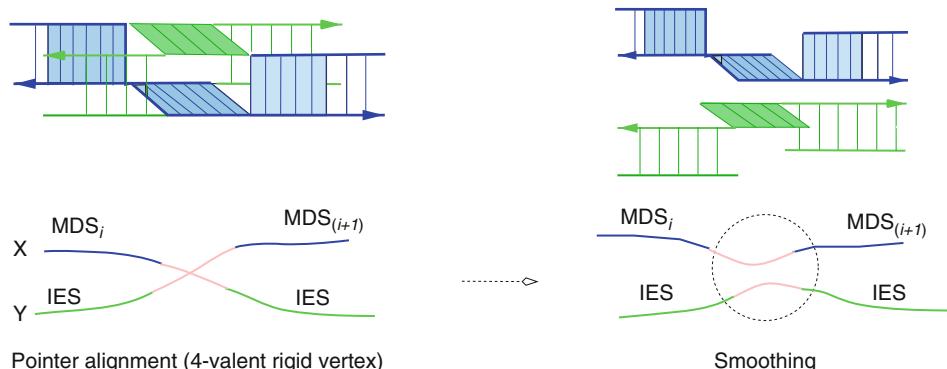
- For each symbol occurring in the code, we place a (disconnected) crossing in the plane and label the crossing with the corresponding symbol. A crossing may be thought of as similar to a vertex with predetermined order of the incident edges of degree 4 in a graph.
- We choose an arbitrary point in the plane to denote as the base point.
- Following a chosen direction, we connect crossings according to the order in the Gauss code. In our example, we would draw arcs from the base point to crossing 2, crossing 2 to crossing 3, crossing 3 to crossing 4, crossing 4 to itself, crossing 4 to crossing 5, and so forth. Each crossing corresponds to a “roof-floor” structure depicted in Fig. 26 left.
- We connect the remaining arc leaving the final crossing back to the base point.

Note that during the construction of the virtual knot diagram, it might happen that we have to cross an already sketched arc. This crossing is not labeled and does not correspond to a required pointer-guided homologous recombination, therefore it is called a “virtual crossing.” The virtual crossings correspond to a cross-over embedding of the DNA in space when one helix crosses over another.

The virtual knot diagram is now relabeled with each crossing receiving the label of its associated pointer and indicating inverted pointers with a bar. The process of assembly is now reduced to one of *smoothing* the crossings of the virtual knot diagram (see Fig. 26 right). The smoothing of a crossing consists of eliminating the crossing by splicing together the arcs of the crossing according to the pointers: if the pointers are not inverted, we splice the arcs together

Fig. 26

Roof-floor structure of a crossing in a virtual knot diagram (from Angeleska et al. (2007)).



following the orientation; if the pointers are inverted, we splice the arcs together opposite to the orientation. The result of a simultaneous smoothing of all crossings is a virtual link diagram with no real crossings remaining; note that the link may be composed of multiple components. If the arcs of the virtual knot diagram are labeled with the respective MDS and IES names (see Angeleska et al. (2007)), then we have the following theorem.

Theorem 20 (Angeleska et al. 2007) *For every labeled virtual knot diagram derived from a representation of a scrambled gene G, there exists a simultaneous smoothing yielding a link with a component C containing a subarc labeled with all MDSs in orthodox macronuclear order.*

Note that the basic mechanism of gene assembly through the smoothing of virtual knot diagrams was already introduced in Ehrenfeucht et al. (2002) in terms of a graph-based model, which was also briefly discussed in [Sect. 5.6](#). This model is based on the notion of recombination graph representing pointers as vertices, and MDSs and IESs as edges between the vertices standing for their flanking pointers. The recombination graph is first subject to a process of *graph folding* yielding a structure similar to the virtual knot diagram. A subsequent step of *graph unfolding* is similar to that of *smoothing* above and yields a representation of the assembled gene and of all the molecules excised during gene assembly. In this way, the approach of virtual knot diagrams from Angeleska et al. (2007) is a translation of the graph-theoretic approach from Ehrenfeucht et al. (2002) into a topological framework. This allows one to apply a rich set of techniques from both graph theory and knot theory to the investigation of gene assembly.

One significant issue that has not yet been addressed concerns the thermodynamics of template-guided recombination. For template-guided recombination to be implemented as suggested in the theoretical models requires some intricate positioning and biochemical operations; specifically, it requires the juxtaposition of three nucleic acid strands in space followed by a strand branch migration process as is observed *in vivo* (see, e.g., Dennis et al. (2004) and Zerbib et al. (1998)) and *in vitro* (see, e.g., Yan et al. (2002) and Yurke et al. (2000)). In order for such a branch migration process to start, it is possibly necessary that some of the cuts in the molecule ([Fig. 25](#) bottom left) appear early in the process. On the other side, it is possible to suppose that the juxtaposition of the molecules is artificially created by as-yet undiscovered enzymatic mechanics. Yet another possibility would be a simple argument indicating the thermodynamic favorability of the process. In the absence of such complex additional enzymatic machinery, template-guided recombination must rely upon diffusion processes within the cell to juxtapose the template and substrate strands. It is a well-known theorem in mathematics that random walks in three-space need never pass through the same point twice, so the likelihood of juxtaposing three molecules, in three-space, seems low. An alternative knot theory-based model presented in Daley et al. (2010) attempts to reduce the complexity of this problem by considering the effects of individual recombinations on the structure of the substrate DNA.

When circular DNA undergoes recombination, it is supercoiled due to twist applied by the recombination machinery (Krasnow et al. 1983; White 1992). Where linear DNA has unbound ends that are free to rotate and relax, thus removing the induced twist, circular DNA cannot relax in this fashion after recombination. Instead, the relaxation of the DNA strand induces supercoiling of the DNA molecule. The connection between linking, twist, and writhe of the DNA has been observed decades ago (White 1992), therefore, multiple recombinations on a closed circle of DNA can lead to a knotted, supercoiled strand.

It is suggested in Daley et al. (2010) that DNA is assembled in a closed circular topology where the twist induced by a template-guided recombination (or multiple recombinations)

causes a new, potentially knotted and supercoiled, topology to form. This new topology could facilitate the juxtaposition of the “next” regions of DNA to be assembled.

8 Discussion

In this chapter, we have discussed a number of topics related to research on the computational nature of gene assembly in ciliates, including the two main models for gene assembly. Among others, we have discussed their mathematical formalizations, invariant properties, template-based DNA recombination, and topology-based models for gene assembly. Due to space restrictions, all lines of research could not be discussed; for the sake of completeness, some of the topics that were not covered are now mentioned briefly.

The organization of the micronuclear genes into broken and shuffled MDSs, separated by IESs, is one of the characteristic features of the ciliates. To explain the evolutionary origin of this organization, a somewhat geometrical hypothesis based on a novel proposal for DNA repair is suggested in Ehrenfench et al. (2007).

Approaches based on formal languages have been introduced for both the intermolecular model and the intramolecular model, leading to very diverse research topics: computability, language equations, closure properties, hierarchies of classes of languages, etc. A typical approach is to consider contextually based applications of string rewriting rules; see, for example, Landweber and Kari (1999) for the intermolecular model. A derivation relation and an axiom are introduced, thus obtaining an acceptance mechanism: Start with a multiset of strings, for example, consisting of several copies of the input string and eventually derive a multiset containing the axiom. It can be proved that such a mechanism is a universal computing device. A similar result can be obtained also based on the intramolecular model, see Ishdorj et al. (2007), where the multisets are replaced with single strings. The idea is to concatenate several copies of the input string rather than having them in a multiset. A generating, rather than accepting, computing device inspired by gene assembly was also considered in Dassow and Vaszil (2006). Defining non-contextual string-based rewriting rules inspired by the molecular operations in either model leads to language operations and to questions related to closure properties, or solutions of language equations, see Daley et al. (2003a), Dassow and Holzer (2005), and Freund et al. (2002), and also Dassow et al. (2002) for a more general framework. Language operations inspired by the template-based DNA recombination were considered in Daley and McQuillan (2005b) and in Domaratzki (2007).

The original, non-contextual, intramolecular and intermolecular operations were generalized to the synchronized insertion and deletion operations on linear strings in Daley et al. (2003a). Let α, β be two nonempty words in an alphabet Σ^* . The synchronized insertion of β into α is defined as $\alpha \oplus \beta = \{uxvxw \mid \alpha = uxw, \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$. While the synchronized deletion of β from α is defined as $\alpha \ominus \beta = \{uxw \mid \alpha = uxvxw, \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$. All language families in the Chomsky hierarchy were shown to be closed under synchronized insertion while only the families of regular and recursively enumerable languages were closed under synchronized deletion. The existence of a solution was shown to be decidable for language equations of the form $L \odot Y = R$ and $X \odot L = R$ where \odot is one of the synchronized insertion or synchronized deletion operations and L, R are regular languages. The same problems are undecidable in the case that L is a context-free language.

More general results considering families of languages defined by reversal-bounded counter machines are also given in Daley et al. (2003a) along with results for a similarly generalized version of the *hi* operation. Generalized versions of the *ld* and *dld* operations are considered in Daley et al. (2003b) while families of languages defined by closure under these generalized operations are examined in Daley et al. (2004).

Two novel classes of codes based on synchronized insertion were defined and studied in Daley and Domaratzki (2007): the synchronized outfix codes (\oplus -codes) defined by the equation $(L \oplus \Sigma^+) \cap L = \emptyset$ and the synchronized hypercodes (\otimes -codes) defined by $(L \otimes \Sigma^+) \cap L = \emptyset$ (where \otimes is the transitive closure of \oplus , namely synchronized scattered insertion). The \oplus -codes and \otimes -codes are shown to be completely disjoint from the regularly studied classes of $*$ -codes and it is demonstrated that it is decidable if a regular language is an \oplus -code while the same property is undecidable for linear context-free languages. It is, however, decidable if an arbitrary context-free language is an \otimes -code while this same property is, unsurprisingly, undecidable for context-sensitive languages.

A different computability approach was developed in Alhazov et al. (2008), where the process of gene assembly is used to solve an NP-complete problem. Conceptually, this is important since gene assembly is confluent, see [Sect. 5](#), that is, it yields a computing device that answers “yes” to all legal inputs. A way around the problem is to make the device highly nondeterministic by extending its set of legal inputs. For example, strings with more than two occurrences of each letter may be allowed. With this modification, a suitably defined model can be introduced to solve the Hamiltonian path problem (HPP) by mimicking gene assembly on an encoding of the input to HPP, see Alhazov et al. (2008). The intermolecular model also leads to solutions to computational problems, see Ishdorj et al. (2008) for a solution to the satisfiability problem. Yet another approach based on Boolean circuits was investigated in Ishdorj and Petre (2008). Algorithmic questions related to finding a gene assembly strategy were considered in Ilie and Solis-Oba (2006).

The topological model of gene assembly with virtual knot diagrams raises new mathematical questions. Considering that a virtual knot diagram could represent the physical structure of the micronuclear DNA at the time of recombination, in Angeleska et al. (2009) it was shown that for every such virtual knot diagram there is an embedding of the molecule in space, and there is smoothing of the vertices (recombination along the pointers) such that the resulting molecule is always unlinked. Further, it was shown that the smoothing guided by the pointers differs from the existent smoothing notions defined earlier for virtual knot diagrams (Kauffman 1999). This opens completely new problems on virtual knot diagrams that have not been studied before.

Research on the computational nature of gene assembly is an example of genuinely interdisciplinary research that contributed to both computer science, through a whole range of novel and challenging models of computation, and to biology, by increasing our understanding of the biological nature of gene assembly – it has even led to formulating biological models of this process based on the notion of template-guided recombination.

Acknowledgments

MD and GR acknowledge support by NSF, grant 0622112. IP acknowledges support by the Academy of Finland, grants 108421 and 203667. NJ has been supported in part by the NSF grants CCF 0523928 and CCF 0726396.

References

- Alhazov A, Petre I, Rogojin V (2008) Solutions to computational problems through gene assembly. *J Nat Comput* 7(3):385–401
- Alhazov A, Petre I, Rogojin V (2009) The parallel complexity of signed graphs: Decidability results and an improved algorithm. *Theor Comput Sci* 410 (24–25):2308–2315
- Alhazov A, Li C, Petre I (2010) Computing the graph-based parallel complexity of gene assembly. *Theor Comput Sci* 411(25):2359–2367
- Angeleska A, Jonoska N, Saito M, Landweber LF (2007) RNA-template guided DNA assembly. *J Theor Biol* 248:706–720
- Angeleska A, Jonoska N, Saito M (2009) DNA recombination through assembly graphs. *Discrete Appl Math* 157(14):3020–3037
- Brijder R (2008) Gene assembly and membrane systems. PhD thesis, University of Leiden
- Brijder R, Hoogeboom H (2008a) The fibers and range of reduction graphs in ciliates. *Acta Inform* 45:383–402
- Brijder R, Hoogeboom HJ (2008b) Extending the overlap graph for gene assembly in ciliates. In: Martín-Vide C, Otto F, Fernau H (eds) LATA 2008: 2nd international conference on language and automata theory and applications, Tarragona, Spain, March 2008. Lecture notes in computer science, vol 5196. Springer, Berlin Heidelberg, pp 137–148
- Brijder R, Hoogeboom H, Rozenberg G (2006) Reducibility of gene patterns in ciliates using the breakpoint graph. *Theor Comput Sci* 356:26–45
- Brijder R, Hoogeboom H, Muskulus M (2008) Strategies of loop recombination in ciliates. *Discr Appl Math* 156:1736–1753
- Cavalcanti A, Clarke TH, Landweber L (2005) MDS_IES_DB: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucl Acids Res* 33:396–398
- Chang WJ, Kuo S, Landweber L (2006) A new scrambled gene in the ciliate *Uroleptus*. *Gene* 368:72–77
- Daley M, Domaratzki M (2007) On codes defined by bio-operations. *Theor Comput Sci* 378(1):3–16
- Daley M, McQuillan I (2005a) On computational properties of template-guided DNA recombination. In: Carbone A, Pierce N (eds) DNA 11: Proceedings of 11th international meeting on DNA-based computers, London, Ontario, June 2005. Lecture notes in computer science, vol 3892. Springer, Berlin, Heidelberg, pp 27–37
- Daley M, McQuillan I (2005b) Template-guided DNA recombination. *Theor Comput Sci* 330 (2):237–250
- Daley M, McQuillan I (2006) Useful templates and iterated template-guided DNA recombination in ciliates. *Theory Comput Syst* 39(5):619–633
- Daley M, Ibarra OH, Kari L (2003a) Closure properties and decision questions of some language classes under ciliate bio-operations. *Theor Comput Sci* 306 (1–3): 19–38
- Daley M, Ibarra OH, Kari L, McQuillan I, Nakano K (2003b) The ld and dld bio-operations on formal languages. *J Autom Lang Comb* 8(3):477–498
- Daley M, Kari L, McQuillan I (2004) Families of languages defined by ciliate bio-operations. *Theor Comput Sci* 320(1):51–69
- Daley M, Domaratzki M, Morris A (2007) Intramolecular template-guided recombination. *Int J Found Comput Sci* 18(6):1177–1186
- Daley M, McQuillan I, Stover N, Landweber LF (2010) A simple topological mechanism for gene descrembling in Stichotrichous ciliates. (under review)
- Dassow J, Holzer M (2005) Language families defined by a ciliate bio-operation: hierarchies and decidability problems. *Int J Found Comput Sci* 16(4):645–662
- Dassow J, Vaszil G (2006) Ciliate bio-operations on finite string multisets. In: Ibarra OH, Dang Z (eds) 10th international conference on developments in language theory, Santa Barbara, CA, June 2006. Lecture notes in computer science, vol 4036. Springer, Berlin Heidelberg, pp 168–179
- Dassow J, Mitrana V, Salomaa A (2002) Operations and languages generating devices suggested by the genome evolution. *Theor Comput Sci* 270(1–2):701–738
- Dennis C, Fedorov A, Kás E, Salomé L, Grigoriev M (2004) RuvAB-directed branch migration of individual Holliday junctions is impeded by sequence heterology. *EMBO J* 23:2413–2422
- Domaratzki M (2007) Equivalence in template-guided recombination. *J Nat Comput* 7(3):439–449
- Ehrenfeucht A, Rozenberg G (2006) Covers from templates. *Int J Found Comput Sci* 17(2):475–488
- Ehrenfeucht A, Petre I, Prescott DM, Rozenberg G (2000) Universal and simple operations for gene assembly in ciliates. In: Mitrana V, Martin-Vide C (eds) Where mathematics, computer science, linguistics and biology meet. Kluwer, Dordrecht, pp 329–342
- Ehrenfeucht A, Petre I, Prescott DM, Rozenberg G (2001a) Circularity and other invariants of gene assembly in ciliates. In: Ito M, Păun G and Yu S (eds) Words, semigroups, and transductions. World Scientific, Singapore, pp 81–97
- Ehrenfeucht A, Prescott DM, Rozenberg G (2001b) Computational aspects of gene (un)scrambling in ciliates. In: Landweber LF, Winfree E (eds)

- Evolution as computation. Springer, Berlin, Heidelberg, New York, pp 216–256
- Ehrenfeucht A, Harju T, Rozenberg G (2002) Gene assembly in ciliates through circular graph decomposition. *Theor Comput Sci* 281:325–349
- Ehrenfeucht A, Harju T, Petre I, Prescott DM, Rozenberg G (2003a) Computation in living cells: gene assembly in ciliates. Springer, Berlin, Heidelberg, New York
- Ehrenfeucht A, Harju T, Petre I, Prescott DM, Rozenberg G (2003b) Formal systems for gene assembly in ciliates. *Theor Comput Sci* 292:199–219
- Ehrenfeucht A, Prescott DM, Rozenberg G (2007) A model for the origin of internal eliminated segments (IESs) and gene rearrangement in stichotrichous ciliates. *J Theor Biol* 244(1):108–114
- Freund R, Martin-Vide C, Mitrana V (2002) On some operations on strings suggested by gene assembly in ciliates. *New Generation Comput* 20(3):279–293
- Harju T, Petre I, Rozenberg G (2004a) Two models for gene assembly. In: Karhumäki J, Păun G, Rozenberg G (eds) Theory is forever. Springer, Berlin, pp 89–101
- Harju T, Petre I, Rozenberg G (2004b) Gene assembly in ciliates: formal frameworks. In: Paun G, Rozenberg G, Salomaa A (eds) Current trends in theoretical computer science (the challenge of the new century). World Scientific, Hackensack, NJ, pp 543–558
- Harju T, Li C, Petre I, Rozenberg G (2006a) Parallelism in gene assembly. *Nat Comp* 5(2):203–223
- Harju T, Petre I, Rogojin V, Rozenberg G (2006b) Simple operations for gene assembly. In: Kari L (ed) Proceedings of 11th international meeting on DNA-based computers, London, Ontario, 2005. Lecture notes in computer science, vol 3892. Springer, Berlin, pp 96–111
- Harju T, Petre I, Rozenberg G (2006c) Modelling simple operations for gene assembly. In: Chen J, Jonoska N, Rozenberg G (eds) Nanotechnology: science and computation. Springer, Berlin, pp 361–376
- Harju T, Li C, Petre I, Rozenberg G (2007) Complexity measures for gene assembly. In: Tuyls K, Westra R, Saeyns Y, Nowé A (eds) Proceedings of the knowledge discovery and emergent complexity in bioinformatics workshop, Ghent, Belgium, May 2006. Lecture notes in bioinformatics, vol 4366. Springer, Berlin, Heidelberg, pp 42–60
- Harju T, Li C, Petre I (2008a) Graph theoretic approach to parallel gene assembly. *Discr Appl Math* 156(18): 3416–3429
- Harju T, Li C, Petre I (2008b) Parallel complexity of signed graphs for gene assembly in ciliates. *Soft Comput Fusion Found Methodol Appl* 12(8):731–737
- Harju T, Petre I, Rogojin V, Rozenberg G (2008c) Patterns of simple gene assembly in ciliates. *Discr Appl Math* 156(14):2581–2597
- Hausmann K, Bradbury PC (eds) (1997) Ciliates: cells as organisms. Vch Pub, Deerfield Beach, FL
- Ilie L, Solis-Oba R (2006) Strategies for DNA self-assembly in ciliates. In: Mao C, Yokomori T (eds) DNA'06: Proceedings of the 12th international meeting on DNA computing, Seoul, Korea, June 2006. Lecture notes in computer science, vol 4287. Springer, Berlin, pp 71–82
- Ishdorj T-O, Petre I (2008) Gene assembly models and boolean circuits. *Int J Found Comput Sci* 19(5):1133–1145
- Ishdorj T-O, Petre I, Rogojin V (2007) Computational power of intramolecular gene assembly. *Int J Found Comp Sci* 18(5):1123–1136
- Ishdorj T-O, Loos, R, Petre I (2008) Computational efficiency of intermolecular gene assembly. *Fund Informaticae* 84(3-4):363–373
- Jahn CL, Klobutcher LA (2000) Genome remodeling in ciliated protozoa. *Ann Rev Microbiol* 56:489–520
- Kari L, Landweber LF (1999) Computational power of gene rearrangement. In: Winfree E, Gifford DK (eds) Proceedings of DNA based computers, V. Massachusetts Institute of Technology, 1999 American Mathematical Society, Providence, RI, pp 207–216
- Kari L, Kari J, Landweber LF (1999) Reversible molecular computation in ciliates. In: Karhumäki J, Maurer H, Păun G, Rozenberg G (eds), Jewels are forever. Springer, Berlin, Heidelberg, New York, pp 353–363
- Kauman LH (1999) Virtual knot theory. *Eur J Combinatorics* 20:663–690
- Krasnow MA, Stasiak A, Spengler SJ, Dean F, Koller T, Cozzarelli NR (1983) Determination of the absolute handedness of knots and catenanes of DNA. *Nature* 304(5926):559–560
- Landweber LF, Kari L (1999) The evolution of cellular computing: Nature's solution to a computational problem. In: Kari L, Rubin H, Wood D (eds) Special issue of Biosystems: proceedings of DNA based computers IV, vol 52(1–3). Elsevier, Amsterdam, pp 3–13
- Landweber LF, Kari L (2002) Universal molecular computation in ciliates. In: Landweber LF, Winfree E (eds) Evolution as computation. Springer, Berlin, Heidelberg, New York, pp 257–274
- Langille M, Petre I (2006) Simple gene assembly is deterministic. *Fund Inf* 72:1–12
- Langille M, Petre I (2007) Sequential vs. parallel complexity in simple gene assembly. *Theor Comput Sci* 395(1):24–30
- Langille M, Petre I, Rogojin V (2010) Three models for gene assembly in ciliates: a comparison. *Comput Sci J Moldova* 18(1):1–26
- Möllenbeck M, Zhou Y, Cavalcanti ARO, Jönsson F, Higgins BP, Chang W-J, Juraneck S, Doak TG, Rozenberg G, Lipps HJ, Landweber LF (2008) The

- pathway to detangle a scrambled gene. *PLoS ONE* 3(6):2330
- Nowacki M, Vijayan V, Zhou Y, Schotanus K, Doak TG, Landweber LF (2008) RNA-mediated epigenetic programming of a genome-rearrangement pathway. *Nature* 451:153–158
- Petre I (2006) Invariants of gene assembly in stichotrichous ciliates. IT, Oldenbourg Wissenschaftsverlag 48(3):161–167
- Petre I, Rogojin V (2008) Decision problems for shuffled genes. *Info Comput* 206(11):1346–1352
- Petre I, Skogman S (2006) Gene assembly simulator. <http://combio.abo.fi/simulator/simulator.php>
- Pevzner P (2000) Computational molecular biology: an algorithmic approach. MIT Press, Cambridge, MA
- Prescott DM (1994) The DNA of ciliated protozoa. *Microbiol Rev* 58(2):233–267
- Prescott DM (1999) The evolutionary scrambling and developmental unscrambling of germline genes in hypotrichous ciliates. *Nucl Acids Res* 27(5):1243–1250
- Prescott DM (2000) Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat Rev Genet* 3:191–198
- Prescott DM, DuBois M (1996) Internal eliminated segments (IESs) of oxytrichidae. *J Eukariot Microbiol* 43:432–441
- Prescott DM, Greslin AF (1992) Scrambled actin I gene in the micronucleus of *Oxytricha nova*. *Dev Gen* 13(1):66–74
- Prescott DM, Ehrenfeucht A, Rozenberg G (2001a) Molecular operations for DNA processing in hypotrichous ciliates. *Eur J Protistol* 37:241–260
- Prescott DM, Ehrenfeucht A, Rozenberg G (2001b) Template-guided recombination for IES elimination and unscrambling of genes in stichotrichous ciliates. *J Theor Biol* 222:323–330
- Setubal J, Meidanis J (1997) Introduction to computational molecular biology. PWS Publishing Company, Boston, MA
- White JH (1992) Geometry and topology of DNA and DNA-protein interactions. In: Sumners et al. (eds) New scientific applications of geometry and topology, American Mathematical Society, Providence, RI, pp 17–38
- Yan H, Zhang X, Shen Z, Seeman NC (2002) A robust DNA mechanical device controlled by hybridization topology. *Nature* 415:62–65
- Yurke B, Turberfield AJ, Mills AP, Simmel FC Jr (2000) A DNA fueled molecular machine made of DNA. *Nature* 406:605–608
- Zerbib D, Mezard C, George H, West SC (1998) Coordinated actions of RuvABC in Holliday junction processing. *J Mol Biol* 281(4):621–630

38 DNA Memory

Masanori Arita¹ · Masami Hagiya² · Masahiro Takinoue³ · Fumiaki Tanaka⁴

¹Department of Computational Biology, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan
arita@k.u-tokyo.ac.jp

²Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
hagiya@is.s.u-tokyo.ac.jp

³Department of Physics, Kyoto University, Kyoto, Japan
takinoue@chem.scphys.kyoto-u.ac.jp

⁴Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
fumi95@is.s.u-tokyo.ac.jp

1	<i>Introduction</i>	1282
2	<i>Sequence Design</i>	1282
3	<i>Memory Operations</i>	1293
4	<i>In Vitro DNA Memory</i>	1305
5	<i>DNA Memory on Surfaces</i>	1307
6	<i>In Vivo DNA Memory</i>	1311
7	<i>Conclusion</i>	1315

Abstract

This chapter summarizes the efforts that have been made so far to build a huge memory using DNA molecules. These efforts are targeted at increasing the size of the address space of a molecular memory and making operations on a specified word in the address space more efficient and reliable. The former issue should be solved by careful design of the base sequences of the address portions. The latter issue depends on the architecture of a molecular memory and the available memory operations. Concrete examples of molecular memories described in this chapter are classified into *in vitro* DNA memory, DNA memory on surfaces, and *in vivo* DNA memory. This chapter also describes the technology for designing base sequences of DNA molecules.

1 Introduction

Soon after Adleman published his seminal work on DNA computing (Adleman 1994) and Lipton (1995) gave a more general framework for solving nondeterministic polynomial time (NP)-complete problems using DNA, Baum wrote a technical comment in *Science* on building a huge memory using DNA molecules (Baum 1995). He claimed that it was possible to build an associative memory vastly larger than the brain. More concretely, he wrote that it was not completely implausible to imagine vessels storing 10^{20} words, which is comparable to standard estimates of brain capacity as 10^{14} synapses each storing a few bits. Among various proposals for constructing a molecular memory, he considered a scheme that stores DNA molecules consisting of an address portion and a data portion. In order for the scheme to work as an ordinary random access memory, each address portion should be accessible only by the sequence that is complementary to the sequence of the address portion. He even mentioned the use of an error correcting code to avoid accidental bonding due to approximate match.

This chapter summarizes the efforts that have been made so far toward achieving Baum's dream. Those efforts are targeted at increasing the size of the address space of a molecular memory and making operations on a specified word in the address space more efficient and reliable. As Baum pointed out, the former issue should be solved by careful design of base sequences of address portions. The latter issue depends on the architecture of a molecular memory and available memory operations. Sequence design is also affected by the required memory operations.

The next section describes the technology for designing base sequences of DNA molecules. It is one of the most fruitful technological achievements in the research field of DNA computing, and can be used in various application fields including biotechnology and nanotechnology. The following section then summarizes a series of memory operations that have been devised for constructing various kinds of molecular memories. Concrete examples of molecular memories are then described in the sections that follow, classified into *in vitro* DNA memory, DNA memory on surfaces, and *in vivo* DNA memory.

2 Sequence Design

This section describes sequence design in DNA computing. Sequence design is the search for sequences satisfying given constraints or having a high value for a given evaluation function.

First, this section describes the thermodynamics of DNA molecules and the constraints to be satisfied in DNA computing. Then, various strategies for sequence design are described.

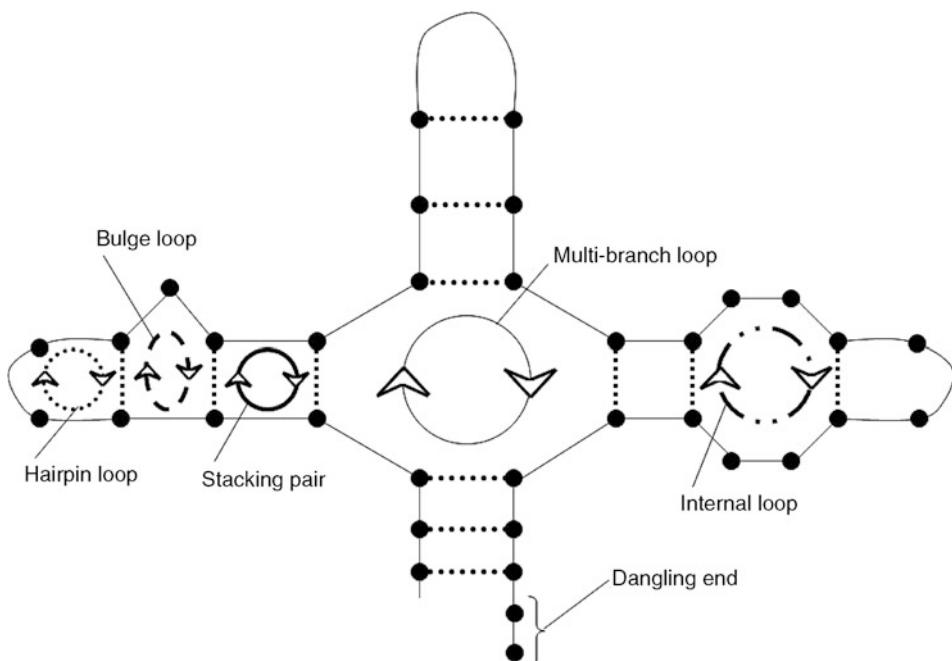
2.1 Thermodynamics of DNA Molecules

Deoxyribonucleic acid (DNA) is a polymer consisting of three units: bases, deoxyribose, and phosphoric acids. In DNA, the bases comprising the nucleotides are adenine (A), thymine (T), cytosine (C), and guanine (G); in RNA, thymine is replaced by uracil (U). The pairs between A and T and between C and G are called Watson–Crick base pairs (Watson and Crick 1953). The former has two hydrogen bonds and the latter has three, making it more stable. Other pairs (called “mismatches”) such as G-T are generally less stable than the Watson–Crick base pairs. On the basis of these selective hydrogen bonds, a DNA sequence and its complementary strand form a double-stranded structure in an antiparallel direction at low temperature (Watson and Crick 1953), which is called hybridization. For example, sequence 5'-AGGCTTACCC-3' hybridizes with its complement 3'-TCCGATGG-5' and forms a double-stranded structure. A double strand melts into two single strands at high temperature, which is called denaturation.

A secondary structure of DNA is defined as a set of base pairs. DNA sequences can form various secondary structures consisting of stacking pairs, various loops, and dangling ends (see [Fig. 1](#)).

■ Fig. 1

Example of DNA secondary structure. The dots represent the bases, while the thin and dotted lines represent the backbones and hydrogen bonds, respectively.



The stability of double-stranded DNA depends on the temperature, the DNA concentration, and the buffer solution. The free energy change (ΔG) is the commonly used indicator of stability for DNA complexes. ΔG is the free energy change when the sequence folds into the structure from a random coil. In particular, ΔG_T denotes the free energy change at temperature T . The lower the free energy, the more stable the structure, so more sequences form into a structure in solution. The free energy of an entire structure is the sum of the free energies of all the local substructures (Zuker and Stiegler 1981). The minimum free energy (ΔG_{\min}) is the minimum value of free energy for any structure. Thus, the structure with the minimum free energy must be the most stable in solution. Stability can be well approximated using the minimum free energy calculation. In fact, the minimum free energy is often used for secondary structure prediction of single-stranded RNA (Zuker and Stiegler 1981). The minimum free energy can be calculated using dynamic programming.

Another indicator for the stability of a double-stranded DNA is the melting temperature, T_m , which is defined as the temperature at which single and double strands are in equilibrium with a 1:1 ratio. A pair of sequences with a high melting temperature is stable, while one with a low melting temperature is unstable. The melting temperature is calculated using (Borer et al. 1974)

$$T_m = \frac{\Delta H}{R \ln(C_t/\alpha) + \Delta S}, \quad (1)$$

where R is the gas constant, C_t is the total oligo concentration. ΔH and ΔS are enthalpy and entropy change from a random coil, respectively. Parameter α is set to one for self-complementary strands and to four for non-self-complementary strands.

Parameters ΔH , ΔS and ΔG are called “thermodynamic parameters” and are calculated on the basis of the nearest-neighbor model (SantaLucia 1998). For example, the ΔG of double-stranded DNA 5'-AGTG-3'/3'-TCAC-5' is calculated using

$$\Delta G \text{ (5'-AGTG-3'/3'-TCAC-5')} = \text{initiation} + \text{symmetry} + \Delta G \text{ (5'-AG-3'/3'-TC-5')} + \Delta G \text{ (5'-GT-3'/3'-CA-5')} + \Delta G \text{ (5'-TG-3'/3'-AC-5')}$$

where “*initiation*” is a contribution by terminal base pairs and the value differs depending on whether the terminal base pair is an A-T pair or G-C pair. The term “*symmetry*” describes whether the sequence is a palindrome or not. The thermodynamic parameters of a substructure, such as 5'-GT-3'/3'-CA-5', are called “nearest-neighbor parameters” or “nearest-neighbor thermodynamics.” Nearest-neighbor parameters are determined by the sequence forming the substructure and must be estimated in advance through thermodynamic analysis. ◉ [Table 1](#) shows an example of reported nearest-neighbor parameters for Watson–Crick base pairs (Allawi and SantaLucia 1997; Tanaka et al. 2004). If the SantaLucia parameters (in the rightmost column of the table) are used, the ΔG of 5'-AGTG-3'/3'-TCAC-5' is calculated as $(0.98 + 1.03) + 0 + (-1.28) + (-1.44) + (-1.45) = -2.16 \text{ kcal/mol}$.

All possible nearest-neighbor parameters have already been estimated for Watson–Crick base pairs (SantaLucia 1998; Sugimoto et al. 1996; SantaLucia et al. 1996), internal mismatches (Aboul-ela et al. 1985), G-A mismatches (Li et al. 1991; Li and Agrawal 1995), G-T mismatches (Allawi and SantaLucia 1997), A-C mismatches (Allawi and SantaLucia 1998a), C-T mismatches (Allawi and SantaLucia 1998b), G-A mismatches (Allawi and SantaLucia 1998c), N-N mismatches (N represents A, T, C, or G) (Peyret et al. 1999), dangling ends (Senior et al. 1988; Bommarito et al. 2000), and bulge loops (Tanaka et al. 2004). The thermodynamic parameters for various loops are calculated approximately as a function of loop length; shorter loops have lower free energy while longer loops have higher free energy.

Table 1

Nearest-neighbor thermodynamic parameters for Watson–Crick base pairs

Sequence	ΔH (Tanaka et al. 2004) (kcal/mol)	ΔS (Tanaka et al. 2004) (eu)	ΔG (Tanaka et al. 2004) (kcal/mol)	ΔG (Allawi and SantaLucia 1997) (kcal/mol)
AA/TT	-8.3 ± 1.1	-23.3 ± 3.1	-1.04 ± 0.07	-1.00 ± 0.01
AT/TA	-8.5 ± 0.8	-24.6 ± 2.4	-0.91 ± 0.04	-0.88 ± 0.04
AC/TG	-9.5 ± 1.1	-26.0 ± 2.7	-1.49 ± 0.09	-1.44 ± 0.04
AG/TC	-8.2 ± 0.9	-22.3 ± 2.7	-1.33 ± 0.06	-1.28 ± 0.03
TA/AT	-6.5 ± 0.6	-19.0 ± 1.7	-0.60 ± 0.04	-0.58 ± 0.06
TC/AG	-8.7 ± 0.8	-23.6 ± 2.4	-1.35 ± 0.03	-1.30 ± 0.03
TG/AC	-8.9 ± 0.4	-24.3 ± 1.3	-1.51 ± 0.05	-1.45 ± 0.06
CC/GG	-8.8 ± 0.6	-21.7 ± 1.7	-1.90 ± 0.04	-1.84 ± 0.04
CG/GC	-11.4 ± 0.9	-29.7 ± 2.5	-2.25 ± 0.06	-2.17 ± 0.05
GC/C _G	-10.1 ± 1.0	-25.4 ± 2.6	-2.28 ± 0.06	-2.24 ± 0.03
Init_AT	4.4 ± 1.4	10.7 ± 3.9	1.19 ± 0.09	1.03 ± 0.05
Init_GC	1.7 ± 0.5	2.4 ± 1.3	1.12 ± 0.03	0.98 ± 0.05
Symmetry	$0 \pm -$	$-1.4 \pm -$	$0.4 \pm -$	$0.4 \pm -$

2.2 Design Constraint

2.2.1 Uniform Melting Temperature

DNA sequences used in DNA computing should hybridize with their complements at the same efficiency to prevent hybridization bias. Thus, making the melting temperature uniform among pairs of complementary sequences is important for obtaining uniform hybridization efficiency. The melting temperature is calculated using formula (1).

Another indicator for the melting temperature is the GC content, which is the ratio of the sum of the number of G and C nucleotides over the total number of nucleotides in the sequence. Double strands with a higher GC content tend to have a higher melting temperature because the hydrogen bond between G and C is more stable than that between A and T. Thus, the GC content can be used to approximately calculate the melting temperature. In DNA computing, sequences are often designed such that their GC content is 50%.

2.2.2 Specific Hybridization

Specific hybridization, which is also called intended/expected/planned hybridization/interaction, denotes the hybridization between an intended pair of sequences while nonspecific hybridization, or unintended/unexpected/unplanned hybridization/interaction, denotes the hybridization between an unintended pair of sequences. Nonspecific hybridization causes various problems in DNA computing, including errors in the computation, failure in the construction of intended DNA nanostructures, and misdiagnosis in DNA microarrays.

Therefore, in DNA computing, multiple sequences need to be designed, such that they do not hybridize nonspecifically with each other, while in RNA secondary structure design, a single sequence needs to be designed that folds into the desired secondary structure (Hofacker 2003; Andronescu et al. 2003, 2004; Dirks et al. 2004). Specific sequences are sequences between which only specific hybridizations can form. Since the number of combinations increases exponentially with the number of sequences, designing specific sequences is an NP-hard problem and is equivalent to the independent set problem (Frutos et al. 1997).

To design specific sequences, we must design sequences among which specific hybridizations are stable and nonspecific ones are unstable. Thus, specific-sequence design requires a metric for stability between two sequences. The most often used criterion for stability is the *h*-measure proposed by Garzon et al. (1997, 1998). The *h*-measure for sequences x_i , x_j is defined as

$$|x_i, x_j| := \min_{-n < k < n} H(x_i, \sigma^k(\bar{x}_j)), \quad (2)$$

where $H(*, *)$ denotes the Hamming distance, σ^k denotes the right (left) shift for $k > 0$ ($k < 0$), k denotes the number of the shift, and \bar{x}_j is the sequence which is Watson–Crick complementary to x_j . The *h*-measure is based on the fact that the stability depends on the number of base pairs: a double strand with a certain number of base pairs is more stable than one with fewer base pairs. Therefore, if the *h*-measure between two sequences is large, the double strand that they form tends to be unstable.

2.2.3 Preventing Secondary Structures

Secondary structures often prevent the sequence from hybridizing to the correct complementary sequence. Thus, we must avoid sequences that form stable secondary structures unless they are specifically designed to form them. The mfold program (Zuker 2003) is commonly used to calculate the stability of secondary structures.

2.2.4 Other Constraints

1. Repeated Bases

Repeated bases have a harmful effect on specific hybridization. For example, guanine-rich motifs can form four-stranded complexes (Sen and Gilbert 1988, 1990). Thus, sequences are often designed not to have repeated bases.

2. Forbidden Subsequences

When a restriction enzyme, which cuts off the sequence at specific sites, is used, the recognition sequence must appear only at an intended site. Otherwise, sequences will be cut off at some unintended site. Therefore, recognition sequences are forbidden except for intended sites.

3. Three-Base Constraint

To prevent nonspecific hybridization, several strategies have been devised that use only three kinds of bases (A, T, and C) for sequence design (Cukras et al. 1999; Faulhammer et al. 2000; Braich et al. 2000, 2002). The reason for omitting base G is that it can form G-C and G-T base-pairings. In addition, Whiplash polymerase chain reaction (PCR) needs to use only three bases because the fourth base is used as stopper sequences to stop the

polymerase extension at the designated position (Hagiya et al. 1999). This constraint is effective to some extent for the specific hybridization. However, omitting G greatly reduces the variety of sequences.

2.3 Strategy for Sequence Design

This subsection summarizes strategies for producing sequences that satisfy the design criteria discussed in the previous section. Many strategies have been proposed for searching appropriate sequences.

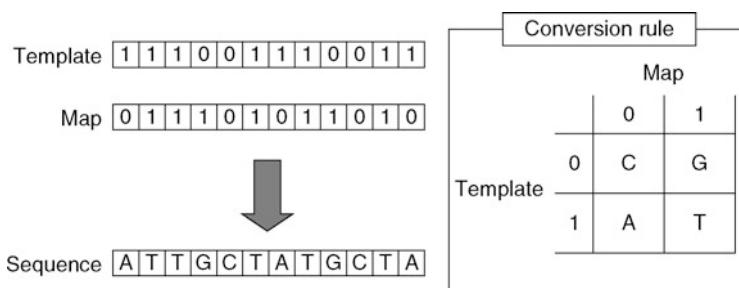
In the coding theory research area, researchers have focused on a classical problem: maximize the size of a set consisting of equi-length code words so that the Hamming distances between them are above a certain threshold. This problem resembles the sequence design problem: design sequences so that the Hamming distances between them are above a certain threshold. Thus, the theoretical analyses and achievements in coding theory can be used for DNA sequence design.

On the basis of coding theory, sequences are produced by crossing two binary codes of length n , namely a template and map. First, the positions for [AT] and [GC] are determined using a template $t = t_1t_2\dots t_n$ ($t_i \in \{0, 1\}$), where 0 indicates [GC] and 1 indicates [AT]. Since the number of occurrences of 0 in the template is equal to the number of G and C in a DNA sequence, the GC content of DNA sequences can be adjusted by controlling the occurrences of 0 in a template. Either A or T is chosen for template positions $t_i = 1$ and either G or C for positions $t_i = 0$ using a map $m = m_1m_2\dots m_n$ ($m_i \in \{0, 1\}$). The conversion rule is to convert 0 in a template and 0 in a map into C, 0 and 1 into G, 1 and 0 into A, and 1 and 1 into T. For example, the sequence “ATTGCTATGCTA” is produced by conversion of codes T=“111001110011” and M=“01101011010” (see Fig. 2).

The problem is to find as many templates and maps as possible such that sequences obtained from them satisfy certain criteria such as uniform GC content or maximal Hamming distance. Frutos et al. proposed a strategy for designing sequences that satisfy a Hamming distance constraint (Frutos et al. 1997). Although the design procedure does not consider frameshift hybridization, their study is a pioneering work in DNA sequence design using coding theory. Li et al. developed a similar design strategy based on the Hamming distance without the frameshift (Li et al. 2002).

Fig. 2

Conversion from two binary codes into a DNA sequence. The bases of the sequence are determined on the basis of the conversion rule.



Marathe et al. clarified the lower and upper bounds on the maximum size of DNA sequences that satisfy the constraints on the basis of the Hamming distance without the frameshift (Marathe et al. 2001). Their study was based on several achievements in coding theory. As Brenneman and Condon pointed out (Brenneman and Condon 2002), however, the upper bound is probably still unknown when the Hamming distance is considered with the frameshift.

Liu et al. proposed a design strategy that considers frameshift hybridization in accordance with the h -measure (Liu et al. 2003). To prevent low h -measure values, they kept equal 01 content in the map. This strategy is based on the observation that deviation in the occurrence of 0 or 1 in the map tends to reduce the h -measure value. Although the heuristics may be partially effective, they are not sufficient because this strategy does not completely guarantee the exclusion of maps, which can cause a low h -measure value.

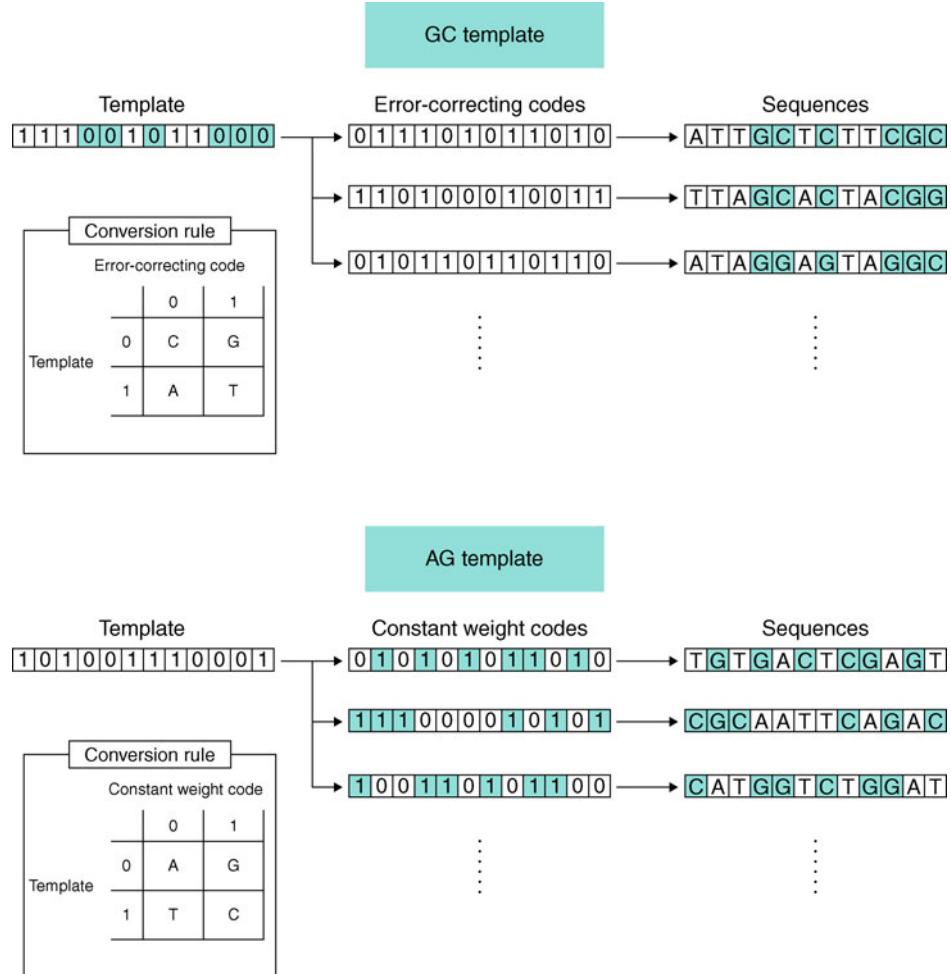
A more sophisticated method, called the “template method,” was proposed by Arita and Kobayashi (2002). In this method, an error-correcting code is used as a map (see  Fig. 3). The error-correcting code is the code for which codewords have at least k mismatches with each other. It has been already investigated in traditional coding theory. For example, a BCH code of length 15 provides at least 128 words with a minimum of five mismatches. Thus, the template method needs to design a template that satisfies a specified constraint. The constraint imposed by Arita and Kobayashi was that the number of mismatches must be larger than a threshold. The template method has two strengths: (i) frameshift hybridization is considered and (ii) mismatches between sequences and concatenations of two sequences are considered. Templates satisfying the specified constraint are found by an exhaustive search, although the number of candidates is dramatically reduced by theoretical consideration. As a result, sets of templates of length up to 30 can be designed.

This template method was later extended by using an AG-template, which determines the position of [AG] rather than [GC] (Kobayashi et al. 2003). The mismatches obtained by using AG-templates outnumber those by using GC-templates. However, using an AG-template does not guarantee uniform GC content. This problem was solved by using a constant weight code in which the codewords have the same number of occurrences of 1 ( Fig. 3). Thus, maps (i.e., constant weight codes) guarantee the uniform GC content instead of templates.

Another important strategy is a *de novo* design proposed by Seeman (1990). The *de novo* design generates sequences for which each subsequence of length l appears at most once. Let one consider a sequence CATGGGAGATGCTTAG as an example of $l = 6$. Any subsequence of length 6 such as CATGGG appears only once in this sequence. The first design step is to choose a sequence of length l that is not included in a given list of prohibited subsequences and to add this sequence to the prohibited list. This sequence is prolonged to the right by concatenating a letter in such a way that the resulting sequence of length $l+1$ does not have any sequence in the prohibited list as its suffix is of length l . Sequences generated by repeating this process are guaranteed not to have two identical subsequences of length l in it. This constraint is motivated by the fact that consecutive complementary bases are likely to form stable structures. However, there is no guarantee that the *de novo* design prevents nonspecific hybridization. For example, although sequences AAAAACAAAAAA and AAAAAGAAAAAA do not share any 6-mer subsequence, the Hamming distance between them is one, resulting in cross-hybridization between one sequence and the complement of the other. Therefore, the sequences produced by the *de novo* design must be checked using another program such as mfold (Zuker 2003) before their practical use.

Fig. 3

Template method with GC template (upper part) and AG template (lower part).

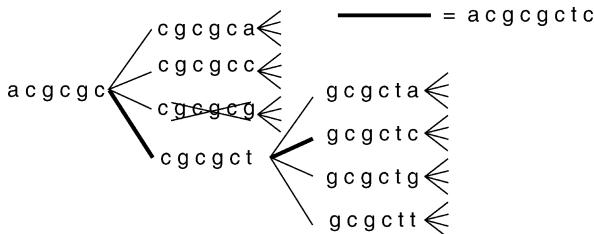


Feldkamp et al. developed a software program with a GUI called *DNASequenceGenerator* for the construction of sets of DNA sequences (Feldkamp et al. 2001). The program can design sequences that satisfy uniqueness (i.e., *de novo* design), uniform melting temperature, and GC content, and that prevent palindromes, start codons and restriction sites. It creates *de novo* sequences using a graph representing DNA sequences (see [Fig. 4](#)). Each path in the graph represents a DNA sequence. The program cuts off from the graph forbidden sequences defined by users and sequences already used. Consequently, *de novo* sequences can be designed.

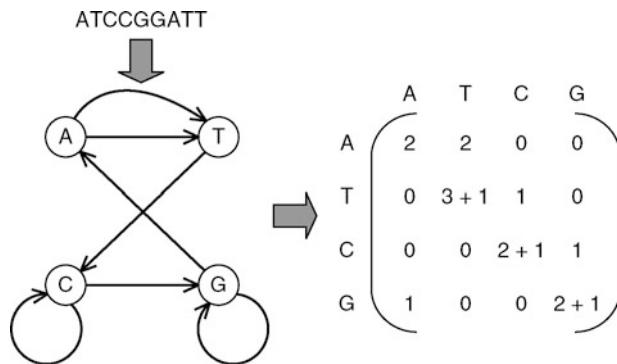
Pancoska et al. developed a design algorithm for sequences with the same melting temperature that uses an Eulerian graph (Pancoska et al. 2004). The algorithm first constructs an Eulerian graph and an adjacency matrix from a given sequence (see [Fig. 5](#)). The key idea is that every Euler path has the same occurrence of each nearest-neighbor parameter. Thus, if we calculate the melting temperature using formula (1), all sequences represented by Euler

Fig. 4

Graph representation of DNA sequences. Each path represents a DNA sequence.

**Fig. 5**

Transformation from a DNA sequence into the Eulerian graph and the adjacency matrix.



paths have the same melting temperature. Consequently, sequences with the same melting temperature can be designed by enumerating all Euler paths and converting them into the corresponding sequences.

With an appropriate evaluation function or set of constraints, the sequence design problem can be formulated as a search for sequences that maximize (minimize) the evaluation function or satisfy the constraints.

The simplest strategy is to generate sequences randomly, and then modify them until they satisfy the constraints. Faulhammer et al. developed a program called PERMUTE that designs sequences on the basis of this strategy (Faulhammer et al. 2000).

A similar strategy is to generate a sequence randomly and then add it to a sequence set if it satisfies all the constraints. This process is repeated until the sequence set contains a predetermined number of sequences. Suyama's group developed a procedure using this strategy for designing orthogonal DNA sequences (Yoshida and Suyama 2000). The sequences are designed to satisfy three constraints: (i) uniform melting temperature, (ii) no mishybridization, and (iii) no formation of stable intra-molecular structures. A disadvantage of this strategy is that it easily falls into a local optimum, resulting in the saturation of the number of sequences successfully designed. With the aim of overcoming this drawback, Kashiwamura et al. developed a modified algorithm called "two-step search" (Kashiwamura et al. 2004). They observed that critical sequences, which have higher scores of Hamming distance with sequences in a sequence set, prevent the design of more sequences. In line with this

analysis, the two-step search generates sequences with Hamming distances as short as possible. This is analogous to packing particles into a limited space, for which an effective strategy is to pack them as tightly as possible.

Ruben et al. developed a program called “PUNCH” that designs sequences using a local search algorithm (Ruben et al. 2001). The strategy is to find sequences based on uniqueness, mishybridization, base equality, and folding scores. The program searches neighbors obtained by mutating previous best sequences randomly. Therefore, this search procedure cannot escape from local optima.

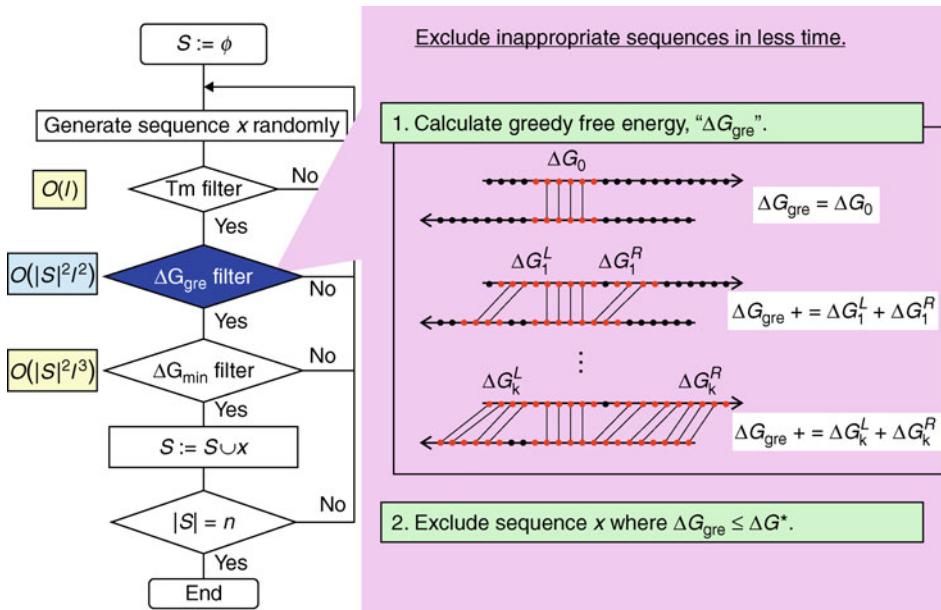
A more sophisticated algorithm is the stochastic local search (SLS) algorithm. The algorithm designs sequences by improving a given sequence set iteratively. First, it chooses a pair of sequences s_1 and s_2 that violate one of the given constraints. Then, it generates a set S_1 of sequences obtained by substituting some bases of s_1 and a set S_2 by substituting some bases of s_2 . With a probability of θ , it selects a sequence s' from $S_1 \cup S_2$ at random. Otherwise, it selects a sequence s' from $S_1 \cup S_2$ such that the number of conflict violations is maximally decreased. If $s' \in S_1$, it replaces s_1 by s' . Otherwise, it replaces s_2 by s' . This procedure is iterated for a predefined number of steps. Using the SLS algorithm, Tulpan et al. found sequence sets that matched or exceeded the best previously known constructions though they did not take into account a frameshift for the Hamming distance (Tulpan et al. 2003; Tulpan and Hoos 2003). They determined that it was effective to use “hybrid randomized neighborhoods” obtained by adding random sequences to neighborhoods by substituting one base (Tulpan and Hoos 2003).

Sequences can also be designed using metaheuristics such as the genetic algorithm (GA). Deaton et al. used GA to design sequences that satisfied various constraints such as the Hamming distances between them being a certain length (Deaton et al. 1996, 1998). Although they considered only Hamming distances without a frameshift, they showed that the Hamming distances for some sequences were identical to the upper bound, called the “Hamming bound.” Arita et al. proposed a design strategy in which an evaluation function calculated as the linear sum of multiple evaluation terms is optimized (Arita et al. 2000). The evaluation function considers restriction sites, GC content, Hamming distance, repetition of the same base, and complete hybridization at the 3'-end. They also utilized a GA for the optimization.

The sequence design methods described above approximate the stability of double strand on the basis of the number of base pairs. Some recently proposed design methods estimate the stability on the basis of the minimum free energy (ΔG_{\min}). This thermodynamic approach approximates the stability with greater accuracy because it takes into account the stability of some loops and the stacking between base pairs. However, it takes longer to calculate the minimum free energy. The time complexity of the minimum free energy calculation by dynamic programming is $O(l^3)$, where l is the sequence length (Lyngso et al. 1999). Therefore, efforts have been made to reduce the computational time. Tanaka et al. proposed approximating the calculation of the minimum free energy in a greedy manner to reduce the time (Tanaka et al. 2005). They defined the problem of designing a pool S containing n sequences of length l for which the minimum free energy is greater than the threshold (ΔG_{\min}^*) in any pairwise duplex of sequences in S and any concatenation of two sequences in S , plus their complements except for the complementary pairs. ➤ *Figure 6* shows a conceptual diagram of this greedy calculation. The approximated value for the minimum free energy, denoted by ΔG_{gre} , is calculated by searching for the most stable continuous base pairs and fixing them iteratively. The time complexity of ΔG_{gre} calculation is $O(l^2)$. Obviously, ΔG_{gre} is an upper bound of ΔG_{\min} (i.e., $\Delta G_{\min} \leq \Delta G_{\text{gre}}$). Thus, ΔG_{gre} can be used to check the sequence before calculating ΔG_{\min} .

Fig. 6

Schematic diagram for sequence design using greedy calculation of ΔG_{\min} . In this diagram, S and I denote a sequence set already designed and the sequence length, respectively. ΔG_{gre} denotes the approximate value of ΔG_{\min} calculated in a greedy manner.



Kawashimo et al. also developed a design procedure based on the thermodynamic approach (Kawashimo et al. 2008, 2009). They reduced the computation time by introducing a sophisticated search order and by limiting the loop size.

2.4 Future Direction of Sequence Design

We have discussed sequence design to date, but research on sequence design continues. What are the future directions?

One will be the development of sequence design that takes the kinetics of DNAs into consideration. Kitajima et al. proposed using a design criterion for rapid hybridization between two complementary sequences (Kitajima et al. 2008). They demonstrated that the rapidity of hybridization depended on the stability of the self-folded secondary structures and on the nucleation capability at the tails of their self-folded secondary structures. Although this finding cannot be directly used for sequence design, it can be used to exclude the sequences for which the hybridization rate is low.

Another direction will be sequence design that considers the dynamics of DNA machinery, in which DNA ruled by the machinery has multiple stable states. In this case, sequences must be designed that can transit from one state to another. To achieve this, we should estimate the energy landscape produced by the DNA sequences. However, it is difficult to predict the energy

landscape because the number of possible structures grows exponentially with the number of sequences. There have been several efforts to estimate the energy landscape efficiently (Uejima and Hagiya 2004; Kubota and Hagiya 2005).

Uejima and Hagiya developed a sequence design method for their multi-state molecular machine that is based on the minimum free energies, the structure transition paths, and the total frequency of the optimal and suboptimal structures (Uejima and Hagiya 2004). The transition paths were estimated using Morgan and Higgs' heuristics. This estimation was expanded by Kubota and Hagiya (2005). They proposed a technique for analyzing the energy landscapes given the minimax path, which is a path on the minimum spanning tree.

In short, sequence designs that consider the above constraints will be developed for more complex DNA machinery.

3 Memory Operations

As mentioned in the introduction of this chapter, one of the most crucial issues for constructing a molecular memory is how to access a specified word in its address space. This section summarizes various techniques that have been developed to implement memory operations on a molecular memory.

To compare various implementations, an associative molecular memory consisting of words that are divided into two portions is considered: the address part and the data part. The address part can be further divided into multiple address digits, comprising a hierarchical address space. For each memory operation, it is first assumed that the address part consists of a single address digit, and then it touches on the possibility of extending the memory operation to a hierarchical memory.

The data part of a memory word can be designed in various ways. It can represent a single bit, that is, 0 or 1. In some molecular memories, the base sequence of a data part is not relevant. For example, a long data part represents bit 1, and a short data part represents bit 0. Data parts can also be empty. In such a molecular memory, existence of a word having a specified address means bit 1 of the address and nonexistence of a word means bit 0. Each address (digit) can have its complementary address (digit). This is typical when variable assignments are represented by a DNA molecule. Each variable and its negation correspond to complementary address digits and they comprise a hierarchical molecular memory.

The data part can represent richer information. For example, if an artificial tag sequence is attached to a gene extracted from a chromosome or cDNA in a library of genes, then the tag sequence is regarded as the address part and the gene sequence as the data part.

This section is organized as follows. In the first subsection, various implementations of the access operation as mentioned above are summarized. In the next sections, the read and write operations are briefly explained. The final section touches on some other operations on a molecular memory.

3.1 Access Operation

This section explains how to access a memory word having a specified address in a molecular memory. If there is no data part in a memory word, success in accessing the word amounts to reading bit 1 of the address, and failure amounts to reading bit 0.

3.1.1 Extraction by Magnetic Beads

As in the seminal work of Adleman, extraction using magnetic beads is a typical access operation on a molecular memory. More generally, any kind of affinity separation based on the address part of a word can be used as an access operation. Affinity separation by magnetic beads is performed as follows.

The address part of a memory word should be represented by a single-stranded segment of a DNA molecule. Typically, a whole word takes the form of a single strand of DNA as in [Fig. 7](#). As Baum pointed out (Baum 1995), in order to avoid unexpected interaction, the data part of a memory word can be double-stranded. Such words comprising a molecular memory are mixed and diluted in an aqueous solution.

DNA molecules containing the sequence complementary to the address part, called probes, are also single-stranded and attached on the surface of magnetic beads as in [Fig. 8](#). The protein called biotin is covalently bonded to the probes. Such modified DNA can be ordered to a DNA synthesizing company as ordinary plain DNA with some additional cost. The surface of the magnetic beads is coated with another kind of protein called streptavidin. Strong interaction between streptavidin and biotin attaches the probes to the surface of the magnetic beads.

The magnetic beads are then put into the aqueous solution of the molecular memory in a test tube. The resulting test tube looks like muddy water, in which the molecular words and the probes attached on the surface of magnetic beads interact, and those words having the address part complementary to the probes are trapped by DNA hybridization. Those with different address parts are still in the solution.

A strong magnet is then placed at the side of the test tube. Not surprisingly, all the magnet beads are pulled together near the magnet ([Fig. 9](#)), and the solution quickly becomes transparent, in which memory words not trapped in the beads are still diluted. The solution is then extracted by a pipet while the beads are fixed to the magnet. Pure water is then poured into the test tube and the magnet is removed. The water is then turned into mud. This process is repeated to separate all memory words that are not trapped in the beads.

Fig. 7

A simple memory word. A single strand of DNA consists of the address part and the data part.



Fig. 8

A strand trapped by a magnetic bead. Probes containing the sequence complementary to the target are attached on the surface of a magnetic bead. A single strand containing the target sequence is trapped by the bead.

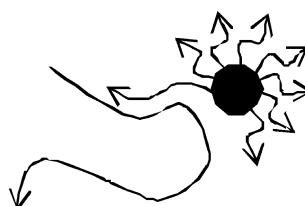
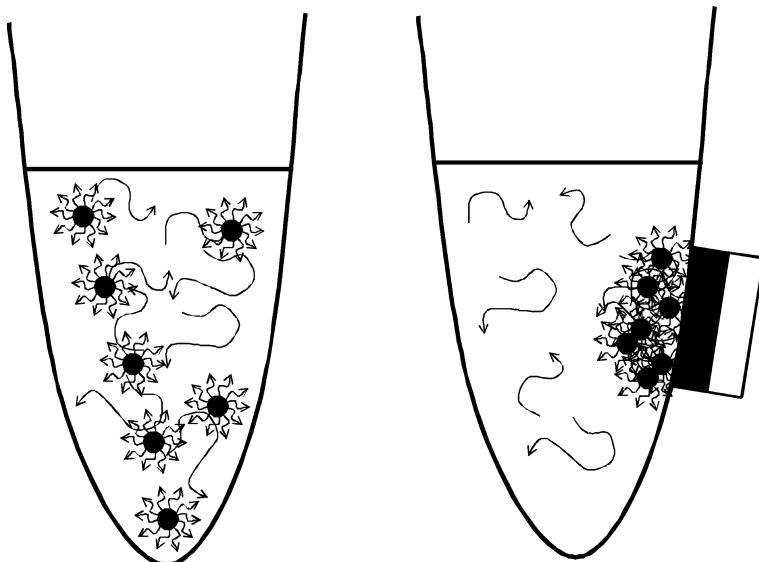
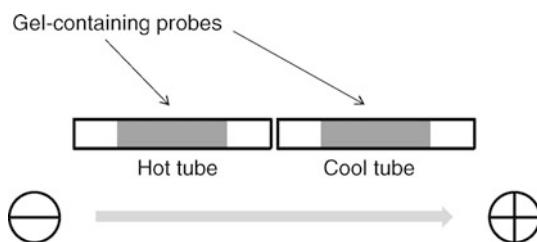


Fig. 9

Magnetic beads. In the left panel, magnetic beads are diluted in the tube to trap the target DNA. In the right panel, magnetic beads with the trapped DNA are collected by a magnet.

**Fig. 10**

A hot tube connected to a cool tube. DNA molecules released from the hot tube are moved and trapped in the cool tube.



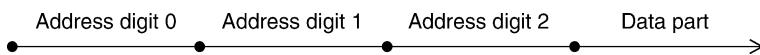
Finally, hot water is poured into the test tube. At this time, since hydrogen bonds in DNA hybridization are broken at high temperatures, the memory words trapped in the beads become free and diluted in the hot water. In this way, memory words having the specified address are separated from those having different addresses.

In the experiment conducted by Adleman et al., in which they solved a 20-variable SAT problem, probes are fixed in a gel filled inside a thin tube (Braich et al. 2002). Probes are chemically modified so that they are fixed in a gel and cannot move.

As in Fig. 10, two such tubes are connected and placed in a specially designed electrophoresis box, in which one tube is heated while the other is cooled down. In the hot tube, trapped DNA molecules are released and moved toward the cool tube according to the voltage

Fig. 11

A word in a molecular memory with a hierarchical address space. Each word consists of multiple address digits (three digits in this figure) and a data part.



of electrophoresis, and trapped in the cool tube if they are complementary to the probes fixed in the cool tube.

A hierarchical molecular memory consists of memory words whose address is comprised of multiple address digits as in [Fig. 11](#). If a single digit has r different values and each address consists of n digits, the whole address space contains r^n different addresses.

It is easy to implement a hierarchical molecular memory whose access operation is implemented by affinity separation as explained so far. The access operation is iterated for each digit, that is, affinity separation by the i th digit is applied to the result for $(i-1)$ th digit. A serious problem of this iterative approach is that at each round of affinity separation, some memory words are lost due to the limitation of the employed experimental technique. The yield of extraction thus exponentially decreases as n increases. This problem is solved by the approach using PCR, since copies of the extracted memory word are made.

3.1.2 Extraction by PCR

As is well known, PCR (polymerase chain reaction) is a method to amplify specified DNA molecules, that is, make their copies. Copied DNA molecules should have the specified sequences at their ends, with which short DNA molecules, called primers, hybridize.

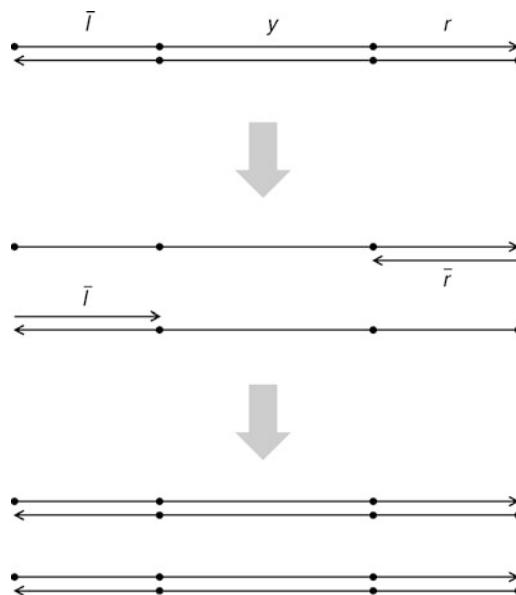
To be more precise, copied DNA molecules are double-stranded, formed by a sequence x and its complement \bar{x} . The sequence x should end with a short sequence r , while the sequence \bar{x} should also end with a short sequence l . Consequently, x should be of the form $\bar{l}yr$ ([Fig. 12](#)). Primers are single-stranded DNA molecules whose sequence is \bar{r} and \bar{l} .

PCR is conducted as follows ([Fig. 12](#)). Double strands consisting of x and \bar{x} are mixed with the pair of primers \bar{r} and \bar{l} in an appropriate buffer, containing polymerase, nucleotides, and adenosine triphosphate (ATP), which is called the polymerization buffer. The concentration of primers is much larger than that of double strands. The solution is then heated up to over 90°C, denaturing double strands into single strands x and \bar{x} . When the solution is cooled down, single strands hybridize with primers rather than their counterparts because they have more chances to contact the primers. During an appropriate duration for incubation, the 3'-end of \bar{r} in the hybrid of x and \bar{r} is extended by polymerase and the hybrid turns into a complete double strand consisting of x and \bar{x} . This process is called polymerization. The hybrid of \bar{x} and \bar{l} similarly turns into the same double strand. In this way, in a cycle of heating up, cooling down, and incubation, the concentration of x and \bar{x} is doubled, and repeating the cycle amplifies the concentration of the double strands.

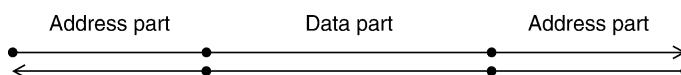
In order to use PCR for accessing memory words in a DNA memory, each word should have its data part surrounded by two primer sequences at its ends as in [Fig. 13](#). These primer sequences comprise the address part, since a memory word can be amplified only if it has both

Fig. 12

PCR (polymerase chain reaction). A double strand is first denatured to single strands at high temperature. If cooled down, they hybridize with short DNA molecules, called primers. Polymerase then extends the primers and two copies of the original double strand are made.

**Fig. 13**

A memory word extractable by PCR. Each word is a double strand having primer sequences as its address.



primer sequences. Therefore, in order to extract a memory word, the solution of the DNA memory is diluted and mixed with the primers in the polymerization buffer. Since PCR only amplifies the target memory word, if the resulting buffer is diluted, the concentration of memory words other than the target becomes less than detectable. This means that the target word is extracted by PCR.

A hierarchical address space can be obtained by introducing multiple primer pairs into a memory word. These primer pairs are nested as in [Fig. 14](#). In order to extract a memory word, PCR is conducted with the outermost primer pair first. The obtained memory words are then processed by the second iteration of PCR with the next outer primer pair. Note that after PCR is conducted with an inner primer pair, the outer primer sequences are lost.

In [Sect. 4.2](#), an actual PCR-based molecular memory is described in detail.

3.1.3 Hybridization and Conformational Change

Instead of extracting a memory word with a specified address, separating it from other words, it is also possible to access a memory word by changing its state, while words with different addresses are unchanged. In this kind of molecular memory, since memory words are not separated or mixed again, they can be fixed on a surface or in a space. In particular, it is possible to make a two-dimensional molecular memory by fixing DNA molecules on a surface of gold or other appropriate material.

If memory words are represented as single strands, simply adding a single-stranded probe whose sequence is complementary to the address of the target changes it into a double strand (☞ Fig. 15). In a simple setting, where a memory word does not have a data part, this process is considered as a write operation. The single-stranded form represents bit 0 and the double-stranded form represents bit 1.

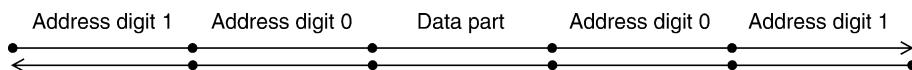
As explained in detail in ☞ Sect. 5, Suyama's group employs hairpin molecules as memory words. The use of hairpins can introduce an additional condition for the access operation. Since hairpin formation is an intra-molecular reaction, it is more efficient than hybridization between a hairpin and its complement, which is an inter-molecular reaction. Therefore, it is necessary to denature a hairpin structure by raising temperature in order for its complement to hybridize with the hairpin (☞ Fig. 16).

They attach hairpins with different addresses to a gold surface, and employ a laser beam to locally raise temperature on the surface. Those hairpins on the laser spot are denatured and if their sequence is complementary to that of single strands in the surrounding solution, they can hybridize with the single strands.

It is not obvious to construct a hierarchical molecular memory based on hybridization, in which a memory word contains multiple address digits. For accessing a target memory word of the form in ☞ Fig. 11, single strands whose sequences are complementary to the address digits of the target are mixed with the whole molecular memory. Then the address part of the target

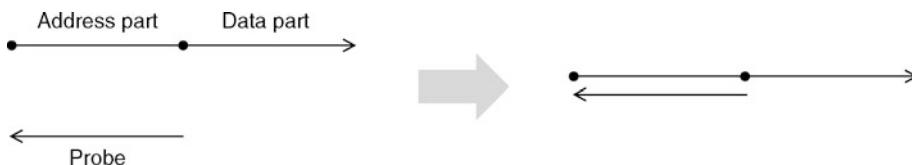
■ Fig. 14

A memory word in a hierarchical molecular memory extractable by PCR. Each word contains nested primer pairs.



■ Fig. 15

Hybridization of a memory word and a probe that is complementary to the address part of the word. The address part is changed from a single strand to a double strand.



word becomes fully double-stranded. It is therefore necessary to distinguish the target word from other words whose address part is not double-stranded or partially double-stranded.

Hagiya's group employed hairpins to construct a hierachal molecular memory based on hybridization (Kameda et al. 2008b). A memory word consists of concatenated hairpins and a single-stranded end as in **Fig. 17**.

A memory word has a single-stranded part to the left of the first hairpin. Single strands called openers are used to open the hairpins successively. The first opener is complementary to the single-stranded part of a word and the stem of the first (leftmost) hairpin. It first hybridizes with the single-stranded part, and then invades into the stem of the first hairpin by branch migration. Eventually, it completely hybridizes with the single-stranded part and the adjacent part of the stem. Consequently the first hairpin is opened, and the other part of the stem is exposed as single-stranded. Therefore, if the second opener, which is complementary to the exposed stem and the adjacent part of the second hairpin, is present, it first

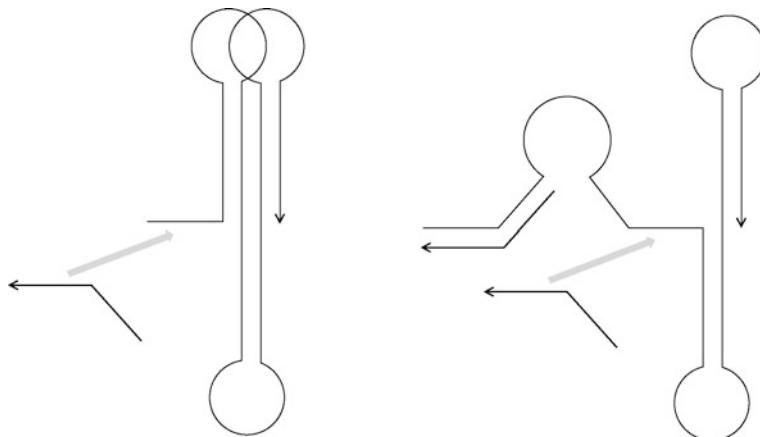
Fig. 16

A hairpin molecule as a memory word. If the temperature is raised, the hairpin is denatured and it can hybridize with a probe that is complementary to the hairpin loop and one side of the stem.



Fig. 17

A memory word consisting of three consecutive hairpins. A single strand called an opener hybridizes with the single-stranded part of the memory word and opens the adjacent (first) hairpin. As a result, the next opener that opens the second hairpin can hybridize with the exposed stem of the first hairpin.



hybridizes with the exposed stem and then invades the stem of the second hairpin by branch migration. Note that without the first opener, the second cannot hybridize with the memory word and open the second hairpin. Therefore, only if all the openers are present, are all the hairpins opened. In particular, the final hairpin is opened only if all the openers are present. This means that a memory word is accessed if and only if the final hairpin is opened.

Hagiya's group actually constructed a memory word consisting of four hairpins and verified that the proposed scheme actually worked (Kameda et al. 2008b). They also constructed a molecular memory consisting of 3^3 addresses by employing the scheme. In this molecular memory, a memory word consists of three hairpins, and there are three variations for each opener that opens one of the hairpins. Consequently, there are 3^3 combinations of three openers. They verified that the final hairpin of each memory word can only be opened by the corresponding combination of three openers.

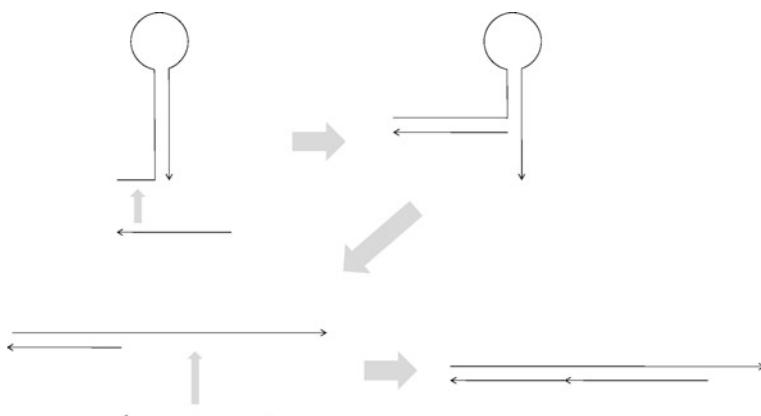
It is also possible to combine schemes described in this section. For example, in the scheme by Suyama's group a hairpin molecule can hybridize with its complement only if the local temperature is raised (Takinoue and Suyama 2004, 2006). If the stem of the hairpin is longer, raising the temperature is not sufficient to denature the hairpin. In this case, a single strand can be used to partially open the stem as in Fig. 18. This is a combination of the scheme of Hagiya's group (Kameda et al. 2008b) and that of Suyama's group (Takinoue and Suyama 2004, 2006).

3.2 Read Operation

In an extraction-based molecular memory, if a memory word consists of only the address part, and if existence and nonexistence of a word means bit 1 and bit 0, respectively, then extracting a memory word, whether by affinity separation or by PCR, amounts to reading the bit of the address of the word.

Fig. 18

A hairpin memory word accessible by the combination of an opener and temperature. If an opener is present, it partially opens the hairpin. If the temperature is raised (from the upper right diagram to the lower left one), then the hairpin is completely opened and it can hybridize with the complementary strand.



If a memory word contains a data part that encodes some additional information, then it is necessary to read the data part by a separate read operation. In an extraction-based molecular memory, the extracted word is given to a read operation such as gel electrophoresis, sequencing, or DNA chip detection. If the length of the data part encodes the information stored in the memory word, gel electrophoresis is the most efficient way to read the data. If the sequence of the data part encodes information, then sequencing is required to determine the bases in the data part.

As sequencing technology is progressing rapidly, it is becoming more and more efficient to read the base sequence of a given DNA molecule. However, it is still time consuming to read out individual bases in DNA. Moreover, sequencing requires a certain number of copies of the given DNA molecule, and additional amplification by PCR is usually necessary. Therefore, PCR-based memories are more suitable for sequencing.

DNA chips are also used if the target memory word is amplified with a fluorescent label. Single strands whose sequences are complementary to possible data parts are attached on a DNA chip. Single-stranded copies of the target word with a fluorescent label are then put on the DNA chip. The fluorescent label is observed at the spot having the sequence complementary to the data part.

Fluorescence resonance energy transfer (FRET) is more useful because the target need not be amplified. It can be used if the extracted target word (or its data part) is single-stranded. A molecular beacon is a hairpin structure having a fluorescent group at its end and a quencher group at the other end (Fig. 19). If the data part is complementary to the loop sequence of the beacon, the hairpin is opened and the beacon hybridizes with the data part. Consequently, the fluorescent group and the quencher group part and fluorescence is observed.

In a hybridization-based molecular memory, since the target memory word is not extracted but is still mixed with other memory words, the methods mentioned above (except that of FRET) cannot be used in general.

In the simplest hybridization-based scheme in which each memory word consists of only a nonhierarchical address part, a molecular beacon corresponding to the address part of a memory word can be used to detect the existence of the memory word.

If each memory word contains a data part in addition to a nonhierarchical address part, a single-stranded probe whose sequence is complementary to the address part can be used together with another probe for the data part. Both probes are labeled with different

Fig. 19

A molecular beacon. A fluorescent group and a quencher group are attached at the terminals of a small hairpin. Since the stem of the hairpin is short, it can hybridize with the strand whose sequence is complementary to the hairpin loop. After hybridization, the fluorescent group parts from the quencher and fluorescence is observed.

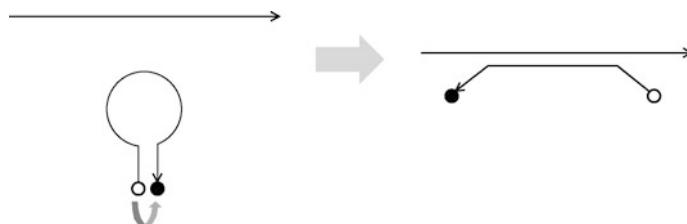
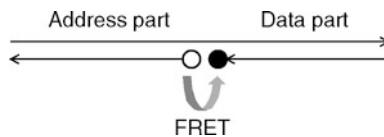


Fig. 20

FRET between an address probe and a data probe. The two probes are complementary to the address part and the data part of a memory word. If they hybridize with the memory word, they come in close proximity and FRET occurs.



fluorescent groups. If both probes hybridize with the memory word, these groups come in close proximity to each other and energy transfer occurs (☞ Fig. 20).

Note that this method cannot be applied to a naïve hierarchical memory in which address digits are simply concatenated. The address digit next to the data part is only significant for energy transfer, and other digits are irrelevant.

The hairpin-based scheme proposed by Hagiya's group solves this problem (Kameda et al. 2008b). In this scheme, the last hairpin is regarded as a data part. Remember that the last hairpin can be opened only after all the preceding hairpins have been opened by their openers. If the last hairpin is opened, since it becomes single-stranded, FRET as in a molecular beacon, can be used to detect this single-stranded part.

So far, we assumed that the sequence (or length) of the data part of a memory word represents the content of the word. In general, however, the read operation of a molecular memory depends on the write operation. For example, in some hybridization-based memory schemes, each memory word consists of only an address part and does not have a data part. The single-stranded and double-stranded states of a memory word are regarded as bit 0 and 1, respectively. If a memory word hybridizes with the corresponding probe and becomes double-stranded, the word is considered to have bit 1, while if it is single-stranded, it is considered to have bit 0. In this kind of scheme, it is not obvious how to read out the single- or double-stranded state of the memory word.

A molecular beacon can be used here because it does not hybridize with the memory word having bit 1, since it is already double-stranded. However, if a beacon hybridizes with the memory word having bit 0, it becomes double-stranded. So this method is destructive. It is also possible to raise the temperature of the whole molecular memory, and collect all the probes attached to memory words. Each probe denotes that the corresponding memory word has had bit 1. This read operation is also destructive. Undestructive read operations require more sophisticated schemes in this kind of hybridization-based memory.

There are some other operations related to the read operation. For example, extraction with respect to the data part is a kind of read operation. They are explained in ☞ Sect. 3.3.

3.3 Write Operation

If existence and nonexistence of a memory word mean bit 1 and bit 0, respectively, then writing bit 1 to an address amounts to simply inserting the memory word of that address into the molecular memory. This operation is always easy. On the other hand, writing bit 0 to an address amounts to deleting the memory word of that address from the molecular memory.

This is easy in a molecular memory based on extraction by affinity separation. It is difficult or impossible to implement the delete operation in a molecular memory based on extraction by PCR or hybridization.

If the data part of a memory word encodes some information, then the write operation should change the data part according to the specified data.

In a molecular memory based on extraction by affinity separation, changing the data part of a memory word having a specified address amounts to deleting the memory word and adding a new memory word having the same address and the specified data. If a molecular memory is based on extraction by PCR, it is also possible to add a new memory word with the specified data, while it is difficult to delete the old memory word with the original data.

In a (nonhierarchical) hybridization-based molecular memory, it is possible to add a new data part to the accessed memory word or delete the original data part. For example, if the new data part is put together with the probe complementary to the address part and the new data part, they form a double strand with the target memory word (without a data part) as in [Fig. 21](#). After ligation is performed, the target word is concatenated with the new data part. On the other hand, it is also possible to cut the target memory word with some restriction enzyme, if it hybridizes with a probe that is complementary to the address part and the adjacent portion of the data part. The double strand formed by the probe and the word is supposed to contain the recognition site of the restriction enzyme ([Fig. 22](#)).

The hairpin-based scheme proposed by Hagiya's group also allows the write operation in a hierarchical molecular memory (Kameda et al. [2008b](#)). After the last hairpin is opened, it is possible to add a specified data part with the help of an appropriate probe.

As explained in [Sect. 3.2](#), if a memory word does not have a data part, hybridization between a memory word and its probe is considered as writing bit 1 onto the word. Although writing bit 1 is easy, reading the bit is difficult. Writing bit 0 is also nontrivial. Using branch migration as in molecular tweezers (Yurke et al. [2000](#)), the strand complementary to the probe can remove it from the memory word ([Fig. 23](#)).

Fig. 21

Writing a new data part to a memory word. The new data part is put together with the probe complementary to the address part and the new data part. After ligation, the new data part is concatenated with the address part.

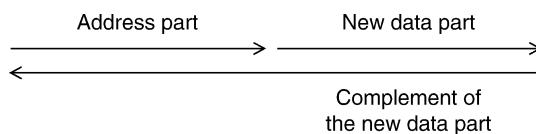


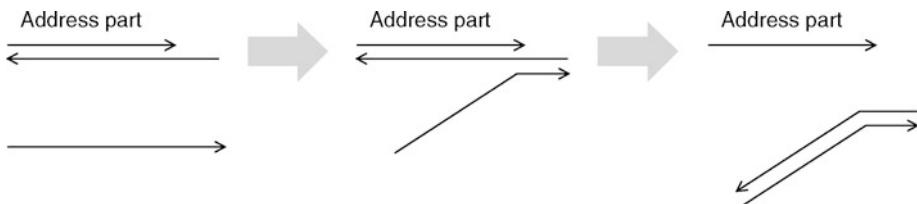
Fig. 22

Deleting the data part of a memory word. The probe is complementary to the address part and the adjacent portion of the data part. The double strand formed by the word and the probe is cut by a restriction enzyme.

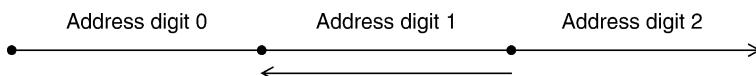


Fig. 23

Removing a probe from a memory word. The strand complementary to the probe hybridizes with the single-stranded portion of the probe and eventually removes it from the memory word.

**Fig. 24**

Writing on an address digit. A probe hybridizes with the address digit 1. This operation is regarded as writing bit 1 to the address digit.



3.4 Other Operations

Due to the features of molecular memories, there are some more operations that can be implemented efficiently.

In a nonhierarchical molecular memory, the address part and the data part are symmetric. Therefore, it is also possible to extract memory words with respect to their data part. This kind of operation is called content-based addressing, and memories that allow such an operation are called associative memories.

In a hierarchical molecular memory, it is also possible to extract memory words based on a specified address digit or on a combination of address digits. For example, when an NP-complete problem like SAT is solved, each candidate of a solution is regarded as a memory word consisting of multiple address digits. Iteration of extraction results in a solution of such a problem. For solving a SAT problem, extraction is performed for each clause of a given Boolean formula. Each extraction operation selects those memory words that contain at least one digit corresponding to a literal in the corresponding clause.

Like extraction, it is also possible to write data onto a specified address digit. For example, in a hybridization-based hierarchical memory, a probe for a particular address digit hybridizes with those memory words having the address digit ([Fig. 24](#)). In the simplest case of this scheme, each address digit has only one possible sequence, and whether or not a probe hybridizes with an address digit is based on whether the address digit is 1 or 0. In this case, the write operation is considered to change the address of a memory molecule from, for example, 000 to 010 ([Fig. 24](#)).

As a result of this operation, each memory word contains a certain number of probes that hybridize with one of its address digits. Another possible operation is separation by counting. Using gel electrophoresis, for example, it is possible to separate memory words according to the number of probes they contain.

The two operations mentioned above, writing bit 1 to an address digit and separation by counting, compose the molecular computing paradigm called aqueous computing (Head and Gal 2001). Note that this scheme is different from the scheme described in [Sect. 4.1](#), where each address digit has two possible sequences.

4 In Vitro DNA Memory

This section briefly touches on molecular memories in an aqueous solution, that is, in a test tube. They are manipulated using the operations mentioned in the previous section.

4.1 Solving NP-Complete Problems

Mainly for solving NP-complete problems, various memory schemes were proposed in the early days of DNA computing.

As mentioned in the previous section, Adleman's group solved a 20-variable SAT problem (Braich et al. 2002). For solving the problem, they constructed a pool consisting of 2^{20} different assignments to 20 Boolean variables. Each assignment can be regarded as a hierarchical memory word with 20 address digits, where each digit has two possible values, true or false. The size of this molecular memory is therefore 2^{20} .

The initial pool of assignments, that is, the whole address space, was constructed in two steps. In the first step, assignments to 10 variables were generated using a mix-and-split synthesis technique (Faulhammer et al. 2000). They used two tubes (or columns) beginning with synthesizing the two sequences for the tenth variable. The results were mixed and then separated into two again. The two sequences for the ninth variable were then attached. The assignments to the first 10 variables and those to the rest were combined using PCR (Stemmer et al. 1995).

The only memory operation employed for solving the problem is extraction with respect to a combination of address digits. For each clause in a given Boolean expression, those memory words that contain an address digit corresponding to a literal in the clause are extracted. If there remain memory words after all the extraction operations are performed, the formula is known to be satisfiable.

4.2 Nested Primer Molecular Memory (NPMM)

Yamamoto et al. constructed a molecular memory consisting of 16^6 addresses (Yamamoto et al. 2008). Their molecular memory is called NPMM, which stands for nested primer molecular memory. Each memory word consists of a data part and an address part. The data part is surrounded by the address part. Each address digit to the left of the data part is paired with a digit to the right, and they comprise a primer pair. In the 16^6 -version of NPMM, there are six address digits, denoted by AL, AR, BL, BR, CL, and CR. The digits AL, BL, and CL are paired with AR, BR, and CR, respectively. The sequence CL is put to the leftmost end of a word and CR is put to the rightmost end. They comprise the first primer pair to extract a memory word. The sequence BL is put to the right of CL, and BR is put to the left of CR. The sequences AL

and AR are put to the left and right of the data part. Therefore, each memory word having the data part, Data, is of the form

$$\text{CL} - \text{BL} - \text{AL} - \text{Data} - \text{AR} - \text{BR} - \text{CR}$$

These memory words are prepared as double strands of DNA. It suffices to repeat three PCR experiments to extract a single memory word.

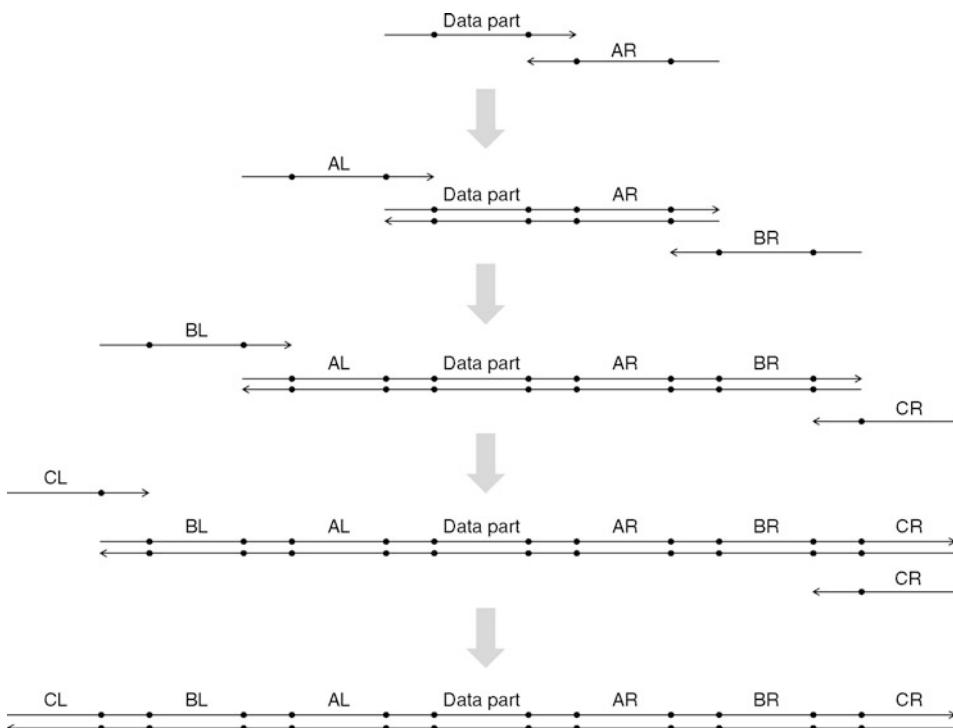
In the 16^6 -version of NPMM, 16 sequences were prepared for each address digit, denoted by AL_i , AR_i , BL_i , etc. Consequently, the entire address space consists of $16^6 = 16,777,216$ addresses. In addition to the 16×6 sequences for the address digits, six linker sequences were inserted between adjacent address digits and between the data part and the address part. As for the data part, three sequences of different lengths were prepared. In total, 105 different sequences were designed, using a two-step search method, which was developed for this molecular memory.

The entire memory was constructed in a style of parallel overlap assembly (Stemmer et al. 1995) as in Fig. 25. The common linker sequences were used as primers in PCR.

In order to verify that the access operation was correctly implemented, Yamamoto et al. picked up one address ($\text{CL}_0, \text{BL}_0, \text{AL}_0, \text{AR}_0, \text{BR}_0, \text{CR}_0$) and inserted a memory word with a longer data part (data40) with the address. The initial memory consists of memory words having a fixed data part (data20). Similarly, they inserted a memory word with the longest

Fig. 25

The construction of NPMM. PCR is iterated four times to construct a molecular memory with six digits.



data part (data60) at another address (CL8, BL8, AL8, AR8, BR8, CR8). Then they accessed 16 different addresses ($CL_i, BL_i, AL_i, AR_i, BR_i, CR_i$) and verified by gel electrophoresis that the access operation was always correct.

One of the most important factors for implementing this kind of molecular memory is to reduce variation of concentrations of memory words. Using real-time PCR, Yamamoto et al. measured the concentrations of the first products in the construction of NPMM, that is, memory words of the form $AL_i - \text{data}20 - AR_j$. They observed that, compared to the least concentration of a memory word, the largest concentration was 2.36 times larger and the average was 1.4. In the 16^6 -version of NPMM with six address digits, it is expected that compared to the least concentration of a memory word, the largest concentration is 173 and the average is 7.53. The molecular memory they actually constructed was supposed to contain 185 molecules for each memory address on average. Therefore, the least number of molecules for a word is expected to be 25.

Yamamoto et al. also estimated the limitation of their approach by analyzing a mathematical model based on the current experimental technologies including PCR. (The numbers in parentheses are their tentative estimations.)

- The total number of molecules should not exceed the number of primers (50 pmol).
- The number of PCR iterations is limited (30 cycles).
- After amplification, the copy number of a target word should be significantly larger than the total number of molecules (1,000 times larger).
- The total number of sequences for address digits is limited (200 sequences).

Using the model, they estimated that it was not possible to construct a molecular memory having more than 170M addresses. In this sense, the molecular memory they constructed was comparable to this limit.

As a possible application of a molecular memory with a huge address space, Hagiya's group proposed to use a molecular memory for DNA ink (Kameda et al. 2008a). By simply diluting a molecular memory, the total number of molecules becomes less than the total number of addresses. Consequently, each memory address may or may not exist in the diluted solution. This amounts to tossing a coin for each address. The diluted solution can be amplified with common primers, and if the primers are removed after amplification, it is practically impossible to completely determine the contents of the memory. Therefore, it can be used for authentication or as a seed for key generation. For example, it can be mixed with ink and used for printing passports or signing secret documents. It is possible to obtain more and more information from the memory by sampling more and more addresses by PCR.

5 DNA Memory on Surfaces

5.1 Introduction

A DNA molecule is a molecular memory stably storing genetic information. The information is described in a DNA base sequence and is accurately read out when DNA is replicated or RNA is transcribed. The specificity of the Watson–Crick base pairing between complementary DNA strands realizes high fidelity in replication and transcription. The specificity of the DNA base pairing is also utilized to construct structures, devices, and computers working in a nanometer scale (Bath and Turberfield 2007; Seeman and Lukeman 2005; Simmel and Dittmer

2005), which are actively developed today: DNA nanostructures (He et al. 2008; Rinker et al. 2008; Rothemund 2006), DNA nanomachines (Goodman et al. 2008; Venkataraman et al. 2007), DNA computers (Benenson et al. 2004; Komiya et al. 2006; Seelig et al. 2006; Takinoue et al. 2008; Zhang et al. 2007), etc.

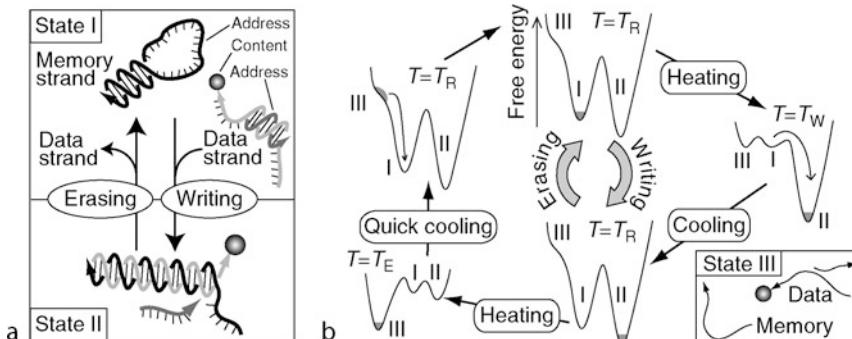
The specificity of the DNA base pairing can achieve even rewritable molecular memories (Shin and Pierce 2004; Takinoue and Suyama 2006). This section introduces one of the rewritable DNA molecular memories. This memory is based on physicochemical reactions of hairpin-shaped DNA molecules; that is, a temperature-controlled conformational transition and DNA hybridization of the hairpin DNA molecules achieve memory addressing and data storing (Takinoue and Suyama 2004, 2006). Although, in life systems, a DNA molecule is a read-only and non-rewritable molecular memory storing information as a base sequence, the physicochemical reactions of the hairpin DNA can realize not only the readout but also the rewriting of the DNA molecular memory. By cooperation between the base sequence information and the physicochemical reactions, the hairpin DNA memory achieves massively parallel and highly selective addressing in a very large memory space.

5.2 Hairpin DNA Memory Dissolved in a Solution

First, the hairpin DNA memory dissolved in a solution is explained. A hairpin DNA memory has two states: a single-stranded hairpin structure (unwritten state, state I of Fig. 26a) and a double-stranded linear structure (written state, state II of Fig. 26a). A memory strand has a hairpin DNA sequence with a memory address in its loop region. A data strand is composed of an address part and a content part. The address part is a linear DNA sequence, whereas the data part can be composed of various substances, such as DNA/RNA sequences, proteins, metal nanoparticles, or quantum dots. The DNA sequence of the address part of the data strand is complementary to that of the hairpin loop of the memory strand, so that the memory and the corresponding data strands can hybridize with each other by base pairing between the

Fig. 26

A hairpin DNA memory dissolved in a solution. (a) Structures of memory and data. State I: unwritten state (hairpin structure). State II: written state (linear structure). (b) Writing and erasing by temperature control. T_R , T_W , and T_E : temperatures of readout, writing, and erasing ($T_R < T_W < T_E$). State III: dissociated state.



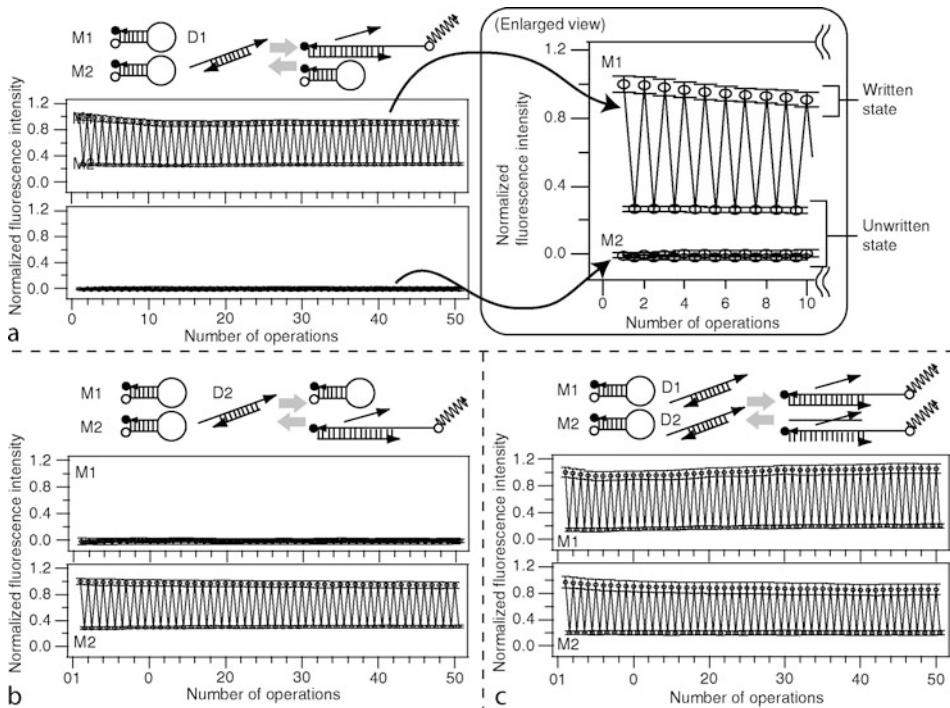
complementary sequences. The address part of the data strand partly forms a double helix with an auxiliary DNA strand to prevent the data strand from spontaneously hybridizing with the memory strand.

The writing and erasing of the hairpin-DNA memory are performed by temperature control of a solution containing memory and data strands. [Figure 26b](#) illustrates the mechanism of memory writing and erasing. Data is written to the memory through a state transition from state I to state II. When a solution containing the memory and the data strands is exposed to a writing temperature (T_W), the hairpin structure of the memory strand opens and the data strand is released from the auxiliary strand, and therefore the data strand hybridizes with the memory strand. Conversely, data is erased through a state transition from state II to state I. When the solution is quickly cooled down to a readout temperature (T_R) after being exposed to an erasing temperature (T_E), the hairpin structure of the memory strand closes before the data strand hybridizes with it again. The intramolecular hairpin formation of the memory strand is faster than the intermolecular hybridization between the memory and the data strands. Once the memory and the data strands are separated, they can hardly hybridize with each other until the solution temperature is raised to T_W .

The writing and erasing of the hairpin DNA memory can be performed even when multiple memory addresses coexist in a memory solution. [Figure 27](#) shows the experimental results of

Fig. 27

Alternate repetition of writing and erasing of hairpin DNA memory. M1, M2: Memory 1, 2. D1, D2: Data 1, 2. (a) Selective writing and erasing of M1-D1, and the enlarged view. (b) Selective writing and erasing of M2-D2. (c) Parallel writing and erasing of M1-D1 and M2-D2. $T_R=25^\circ\text{C}$, $T_W = 50^\circ\text{C}$, $T_E = 90^\circ\text{C}$. Experimental detail: reference Takinoue and Suyama (2006).



selective and parallel writing and erasing of the hairpin DNA memory in the case of the coexistence of two memory addresses. In **Fig. 27a** and **b**, Data 1 (D1) and Data 2 (D2) are selectively written to the memories with the corresponding address. **Figure 27c** shows that D1 and D2 are written to the memories with the corresponding address in parallel. The hairpin DNA memory works very stably as shown in the 50-times repeated writing and erasing operations in **Fig. 27**.

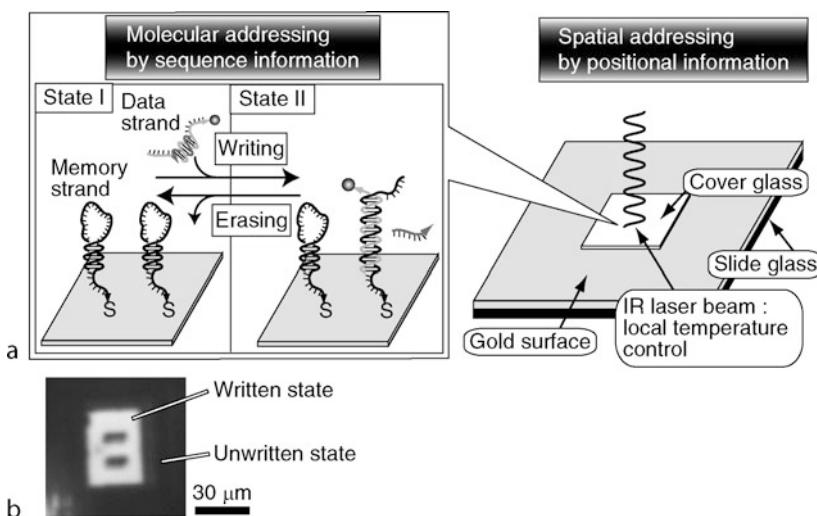
5.3 Hairpin DNA Memory Immobilized on a Surface

The functionality of the hairpin DNA memory can be extended by immobilizing hairpin DNA memory strands onto a solid surface. The immobilization allows spatial addressing of the memory by physically positioning memory locations on the surface, as well as molecular addressing of the memory by hybridization between memory and data strands (**Fig. 28**). In the memory substrate of **Fig. 28a**, the memory strand is uniformly immobilized on the surface of a thin gold layer via a thiol-Au bond. The spatial addressing is achieved through local heating of the memory substrate by IR laser beam irradiation. No irradiation, weak irradiation, and strong irradiation of the IR laser beam correspond to the temperature controls T_R , T_W and T_E in **Fig. 26b**. On the other hand, molecular addressing is performed through hybridization of a data strand with the immobilized memory strand. The reaction is the same as the reaction shown in **Fig. 26a**.

Fig. 28b shows a fluorescence microscope image of a hairpin DNA memory immobilized on a surface. The white and black areas indicate the written and unwritten states,

Fig. 28

A hairpin DNA memory immobilized on a surface. **(a)** Hairpin DNA memories are immobilized on a gold surface. Writing and erasing are performed by IR laser beam irradiation. **(b)** A fluorescence microscope image of the hairpin DNA memory. White area: written state. Black area: unwritten state.



respectively. Local irradiation with the IR laser beam allows data storing with microscale spatial patterning. The memory addressing is not limited only to a single memory address. Selective and parallel writing and erasing for multiple memory addresses can also be realized. In addition, this system is stable for many repetitions of the writing and erasing.

5.4 Conclusion

This section explained the hairpin DNA memory and its immobilization on a solid surface. Hybridization of data strands with memory strands and the structural transition of hairpin-shaped memory strands realize molecular addressing and data storing to the memory. Immobilization makes possible spatial addressing to the memory using locations on the surface.

Hairpin DNA memory immobilized on a solid surface has a lot of possibilities for applications in micro and nanotechnology. In information technology, it will provide a large amount of memory space and massively parallel addressing to store plenty of information, especially molecular information such as molecular computing outputs. It will also provide a universal plate for assembling DNA nanostructures, nanoparticles, and other functional molecules by both molecular addressing and spatial addressing. The memory substrate can be extended to a fundamental technique for a molecular imaging plate to directly store the spatial distribution of mRNAs and proteins in tissues and cells. No silicon-based memory devices can succeed in these applications.

6 In Vivo DNA Memory

6.1 Introduction

The perfect design of in vivo memory requires substantial knowledge of cellular and molecular mechanisms. Ideally, the memory function should be isolated from the native physiological functions to avoid unexpected interactions among sequences. However, it is difficult to meet this requirement, especially if memory mechanisms are implemented with standard nucleic acids. Knowledge of the mechanisms of DNA transcription or translation remains elementary; previous attempts to computationally estimate transcribed regions (or gene structures) never matched the findings of tiling arrays, that as much as 70% of the human genome is transcribed (Kapranov et al. 2002). It is also unknown why bacterial genomes can detect and excise foreign DNA sequences with significantly biased GC contents. Due to these difficulties, in vivo memories have not been as widely investigated as in vitro approaches.

6.2 Encryption into Non-coding Regions

The simplest way to encode information into DNA is to apply a code table translating alphabet letters and digits to DNA triplets.  [Table 2](#) was used by Clelland et al. to encode the message “June 6 Invasion: Normandy” (all in capitals) into a short DNA fragment in vitro (Clelland et al. 1999). A similar translation table was used in other approaches, such as encoding a song phrase of “It’s a small world” into the genomes of *Escherichia coli* and *Deinococcus radiodurans* (Leier et al. 2000; Wong et al. 2003). This straightforward method has a fatal flaw; DNA

Table 2

Direct encryption table used in Clelland et al. (1999)

A = CGA	B = CCA	C = GTT	D = TTG	E = GGT	G = TTT	H = CGC	I = ATG
J = AGT	K = AAG	L = TGC	M = TCC	N = TCT	O = GGC	Q = AAC	R = TCA
S = ACG	T = TTC	U = CTG	V = CCT	W = CCG	X = CTA	Y = AAA	_ = ATA
, = TCG	. = GAT	: = GCT	U ^a = ACT	1 = ACC	2 = TAG	3 = GCA	4 = GAG
5 = AGA	7 = ACA	8 = AGG	9 = GCG				

^a In the original paper, symbols F, P, Z, G are missing and U is duplicated.**Table 3**

An example of comma-free words of DNA triplets that also consider reverse strands. The maximum number of words is 10

ACA	AGA	AAU	ACC	CGA	ACU	GCA	AGG	AGU	CGG
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

sequences are retrieved by polymerase chain reaction (PCR) using short primer sequences, but such sequences may appear in the encoded message. It is an absolute requirement for primer sequences and their reverse complements to never occur in any combination of encrypted sequences irrespective of frame shifts. This condition is met by the design of comma-free codes.

A code is said to be comma-free if separator symbols are unnecessary to parse its words unambiguously. The concept of comma-freeness was first introduced by Francis Crick, as an elegant (but mistaken) association between codons and 20 amino acids (Crick et al. 1957). Indeed, the maximum number of comma-free DNA triplets is exactly 20 out of $4^3 = 64$. (It is easy to say that in this association there is no room for the “stop” sign. It is probably because we now know the correct codon table.)

How many comma-free words can be designed with DNA quadruplets or pentuplets? Obviously, any circular permutation of a comma-free word cannot be another word in the same code. Soon after Crick offered his beautiful hypothesis, Solomon Golomb proved that the maximum number of comma-free words of length k using n letters is $(n^k - n)/k$ (k : prime) (Golomb et al. 1958). When the comma-free condition for reverse complements is also considered, the number is halved (Table 3). Since encryption of alphabet letters, digits, and primer sequences requires at least 40 letters, the necessary comma-free words can be no shorter than DNA pentuplets.

DNA pentuplets may raise different types of complication. The first is a larger bias for GC content, resulting in a broader range of melting temperatures. Since the melting temperature should be the same for all primer sequences, words for primers must be carefully chosen from comma-free DNAs. The second complication is vulnerability to mutations or sequencing errors. Enhancement of the error-correcting ability further lengthens the DNA words, and practical codes may become as long as 12 base-pairs (Arita and Kobayashi 2002). The third and the most serious complication is the possible interaction with native genome sequences. Especially in eukaryotic cells, any noncoding or intergenic region may be functional and it is impossible to guarantee the safety of inserting a long sequence into their genome.

6.3 Encryption into Coding Regions

A better way to achieve *in vivo* memories is to override artificial information on sequences whose biological information is well known. If we can superimpose a message in protein coding regions in a way that the message does not affect amino acid sequences to be translated, then the message can be used as a secret signature, or *DNA watermark*. This idea can be realized easily.

The translation table of amino acids specifies only 21 patterns (20 amino acids and the stop sign) using the 64 codons; the third position of the codons for frequently used amino acids may change without changing the translation result. This is called a *wobbled* codon (❷ [Table 4](#)).

Arita and Ohashi proposed association of a single bit for each codon where a value of 0 indicates the presence of the normal (wild-type) bases at the third position, and a value of one indicates altered bases at this position (wobbled codon by mutation) (Arita and Ohashi 2004).

In this scheme, the normal sequence of an organism is considered to carry information 000...0, and once some of its codons are wobbled, those positions are interpreted as one. The decoding of information requires the sequencing of the protein coding regions, for example, using PCR. Its obvious disadvantage is therefore little storage capacity; the sequence length that can be PCR-amplified is limited.

When wobbling between base U↔C and base A↔G is utilized in the standard codon table (❷ [Table 4](#)), a single bit can be overridden for 60 codons out of 64. Exceptions are UGA, UGG, AUA, and AUG, whose wobbling counterparts code different amino acids or a stop codon. Such codons are skipped in the encoding. For example, if alphabet letters and digits are encoded using the ASCII code (7 bits), one character can be watermarked in approximately

❸ **Table 4**

Codon usage chart of a standard eukaryotic genome. “Stop” indicates stop codons. Bold letters are codons that cannot be used for encoding artificial information (see main text)

UUU Phe	UUA Leu	UAU Tyr	UAA Stop	GAA Glu	GAU Asp	GUG Val	GUU Val
UUC Phe	UUG Leu	UAC Tyr	UAG Stop	GAG Glu	GAC Asp	GUA Val	GUC Val
UCU Ser	UCA Ser	UGU Cys	UGA Stop	GGA Gly	GGU Gly	GCA Ala	GCU Ala
UCC Ser	UCG Ser	UGC Cys	UGG Trp	GGG Gly	GGC Gly	GCG Ala	GCC Ala
CCC Pro	CCG Pro	CGC Arg	CGG Arg	AGG Arg	AGC Ser	ACG Thr	ACC Thr
CCU Pro	CCA Pro	CGU Arg	CGA Arg	AGA Arg	AGU Ser	ACA Thr	ACU Thr
CUC Leu	CUA Leu	CAC His	CAG Gln	AAG Lys	AAC Asn	AUA Ile	AUC Ile
CUU Leu	CUG Leu	CAU His	CAA Gln	AAA Lys	AAU Asn	AUG Met	AUU Ile

Table 5

Translation from binaries to DNA bases

00 → T	01 → G	10 → C	11 → A
--------	--------	--------	--------

every $7 \times 3 = 21$ base-pairs (this is an approximation because we may need to skip some codons). Using 6 bits for each letter, Arita and Ohashi demonstrated the encoding of “KEIO,” the name of their university, into an essential gene of *Bacillus subtilis* (*ftzZ*). The watermarked bacterium did not exhibit any superficial anomaly with respect to colony morphology, growth rate, and the level of protein expression.

Their idea was later extended by Heider and Barnekow (2007). Instead of associating a single bit for all codons, they first translate binary information into DNA bases according to [Table 5](#), and the DNA bases are embedded at the third position of codons one by one. If the substitution produces a wobbled codon, the embedding takes place. If it does not (i.e., if it alters the amino acid in [Table 4](#) or if it is identical to the native sequence), the codon is skipped and the next codon is tested. This method can assign 2 bits for each embedded letter as in [Table 5](#), although the total length of the embedded sequence is information- and genome-dependent: every time a base to be embedded is identical to the third position of the target codon, or if the substitution does not produce a wobbled codon, then the encryption span becomes one codon longer. The actual encoding length is therefore unpredictable.

They also proposed the use of several cryptographic algorithms and error-correcting codes in the translation of the original information into the binary bits. Although they did not demonstrate this error-correcting mechanism, they tested the encoding method in the *Vam7* gene of *Saccharomyces cerevisiae*, a diploidal eukaryote (Heider and Barnekow 2008). The watermarked yeast again exhibited no superficial anomalies including sporulation. The issue of sexual reproduction with recombination, a notable characteristic of eukaryotic cells, was not addressed. These authors subsequently proposed the use of Y-chromosomal or mitochondrial sequences in future trials (Heider and Kessler 2008).

6.4 Conclusion

The use of error-correcting codes or cryptographic algorithms such as RSA is essentially irrelevant to the implementation of DNA memories because the technology under focus is to address how and where to encode information in genomic DNA. We must remember a simple way of implementing the error-correcting ability, that is, the encoding of any of the above code multiple times at different locations in a genome. In fact, using *Bacillus subtilis*, Yachie et al. demonstrated an application of a multiple sequence alignment for embedded memory fragments, each of which was retrieved by PCR experimentally (Yachie et al. 2007). They also proposed whole genome sequencing for retrieving memory sequences. Having seen the recent advances in high-speed genome sequencing, we no longer dismiss such a proposal as unrealistic.

Research on synthetic biology may lead to another type of information storage in vivo. Gardner et al. used two DNA-binding regulatory genes, each of which represses the other, to implement a bistable, genetic toggle switch (Gardner et al. 2000). This switch can be regarded as a single-bit memory although the reliability of keeping the memory bit under cell replication is not clear. In their system, toggling was performed by externally adding

a chemical agent that blocks the DNA-binding ability of a protein used. Although their approach is obviously not scalable, it exemplifies the advantage of read-write, visual memory. Since it is not difficult to visualize the cellular state by using several different fluorescent proteins, one interesting application area would be scalable, multi-bit, visual *in vivo* memory.

7 Conclusion

We hope the reader now realizes that there exist a variety of models and methods to store information in terms of molecules. In particular, types of DNA memories range from those in solution or on surfaces to those stored *in vivo*. However, they all require careful design of base sequences of DNA. In particular, DNA memories in solution and on surfaces rely on selective hybridization between complementary sequences of DNA, and efforts to construct a reasonably large memory space in those types have led to the theory and practice of sequence design explained in Sect. 2. Such theory and practice can also be applied to molecular machines or nanostructures constructed from DNA.

Many of the achievements reported in this chapter, including NPMM and hairpin-based memories, were supported by the JST-CREST project on molecular memory.

References

- Aboul-ela F, Koh D, Tinoco I, Martin F (1985) Base-base mismatches. Thermodynamics of double helix formation for dCA3XA3G + dCT3YT3G (X, Y = A,C, G,T). *Nucleic Acids Res* 13:4811–4824
- Adleman L (1994) Molecular computation of solutions to combinatorial problems. *Science* 266:1021–1024
- Allawi H, SantaLucia J (1997) Thermodynamics and NMR of internal G.T mismatches in DNA. *Biochemistry* 36:10581–10594
- Allawi H, SantaLucia J (1998a) Nearest-neighbor thermodynamics of internal A.C mismatches in DNA: sequence dependence and pH effects. *Biochemistry* 37:9435–9444
- Allawi H, SantaLucia J (1998b) Thermodynamics of internal C.T mismatches in DNA. *Nucleic Acids Res* 26:2694–2701
- Allawi H, SantaLucia J (1998c) Nearest neighbor thermodynamic parameters for internal G.A mismatches in DNA. *Biochemistry* 37:2170–2179
- Andronescu M, Aguirre-Hernandez R, Condon AE, Hoos HH (2003) RNAsoft: a suite of RNA secondary structure prediction and design software tools. *Nucleic Acids Res* 31:3416–3422
- Andronescu M, Fejes AP, Hutter F, Hoos HH, Condon AE (2004) A new algorithm for RNA secondary structure design. *J Mol Biol* 336:607–624
- Arita M, Kobayashi S (2002) DNA sequence design using templates. *New Gen Comput* 20:263–277
- Arita M, Ohashi Y (2004) Secret signatures inside genomic DNA. *Biotechnol Prog* 20:1605–1607
- Arita M, Nishikawa A, Hagiya M, Komiya K, Gouzu H, Sakamoto K (2000) Improving sequence design for DNA computing. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2000) Orlando, pp 875–882
- Bath J, Turberfield AJ (2007) DNA nanomachines. *Nat Nanotechnol* 2:275–284
- Baum EB (1995) Building an associative memory vastly larger than the brain. *Science* 268:583–585
- Benenson Y, Gil B, Ben-Dor U, Adar R, Shapiro E (2004) An autonomous molecular computer for logical control of gene expression. *Nature* 429: 423–429
- Bommarito S, Peyret N, SantaLucia J (2000) Thermodynamic parameters for DNA sequences with dangling ends. *Nucleic Acids Res* 28:1929–1934
- Borer P, Dengler B, Tinoco I, Uhlenbeck O (1974) Stability of ribonucleic acid double-stranded helices. *J Mol Biol* 86:843–853
- Braich RS, Chelyapov N, Johnson C, Rothemund PWK, Adleman L (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* 296:499–502
- Braich RS, Johnson C, Rothemund PWK, Hwang D, Chelyapov N, Adleman L (2000) Solution of a satisfiability problem on a gel-based DNA computer.

- Proceedings of the 6th International Workshop on DNA-Based Computers, LNCS 2054: Berlin, pp 27–42
- Brenneman A, Condon AE (2002) Strand design for biomolecular computation. *Theor Comput Sci* 287: 39–58
- Clelland CT, Risca V, Bancroft C (1999) Hiding messages in DNA microdots. *Nature* 399:533–534
- Crick FH, Griffith JS, Orgel LE (1957) Codes without commas. *Proc Natl Acad Sci USA* 43:416–421
- Cukras A, Faulhammer D, Lipton R, Landweber L (1999) Chess games: a model for RNA based computation. *Biosystems* 52:35–45
- Deaton R, Murphy RC, Rose JA, Garzon M, Franceschetti DT, Stevens SEJ (1996) Genetic search for reliable encodings for DNA-based computation. In First genetic programming conference, Stanford, pp 9–15
- Deaton R, Murphy RC, Garzon M, Franceschetti DR, Stevens SEJ (1998) Good encodings for DNA-based solutions to combinatorial problems. Proceedings of the second annual meeting on DNA based computers, DIMACS: series in discrete mathematics and theoretical computer science, vol 44, Princeton, pp 247–258
- Dirks RM, Lin M, Winfree E, Pierce NA (2004) Paradigms for computational nucleic acid design. *Nucleic Acids Res* 32:1392–1403
- Faulhammer D, Cukras A, Lipton R, Landweber L (2000) Molecular computation: RNA solutions to chess problems. *Proceedings of the Natl Acad Sci USA* 97:1385–1389
- Feldkamp U, Saghafi S, Rauhe H (2001) DNASequence-Generator – a program for the construction of DNA sequences. Proceedings 7th International workshop on DNA-based computers, LNCS 2340, Tampa, pp 23–32
- Frutos A, Liu Q, Thiel A, Sanner A, Condon AE, Smith L, Corn RM (1997) Demonstration of a word design strategy for DNA computing on surfaces. *Nucleic Acids Res* 25:4748–4757
- Gardner TS, Cantor CR, Collins JJ (2000) Construction of a genetic toggle switch in *Escherichia coli*. *Nature* 403:339–342
- Garzon M, Deaton R, Neather P, Franceschetti DR, Murphy RC (1997) A new metric for DNA computing. In Proceedings of the 2nd Genetic Programming Conference, San Mateo, pp 472–478
- Garzon M, Deaton R, Nino L, Stevens SEJ, Wittner M (1998) Encoding genomes for DNA computing. In Proceedings of the 3rd Genetic Programming Conference, San Mateo, pp 684–690
- Golomb S, Gordon B, Welch L (1958) Comma-free codes. *Can J Math* 10:202–209
- Goodman RP, Heilemann M, Doose S, Erben CM, Kapanidis AN, Turberfield AJ (2008) Reconfigurable, braced, three-dimensional DNA nanostructures. *Nat Nanotechnol* 3:93–96
- Hagiya M, Arita M, Kiga D, Sakamoto K, Yokoyama S (1999) Towards parallel evaluation and learning of boolean μ -formulas with molecules. *DIMACS Series Discrete Math Theor Comput Sci* 48:57–72
- He Y, Ye T, Su M, Zhang C, Ribbe AE, Jiang W, Mao C (2008) Hierarchical self-assembly of DNA into symmetric supramolecular polyhedra. *Nature* 452:198–201
- Head T, Gal S (2001) Aqueous computing: writing into fluid memory. *Bull EATCS* 75:190–198
- Heider D, Barnekow A (2007) DNA-based watermarks using the DNA-crypt algorithm. *BMC Bioinformatics* 8:176
- Heider D, Barnekow A (2008) DNA watermarks: a proof of concept. *BMC Mol Biol* 9:40
- Heider D, Kessler D, Barnekow A (2008) Watermarking sexually reproducing diploid organisms. *Bioinformatics* 24:1961–1962
- Hofacker IL (2003) Vienna RNA secondary structure server. *Nucleic Acids Res* 31:3429–3431
- Kameda A, Kashiwamura S, Yamamoto M, Ohuchi A, Hagiya M (2008a) Combining randomness and a high-capacity DNA memory. 13th International meeting on DNA computing, DNA13, LNCS 4848, Berlin, pp 109–118
- Kameda A, Yamamoto M, Ohuchi A, Yaegashi S, Hagiya M (2008b) Unravel four hairpins! *Nat Comput* 7:287–298
- Kapranov P, Cawley SE, Drenkow J, Bekiranov S, Strausberg RL, Fodor SPA, Gingras TR (2002) Large-scale transcriptional activity in chromosomes 21 and 22. *Science* 296:916–919
- Kashiwamura S, Kameda A, Yamamoto M, Ohuchi A (2004) Two-step search for DNA sequence design. *IEICE TRANSACTIONS Fundam Electron Commun Comput Sci* E87-A:1446–1453
- Kawashimo S, Ng YK, Ono H, Sadakane K, Yamashita M (2009) Speeding up local-search type algorithms for designing DNA sequences under thermodynamical constraints. Proceedings of the 14th International meeting on DNA computing, LNCS 5347, Prague, pp 168–179
- Kawashimo S, Ono H, Sadakane K, Yamashita M (2008) Dynamic neighborhood searches for thermodynamically designing DNA sequence. 13th International meeting on DNA computing, DNA13, LNCS 4848, Memphis, pp 130–139
- Kitajima T, Takinoue M, Shohda K, Suyama A (2008) Design of code words for DNA computers and nanostructures with consideration of hybridization kinetics. 13th International meeting on DNA computing, DNA13, LNCS 4848, Berlin, pp 119–129
- Kobayashi S, Kondo T, Arita M (2003) On template method for DNA sequence design. Proceedings of

- the 8th International workshop on DNA-based computers, LNCS 2568, Berlin, pp 205–214
- Komiya K, Sakamoto K, Kameda A, Yamamoto M, Ohuchi A, Kiga D, Yokoyama S, Hagiya M (2006) DNA polymerase programmed with a hairpin DNA incorporates a multiple-instruction architecture into molecular computing. *Biosystems* 83:18–25
- Kubota M, Hagiya M (2005) Minimum basin algorithm: an effective analysis technique for DNA energy landscapes. Proceedings of the 10th International workshop on DNA computing, LNCS 3384, New York, pp 202–214
- Leier A, Richter C, Banzhaf W, Rauh H (2000) Cryptography with DNA binary strands. *Biosystems* 57:13–22
- Li M, Lee HJ, Condon AE, Corn RM (2002) DNA word design strategy for creating sets of non-interacting oligonucleotides for DNA microarrays. *Langmuir* 18:805–812
- Li Y, Agrawal S (1995) Oligonucleotides containing G.A pairs: effect of flanking sequences on structure and stability. *Biochemistry* 34:10056–10062
- Li Y, Zon G, Wilson W (1991) Thermodynamics of DNA duplexes with adjacent G.A mismatches. *Biochemistry* 30:7566–7572
- Lipton RJ (1995) DNA solution of hard computational problems. *Science* 268:542–545
- Liu W, Wang S, Gao L, Zhang F, Xu J (2003) DNA sequence design based on template strategy. *J Chem Inf Comput Sci* 43:2014–2018
- Lyngso R, Zuker M, Pedersen C (1999) Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics* 15:440–445
- Marathe A, Condon AE, Corn RM (2001) On combinatorial DNA word design. *J Comput Biol* 8:201–219
- Pancoska P, Moravek Z, Moll UM (2004) Rational design of DNA sequences for nanotechnology, microarrays and molecular computers using Eulerian graphs. *Nucleic Acids Res* 32:4630–4645
- Peyret N, Seneviratne P, Allawi H, SantaLucia J (1999) Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A.A, C.C, G.G, and T.T mismatches. *Biochemistry* 38:3468–3477
- Rinker S, Ke Y, Liu Y, Chhabra R, Yan H (2008) Self-assembled DNA nanostructures for distance-dependent multivalent ligand-protein binding. *Nat Nanotechnol* 3:418–422
- Rothemund PWK (2006) Folding DNA to create nano-scale shapes and patterns. *Nature* 440:297–302
- Ruben AJ, Freeland SJ, Landweber LF (2001) PUNCH: An evolutionary algorithm for optimizing bit set selection. Proceedings of the 7th International workshop on DNA-based computers, LNCS 2340, Tampa, pp 150–160
- SantaLucia J (1998) A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc Natl Acad Sci USA* 95:1460–1465
- SantaLucia J, Allawi H, Seneviratne P (1996) Improved nearest-neighbor parameters for predicting DNA duplex stability. *Biochemistry* 35:3555–3562
- Seelig G, Soloveichik D, Zhang DY, Winfree E (2006) Enzyme-free nucleic acid logic circuits. *Science* 314:1585–1588
- Seeman N (1990) De novo design of sequences for nucleic acid structural engineering. *J Biomol Struct Dyn* 8:573–581
- Seeman NC, Lukeman PS (2005) Nucleic acid nanostructures: bottom-up control of geometry on the nanoscale. *Rep Prog Phys* 68:237–270
- Sen D, Gilbert W (1988) Formation of parallel four-stranded complexes by guanine-rich motifs in DNA and its implications for meiosis. *Nature* 334:364–366
- Sen D, Gilbert W (1990) A sodium-potassium switch in the formation of four-stranded G4-DNA. *Nature* 344:410–414
- Senior M, Jones R, Breslauer K (1988) Influence of dangling thymidine residues on the stability and structure of two DNA duplexes. *Biochemistry* 27:3879–3885
- Shin JS, Pierce NA (2004) Rewritable memory by controllable nanopatterning of DNA. *Nano Lett* 4:905–909
- Simmel FC, Dittmer WU (2005) DNA nanodevices. *Small* 1:284–299
- Stemmer WP, Crameri A, Ha KD, Brennan TM, Heyneker HL (1995) Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides. *Gene* 164:49–53
- Sugimoto N, Nakano S, Yoneyama M, Honda K (1996) Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes. *Nucleic Acids Res* 24:4501–4505
- Takinoue M, Suyama A (2006) Hairpin-DNA memory using molecular addressing. *Small* 2:1244–1247
- Takinoue M, Suyama A (2004) Molecular reactions for a molecular memory based on hairpin DNA. *Chem-Bio Inform J* 4:93–100
- Takinoue M, Kiga D, Shohda K, Suyama A (2008) Experiments and simulation models of a basic computation element of an autonomous molecular computing system. *Phys Rev E* 78:041921
- Tanaka F, Kameda A, Yamamoto M, Ohuchi A (2004) Thermodynamic parameters based on a nearest-neighbor model for DNA sequences with a single-bulge loop. *Biochemistry* 43:7143–7150
- Tanaka F, Kameda A, Yamamoto M, Ohuchi A (2005) Design of nucleic acid sequences for DNA computing based on a thermodynamic approach. *Nucleic Acids Res* 33:903–911
- Tulpan D, Hoos H (2003) Hybrid randomised neighbourhoods improve stochastic local search for DNA code design. In: Advances in artificial intelligence: 16th conference of the Canadian society for

- computational studies of intelligence, Berlin, vol 2671, pp 418–433
- Tulpan D, Hoos H, Condon A (2003) Stochastic local search algorithms for DNA word design. Proceedings of the 8th International workshop on DNA-based computers, LNCS 2568, Berlin, pp 229–241
- Uejima H, Hagiyama M (2004) Secondary structure design of multi-state DNA machines based on sequential structure transitions. Proceedings of the 9th International workshop on DNA-based computers, LNCS 2943, Berlin, pp 74–85
- Venkataraman S, Dirks RM, Rothemund PWK, Winfree E, Pierce NA (2007) An autonomous polymerization motor powered by DNA hybridization. *Nat Nanotechnol* 2:490–494
- Watson J, Crick F (1953) Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature* 171:737–738
- Wong P, Wong K-K, Foote H (2003) Organic data memory using the DNA approach. *Commun ACM* 46(1):95–98
- Yachie N, Sekiyama K, Sugahara J, Ohashi Y, Tomita M (2007) Alignment-based approach for durable data storage into living organisms. *Biotechnol Prog* 23:501–505
- Yamamoto M, Kashiwamura S, Ohuchi A, Furukawa M (2008) Large-scale DNA memory based on the nested PCR. *Nat Comput* 7:335–346
- Yoshida H, Suyama A (2000) Solution to 3-SAT by breadth first search. *DIMACS Ser Discrete Math Theor Comput Sci* 54:9–22
- Yurke B, Turberfield AJ, Mills AP, Simmel FC, Neumann JL (2000) A DNA-fuelled molecular machine made of DNA. *Nature* 406:605–608
- Zhang DY, Turberfield AJ, Yurke B, Winfree E (2007) Engineering entropy-driven reactions and networks catalyzed by DNA. *Science* 318:1121–1125
- Zuker M, Stiegler P (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res* 9:133–148
- Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res* 31:3406–3415

39 Engineering Natural Computation by Autonomous DNA-Based Biomolecular Devices

John H. Reif¹ · Thomas H. LaBean²

¹Department of Computer Science, Duke University, Durham, NC, USA
reif@cs.duke.edu

²Department of Computer Science and Department of Chemistry and
Department of Biomedical Engineering, Duke University, Durham,
NC, USA
thomas.labean@duke.edu

1	<i>Introduction</i>	1320
2	<i>Introducing DNA, Its Structure and Its Manipulation</i>	1322
3	<i>Adleman's Initial Demonstration of a DNA-Based Computation</i>	1326
4	<i>Self-assembled DNA Tiles and Lattices</i>	1327
5	<i>Autonomous Finite State Computation Using Linear DNA Nanostructures</i>	1331
6	<i>Assembling Patterned and Addressable 2D DNA Lattices</i>	1333
7	<i>Error Correction and Self-repair at the Molecular-Scale</i>	1338
8	<i>Three-Dimensional DNA Lattices</i>	1340
9	<i>From Nucleic Detection Protocols to Autonomous Computation</i>	1343
10	<i>Autonomous Molecular Transport Devices Self-assembled from DNA</i>	1346
11	<i>Conclusions and Challenges</i>	1349

Abstract

This chapter overviews the past and current state of a selected part of the emerging research area of DNA-based biomolecular devices. We particularly emphasize molecular devices that are: *autonomous*, executing steps with no exterior mediation after starting; and *programmable*, the tasks executed can be modified without entirely redesigning the nanostructure.

We discuss work in this area that makes use of synthetic DNA to self-assemble into DNA nanostructure devices. Recently, there has been a series of impressive experimental results that have taken the technology from a state of intriguing possibilities into demonstrated capabilities of quickly increasing scale. We discuss various such programmable molecular-scale devices that achieve: computation, 2D patterning, amplified sensing, and molecular or nanoscale transport.

This article is written for a general audience, and particularly emphasizes the interdisciplinary aspects of this quickly evolving and exciting field.

1 Introduction

1.1 Autonomous Programmable Molecular Devices

This chapter introduces the reader to a rapidly evolving and very exciting topic of nanoscience, namely, self-assembled DNA nanostructures, and their use in molecular computation and molecular transport. To tighten the focus, and to emphasize techniques that seem most promising, the discussion will be limited to processes that are *autonomous* (executing steps with no exterior mediation after starting), and *programmable* (the tasks executed can be modified without entirely redesigning the nanostructure).

1.2 Questions About Biomolecular Devices

We will see that DNA self-assembly processes can be made computational-based and programmable, and it seems likely that computer science techniques will be essential to the further development of this emerging field of biomolecular computation. This chapter particularly illustrates the way in which computer science techniques and methods impact on this emerging field. Some of the key questions one might ask are given below.

- What is the theoretical basis for these devices?
- How will such devices be designed and implemented?
- How can we simulate them prior to manufacture?
- How can we optimize their performance?
- How will such devices be manufactured?
- How much do the devices cost?
- How scalable is the device design?
- How will efficient I/O be accomplished?
- How will they be programmed?

- What efficient algorithms can be programmed?
- What will be their applications? What can they control?
- How can we correct errors or repair them?

Note that these questions are exactly the sort of questions that computer scientists routinely ask about conventional computing devices. The discipline of computer science has developed a wide variety of techniques to address such basic questions, and we will later point out some which have an important impact on molecular-scale devices.

1.3 Why Bottom-Up Self-assembly?

Construction of molecular-scale structures and devices is one of the key challenges facing science and technology in the twenty-first century. This challenge is at the core of the emerging discipline of nanoscience. A key deficiency is the need for robust, error-free methods for self-assembly of complex devices out of large numbers of molecular components. This key challenge requires novel approaches.

For example, the macroelectronics industry is now reaching the limit of miniaturization possible by top-down lithographic fabrication techniques. New bottom-up methods are needed for self-assembling complex, aperiodic structures for nanofabrication of molecular electronic circuits that are significantly smaller than conventional electronics.

1.4 Why Use DNA for Assembly of Biomolecular Devices?

The particular molecular-scale devices that are the topic of this chapter are known as DNA nanostructures. As will be explained, DNA nanostructures have some unique advantages among nanostructures: they are relatively easy to design, fairly predictable in their geometric structures, and have been experimentally implemented in a growing number of labs around the world. They are primarily made up of synthetic DNA. A key principle in the study of DNA nanostructures is the use of self-assembly processes to actuate the molecular assembly. Conventional electronics fabrication is reaching the limit of miniaturization possible by top-down techniques. Since self-assembly operates naturally at the molecular scale, it does not suffer from the limitation in scale reduction that so restricts lithography or other more conventional top-down manufacturing techniques.

In attempting to understand the modern development of DNA self-assembly, it is interesting to recall that mechanical methods for computation date back to the very onset of computer science, for example, to the cog-based mechanical computing machine of Babbage. Lovelace stated in 1843 that Babbage's "Analytical Engine weaves algebraic patterns just as the Jacquard-loom weaves flowers and leaves." In some of the recently demonstrated methods for biomolecular computation described here, computational patterns were essentially woven into molecular fabric (DNA lattices) via carefully controlled and designed self-assembly processes.

1.5 The Dual Role of Theory and Experimental Practice

In many cases, self-assembly processes are programmable in ways analogous to more conventional computational processes. We will overview theoretical principles and techniques

(such as tiling assemblies and molecular transducers) developed for a number of DNA self-assembly processes that have their roots in computer science theory (e.g., abstract tiling models and finite state transducers).

However, the area of DNA self-assembled nanostructures and molecular robotics is by no means simply a theoretical topic – many dramatic experimental demonstrations have already been made and a number of these will be discussed. The complexity of these demonstrations has increased at an impressive rate (even in comparison to the rate of improvement of silicon-based technologies).

1.6 Applications of Autonomous Programmable Molecular Devices

This chapter discusses the accelerating scale of complexity of DNA nanostructures (such as the number of addressable pixels of 2D patterned DNA nanostructures) and provides some predictions for the future. Molecular-scale devices using DNA nanostructures have been engineered to have various capabilities, ranging from (1) execution of molecular-scale computation, (2) use as scaffolds or templates for the further assembly of other materials (such as scaffolds for various hybrid molecular electronic architectures or perhaps high-efficiency solar cells), (3) robotic movement and molecular transport (akin to artificial, programmable versions of cellular transport mechanisms), (4) exquisitely sensitive molecular detection and amplification of single molecular events, and (5) transduction of molecular sensing to provide drug delivery. Error-correction techniques for correct assembly and repair of DNA self-assemblies have also been recently developed. Computer-based design and simulation are also essential to the development of many complex DNA self-assembled nanostructures and systems.

1.7 The Operation of Programmable Molecular Devices

While we will describe a wide variety of methods for executing molecular computation, these methods can be partitioned into two basic classes:

- *Distributed parallel molecular computations* execute a computation for which they require multiple distinct molecules that interact to execute steps of each computation. An example is the tiling assembly computations described in [Sect. 5](#).
- *Local molecular computations* execute computations within a single molecule, possibly in parallel with other molecular computing devices. An example is Whiplash polymerase chain reaction (PCR) described in [Sect. 9.1](#).

2 Introducing DNA, Its Structure and Its Manipulation

2.1 Introducing DNA

In general, nanoscience research is highly interdisciplinary. In particular, DNA self-assembly uses techniques from multiple disciplines such as biochemistry, physics, chemistry, and material science, as well as computer science and mathematics. While this makes the topic

quite intellectually exciting, it also makes it challenging for a typical computer science reader. Having no training in biochemistry, one must obtain a coherent understanding of the topic from a short article. For this reason, this chapter was written with the expectation that the reader is a computer scientist with little background knowledge of chemistry or biochemistry. On the other hand, a reader with a basic knowledge of DNA, its structure and its enzymes can skip this section and proceed to the next.

2.2 DNA and Its Structure

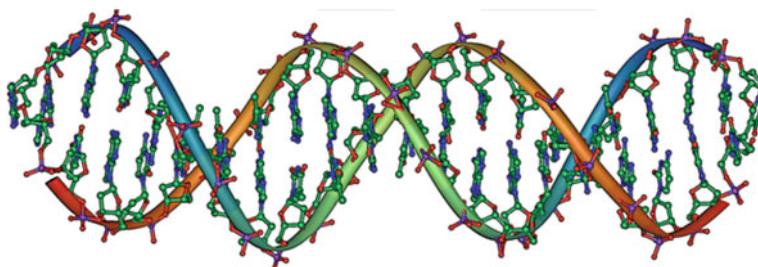
A brief introduction to DNA is given here. *Single stranded DNA* (denoted ssDNA) is a linear polymer consisting of a sequence of nucleotide bases spaced along a backbone with chemical directionality (i.e., the 5-prime and 3-prime ends of the backbone are chemically distinct). By convention, the base sequence is listed starting from the 5-prime end of the polymer and ending at the 3-prime end (these names refer to particular carbon atoms in the deoxyribose sugar units of the sugar-phosphate backbone, the details of which are not critical to the present discussion). The consecutive monomer units (base + sugar + phosphate) of an ssDNA molecule are connected via covalent bonds. There are four types of DNA bases: adenine, thymine, guanine, and cytosine typically denoted by the symbols A, T, G, and C, respectively. These bases form the alphabet of DNA; the specific sequence comprises DNA's information content. The bases are grouped into *complementary pairs* (G, C) and (A, T).

The most basic DNA operation is *hybridization* where two ssDNA oriented in opposite directions can bind to form a double-stranded *DNA helix* (dsDNA) by pairing between complementary bases. DNA hybridization occurs in a physiologic-like buffer solution with appropriate temperature, pH, and salinity (● Fig. 1).

Since the binding energy of the pair (G, C) is approximately half-again the binding energy of the pair (A, T), the association strength of hybridization depends on the sequence of complementary bases, and can be approximated by known software packages. The *melting temperature* of a DNA helix is the temperature at which half of all the molecules are fully hybridized as double helix, while the other half are single-stranded. The kinetics of the DNA hybridization process is quite well understood; it often occurs in a (random) zipper-like manner, similar to a biased one-dimensional random walk.

■ Fig. 1

Structure of a DNA double helix. (Image by Michael Ströck and released under the GNU Free Documentation License (GFDL).)



Whereas ssDNA is a relatively floppy molecule, dsDNA is quite stiff (over lengths of less than 150 or so bases) and has the well characterized double helix structure. The exact geometry (angles and positions) of each segment of a double helix depends slightly on the component bases of its strands and can be determined from known tables. There are about 10.5 bases per full rotation on this helical axis. A *DNA nanostructure* is a multimolecular complex consisting of a number of ssDNA that have partially hybridized along their subsegments.

2.3 Manipulation of DNA

Some techniques and known enzymes used for manipulation of DNA nanostructures are listed here.

Strand displacement, which is the displacement of a DNA strand from a prior hybridization with another complementary strand, is a key process in many of the DNA protocols for running DNA autonomous devices. (Figure 2a) illustrates the displacement of DNA strand via branch migration.

In addition to the hybridization reaction described above, there is a wide variety of known enzymes and other proteins used for manipulation of DNA nanostructures that have predictable effects. (Interestingly, these proteins were discovered in natural bacterial cells and tailored for laboratory use.) These include:

- *Restriction enzymes*, some of which can cut (or nick, which is to cut only one strand) strands of a DNA helix at locations determined by short specific DNA base sequences (Fig. 3).
- *Ligase enzymes* that can heal nicks in a DNA helix (i.e., form covalent bonds to join two backbones).
- *Polymerase enzymes*, which, given an initial “primer” DNA strand hybridized onto a segment of a template DNA strand, can extend the primer strand in the 5' to 3' direction by appending free DNA nucleotides complementary to the template's nucleotides (see Fig. 2b). Certain polymerase enzymes (e.g., phi-29) can, as a side effect of their polymerization reaction, efficiently displace previously hybridized strands.
- In addition, *Deoxyribozymes (DNAzymes)* is a class of nucleic acid molecules that possess enzymatic activity – they can, for example, cleave specific target nucleic acids. Typically, they are discovered by in vivo evolution search. They have had some use in DNA computations (e.g., see Stojanovic and Stefanovic 2003).

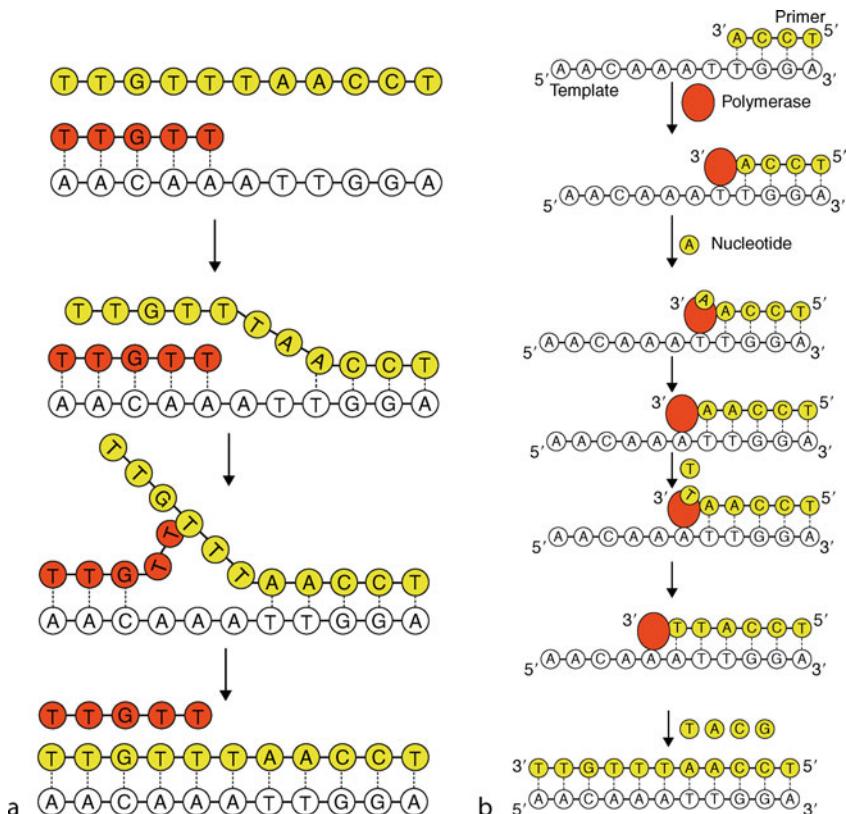
Besides their extensive use in other biotechnology, the above reactions, together with hybridization, are often used to execute and control DNA computations and DNA robotic operations. The restriction enzyme reactions are programmable in the sense that they are site specific, only executed as determined by the appropriate DNA base sequence. The latter two reactions, using ligase and polymerase, require the expenditure of energy via consumption of adenosine triphosphate (ATP) molecules, and thus can be controlled by ATP concentration.

2.4 Why Use DNA to Assemble Molecular-Scale Devices?

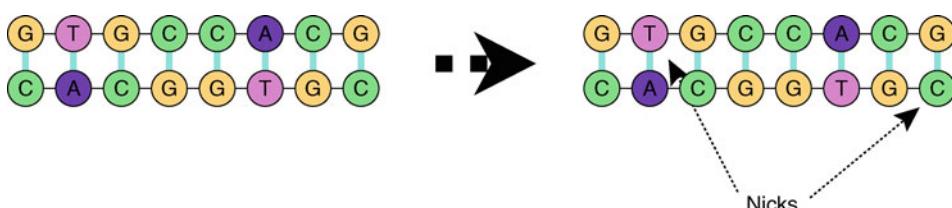
Listed below are some reasons why DNA is uniquely suited for the assembly of molecular-scale devices.

Fig. 2

(a) (Left) Strand displacement of dsDNA via a branch migration hybridization reaction: Figure illustrates DNA strand displacement of a DNA strand (indicated in red) induced by the hybridization of a longer strand (indicated in yellow), allowing the structure to reach a lower energy state. (b) (Right) Extension of primer strand bound to the template by DNA polymerase.

**Fig. 3**

Example of restriction enzyme cuts of a doubly stranded DNA subsequence.



There are many advantages of DNA as a material for building things at the molecular scale.

(a) From the perspective of design, the advantages are:

- The structure of most complex DNA nanostructures can be reduced to determining the structure of short segments of dsDNA. The basic geometric and thermodynamic properties of dsDNA are well understood and can be predicted by available software systems from key relevant parameters like sequence composition, temperature, and buffer conditions.
- The design of DNA nanostructures can be assisted by software. To design a DNA nanostructure or device, one needs to design a library of ssDNA strands with specific segments that hybridize to (and only to) specific complementary segments on other ssDNA. There are a number of software systems (developed at NYU, Caltech, Arizona State, and Duke University) for the design of the DNA sequences composing DNA tiles and for optimizing their stability, which employ heuristic optimization procedures for this combinatorial sequence design task (see [Sect. 4.4](#) for more details).

(b) From the perspective of experiments, the advantages are:

- The solid-phase chemical synthesis of custom ssDNA is now routine and inexpensive; a test tube of ssDNA consisting of any specified short sequence of bases (<150) can be obtained from commercial sources for modest cost (about half a US dollar per base at this time); it will contain a very large number (typically at least 10^{12}) identical ssDNA molecules. The synthesized ssDNA can have errors (premature termination of the synthesis is the most frequent error), but can be easily purified by well-known techniques (e.g., electrophoresis as mentioned below).
- The assembly of DNA nanostructures is a very simple experimental process: in many cases, one simply combines the various component ssDNA into a single test tube with an appropriate buffer solution at an initial temperature above the melting temperature, and then slowly cools the test tube below the melting temperature.
- The assembled DNA nanostructures can be characterized by a variety of techniques. One such technique is electrophoresis. It can provide information about the relative molecular mass of DNA molecules, as well as some information regarding their assembled structures. Other techniques like atomic force microscopy (AFM) and transmission electron microscopy (TEM) provide images of the actual assembled DNA nanostructures on 2D surfaces.

3 Adleman's Initial Demonstration of a DNA-Based Computation

3.1 Adleman's Experiment

The field of DNA computing began in 1994 with a laboratory experiment (Adleman 1994, 1998). The goal of the experiment was to find a Hamiltonian path in a graph, which is a path that visits each node exactly once. To solve this problem, a set of ssDNA was designed based on the set of edges of the graph. When combined in a test tube and cooled, they self-assembled into dsDNA. Each of these DNA nanostructures was a linear DNA helix that corresponded to a path in the graph. If the graph had a Hamiltonian path, then one (or a subset) of these DNA nanostructures encoded the Hamiltonian path. By conventional biochemical extraction

methods, Adleman was able to isolate only DNA nanostructures encoding Hamiltonian paths, and by determining their sequence, the explicit Hamiltonian path. It should be mentioned that this landmark experiment was designed and experimentally demonstrated by Adleman alone, a computer scientist with limited training in biochemistry.

3.2 The Non-scalability of Adleman's Experiment

While this experiment founded the field of DNA computing, it was not scalable in practice, since the number of different DNA strands needed increased exponentially with the number of nodes of the graph. Although there can be an enormous number of DNA strands in a test tube (10^{15} or more, depending on solution concentration), the size of the largest graph that could be solved by his method was limited to at most a few dozen nodes. This is not surprising, since finding the Hamiltonian path is an NP-complete problem, whose solution is likely to be intractable using conventional computers. Even though DNA computers operate at the molecular-scale, they are still equivalent to conventional computers (e.g., deterministic Turing machines) in computational power. This experiment taught a healthy lesson to the DNA computing community (which is now well recognized): to carefully examine scalability issues and to judge any proposed experimental methodology by its scalability.

3.3 Autonomous Biomolecular Computation

Shortly following Adleman's experiment, there was a burst of further experiments in DNA computing, many of which were quite ingenious. However, almost none of these DNA computing methods were autonomous, and instead required many tedious laboratory steps to execute. In retrospect, one of the most notable aspects of Adleman's experiment was that the self-assembly phase of the experiment was completely autonomous – it required no exterior mediation. This autonomous property makes an experimental laboratory demonstration much more feasible as the scale increases. The remaining article mostly discusses autonomous devices for biomolecular computation based on self-assembly.

4 Self-assembled DNA Tiles and Lattices

4.1 DNA Nanostructures

Recall that a DNA nanostructure is a multimolecular complex consisting of a number of ssDNA that have partially hybridized along their subsegments. The field of DNA nanostructures was pioneered by Seeman (Robinson and Seeman 1987; Sherman and Seeman 2004).

Particularly useful types of motifs often found in DNA nanostructures include:

A *stem-loop* (a), where ssDNA loops back to hybridize on itself (that is, one segment of the ssDNA (near the 5' end) hybridizes with another segment further along (nearer the 3' end) on the same ssDNA strand). The shown stem consists of the dsDNA region with sequence CACGGTGC on the bottom strand. The shown loop consists of the ssDNA region with sequence TTTT. Stem-loops are often used to form patterns on DNA nanostructures: a *sticky end* (b), where unhybridized ssDNA protrudes from the end of a double helix. The sticky end

shown (ATCG) protrudes from dsDNA (CACG on the bottom strand). Sticky ends are often used to combine two DNA nanostructures together via hybridization of their complementary ssDNA. **Figure 4** shows the antiparallel nature of dsDNA with the 5' end of each strand pointing toward the 3' end of its partner strand.

A *Holliday junction*, as illustrated in **Fig. 5**, where two parallel DNA helices form a junction with one strand of each DNA helix (blue and red) crossing over to the other DNA helix. Holliday junctions are often used to tie together various parts of a DNA nanostructure.

Fig. 4

DNA stem-loop and a DNA sticky end.

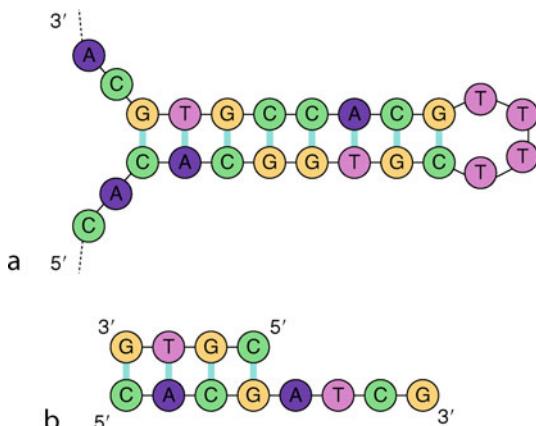
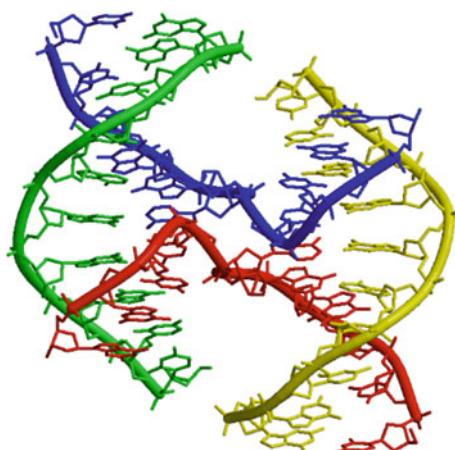


Fig. 5

DNA Holliday junction. (Created by Miguel Ortiz-Lombardía, CNIO, Madrid, Spain.)



4.2 Computation By Self-assembly

The most basic way that computer science ideas have impacted DNA nanostructure design is via the pioneering work by theoretical computer scientists on a formal model of 2D tiling due to Wang in 1961, which culminated in a proof by Berger in 1966 that universal computation could be done via tiling assemblies. Winfree et al. (1998) was the first to propose applying the concepts of computational tiling assemblies to DNA molecular constructs. His core idea was to use tiles composed of DNA to perform computations during their self-assembly process. To understand this idea, we will need an overview of DNA nanostructures, as presented in [Sect. 4.3](#).

4.3 DNA Tiles and Lattices

A *DNA tile* is a DNA nanostructure that has a number of sticky ends on its sides, which are termed *pads*. A DNA lattice is a DNA nanostructure composed of a group of DNA tiles that are assembled together via hybridization of their pads. Generally, the strands composing the DNA tiles are designed to have a melting temperature above those of the pads, ensuring that when the component DNA molecules are combined together in solution, first the DNA tiles assemble, and only then, as the solution is further cooled, do the tiles bind together via hybridization of their pads.

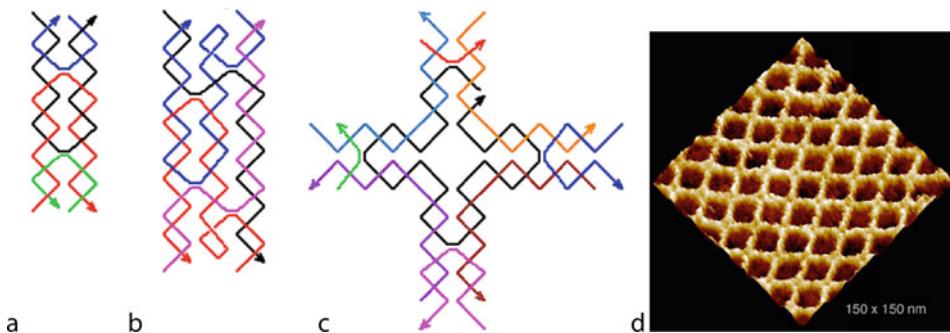
[Figure 6](#) describes some principal DNA tiles (also see LaBean et al. 2006).

Seeman and Winfree, in 1998, developed a family of DNA tiles known collectively as DX tiles (see [Fig. 6a](#)) that consisted of two parallel DNA helices linked by immobile Holliday junctions. They demonstrated that these tiles formed large 2D lattices, as viewed by AFM.

Subsequently, other DNA tiles were developed to provide more complex strand topology and interconnections, including a family of DNA tiles known as TX tiles (see [Fig. 6b](#)) composed of three DNA helices. Both the DX tiles and the TX tiles are rectangular in shape, where two opposing edges of the tile have pads consisting of ssDNA sticky ends of the component strands. In addition, TX tiles have topological properties that allow for strands to propagate in useful ways through tile lattices (this property is often used for aid in patterning DNA lattices as described below).

Fig. 6

DNA tiles. (a) DX tile. (b) TX tile. (c) Cross tile. (d) AFM image of 2D DNA lattice of cross tiles.



Other DNA tiles known as *Cross tiles* (see [Fig. 6c](#)) (Yan et al. 2003b) are shaped roughly square (or more accurately, square cruciform), and have pads on all four sides, allowing for binding of the tile directly with neighbors in all four directions in the lattice plane. [Figure 6d](#) shows an AFM image of a 2D DNA lattice using Cross tiles.

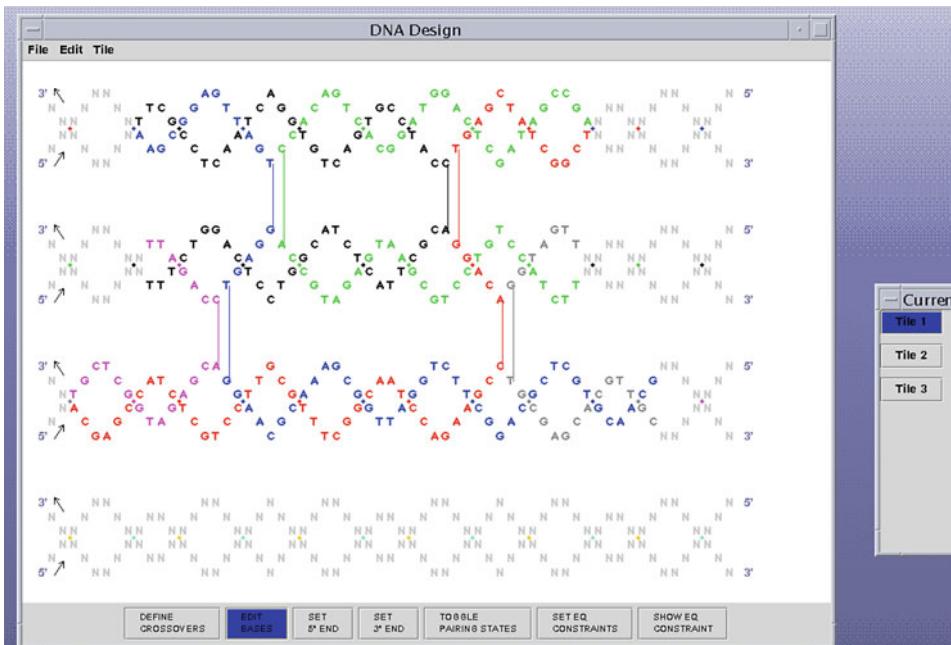
To program a tiling assembly, the pads of tiles are designed so that tiles assemble together as intended. Proper designs ensure that only the adjacent pads (two pairs of sticky ends in the case of Cross tiles) of neighboring tiles are complementary, so only those pads hybridize together.

4.4 Software for the Design of DNA Tiles

A number of prototype computer software systems have been developed for the design of the DNA sequences composing DNA tiles, and for optimizing their stability. [Figure 7](#) gives a screen shot of a software system known as TileSoft (Yin et al. 2004a, b), developed jointly by Duke and Caltech, which provides a graphically interfaced sequence optimization system for designing DNA secondary structures. A more recent commercial product, NanoEngineer, with enhanced capabilities for DNA design and a more sophisticated graphic interface, was developed by Nanorex, Inc.

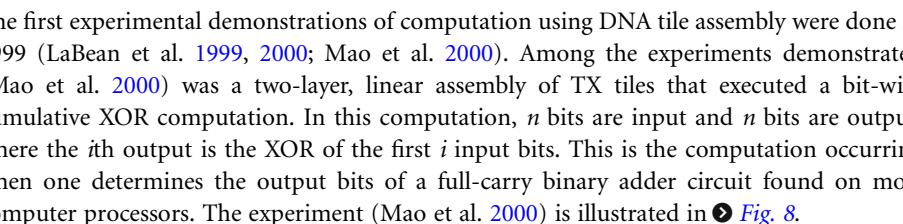
Fig. 7

TileSoft: sequence optimization software for designing DNA secondary structures.



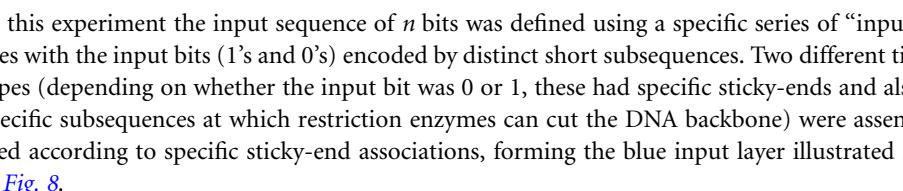
5 Autonomous Finite State Computation Using Linear DNA Nanostructures

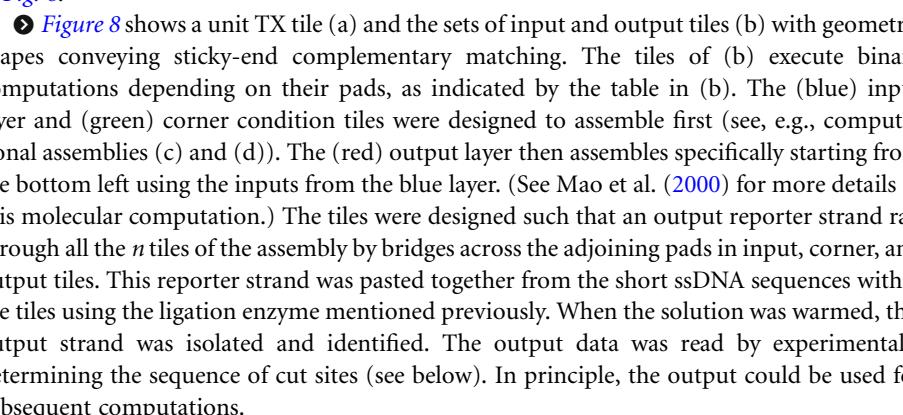
5.1 The First Experimental Demonstration of Autonomous Computations Using Self-assembly of DNA Nanostructures

The first experimental demonstrations of computation using DNA tile assembly were done in 1999 (LaBean et al. 1999, 2000; Mao et al. 2000). Among the experiments demonstrated (Mao et al. 2000) was a two-layer, linear assembly of TX tiles that executed a bit-wise cumulative XOR computation. In this computation, n bits are input and n bits are output, where the i th output is the XOR of the first i input bits. This is the computation occurring when one determines the output bits of a full-carry binary adder circuit found on most computer processors. The experiment (Mao et al. 2000) is illustrated in  Fig. 8.

These experiments (LaBean et al. 1999, 2000; Mao et al. 2000) provided initial answers to some of the most basic questions of how autonomous molecular computation might be done:

- *How can one provide data input to a molecular computation using DNA tiles?*

In this experiment the input sequence of n bits was defined using a specific series of “input” tiles with the input bits (1’s and 0’s) encoded by distinct short subsequences. Two different tile types (depending on whether the input bit was 0 or 1, these had specific sticky-ends and also specific subsequences at which restriction enzymes can cut the DNA backbone) were assembled according to specific sticky-end associations, forming the blue input layer illustrated in  Fig. 8.

 Figure 8 shows a unit TX tile (a) and the sets of input and output tiles (b) with geometric shapes conveying sticky-end complementary matching. The tiles of (b) execute binary computations depending on their pads, as indicated by the table in (b). The (blue) input layer and (green) corner condition tiles were designed to assemble first (see, e.g., computational assemblies (c) and (d)). The (red) output layer then assembles specifically starting from the bottom left using the inputs from the blue layer. (See Mao et al. (2000) for more details of this molecular computation.) The tiles were designed such that an output reporter strand ran through all the n tiles of the assembly by bridges across the adjoining pads in input, corner, and output tiles. This reporter strand was pasted together from the short ssDNA sequences within the tiles using the ligation enzyme mentioned previously. When the solution was warmed, this output strand was isolated and identified. The output data was read by experimentally determining the sequence of cut sites (see below). In principle, the output could be used for subsequent computations.

The next question of concern is:

- *How can one execute a step of computation using DNA tiles?*

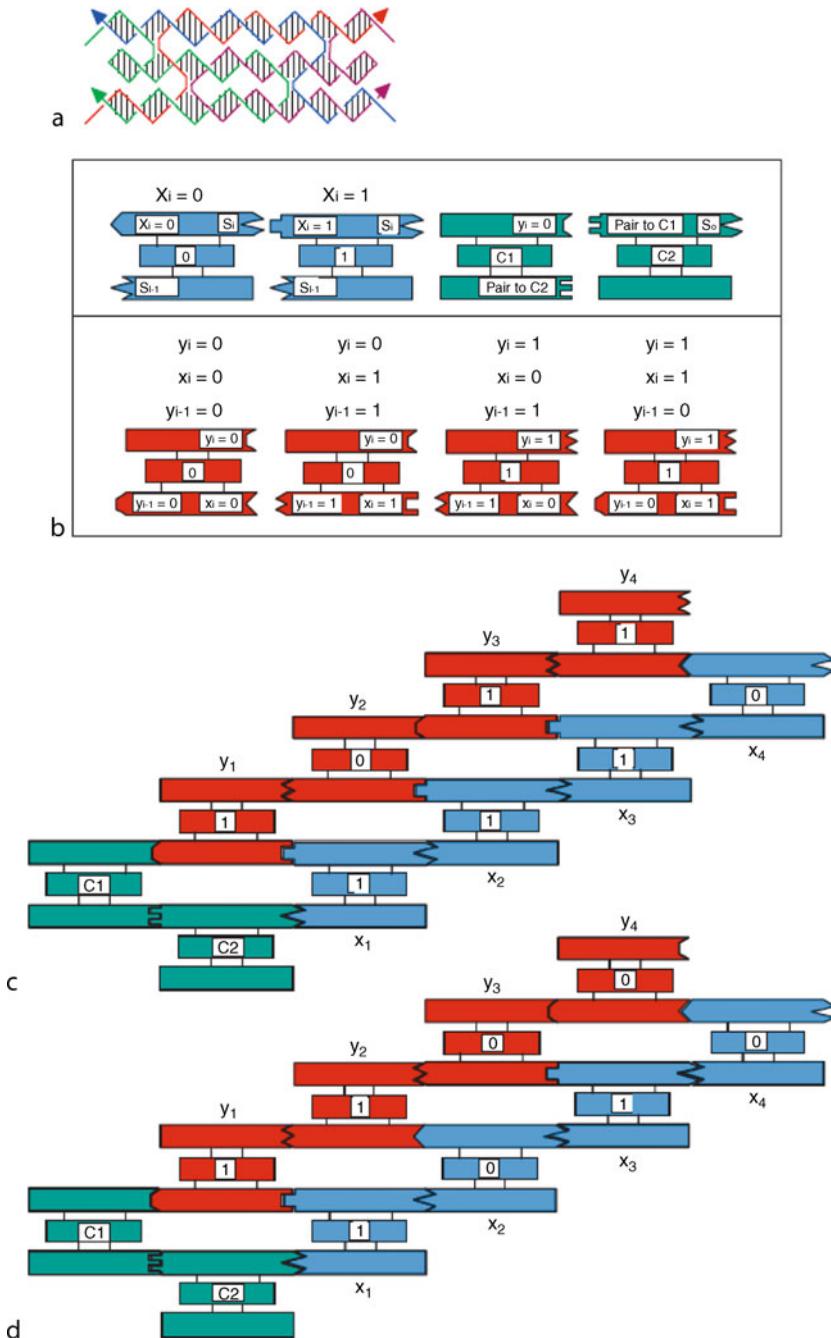
To execute steps of computation, the TX tiles were designed to have pads at one end that encoded the cumulative XOR value. Also, since the reporter strand segments ran through each such tile, the appropriate input bit was also provided within its structure. These two values implied the opposing pad on the other side of the tile to be the XOR of these two bits.

A final question of concern is:

- *How can one determine and/or display the output values of a DNA tiling computation?*

Fig. 8

Sequential Boolean computation via a linear DNA tiling assembly. (Adapted with permission from Mao et al. (2000).) (a) TX tile used in assembly. (b) Set of TX tiles providing logical programming for computation. (c) and (d) Example resulting computational tilings.



The output in this case was read by determining which of the two possible cut sites (endonuclease cleavage sites) were present at each position in the tile assembly. This was executed by first isolating the reporter strand, then digesting separate aliquots with each endonuclease separately and the two together, and finally these samples were examined by gel electrophoresis and the output values were displayed as banding patterns on the gel.

Another method for output (presented below) is the use of AFM observable patterning. The patterning was made by designing the tiles computing a bit 1 to have a stem loop protruding from the top of the tile. The sequence of this molecular patterning was clearly observable under appropriate AFM imaging conditions.

Although only very simple computations, the experiments of LaBean et al. 1999, 2000; Mao et al. 2000; and Yan et al. 2003b did demonstrate for the first time methods for autonomous execution of a sequence of finite-state operations via algorithmic self-assembly, as well as for providing inputs and for outputting the results. Further DNA tile assembly computations will be presented below.

5.2 Autonomous Finite-State Computations via Disassembly of DNA Nanostructures

An alternative method for autonomous execution of a sequence of finite-state transitions was subsequently developed by Shapiro and Benenson (2006). Their technique essentially operated in the reverse of the assembly methods described above, and instead can be thought of as disassembly. They began with a linear double-stranded DNA nanostructure whose sequence encoded the inputs, and then they executed a series of steps that digested the DNA nanostructure from one end. On each step, a sticky end at one end of the nanostructure encoded the current state, and the finite transition was determined by hybridization of the current sticky end with a small “rule” nanostructure encoding the finite-state transition rule. Then a restriction enzyme, which recognized the sequence encoding the current input as well as the current state, cut the appended end of the linear DNA nanostructure, to expose a new sticky end encoding the next state (► Fig. 9).

The hardware–software complex for this molecular device is composed of dsDNA with an ssDNA overhang (shown at the top left ready to bind with the input molecule) and a protein restriction enzyme (shown as gray pinchers).

This ingenious design is an excellent demonstration that there is often more than one way to do any task at the molecular scale. Adar et al. (2004) (see the conclusion, ► Sect. 11 for further discussion) demonstrated in the test tube a potential application of such a finite-state computing device to medical diagnosis and therapeutics.

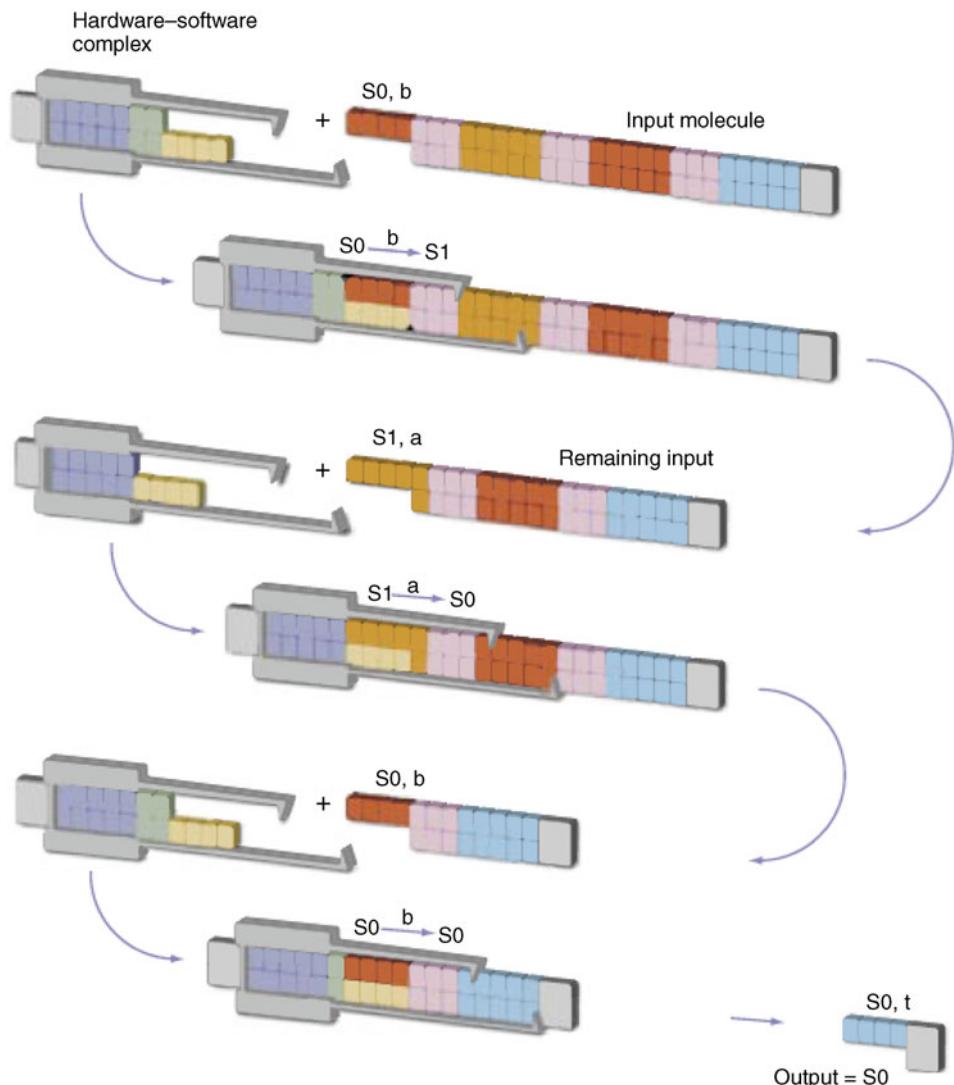
6 Assembling Patterned and Addressable 2D DNA Lattices

One of the most appealing applications of tiling computations is their use to form patterned nanostructures to which other materials can be selectively bound.

An *addressable* 2D DNA lattice is one that has a number of sites with distinct ssDNA. This provides a superstructure for selectively attaching other molecules at addressable locations. Examples of addressable 2D DNA lattices will be given in ► Sect. 6.2.

Fig. 9

Autonomous finite-state computations via disassembly of a double-stranded DNA nanostructure. (Figure adapted with permission from Shapiro and Benenson (2006).)



As discussed below, there are many types of molecules to which one can attach DNA. Known attachment chemistry allows them to be tagged with a given sequence of ssDNA. Each of these DNA-tagged molecules can then be assembled by hybridization of their DNA tags to a complementary sequence of ssDNA located within an addressable 2D DNA lattice. In this way, we can program the assembly of each DNA-tagged molecule onto a particular site of the addressable 2D DNA lattice.

6.1 Attaching Materials to DNA

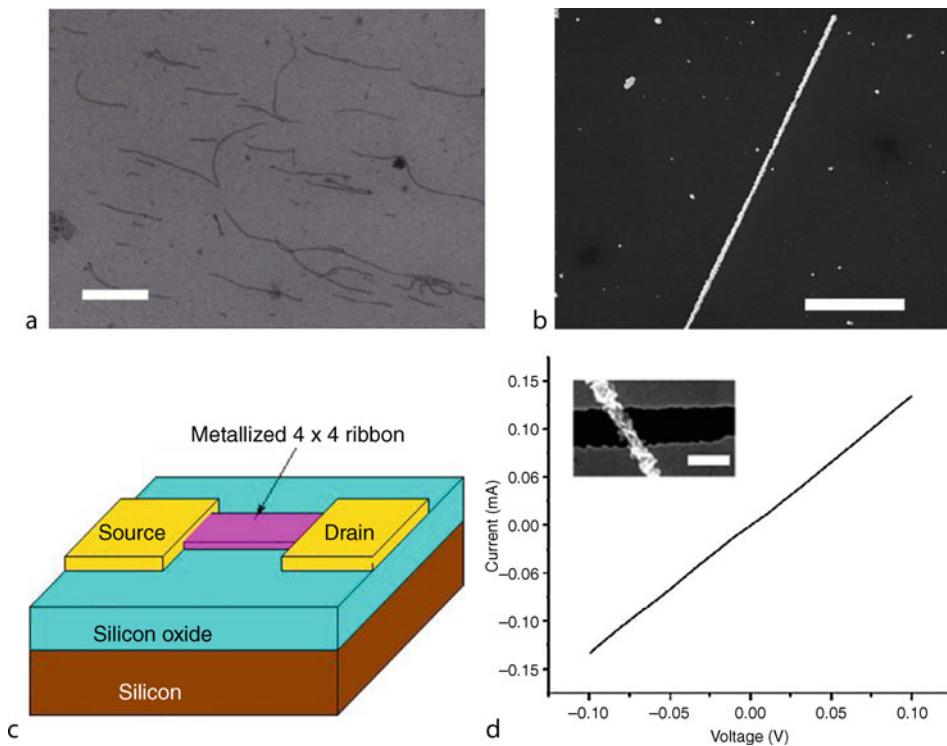
There are many materials that can be made to directly or indirectly bind to specific segments of DNA using a variety of known attachment chemistries. Materials that can directly bind to specific segments of DNA include other (complementary) DNA, RNA, proteins, peptides, and various other materials. Materials that can be made to indirectly bind to DNA include a variety of metals (e.g., gold) that bind to sulfur compounds, carbon nanotubes (via various attachment chemistries), etc. These attachment technologies provide for a molecular-scale method of attaching heterogeneous materials to DNA nanostructures. For example, it can potentially be used for attaching molecular electronic devices to a 2D or 3D DNA nanostructure. Yan et al. (2003c) and Park et al. (2006b) describes conductive wires fabricated from self-assembled DNA tubes plated with gold or silver, as illustrated in [Fig. 10](#).

6.2 Methods for Programmable Assembly of Patterned 2D DNA Lattices

The first experimental demonstration of 2D DNA lattices by Winfree and Seeman provided very simple patterning by repeated stripes determined by a stem loop projecting from every

Fig. 10

Conductive wires fabricated from self-assembled DNA tubes plated with silver. (a) DNA tubes prior to plating. (b) DNA tubes after silver plating. (c) Illustration of conductivity test on silicon oxide substrate. (d) TEM image of conductivity test on silicon oxide substrate.



DNA tile on an odd column. This limited sort of patterning needed to be extended to large classes of patterns.

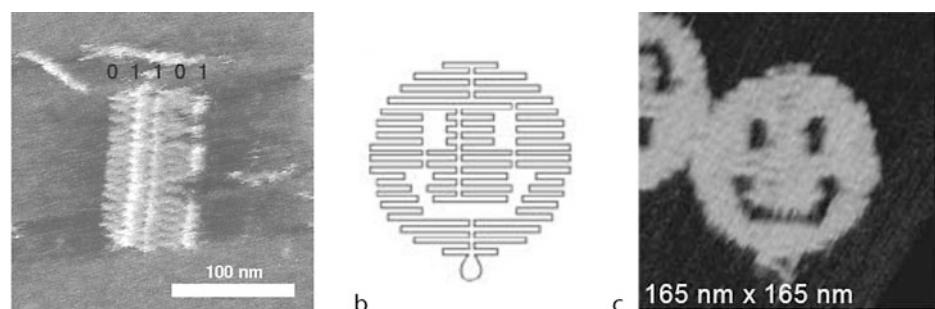
In particular, the key capability needed is a programmable method for forming distinct patterns on 2D DNA lattices, without having to completely redesign the lattice to achieve any given pattern. There are at least three methods for assembling patterned 2D DNA lattices that now have been experimentally demonstrated, as described in the next few sections.

6.2.1 Methods for Programmable Assembly of Patterned 2D DNA Lattices by Use of Scaffold Strands

A *scaffold strand* is a long ssDNA around which shorter ssDNA assemble to form structures larger than individual tiles. Scaffold strands were used to demonstrate programmable patterning of 2D DNA lattices in Yan et al. (2003a) by propagating 1D information from the scaffold into a second dimension to create AFM observable patterns. The scaffold strand weaves through the resulting DNA lattice to form the desired distinct sequence of 2D barcode patterns (left panel of ▶ Fig. 11). In this demonstration, identical scaffold strands ran through each row of the 2D lattices, using short stem loops extending above the lattice to form pixels. This determined a bar code sequence of stripes over the 2D lattice that was viewed by AFM. In principle, this method may be extended to allow for each row's patterning, to be determined by a distinct scaffold strand, defining an arbitrary 2D pixel image. A spectacular experimental demonstration of patterning via scaffold strand is also known as *DNA origami* (Rothenmund 2006). This approach makes use of a long strand of “scaffold” ssDNA (such as from the sequence of a viral phage) that has only weak secondary structure and few long repeated or complementary subsequences. To this is added a large number of relatively short “staple” ssDNA sequences, with subsequences complementary to certain subsequences of the scaffold ssDNA. These staple sequences are chosen so that they bind to the scaffold ssDNA by hybridization and induce the scaffold ssDNA to fold together into a DNA nanostructure. A schematic trace of the scaffold strand is shown in the middle panel of ▶ Fig. 11, and an AFM image of the resulting assembled origami is shown in the right panel of ▶ Fig. 11. This landmark work of Rothenmund (2006)

■ Fig. 11

Methods for programmable assembly of patterned 2D DNA lattices by use of scaffold strands. (a) Barcode patterning. (b) DNA origami design. (c) AFM imaging of DNA origami.



very substantially increases the scale of 2D patterned assemblies to hundreds of molecular pixels (that is, stem loops viewable via AFM) within a square area less than 100 nm on a side. In principle, this “molecular origami” method with staple strands can be used to form arbitrary complex 2D patterned nanostructures as defined. DNA origami has now been extended to simple 3D filaments that were used to partially orient membrane proteins in structural studies employing solution NMR (Douglas et al. 2007).

6.2.2 Programmable Assembly of Patterned 2D DNA Lattices by Computational Assembly

Another very promising method is to use the DNA tile’s pads to program a 2D computational assembly. Recall that computer scientists in the 1970s showed that any computable 2D pattern can be so assembled. Winfree’s group has experimentally demonstrated various 2D computational assemblies, and furthermore provided AFM images of the resulting nanostructures.

☞ *Figure 12* gives a modulo-2 version of Pascal’s Triangle (known as the Sierpinski Triangle) (Rothenmund et al. 2004), where each tile determines and outputs to neighborhood pads the XOR of two of the tile pads. Example AFM images (scale bars = 100 nm) of the assembled structures are shown in the three panels of ☞ *Fig. 12*.

☞ *Figure 13* gives Rothenmund’s and Winfree’s design for a self-assembled binary counter, starting with 0 at the first row, and on each further row being the increment by 1 of the row below. The pads of the tiles of each row of this computational lattice were designed in a similar way to that of the linear XOR lattice assemblies described in the prior section. The resulting 2D counting lattice is found in MUX designs for address memory, and so this patterning may have major applications to patterning molecular electronic circuits (Barish et al. 2005).

■ **Fig. 12**

Programmable assembly of Sierpinski triangle by use of computational assembly. (Figure adapted with permission from Rothenmund et al. (2004).)

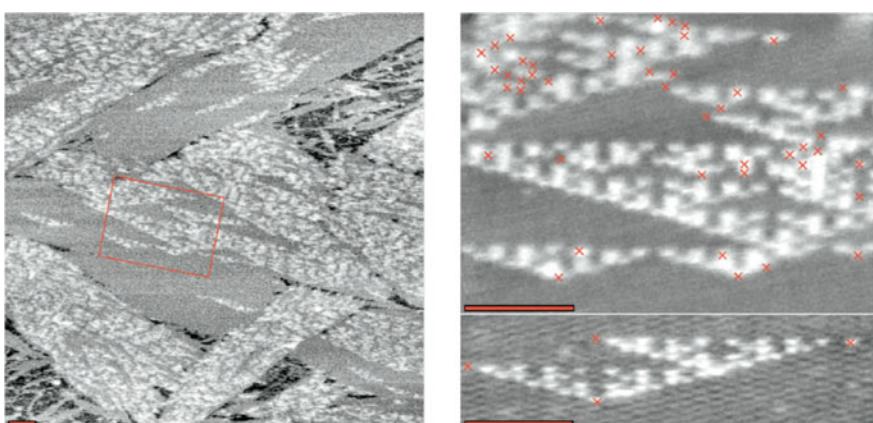
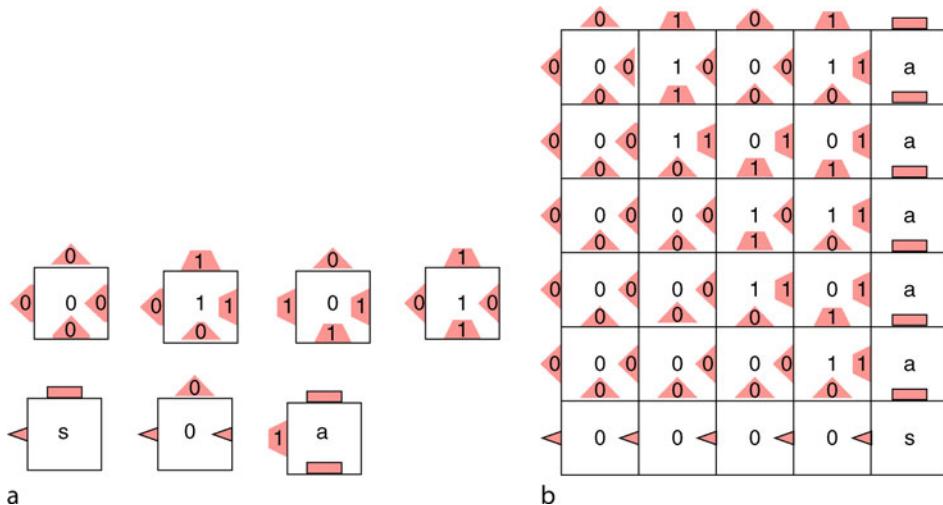


Fig. 13

Rothenmund's and Winfree's design for a self-assembled binary counter using tilings. (a) Tiles used and (b) binary counter assembly.



6.2.3 Programmable Assembly of Patterned 2D DNA Lattices by Hierarchical Assembly

A further approach, known as hierarchical assembly (Park et al. 2006a), is to assemble DNA lattices in multiple stages. **Figure 14** gives three examples of preprogrammed patterns displayed on addressable DNA tile lattices. Tiles are assembled prior to mixing with other preformed tiles. Unique ssDNA pads direct tiles to designed locations. White pixels are “turned on” by binding a protein (avidin) at programmed sites as determined in the tile assembly step by the presence or absence of a small molecule (biotin) appended to a DNA strand within the tile. Addressable, hierarchical assembly has been demonstrated for only modest size lattices to date, but has considerable potential particularly in conjunction with the above methods for patterned assembly.

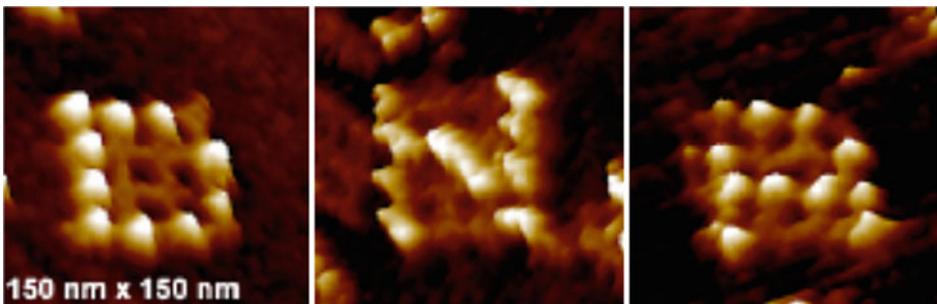
7 Error Correction and Self-repair at the Molecular-Scale

7.1 The Need for Error Correction at the Molecular-Scale

In many of the self-assembled devices described here, there can be significant levels of error. These errors occur both in the synthesis of the component DNA, and in the basic molecular processes that are used to assemble and modify the DNA nanostructures, such as hybridization and the application of enzymes. There are various purification and optimization procedures developed in biochemistry for minimization of many of these types of errors. However, there remains a need for the development of methods for decreasing the errors of assembly and for self-repair of DNA tiling lattices comprising a large number of tiles. A number of

Fig. 14

2D patterns by hierarchical assembly. (Figure adapted with permission from Park et al. (2006a).) AFM images of characters D, N, and A.



techniques have been proposed for decreasing the errors of a DNA tiling assembly, by providing increased redundancy, as described below.

7.2 Winfree's Proofreading Scheme for Error-Resilient Tilings

Winfree and Bekbolatov (2004) developed a “proofreading” method of replacing each tile with a subarray of tiles that provide sufficient redundancy to quadratically reduce errors, as illustrated in [Fig. 15](#). This method, however, increased the area of the assembly by a factor of 4.

7.3 Reif's Compact Scheme for Error-Resilient Tilings

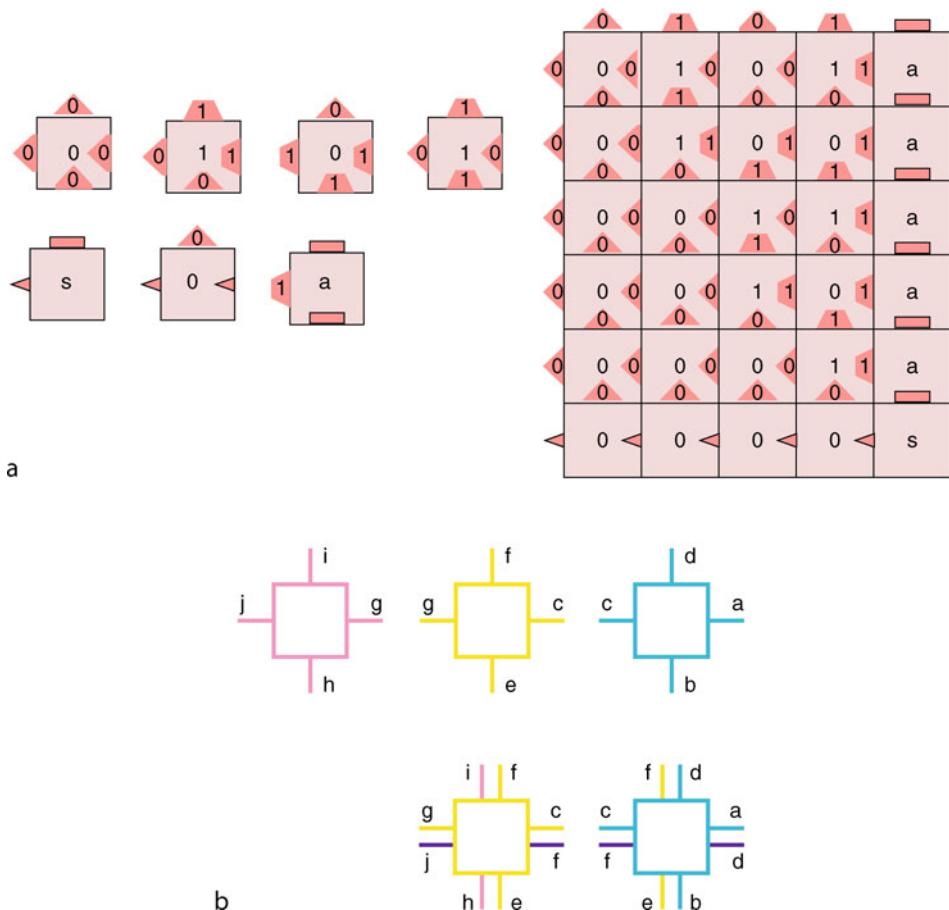
Reif et al. (2006) proposed a more compact method for decreasing assembly errors, as illustrated in [Fig. 16](#). This method modifies the pads of each tile, so that essentially each tile executes both the original computation required at that location, as well as the computation of a particular neighbor, providing a quadratic reduction of errors without increasing the assembly size. The experimental testing of these and related error-reduction methods is ongoing. It seems possible that other error-correction techniques (such as error-correcting codes) developed in computer science may also be utilized.

7.4 Activatable Tiles for Reducing Errors

The uncontrolled assembly of tiling assemblies in reverse directions is potentially a major source of errors in computational tiling assemblies, and a roadblock in the development of applications of large patterned computational DNA lattices. Methods for controlled directional assembly of tiling assemblies would eliminate these errors. Majumder et al. (2007) have recently developed novel designs for an enhanced class of error-resilient DNA tiles (known as *activatable tiles*) for controlled directional assembly of tiles. While conventional DNA tiles store no state, the activatable tiling systems make use of a powerful DNA polymerase enzyme that allows the tiles to make a transition between active (allowing assembly) and inactive states. A *protection-deprotection* process strictly enforces the direction of tiling assembly growth, so that the assembly process is robust against entire classes of growth errors.

Fig. 15

Winfree's proofreading scheme for error-resilient tilings. (a) Original tiles and (b) error-resilient tiles.



Initially, prior to binding with other tiles, the tile will be in an inactive state, where the tile is protected from binding with other tiles and thus prevents lattice growth in the (unwanted) reverse direction. After appropriate binding, other tiles bind to this tile, and then the tile makes a transition to an active state, allowing further growth.

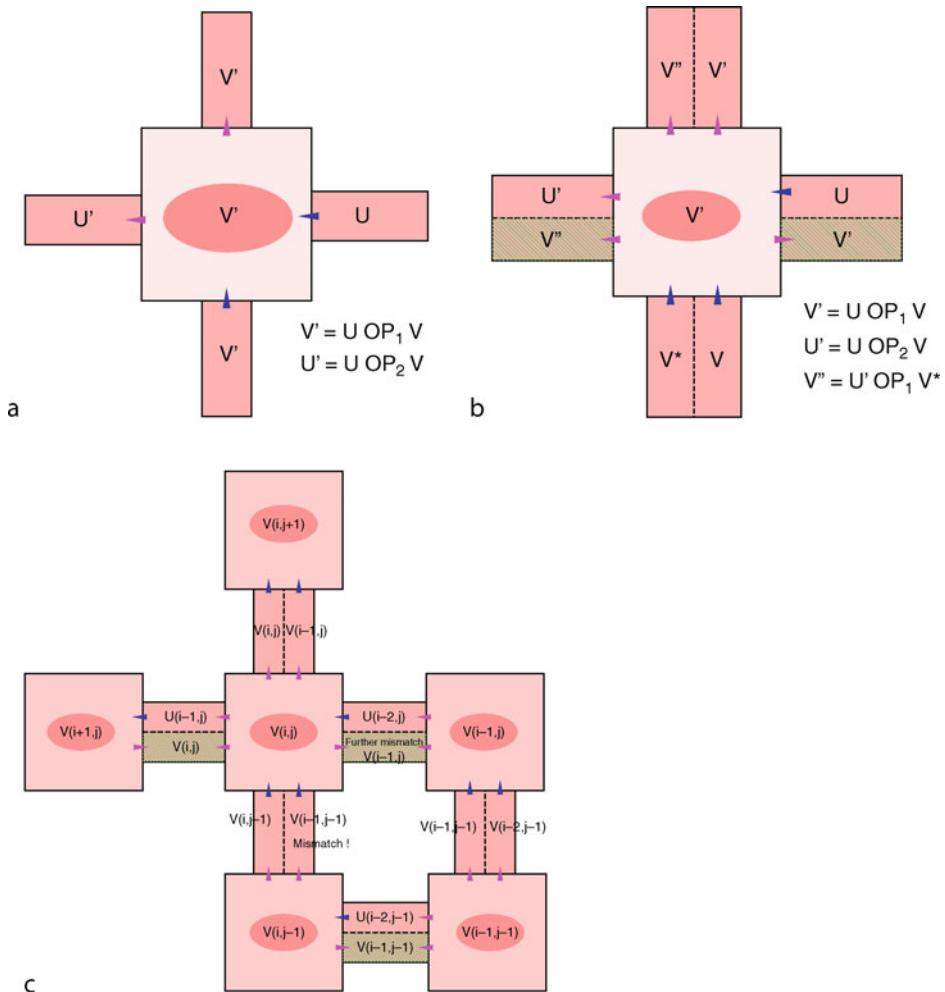
8 Three-Dimensional DNA Lattices

8.1 Potential Applications of Three-Dimensional DNA Lattices

Most of the DNA lattices described in this article have been limited to 2D sheets. It appears to be much more challenging to assemble 3D DNA lattices of high regularity. There are some very important applications to nanoelectronics and biology if this can be done, as described below.

Fig. 16

A compact scheme for error-resilient tilings. (a) Original tile. (b) Error-resilient tiling. (c) A single pad mismatch causes another pad mismatch so destabilizing the assembly.



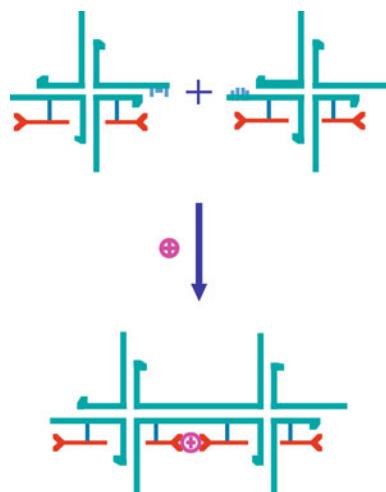
8.2 Scaffolding of 3D Nanoelectronic Architectures

The density of conventional nanoelectronics is limited by lithographic techniques to only a small number of layers. The assembly of even quite simple 3D nanoelectronic devices such as memory would provide much improvement in density. [Figure 17](#) shows DNA (cyan) and protein (red) organizing functional electronic structures.

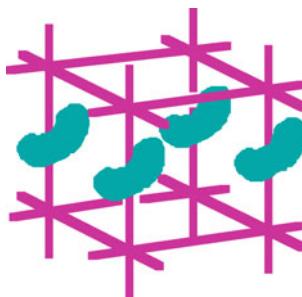
It has been estimated that at least one half of all natural proteins cannot be readily crystallized, and have unknown structure, and determining these structures would have a major impact in the biological sciences. Suppose a 3D DNA lattice can be assembled with sufficient regularity and with regular interstices (say within each DNA tile comprising the

Fig. 17

Scaffolding of 3D nanoelectronic architectures. (Adapted with permission from Robinson and Seeman (1987).)

**Fig. 18**

Scaffolding of proteins into regular 3D arrays. (Adapted with permission from Robinson and Seeman (1987).)



lattice). Then a given protein might be captured within each of the lattice's interstices, allowing it to be in a fixed orientation at each of its regularly spaced locations in 3D (see **Fig. 18**). This would allow the protein to be arranged in 3D in a regular way to allow for X-ray crystallography studies of its structure. This visionary idea is due to Seeman. So far, there has been only limited success in assembling 3D DNA lattices, and they do not yet have the degree of regularity (down to 2 or 3 Å) required for the envisioned X-ray crystallography studies. However, given the success up to now for 2D DNA lattices, this seems eventually achievable.

9 From Nucleic Detection Protocols to Autonomous Computation

9.1 The Detection Problem

A fundamental task of many biochemical protocols is to sense a particular molecule and then amplify the response. In particular, the detection of specific strands of RNA or DNA is an important problem for medicine. Typically, a protocol for nucleic detection is specialized to a subsequence of single-stranded nucleic acid (DNA or RNA oligonucleotide) to be detected. Given a sample containing a very small number of nucleic strand molecules to be detected, a detection protocol must amplify this to a much larger signal. Ideally, the detection protocol is exquisitely sensitive, providing a response from the presence of only a few of the target molecules.

There are a number of novel methods for doing DNA computation that can be viewed as being derived from protocols for detection of DNA. Therefore, understanding the variety of detection protocols can provide insight into these methods used for DNA computation.

9.2 Methods for Autonomous Molecular Computation Derived from PCR

9.2.1 The Polymerase Chain Reaction (PCR)

The original and still the most frequently used method for DNA detection is the *polymerase chain reaction (PCR)* (Saiki et al. 1985), which makes use of DNA polymerase to amplify a strand of DNA by repeated replication, using rounds of thermal cycling. (Recall that given an initial “primer” DNA strand hybridized onto a segment of a template DNA strand, polymerase enzyme can extend the primer strand by appending free DNA nucleotides complementary to the template’s nucleotides.) In addition to DNA polymerase, the protocol requires a pair of “primer” DNA strands, which are extended by the DNA polymerase, each followed by heating and cooling, to allow displacement of the product strands.

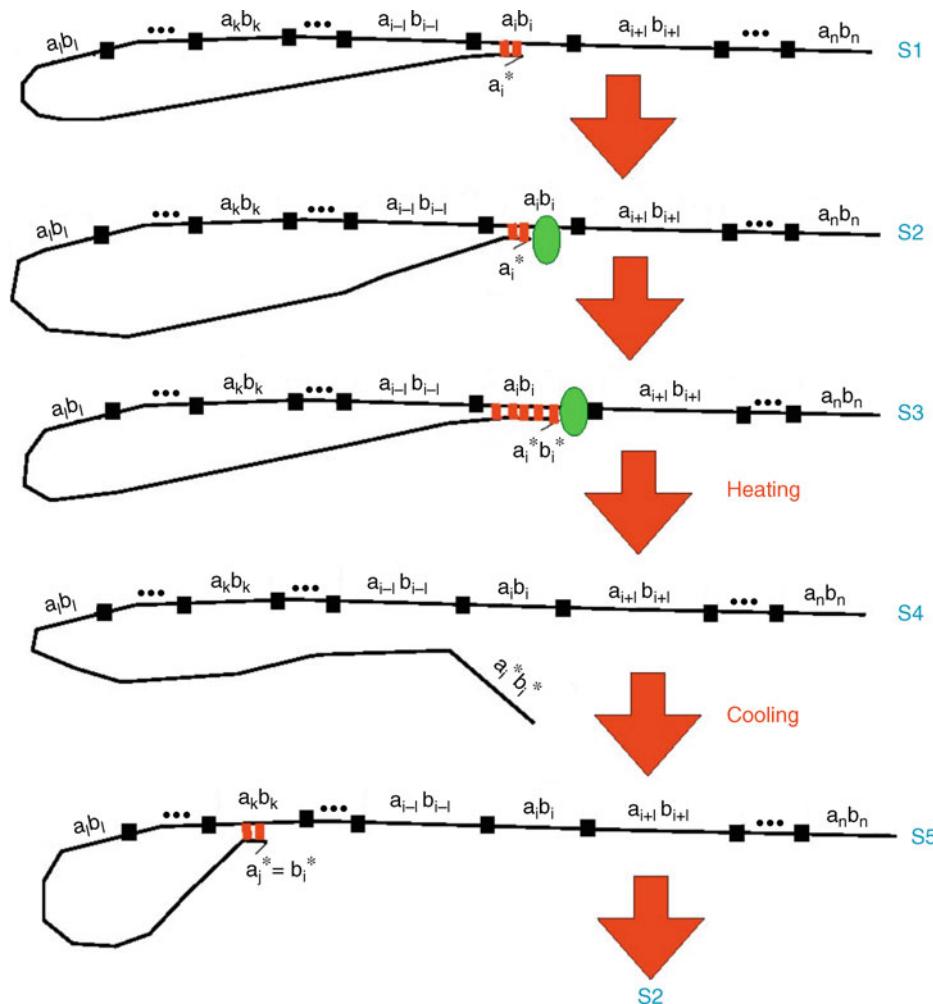
9.2.2 Whiplash PCR: A Method for Local Molecular Computation

A method for DNA computation, known as Whiplash PCR (Sakamoto et al. 1999) makes use of a strand of DNA that essentially encodes a “program” describing state transition rules of a finite state computing machine; the strand is comprised of a sequence of “rule” subsequences (each encoding a state transition rule), and each separated by stopper sequences (which can stop the action of DNA polymerase). On each step of the computation, the 3' end of the DNA strand has a final sequence encoding a state of the computation. A computation step is executed when this 3' end hybridizes to a portion of a “rule” subsequence, and the action of DNA polymerase extends the 3' end to a further subsequence encoding a new state (Fig. 19).

Whiplash PCR is interesting, since it executes a local molecular computation (recall that a molecular computation is local if the computation is within a single molecule, possibly in

Fig. 19

Whiplash PCR: Repeated rounds of heating and cooling allow for the execution of a finite state transition machine with state transitions $a_i \Rightarrow b_i$ encoded by the DNA strand sequence. Each of the state transitions is executed in the cool stages, where if the current state is a_i , the 3' end of the DNA strand has the complement at its 3' end, which hybridizes to a sequence a_i^* and the state transition via polymerization extension step at the 3' end, allowing a transition to a state b_i . The heating stage allows for displacement of the extended 3' end.



parallel with other molecular computing devices). In contrast, most methods for autonomous molecular computation (such as those based on the self-assembly of tiles) provide only a capability for distributed parallel molecular computation since to execute a computation they require multiple distinct molecules that interact to execute steps of each computation.

9.3 Isothermal and Autonomous PCR Detection and Whiplash PCR Computation Protocols

Neither the original PCR protocol nor the Whiplash PCR of Sakamoto et al. (1998) executes autonomously – they require thermal cycling for each step of their protocols. Walker et al. (1992a, b) developed isothermal (requiring no thermal cycling) methods for PCR known as strand displacement amplification (SDA) in which strands displaced from DNA polymerase are used for the further stages of the amplification reaction. Reif and Majumder recently developed (Reif and Majumder 2009) an autonomously executing version of Whiplash PCR (known as isothermal reactivating Whiplash PCR) that makes use of a strand-displacing polymerization enzyme (recall, however, that certain polymerase enzymes such as phi-29 can, as a side effect of their polymerization reaction, displace previously hybridized strands) with techniques to allow the reaction to process isothermally. In summary, an isothermal variant (strand-displacement PCR) of the basic PCR detection protocol provided insight on how to design an autonomous method for DNA computation. Like Whiplash PCR, this new isothermal reactivating Whiplash PCR provides for local molecular computation.

9.4 Autonomous Molecular Cascades and Related Hybridization Reactions for Autonomous DNA Computation

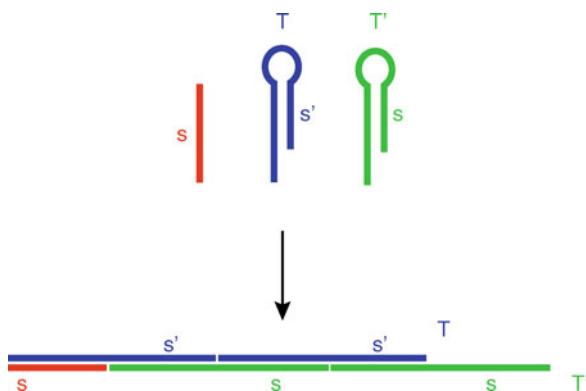
9.4.1 Autonomous Molecular Cascades for DNA Detection

Dirks and Pierce (2004) demonstrated an isothermal, enzyme-free (most known detection protocols require the use of protein enzymes) method for highly sensitive detection of a particular DNA strand. This protocol makes a triggered amplification by the hybridization chain reaction briefly illustrated in Fig. 20.

The protocol made use of multiple copies of two distinct DNA nanostructures T and T' that are initially added to a test tube. When ssDNA sequence S is added to the test tube, S initially has a hybridization reaction with a part of T, thus exposing a second ssDNA S' that had

Fig. 20

Autonomous molecular cascade for signal amplification.



been previously hidden within the nanostructure of T. Next, S' has a hybridization reaction with a part of T', thus exposing a second copy of S that had been previously hidden within the nanostructure of T'. The second copy of S then repeats the process of other similar (but so far unaltered) copies of T and T', allowing a cascade effect to occur completely autonomously such autonomous molecular cascade devices have applications to a variety of medical applications, where a larger response (e.g., a cascade response) is required in response to one of the multiple molecular detection events.

9.4.2 Hybridization Reactions for Autonomous DNA Computation

Winfree (Zhang et al. 2007) developed a general methodology for designing systems of DNA molecules by the use of catalytic reactions that are driven by entropy. In particular, Zhang et al. (2007) demonstrates a general, powerful scheme for executing any Boolean circuit computation via cascades of DNA hybridization reactions. The unique common property of the above detection protocol (Dirks and Pierce 2004) and the molecular computations of Zhang et al. (2007) are their use of only hybridization, making no use of restriction enzyme or any other protein enzymes.

Following on this work, Yin et al. (2008) developed an elegant and highly descriptive labeled diagram scheme (with nodes indicating inputs, products, etc.) for illustrating the programming of biomolecular self-assembly and reaction pathways.

9.5 Autonomous Detection Protocols and Molecular Computations Using DNAzyme

In addition, Chen et al. (2004) demonstrated a novel method for DNA detection, which made use of a dual set of DNAzyme (recall a DNAzyme is a DNA molecule that possesses enzymatic activity, in particular cutting particular single-stranded DNA) that provided for amplified DNA detection. This led to the DNAzyme-based autonomous DNA walker (Tian et al. 2005) described in  Sect. 10.4.2.

10 Autonomous Molecular Transport Devices Self-assembled from DNA

10.1 Molecular Transport

Many molecular-scale tasks may require the transport of molecules and there are a number of other tasks that can be done at the molecular scale that would be considerably aided by an ability to transport within and/or along nanostructures. For example, regarding the importance of molecular transport in nanoscale systems, consider the cell, which uses protein motors fueled by ATP to do this.

10.2 Nonautonomous DNA Motor Devices

In the early 2000s a number of researchers developed and demonstrated motors composed of DNA nanostructures; for example, Yurke et al. (2000) demonstrated a DNA actuator powered by DNA hybridization (complementary pairing between DNA strands). However, all of these

DNA motor devices required some sort of externally mediated changes (such as temperature-cycling, addition or elimination of a reagent, etc.) per work cycle of the device, and so did not operate autonomously.

10.3 The Need for Autonomous Molecular Transport

Almost all of the conventionally scaled motors used by mankind run without external mediation, and almost all natural systems for molecular motors are also autonomous (e.g., the cell's protein motors are all autonomous). The practical applications of molecular devices requiring externally mediated changes per work cycle are quite limited. So it is essential to develop autonomous DNA devices that do not require external mediation while executing movements.

10.4 Autonomous DNA Walkers

Reif (2003) first described the challenge of autonomous molecular transport devices which he called "DNA walkers" that traversed DNA nanostructures, and proposed two designs that gave bidirectional movement. In 2004, Sherman and Seeman demonstrated a DNA walker (Sherman and Seeman 2004), but it was nonautonomous since it required external mediation for every step it made.

10.4.1 Restriction Enzyme-Based Autonomous DNA Walkers

The first autonomous DNA walker was experimentally demonstrated in 2004 by Yin et al. (2004a, b). It employed restriction enzymes and ligase; see Yin et al. (2005) for its detailed general design (☞ Fig. 21).

The device is described in ☞ Fig. 19.

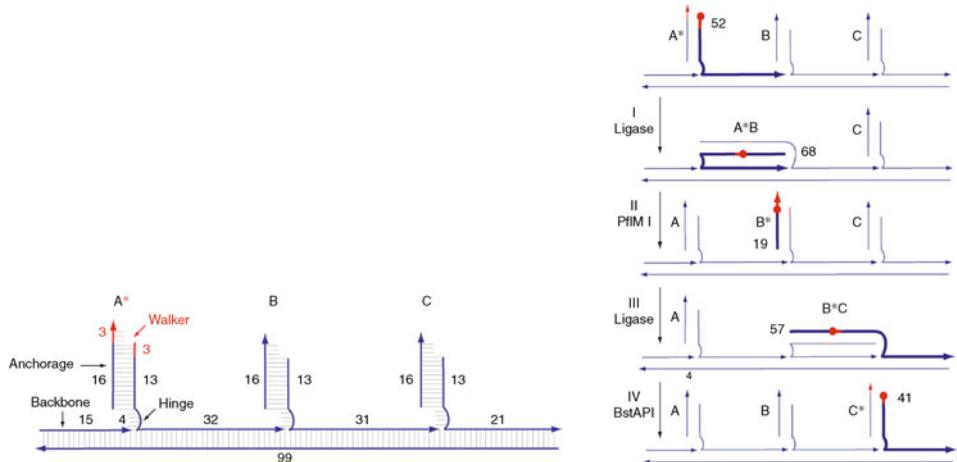
- Initially, a linear DNA nanostructure (the "road") with a series of attached ssDNA strands (the "steps") is self-assembled. Also, a fixed-length segment of DNA helix (the "walker") with short sticky ends (its "feet") are hybridized to the first two steps of the road.
- Then the walker proceeds to make a sequential movement along the road, where at the start of each step the feet of the walker are hybridized to two further consecutive steps of the road.
- Then a restriction enzyme cuts the DNA helix where the backward foot is attached, exposing a new sticky end forming a new replacement foot that can hybridize to the next step that is free, which can be the step just after the step where the other foot is currently attached. A somewhat complex combinatorial design for the sequences composing the steps and the walker ensures that there is unidirectional motion forward, along the road.

10.4.2 DNAzyme-Based Autonomous DNA Walkers

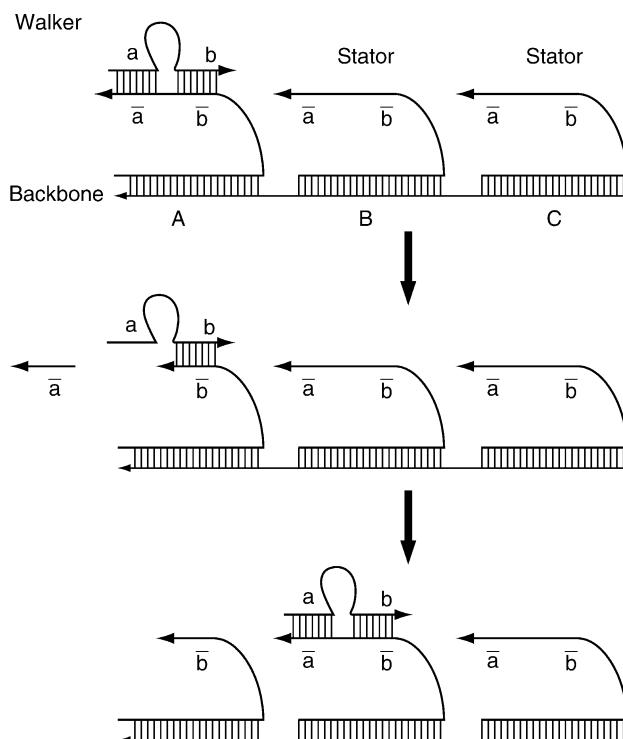
Subsequently in 2005 Mao's group (Tian et al. 2005) demonstrated an autonomous DNA walker that made use of a DNAzyme motor (Chen et al. 2004) that used the cuts provided by the enzymatic activity of DNAzyme to progress along a DNA nanostructure (☞ Fig. 22).

Fig. 21

Autonomous molecular transport devices self-assembled from DNA.

**Fig. 22**

Mao's DNAzyme walker.



Bath and Turberfield (2007) also give an extensive survey of these and further recent DNA motor and walker devices.

10.5 Programmable Autonomous DNA Devices: Nanobots

There are some important applications of these autonomous DNA walkers including transport of molecules within large self-assembled DNA nanostructures. However, the potential applications may be vastly increased if they can be made to execute computations while moving along a DNA nanostructure. This would allow them, for example, to make programmable changes to their state and to make movements programmable. Such programmable autonomous DNA walker devices will be called “programmable DNA nanobots.” Yin et al. (2006) describe an extension of the design of the restriction enzyme-based autonomous DNA walker (Yin et al. 2004a, b) described above in [Sect. 10.4.1](#), to allow programmed computation while moving along a DNA nanostructure.

Another DNA nanobot design (see [Fig. 23](#)) for programmed computation while moving along a DNA nanostructure was developed by Sahu and Reif (2008) using in this case an extension of the design of the DNAzyme-based autonomous DNA walker (Tian et al. 2005) also described below ([Fig. 24](#)).

It remains a challenge to experimentally demonstrate these.

11 Conclusions and Challenges

11.1 What Was Covered and What Was Missed: Further Reading

This chapter has covered most of the major known techniques and results for autonomous methods for DNA-based computation and transport.

However, there is a much larger literature of DNA-based computation that includes methods that are nonautonomous, but otherwise often ingenious and powerful. As just one notable example, Stojanovic demonstrated a deoxyribozyme-based molecular automaton (Stojanovic and Stefanovic 2003) and demonstrated its use to play the optimal strategy for a simple game.

Fig. 23

Reif and Sahu’s DNA nanobot: Figure illustrates the implementation of a state transition through DNAzymes. $D_{0,s1}$ in the transition machinery for state transition at 0 combines with input nanostructure when active input symbol encoded by the sticky end is 0. When the active input symbol encoded by the sticky end is 1, $D_{1,s1}$ in the transition machinery for state transition at 1 combines with the input nanostructure.

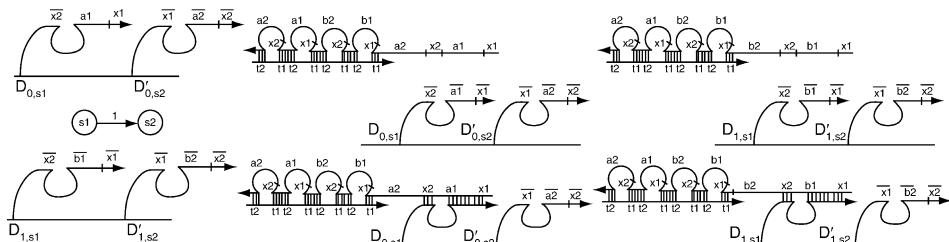
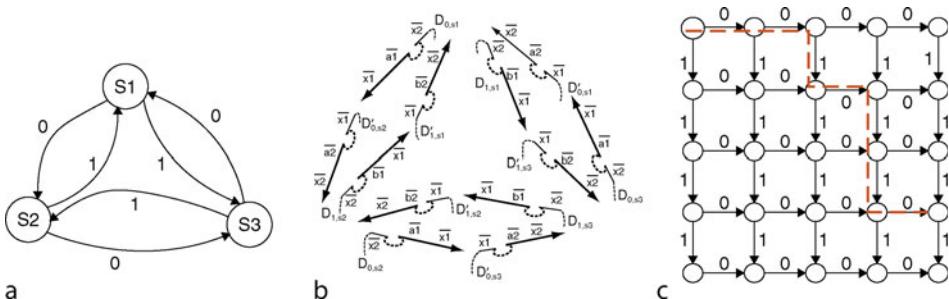


Fig. 24

Use of Reif and Sahu's DNA nanobot to execute a programmed traversal of a grid DNA nanostructure: (a) The DNAzyme implementation of the finite state machine shown on the left. (b) Illustration of programmable routing in two dimensions. (a) State transition diagram. (b) DNAzyme strands programming transitions. (c) Programmed traversal of DNA nanostructure grid.



Other excellent surveys of DNA nanostructures and devices have been given by Seeman (Sherman and Seeman 2004; Sha et al. 2005), Mao (Deng et al. 2006), de Castro (2006), LaBean and Li (2007), Yan and Liu (2006), and Bath and Turberfield (2007).

11.2 Future Challenges for Self-assembled DNA Nanostructures

There are a number of key challenges still confronting this emerging field:

Experimentally demonstrate:

1. *Complex, error-free DNA patterning to the scale of, say, at least 10,000 pixels – as required, say, for a functional molecular electronic circuit for a simple processor.*

Note: This would probably entail the use of a DNA tiling error correction method as well as a significant improvement over existing DNA patterning techniques.

2. *A programmable DNA nanobot autonomously executing a task critical to nano-assembly.*

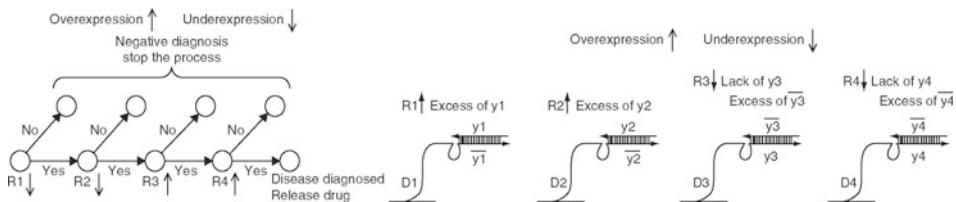
Note: The first stage might be a DNA walker that can be programmed to execute various distinct, complex traversals of a 2D DNA nanostructure, and to load and unload molecules at chosen sites on the nanostructure.

3. *An application of self-assembled DNA nanostructures to medical diagnosis.*

Adar et al. (2004) were the first to propose and to demonstrate in the test tube a finite-state computing DNA device for *medical diagnosis*: the device detects RNA levels (either over- or underexpression of particular RNA), computes a diagnosis based on a finite-state computation, and then provides an appropriate response (e.g., the controlled release of a single-stranded DNA that either promotes or interferes with expression). They demonstrated, in the test tube, a potential application of such a finite-state computing device to medical diagnosis and therapeutics. Sahu and Reif (2008) described a DNAzyme-based autonomous DNA nanobot (see **Sect. 10.4**) that can also function as a finite-state computing DNA device for medical diagnosis (**Fig. 25**).

Fig. 25

A finite-state computing DNA device for medical diagnosis based on Reif and Sahu's DNAzyme-based autonomous DNA nanobot. A state diagram for a DNAzyme doctor nanobot that controls the release of a "drug" RNA on the basis of the RNA expression tests for a disease. The figure shows the consequences of overexpression and underexpression of different RNAs on the concentrations of the respective characteristic sequences. The overexpression of $R1$ and $R2$ results in excess of y_1 and y_2 , respectively, and they block the path of the input nanostructure by hybridizing with $D1$ and $D2$. Similarly, underexpression of $R3$ and $R4$ results in excess of \bar{y}_3 and \bar{y}_4 , respectively, to block the path of input nanostructure.



It remains a challenge to apply such a finite-state computing DNA device for medical diagnosis within the cell, rather than in the test tube.

Acknowledgments

Thanks to Nikhil Gopalkrishnan, Urmi Majumder, and Sudheer Sahu for their very useful comments on this article. Supported by NSF Grants CCF-0829797 and CCF-0829798.

References

- Adar R, Benenson Y, Linshiz G, Rozner A, Tishby N, Shapiro E (2004) Stochastic computing with biomolecular automata. Proc Natl Acad Sci USA 101:9960–9965
- Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266:1021–1024
- Adleman L (Aug 1998) Computing with DNA. Sci Am 279(2):34–41
- Barish RD, Rothemund PWK, Winfree E (2005) Two computational primitives for algorithmic self-assembly: copying and counting. NanoLetters 5(12):2586–2592
- Bath J, Turberfield A (2007) DNA nanomachines. Nat Nanotechnol 2:275–284
- Chen Y, Wang M, Mao C (2004) An autonomous DNA nanomotor powered by a DNA enzyme. Angew Chem Int Ed 43:3554–3557
- de Castro LN (2006) Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications. Chapman & Hall/CRC Computer and Information Sciences
- Deng Z, Chen Y, Tian Y, Mao C (2006) A fresh look at DNA nanotechnology, In: Chen J, Jonoska N, Rozenberg G (eds) Nanotechnology: science and computation, Springer series in Natural Computing, pp 23–34
- Dirks RM, Pierce NA (2004) Triggered amplification by hybridization chain reaction. Proc Natl Acad Sci USA 101:15275–15278
- Douglas SM, Chou JJ, Shih WM (2007) DNA-nano-tube-induced alignment of membrane proteins for NMR structure determination. PNAS 104:6644–6648
- LaBean T, Li H (2007) Using DNA for construction of novel materials. Nano Today 2:26–35
- LaBean TH, Goethel KV, Reif JH (2006). Self-assembling DNA nanostructures for patterned molecular assembly, invited chapter in textbook. In: Mirkin CA,

- Niemeyer CM (eds) Nanobiotechnology. Wiley, Weinheim, Germany
- LaBean TH, Winfree E, Reif JH (1999) Experimental progress in computation by self-assembly of DNA tilings. In: Winfree E, Gifford DK (eds) DIMACS series in discrete mathematics and theoretical computer science, Proceedings of the 5th DIMACS workshop on DNA based computers, Vol 54. MIT, Cambridge, MA, pp 123–140
- LaBean TH, Yan H, Kopatsch J, Liu F, Winfree E, Reif JH, Seeman NC (2000) The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. *J Am Chem Soc (JACS)* 122:1848–1860
- Majumder U, LaBean TH, Reif JH (2007) Activatable tiles: compact, robust programmable assembly and other applications, In: Garzon M, Yan H (eds) DNA computing: DNA13. Springer, Lecture notes in computer science (LNCS), Vol 4848. Springer, Berlin, Germany, pp 15–25
- Mao C, LaBean, Reif TH, Seeman JH (Sept 28, 2000) Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature* 407:493–495
- Park SH, Pistol C, Ahn SJ, Reif JH, Lebeck AR, Dwyer C, LaBean TH (Jan 23, 2006a) Finite-size, fully addressable DNA tile lattices formed by hierarchical assembly procedures. *Angew Chem [Int Ed]* 45(5):735–739
- Park S-H, Prior MW, LaBean TH, Finkelstein G (2006b) Optimized fabrication and electrical analysis of silver nanowires templated on DNA molecules. *Appl Phys Lett* 89:033901
- Reif JH (2003) The design of autonomous DNA nano-mechanical devices: walking and rolling DNA. In: Hagiya M, Ohuchi A (eds) DNA based computers (DNA8), Sapporo, Japan, June 10–13, 2002. Lecture notes in computer science, No. 2568, Springer, New York, pp 22–37. Published in *Natural Computing*, DNA8 special issue, Vol 2, pp 439–461
- Reif JH, Majumder U (2009) Isothermal reactivating whiplash PCR for locally programmable molecular computation, Fourteenth international meeting on DNA based computers (DNA14). In: Goel A, Simmel FC (eds) Prague, Czech Republic (June, 2008). Lecture notes for computer science (LNCS), New York, NY, Springer, New York. Invited Paper, Special issue in *Natural Computing*
- Reif JH, Sahu S, Yin P (2006) Compact error-resilient computational DNA tiling assemblies. In: Chen J, Jonoska N, Rozenberg G (eds) Nanotechnology: science and computation, Springer series in *Natural Computing*, New York, pp 79–104
- Robinson BH, Seeman NC (1987) Protein Eng 1:295–300
- Rothemund PWK (March 16, 2006) Folding DNA to create nanoscale shapes and patterns. *Nature* 440:297–302
- Rothemund PWK, Papadakis N, Winfree E (Dec 2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol* 2(12)
- Sahu S, Reif JH (2008) DNA-based self-assembly and nanorobotics. VDM Verlag Dr. Mueller e. K, Saarbrücken, Germany, 128 p. ISBN-10: 363909770X, ISBN-13: 978-3639097702
- Saiki RK, Scharf S, Faloona F, Mullis KB, Horn GT, Erlich HA, Arnheim N (Dec 20, 1985) Enzymatic amplification of beta-globin genomic sequences and restriction site analysis for diagnosis of sickle cell anemia. *Science* 230(4732):1350–1354
- Sakamoto K, Kiga D, Momiya K, Gouzu H, Yokoyama S, Ikeda S, Sugiyama H, Hagiya M (1998) State transitions with molecules. In: Proceedings of the 4th DIMACS meeting on DNA based computers, held at the University of Pennsylvania, June 16–19, 1998. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society, Providence, RI
- Sha R, Zhang X, Liao S, Constantinou PE, Ding B, Wang T, Garibotti AV, Zhong H, Israel LB, Wang X, Wu G, Chakraborty B, Chen J, Zhang Y, Yan H, Shen Z, Shen W, Sa-Ardyen P, Kopatsch J, Zheng J, Lukeman PS, Sherman WB, Mao C, Jonoska N, Seeman NC (2005) Structural DNA nanotechnology: molecular construction and computation. In: Calude CS et al. (eds) UC 2005, LNCS 3699, Springer, Berlin, Germany, pp 20–31
- Shapiro E, Benenson Y (May 2006) Bringing DNA computers to life. *Sci Am* 45–51
- Sherman WB, Seeman NC (2004) A precisely controlled DNA biped walking device. *Nano Lett Am Chem Soc* 4:1203
- Stojanovic MN, Stefanovic D (2003) A deoxyribozyme-based molecular automaton. *Nat Biotechnol* 21:1069–1074
- Tian Y, He Y, Chen Y, Yin P, Mao C (2005) Molecular devices – a DNAzyme that walks processively and autonomously along a one-dimensional track. *Angew Chem Int Ed* 44:4355–4358
- Walker GT, Fraiser MS, Schram JL, Little MC, Nadeau JG, Malinowski DP (Apr 1992a) Strand displacement amplification – an isothermal, in vitro DNA amplification technique. *Nucleic Acids Res* 20(7):1691–1696
- Walker GT, Little MC, Nadeau JG, Shank DD (Jan 1992b) Isothermal in vitro amplification of DNA by a restriction enzyme/DNA polymerase system. *Proc Natl Acad Sci USA* 89(1):392–396
- Winfree E, Bekbolatov R (2004) Proofreading tile sets: error correction for algorithmic self-assembly, in *DNA computers 9. LNCS (2943):126–144*
- Winfree E, Yang X, Seeman NC (1998) Universal computation via self-assembly of DNA: some theory and

- experiments. In: DNA based computers II, American Mathematical Society, Providence, RI, pp 191–213
- Yan H, Liu Y (2006) DNA nanotechnology: an evolving field, invited chapter in nanotechnology: science and computation. In: Chen J, Jonoska N, Rozenberg G (eds) Natural computing series. Springer, Berlin, Germany, pp 35–53
- Yan H, Feng L, LaBean TH, Reif J (2003a) DNA nanotubes, parallel molecular computations of pairwise exclusive-or (XOR) using DNA “String Tile”. Self-Assembly J Am Chem Soc (JACS) 125(47): 14246–14247
- Yan H, LaBean TH, Feng L, Reif JH (2003b) Directed nucleation assembly of barcode patterned DNA lattices. PNAS 100(14):8103–8108
- Yan H, Park SH, Finkelstein G, Reif JH, LaBean TH (Sept 26, 2003c) DNA-templated self-assembly of protein arrays and highly conductive nanowires. Science 301:1882–1884
- Yin P, Choi HMT, Calvert CR, Pierce NA (2008) Programming biomolecular self-assembly pathways. Nature 451:318–322
- Yin P, Guo B, Belmore C, Palmeri W, Winfree E, LaBean TH, Reif JH (June 7–10, 2004a) TileSoft: sequence optimization software for designing DNA secondary structures, abstract, preliminary proceedings, Tenth international meeting on DNA based computers (DNA10), Milan, Italy
- Yin P, Turberfield AJ, Sahu S, Reif JH (2005) Designs for autonomous unidirectional walking DNA devices, tenth international meeting on DNA based computers (DNA10). In: Ferretti C, Mauri G, Zandron C (eds) Milan, Italy, June 7–10, 2004. Lecture notes in computer science, Vol 3384. Springer, New York, pp 410–425
- Yin P, Turberfield AJ, Reif JH (2006) Design of autonomous DNA cellular automata, Eleventh international meeting on DNA based computers (DNA11). In: Carbone A, Pierce N (eds) London, UK (June 2005). Springer Lecture notes in computer science (LNCS), New York, NY, Vol 3892, Springer, New York, pp 399–416
- Yin P, Yan H, Daniel XG, Turberfield AJ, Reif JH (Sept 20, 2004b) A unidirectional DNA walker moving autonomously along a linear track, Angew Chem [Int Ed], 43(37):4906–4911
- Yurke B, Turberfield AJ, Mills AP Jr, Simmel FC, Neumann JL (2000) A DNA-fuelled molecular machine made of DNA. Nature 406:605–608
- Zhang DY, Turberfield AJ, Yurke B, Winfree E (2007) Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318:1121–1125

40 Membrane Computing

Gheorghe Păun

Institute of Mathematics of the Romanian Academy, Bucharest, Romania

Department of Computer Science and Artificial Intelligence, University
of Seville, Spain

gpaun@us.es

george.paun@imar.ro

1	<i>Introduction</i>	1356
2	<i>Cell-Like P Systems</i>	1357
3	<i>Tissue-Like P Systems</i>	1369
4	<i>Spiking Neural P Systems</i>	1371
5	<i>Computing Power</i>	1373
6	<i>Computational Efficiency</i>	1374
7	<i>Applications</i>	1375
8	<i>Closing Remarks</i>	1376

Abstract

This chapter introduces the basic ideas, results, and applications of membrane computing, a branch of natural computing inspired by the structure and the functioning of the living cell, as well as by the cooperation of cells in tissues, colonies of cells, and neural nets.

1 Introduction

Membrane computing is a branch of natural computing introduced by Păun (2000). Its main goal is to abstract computing models from the architecture and the functioning of living cells, as well as from the organization of cells in tissues, organs (brain included) or other higher-order structures such as colonies of cells (e.g., bacteria). The initial goal was to learn from cell biology something possibly useful to computer science, and the area quickly developed in this direction. Several classes of computing models, called P systems, were defined in this context and mainly investigated from mathematical and computer science points of view. Soon, other applications appeared, first in biology, then also in linguistics, computer science, economics, approximate optimization, etc. This chapter will introduce the main classes of P systems, with simple examples illustrating their structure and functioning, and then will briefly discuss some types of results and applications.

Consider that a cell is a hierarchical structure of compartments defined by membranes, with solutions of chemicals swimming in compartments and proteins bound on membranes, which also define selective channels among compartments. The computing model abstracted in this context is a distributed one, with (virtual) membranes defining its structure, with “reactions” taking place in compartments, and with a precise way of communicating among compartments. Thus, the main ingredients of a P system are (1) the membrane structure, (2) the multisets of objects placed in the compartments of the membrane structure, and (3) the rules for processing the objects and the membranes. The rules are used for changing the current configuration of the device and in this way we get computations, sequences of transitions among configurations starting with an initial configuration and, when halting, providing a result of the computation. Many classes of P systems have been defined, with biological, mathematical, or computer science motivation; these variants concern the arrangement of membranes, the form of rules, the way of using the rules, the way of defining the result of a computation, and so on. The main classes of P systems (cell-like, tissue-like, neural-like) will be discussed below.

Most of these classes of P systems are rather powerful from a computational point of view. They are Turing complete, even when using ingredients of a reduced complexity – a small number of membranes, rules of simple forms, ways of controlling the use of rules directly inspired from biology. Certain classes of P systems are also efficient. They can be used for solving computationally hard problems – typically NP-complete problems, in a feasible time, which are typically polynomial.

Membrane computing proved to be rather appealing as a modeling framework, especially for handling discrete (biological or biomedical, economic, etc.) processes. They have many attractive features: easy understandability, scalability and programmability, inherent compartmentalization, etc.

Membrane computing is a young research area, but already its bibliography is rather large. A comprehensive presentation can be found in the 2002 monograph (Păun 2002), with several

applications presented in Ciobanu et al. (2006), where a friendly introduction to membrane computing is given in the first chapter. An up-to-date coverage of membrane computing can be found in a recent handbook (Păun et al. 2009). The complete bibliography of the domain, which at the end of 2008 totaled around 1,250 titles, can be found on the P Systems website at www.ppage.psystems.eu. In general, the reader can refer to these bibliographical sources, as this chapter only specifies a few references.

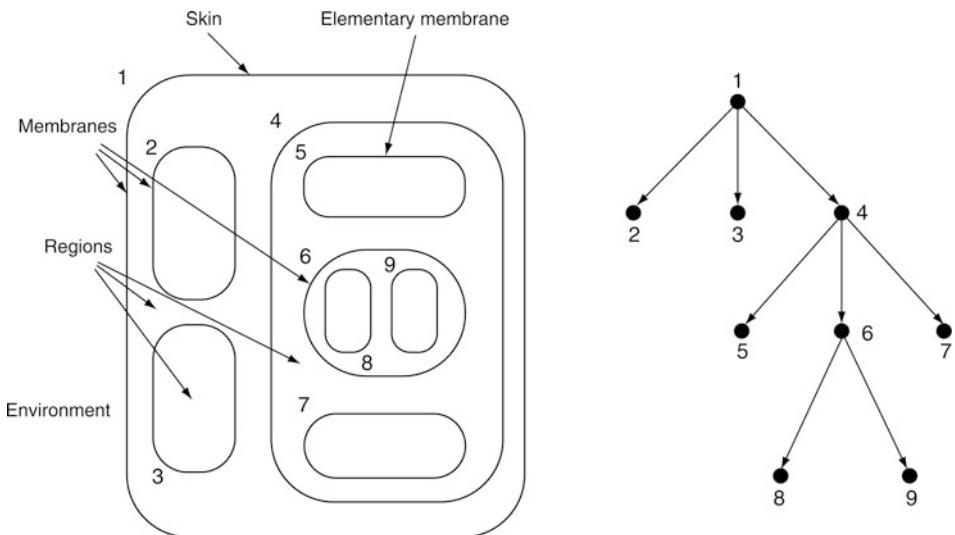
2 Cell-Like P Systems

In this section we discuss in some detail the first introduced and most investigated type of P system, inspired by the structure and functioning of the biological cell. Because the ingredients used in this model are also present, with variations, in other types of P systems, we discuss separately the main ingredients: membrane structure, types of rules, ways of using the rules and of defining the results.

2.1 Membrane Structure

One starts from the observation that the (eukaryotic) cell is compartmentalized by means of membranes, so that one considers hierarchical arrangements of membranes as suggested (in a 2D representation) in the left side of Fig. 1. Here we distinguish an external membrane (corresponding to the plasma membrane and usually called the *skin* membrane) as well as several membranes placed inside the skin membrane (corresponding to the membranes present in a cell, around the nucleus, in Golgi apparatus, vesicles, mitochondria, etc.). A membrane without any other membrane inside it is said to be *elementary*. Each membrane determines a compartment, called a *region*, which is the space delimited by it from above and

Fig. 1
A membrane structure and its tree representation.



from below by the membranes placed directly inside, if any exist. Clearly, the correspondence between the membrane and region is one-to-one; that is why the terms are sometimes used interchangeably.

Usually, the membranes are identified by *labels* from a given set. In [Fig. 1](#), we use numbers, starting with number 1 assigned to the skin membrane. (This is the standard labeling, but the labels can be more informative “names” associated with the membranes.) Also, in the figure the labels are assigned in a one-to-one manner to membranes, but this is possible only in the case of membrane structures that cannot grow (indefinitely); otherwise, several membranes would have the same label (such cases will be seen later). Due to the membrane–region correspondence, we identify by the same label a membrane and its associated region.

This hierarchical structure of membranes can be represented by a rooted unordered tree, with labeled nodes; the tree that describes the membrane structure in [Fig. 1](#) is given in the right-hand side of the same figure. The root of the tree is associated with the skin membrane and the leaves are associated with the elementary membranes.

Directly suggested by the tree representation is the symbolic representation of a membrane structure, by strings of labeled matching parentheses. For instance, a string corresponding to the structure from [Fig. 1](#) is the following:

$$[1 [2]_2 [3]_3 [4 [5]_5 [6 [8]_8 [9]_9]_6 [7]_7]_4]_1 \quad (*)$$

Because the tree is not ordered (and also corresponds to a biological reality), the membranes of the same level can float around. Therefore, by swapping two subexpressions placed at the same level, we get an expression that represents the same membrane structure. For instance, in the previous case, the expression

$$[1 [3]_3 [4 [6 [8]_8 [9]_9]_6 [7]_7 [5]_5]_4 [2]_2]_1$$

is a representation of the same membrane structure, equivalent to [\(*\)](#).

2.2 Multisets

In the compartments of a cell, there are various chemicals swimming in water (at this stage, we ignore the chemicals, mainly proteins, bound on membranes). Because in many cases for the biochemistry taking place in a cell “numbers matter”, the natural data structure to use in this framework is the *multiset*, the set with multiplicities associated with its elements. For this reason, some basic elements related to multisets and to their representation by strings, which is standard in membrane computing, are discussed here.

A *multiset* over a given set U is a mapping $M : U \rightarrow \mathbb{N}$, where \mathbb{N} is the set of nonnegative integers. For $a \in U$, $M(a)$ is the multiplicity of a in M . If the set U is finite, $U = \{a_1, \dots, a_n\}$, then the multiset M can be explicitly given in the form $\{(a_1, M(a_1)), \dots, (a_n, M(a_n))\}$, thus specifying for each element of U its multiplicity in M . In membrane computing, the usual way to represent a multiset $M = \{(a_1, M(a_1)), \dots, (a_n, M(a_n))\}$ over a finite set $U = \{a_1, \dots, a_n\}$ is by using strings: $w = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)}$ and all permutations of w represent M ; the empty multiset is represented by λ , the empty string. The total multiplicity of elements of a multiset (this is also called the *weight* of the multiset) clearly corresponds to the length of a string representing it. This compact representation is so frequent in membrane computing that we can usually say “the multiset w ” instead of “the multiset represented by string w .”

A few basic notions about multisets are useful in membrane computing, but they are not discussed here; the reader can refer to Calude et al. (2001) for details.

2.3 Evolution Rules

The main way the chemicals present in the compartments of a cell evolve is by means of reactions that consume certain chemicals and produce other chemicals. In what follows, the chemicals are considered to be unstructured and hence are described by symbols from a given alphabet; these symbols are called *objects*. Later, structured objects described by strings will also be considered.

Thus, in each compartment of a membrane structure, we consider a multiset of objects over a given alphabet. Corresponding to the biochemical reactions, the main type of rules for evolving these multisets are multiset rewriting rules (simply, evolution rules). Later, rules for moving objects across membranes, called communication rules, as well as rules for handling membranes will also be considered.

The multiset-rewriting rules are of the form $u \rightarrow v$, where u and v are multisets of objects. An example is $aab \rightarrow abcc$ (remember that the multisets are represented by strings). However, in order to make the compartments cooperate, objects have to be moved across membranes, and for this we add *target indications* to the objects produced by a rule as above (to the objects from multiset v). These indications are *here*, *in*, and *out*, with the meanings that an object associated with the indication *here* remains in the same region, one associated with the indication *in* goes immediately into an adjacent lower membrane that is nondeterministically chosen, and *out* indicates that the object has to exit the membrane, thus becoming an element of the region surrounding it. For instance, we can have $aab \rightarrow (a, \text{here})(b, \text{out})(c, \text{here})(c, \text{in})$. Using this rule in a given region of a membrane structure means to consume two copies of a and one of b (they are removed from the multiset of that region), and one copy of a , one of b , and two of c are produced; the resulting copy of a remains in the same region, and the same happens with one copy of c (indications *here*), while the new copy of b exits the membrane, going to the surrounding region (indication *out*), and one of the new copies of c enters one of the child membranes, nondeterministically chosen. If no such child membrane exists, that is, the membrane with which the rule is associated is elementary, then the indication *in* cannot be followed, and the rule cannot be applied. In turn, if the rule is applied in the skin region, then b will exit into the environment of the system (and it is “lost” there, since it can never come back). In general, the indication *here* is not specified when giving a rule.

The evolution rules are classified according to the complexity of their left-hand side. A rule with at least two objects in its left-hand side is said to be *cooperative*; a particular case is that of *catalytic* rules of the form $ca \rightarrow cv$, where c is an object (called catalyst), which assists the object a to evolve into the multiset v ; rules of the form $a \rightarrow v$, where a is an object, are called *noncooperative*.

The rules can also contain other ingredients or indications about their use. For instance, a rule of the form $u \rightarrow v\emptyset$ entails the dissolution of the membrane where it is used: if the rule is applied, then the corresponding membrane disappears and its contents, object and membranes alike, are left free in the surrounding membrane; the rules of the dissolved membrane disappear with the membrane. The skin membrane is never dissolved.

The rules can also be of the form $u \rightarrow v|_p$, where p is an object; it is said that p is a *promoter* and the rule can be applied only if p is present in the region where the rule is applied.

Similarly, we have rules with *inhibitors*: $u \rightarrow v|_{\neg p}$ can be applied only if the object p is not present in the region.

Very important are also the communication rules and the rules for evolving the membrane structure, but, before introducing them we discuss another central feature, the way of using the rules.

2.4 Ways of Using the Rules

The most investigated way of applying the rules in the regions of a P system is described by the phrase: *in the maximally parallel manner, nondeterministically choosing the rules and the objects*.

This means the following: we assign objects to rules, nondeterministically choosing the objects and the rules until no further assignment is possible. Mathematically stated, we look to the *set* of rules and try to find a *multiset* of rules by assigning multiplicities to rules with two properties: (1) the multiset of rules is *applicable* to the multiset of objects available in the respective region; that is, there are enough objects to apply the rules a number of times as indicated by their multiplicities; and (2) the multiset is *maximal*, that is, no further rule can be added to it (no multiplicity of a rule can be increased) such that the obtained multiset is still applicable.

Thus, an evolutionary step in a given region consists of finding a maximal applicable multiset of rules, removing from the region all objects specified in the left-hand sides of the chosen rules (with multiplicities as indicated by the rules and by the number of times each rule is used), producing the objects from the right-hand sides of the rules, and then distributing these objects as indicated by the targets associated with them. If at least one of the rules introduces the dissolving action δ , then the membrane is dissolved and its contents become part of the parent membrane, provided that this membrane was not dissolved at the same time; otherwise we stop at the first upper membrane that was not dissolved (the skin membrane at least remains intact).

The maximal parallelism both corresponds to the apparent simultaneity of reactions taking place in a cell and it is also very useful from a mathematical point of view, but there are several other possibilities, in general closer to the biological reality. *Partial parallelism* is one of them: in each step, one uses a specified number of rules, or at least/at most a specified number of rules. In the limit, we have the *sequential* use of rules: only one rule is used in each region, nondeterministically chosen.

An interesting case is that of *minimal parallelism*. The intuition is that each region that *can* evolve (using at least one rule) *must* evolve (by means of at least one rule). More specifically, we start by nondeterministically choosing one rule from each region where a rule can be used; this combination of rules or any applicable supermultiset of it can be used in evolving the system. Note that in comparison with the maximal parallelism, a considerable degree of nondeterminism is possible, as many choices of multisets of rules can be applied (in particular, any maximal multiset).

2.5 Computations and Results of Computations

A membrane structure and the multisets of objects from its regions identify a *configuration* of a P system. Note that both objects and rules are associated with regions; they are *localized*.

By using the rules in one of the ways suggested above, we pass to another configuration; such a step is called a *transition*. Note that in all cases considered in the previous section, the evolution of the system is synchronized; a global clock exists that marks the time uniformly for all regions and all regions evolve in each time unit. A sequence of transitions constitutes a *computation*. A computation is successful if it halts – it reaches a configuration where no rule can be applied to the existing objects. With a halting computation we can associate a *result* in various ways. One possibility is to count the objects present in the halting configuration in a specified elementary membrane; this is called the *internal output*. We can also count the objects that leave the system during the computation; this is called the *external output*. In both cases, the result is a number. If we distinguish among different objects, then we can have as the result a vector of natural numbers. The objects that leave the system can also be arranged in a sequence according to the moments when they exit the skin membrane, and in this case the result is a string.

Total halting as mentioned above is only one of the ways of defining successful computations and the most investigated in membrane computing (also because it is similar to the way Turing machines provide a result). Another possibility is *partial halting*: The computation stops when at least one region of the system halts; it cannot use any of its rules. Then, we can also use *signals* of various kinds for marking the step when a result is provided by a computation; for instance, if a specified object exits the system, then we count the number of objects in a given membrane in order to get a result.

An important issue is the way we use the system. In the previous discussion, it was assumed that one can start from an initial configuration and proceed through transitions (according to a given way of using the rules). Because of the nondeterminism of the application of rules, one can get several successful computations, and hence several results. This is the *generative* way of using a P system, and in this way a P system *computes* (or *generates*) a set of numbers, or a set of vectors of numbers, or a language. Another important possibility is the *accepting* case: we start from the initial configuration, where some objects codify an input, and that input is accepted if and only if the computation halts (or a specified object, say *yes*, is sent to the environment). A more general case is that of *computing* a function: We start with the argument introduced in the initial configuration and the value of the function is obtained at the end of the computation.

2.6 A Formal Definition and an Example

We now have all the prerequisites for formally defining a cell-like P system. Such a system is a construct

$$\Pi = (O, \mu, w_1, \dots, w_m, R_1, \dots, R_m, i_{in}, i_{out})$$

where O is the alphabet of objects, μ is the membrane structure (with m membranes), given as an expression of labeled parentheses, w_1, \dots, w_m are (strings over O representing) multisets of objects present in the m regions of μ at the beginning of a computation, R_1, \dots, R_m are finite sets of evolution rules associated with the regions of μ , and i_{in}, i_{out} are the labels of input and output membranes, respectively. If the system is used in the generative mode, then i_{in} is omitted; if the system is used in the accepting mode, then i_{out} is omitted. If the system is a catalytic one, then a subset C of O is specified, containing the catalysts. The number m of membranes in μ is called the *degree* of Π .

In the generative case, the set of numbers computed by Π when using the rules in mode *mode*, one of those mentioned in [Sect. 2.4](#), is denoted by $N_{\text{mode}}(\Pi)$; in what follows, the rules are always used in the maximally parallel way, hence the subscript is omitted. The family of all sets $N(\Pi)$ computed by systems Π of degree at most $m \geq 1$ and using rules of *types-of-rules* forms is denoted by $NOP_m(\text{types-of-rules})$; if there is no bound on the degree of systems, then the subscript m is replaced with $*$.

The architecture and the functioning of a cell-like P system is illustrated with a simple example of a computing P system. Formally, it is given as follows:

$$\Pi = (O, C, \mu, w_1, w_2, R_1, R_2, i_o)$$

where

$$O = \{a, b_1, b'_1, b_2, c, e\} \text{ (the set of objects)}$$

$$C = \{c\} \text{ (the set of catalysts)}$$

$$\mu = [\begin{smallmatrix} 1 & [\begin{smallmatrix} 2 & 2 \end{smallmatrix}]_2 \end{smallmatrix}]_1 \text{ (membrane structure)}$$

$$w_1 = c \text{ (initial objects in region 1)}$$

$$w_2 = \lambda \text{ (initial objects in region 2)}$$

$$R_1 = \{a \rightarrow b_1 b_2, cb_1 \rightarrow cb'_1, b_2 \rightarrow b_2 e_{\text{in}}|_{b_1}\} \text{ (rules in region 1)}$$

$$R_2 = \emptyset \text{ (rules in region 2)}$$

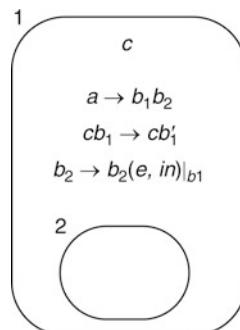
$$i_o = 2 \text{ (the output region)}$$

[Fig. 2](#) indicates the initial configuration (the rules included) of the system Π . It computes a function, namely $n \rightarrow n^2$, for any natural number $n \geq 1$. Besides catalytic and non-cooperating rules, the system also contains a rule with promoters, $b_2 \rightarrow b_2(e, in)|_{b_1}$: the object b_2 evolves to b_2e only if at least one copy of object b_1 is present in the same region.

We start with only one object in the system, the catalyst c . If we want to compute the square of a number n , then we have to input n copies of the object a in the skin region of the system. In that moment, the system starts working by using the rule $a \rightarrow b_1 b_2$, which has to be applied in parallel to all copies of a ; hence, in one step, all objects a are replaced by n copies of b_1 and n copies of b_2 . From now on, the other two rules from region 1 can be used. The catalytic rule $cb_1 \rightarrow cb'_1$ can be used only once in each step, because the catalyst is present in only one copy. This means that in each step one copy of b_1 gets primed. Simultaneously (because of the

Fig. 2

A P system with catalysts and promoters.



maximal parallelism), the rule $b_2 \rightarrow b_2(e, in)|_{b_1}$ should be applied as many times as possible — n times, because we have n copies of b_2 . Note the important difference between the promoter b_1 , which allows us to use the rule $b_2 \rightarrow b_2(e, in)|_{b_1}$, and the catalyst c : the catalyst is involved in the rule and is counted when applying the rule, while the promoter makes possible the use of the rule but it is not counted; the same (copy of an) object can promote any number of rules. Moreover, the promoter can evolve at the same time by means of another rule (the catalyst is never changed).

In this way, in each step we change one b_1 to b'_1 and we produce n copies of e (one for each copy of b_2); the copies of e are sent to membrane 2 (the indication in from the rule $b_2 \rightarrow b_2(e, in)|_{b_1}$). The computation should continue as long as there are applicable rules. This means exactly n steps: in n steps, the rule $cb_1 \rightarrow cb'_1$ will exhaust the objects b_1 and in this way neither this rule can be applied, nor $b_2 \rightarrow b_2(e, in)|_{b_1}$, because its promoter no longer exists. Consequently, in membrane 2, considered as the output membrane, we get n^2 copies of object e .

Note that the computation is deterministic, always the next configuration of the system is unique, and that, changing the rule $b_2 \rightarrow b_2(e, in)|_{b_1}$ with $b_2 \rightarrow b_2(e, out)|_{b_1}$, the n^2 copies of e will be sent to the environment; hence we can read the result of the computation outside the system, and in this case membrane 2 is useless.

2.7 Symport and Antiport Rules

An important part of the cell activity is related to the passage of substances through membranes, and one of the most interesting ways to handle this transmembrane communication is by coupling molecules as suggested by cell biology. The process by which two molecules pass together across a membrane (through a specific protein channel) is called *symport*; when the two molecules pass simultaneously through a protein channel in opposite directions the process is called *antiport*.

These operations can be formalized by considering symport rules of the form (x, in) and (x, out) , and antiport rules of the form $(z, out; w, in)$, where x, z , and w are multisets of arbitrary size; one says that the length of x , denoted by $|x|$, is the *weight* of the symport rule, and $\max(|z|, |w|)$ is the *weight* of the antiport rule. Thus, we obtain the following important class of P systems.

A P system with symport/antiport rules is a construct of the form

$$\Pi = (O, \mu, w_1, \dots, w_m, E, R_1, \dots, R_m, i_{in}, i_{out})$$

where all components $O, \mu, w_1, \dots, w_m, i_{in}, i_{out}$ are as in a P system with multiset rewriting rules, $E \subseteq O$, and R_1, \dots, R_m are finite sets of symport/antiport rules associated with the m membranes of μ . The objects of E are supposed to be present in the environment of the system with an arbitrary multiplicity. (Note that the symport/antiport rules do not change the number of objects, but only their place, which is why we need a supply of objects in the environment; this supply is inexhaustible, that is, it does not matter how many objects are introduced in the system because arbitrarily many still remain in the environment.)

As above, the rules can be used in various ways, but here we consider again the nondeterministic maximally parallel manner. Transitions, computations, and halting computations will be defined in the usual way. The number of objects present in region i_{out} in the halting configuration is said to be computed by the system by means of that computation; the set of all numbers computed in this way by Π is denoted by $N(\Pi)$. The family of sets $N(\Pi)$ computed

by systems Π of degree at most $m \geq 1$, using symport rules of degree at most $p \geq 0$ and antiport rules of degree at most $q \geq 0$ (the cases $p = 0$ or $q = 0$ correspond to not having symport or antiport rules, respectively), is denoted by $NOP_m(\text{sym}_p, \text{anti}_q)$.

The definition of P systems is illustrated with symport/antiport with an example of more general interest: a construction for simulating a *register machine* is given. In this way, one of the widely used proof techniques for the universality results in this area is also introduced.

A (nondeterministic) *register machine* is a device $M = (m, B, l_0, l_h, R)$, where $m \geq 1$ is the number of registers, B is the (finite) set of instruction labels, l_0 is the initial label, l_h is the halting label, and R is the finite set of instructions labeled (and hence uniquely identified) by elements from B . The labeled instructions are of the following forms:

- $l_1 : (\text{add}(r), l_2, l_3)$, $1 \leq r \leq m$ (add 1 to register r and go nondeterministically to one of the instructions with labels l_2, l_3),
- $l_1 : (\text{sub}(r), l_2, l_3)$, $1 \leq r \leq m$ (if register r is not empty, then subtract 1 from it and go to the instruction with label l_2 , otherwise go to the instruction with label l_3).

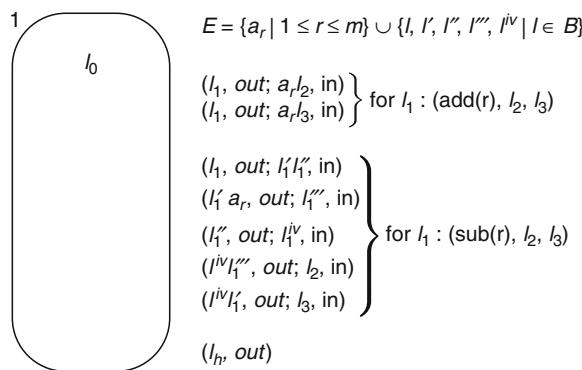
A register machine generates a set of natural numbers in the following manner: we start computing with all m registers empty, with the instruction labeled l_0 ; if the label l_h is reached, then the computation *halts* and the value of register 1 is the number generated by the computation (all other registers can be assumed to be empty at that time). The set of all natural numbers generated in this way by M is denoted by $N(M)$. It is known that nondeterministic register machines (with three registers) can compute any set of Turing computable sets of natural numbers.

A register machine can be easily simulated by a P system with symport/antiport rules. The idea is illustrated in Fig. 3, where we have represented the initial configuration of the system, the rules associated with the unique membrane, and the set E of objects present in the environment.

The value of each register r is represented by the multiplicity of object $a_r, 1 \leq r \leq m$, in the unique membrane of the system. The labels from B , as well as their primed versions, are also objects of our system. We start with the unique object l_0 present in the system. In the presence of a label object l_1 we can simulate the corresponding instruction $l_1 : (\text{add}(r), l_2, l_3)$ or $l_1 : (\text{sub}(r), l_2, l_3)$.

Fig. 3

An example of symport/antiport P system.



The simulation of an `add` instruction is clear, so we discuss only a `sub` instruction. The object l_1 exits the system in exchange for the two objects $l'_1 l''_1$ (rule $(l_1, out; l'_1 l''_1, in)$). In the next step, if any copy of a_r is present in the system, then l'_1 has to exit (rule $(l'_1 a_r, out; l'''_1, in)$), thus diminishing the number of copies of a_r by one, and bringing inside the object l'''_1 ; if no copy of a_r is present, which corresponds to the case when the register r is empty, then the object l'_1 remains inside. Simultaneously, rule $(l''_1, out; l''_1, in)$ is used, bringing inside the “checker” l''_1 . Depending on what this object finds in the system, either l'''_1 or l'_1 , it introduces the label l_2 or l_3 , respectively, which corresponds to the correct continuation of the computation of the register machine.

When the object l_h is introduced, it is expelled into the environment and the computation stops. Clearly, the (halting) computations in Π directly correspond to (halting) computations in M ; hence $N(M) = N(\Pi)$.

2.8 P Systems with Active Membranes

In the systems discussed above, the membrane structure is static, the membranes can be at most dissolved, which is not quite similar to what happens in biology, where membranes can be created or divided, and operations of exocytosis, endocytosis, phagocytosis change the membrane structure. In this section only the operation of membrane division is considered – actually, a case when the membranes play a direct part in the rules is considered.

This leads to a very important class of P systems, the main one used in devising polynomial solutions to computationally hard problems, namely the class of *P systems with active membranes*, which are constructs of the form

$$\Pi = (O, H, \mu, w_1, \dots, w_m, R)$$

where O, w_1, \dots, w_m are as in a P system with multiset rewriting rules or with symport/antiport rules, H is a finite set of *labels* for membranes, μ is a membrane structure of degree $m \geq 1$, with polarizations associated with membranes, that is “electrical charges” $\{+, -, 0\}$, and R is a finite set of rules, of the following forms:

- (a) $[_h a \rightarrow v]_h^e$, for $h \in H, e \in \{+, -, 0\}, a \in O, v \in O^*$ (object evolution rules);
- (b) $a[_h]_h^{e_1} \rightarrow [_h b]_h^{e_2}$, for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$ (in communication rules);
- (c) $[_h a]_h^{e_1} \rightarrow [h]_h^{e_2} b$, for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$ (out communication rules);
- (d) $[_h a]_h^e \rightarrow b$, for $h \in H, e \in \{+, -, 0\}, a, b \in O$ (dissolving rules);
- (e) $[_h a]_h^{e_1} \rightarrow [_h b]_h^{e_2} [_h c]_h^{e_3}$, for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$

(division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, and possibly of different polarizations; the object specified in the rule is replaced in the two new membranes possibly by new objects; the remaining objects are duplicated and may evolve in the same step by rules of type (a)).

Note that each rule has specified the membrane where it is applied; that is why a global set of rules R is considered. The rules are applied in a maximally parallel manner, with the following details: a membrane can be subject to only one rule of types (b)–(e) (the rules of type (a) are not considered to involve the membrane where they are applied); inside each membrane, the rules of type (a) are applied in parallel; each copy of an object is used by only one rule of any type. The rules are used in a bottom-up manner: we use first the rules of type (a), and then the rules of other types; in this way, in the case of dividing membranes, the

result of using first the rules of type (a) is duplicated in the newly obtained membranes). As usual, only halting computations give a result in the form of the number (or the vector) of objects expelled into the environment during the computation.

The set H of labels has been specified because it is possible to allow the change of membrane labels. For instance, a division rule can be of the more general form

$$(e') [_{h_1} a]^{e_1}_{h_1} \rightarrow [_{h_2} b]^{e_2}_{h_2} [_{h_3} c]^{e_3}_{h_3} \quad \text{for } h_1, h_2, h_3 \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O.$$

The change of labels can also be considered for rules of types (b) and (c). Also, we consider the possibility of dividing membranes into more than two copies, or even of dividing nonelementary membranes (in such a case, all inner membranes are duplicated in the new copies of the membrane).

P systems with active membranes can be used for computing numbers, but their main usefulness is in devising polynomial-time solutions to computationally hard problems, by a time–space trade-off. The space is created both by duplicating objects and by dividing membranes. This possibility is illustrated by the way of using membrane division for the generation of all 2^n truth assignments possible for n propositional variables; this is the basic step in solving SAT (satisfiability of propositional formulas in the conjunctive normal form) in polynomial time in this framework.

Assume that we have the variables x_1, x_2, \dots, x_n ; we construct the following system (of degree 2):

$$\begin{aligned} \Pi &= (O, \{1, 2\}, [1]_2^0[2]_1^0, \lambda, a_1 a_2 \dots a_n c_1, R) \\ O &= \{a_i, c_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{\text{check}\} \\ R &= \{[2]a_i[2]^0 \rightarrow [2]t_i[2]^0[2]f_i[2]^0 \mid 1 \leq i \leq n\} \\ &\quad \cup \{[2]c_i \rightarrow c_{i+1}[2]^0 \mid 1 \leq i \leq n-1\} \\ &\quad \cup \{[2]c_n \rightarrow \text{check}[2]^0, [2]\text{check}[2]^0 \rightarrow \text{check}[2]_2^+\} \end{aligned}$$

We start with the objects a_1, \dots, a_n in the inner membrane and we divide this membrane repeatedly by means of the rules $[2]a_i[2]^0 \rightarrow [2]t_i[2]^0[2]f_i[2]^0$; note that the object a_i used in each step is nondeterministically chosen, but each division replaces that object by t_i (for *true*) in one membrane and with f_i (for *false*) in the other membrane; hence after n steps the configuration obtained is the same regardless of the order of expanding the objects. Specifically, we get 2^n membranes with label 2, with each one containing a truth assignment for the n variables. Simultaneously with the division, we have to use the rules of type (a) that update the “counter” c ; hence at each step we increase by one the subscript of c . Therefore, when all variables have been expanded, we get the object *check* in all membranes (the rule of type (a) is used first, and after that the result is duplicated in the newly obtained membranes). In step $n+1$, this object exits each copy of membrane 2, changing its polarization to positive; this is meant to signal the fact that the generation of all truth assignments is completed, and we can start checking the truth values of (the clauses of) the propositional formula.

The previous example was also chosen to show that the polarizations of membranes are not used while generating the truth assignments, though they might be useful after that; until now, this has been the case in all polynomial-time solutions to NP-complete problems obtained in this framework. An important *open problem* in this area is whether or not the polarizations can be avoided. This can be done if other ingredients are considered, such as label changing or division of nonelementary membranes, but without adding such features the best result obtained so far is that the number of polarizations can be reduced to two.

2.9 Structuring the Objects

In the previous classes of P systems, the objects were considered atomic, identified only by their name, but in a cell many chemicals are complex molecules (e.g., proteins, DNA molecules, other large macromolecules), whose structures can be described by strings or more complex data, such as trees, arrays, etc. Also from a mathematical point of view, it is natural to consider P systems with string objects.

Such a system has the form

$$\Pi = (V, T, \mu, M_1, \dots, M_m, R_1, \dots, R_m)$$

where V is the alphabet of the system, $T \subseteq V$ is the terminal alphabet, μ is the membrane structure (of degree $m \geq 1$), M_1, \dots, M_m are finite sets of strings present in the m regions of the membrane structure, and R_1, \dots, R_m are finite sets of string-processing rules associated with the m regions of μ .

We have given the system in general form, with a specified terminal alphabet (we say that the system is *extended*; if $V = T$, then the system is said to be *non-extended*), and without specifying the type of rules. These rules can be of various forms, but we consider here only two cases: rewriting and splicing.

In a *rewriting P system*, the string objects are processed by rules of the form $a \rightarrow u(\text{tar})$, where $a \rightarrow u$ is a context-free rule over the alphabet V and tar is one of the target indications here, in, and out. When such a rule is applied to a string x_1ax_2 in a region i , we obtain the string x_1ux_2 , which is placed in region i , in any inner region, or in the surrounding region, depending on whether tar is here, in, or out, respectively. The strings that leave the system do not come back; if they are composed only of symbols from T , then they are considered as generated by the system. The language of all strings generated in this way is denoted by $L(\Pi)$.

In a *splicing P system*, we use splicing rules like those in DNA computing, that is, of the form $u_1\#u_2\$u_3\#u_4$, where u_1, u_2, u_3 , and u_4 are strings over V . For four strings $x, y, z, w \in V^*$ and a rule $r : u_1\#u_2\$u_3\#u_4$, we write

$$(x, y) \vdash_r (z, w) \text{ if and only if } \begin{aligned} x &= x_1u_1u_2x_2, & y &= y_1u_3u_4y_2, \\ z &= x_1u_1u_4y_2, & w &= y_1u_3u_2x_2, \\ &\text{for some } x_1, x_2, y_1, y_2 \in V^* \end{aligned}$$

We say that x and y are spliced at the sites u_1u_2 and u_3u_4 , respectively, and the result of the splicing (obtained by recombining the fragments obtained by cutting the strings as indicated by the sites) is the strings z and w .

In our case, we add target indications to the two resulting strings, that is, we consider rules of the form $r : u_1\#u_2\$u_3\#u_4(\text{tar}_1, \text{tar}_2)$, with tar_1 and tar_2 one of here, in, and out. The meaning is as standard: After splicing the strings x, y from a given region, the resulting strings z, w are moved to the regions indicated by $\text{tar}_1, \text{tar}_2$, respectively. The language generated by such a system consists again of all strings over T sent into the environment during the computation, without considering only halting computations.

An example of a rewriting or a splicing P system is not given here, but an important extension of rewriting rules — namely, *rewriting with replication* — is introduced. In such systems, the rules are of the form $a \rightarrow (u_1, \text{tar}_1) \parallel (u_2, \text{tar}_2) \parallel \dots \parallel (u_n, \text{tar}_n)$, with the meaning that by rewriting a string x_1ax_2 we get n strings, $x_1u_1x_2, x_1u_2x_2, \dots, x_1u_nx_2$, which have to be moved in the regions indicated by targets $\text{tar}_1, \text{tar}_2, \dots, \text{tar}_n$, respectively.

The replicated rewriting is important because it provides the possibility to replicate strings, thus enlarging the workspace. Indeed, this is one of the frequently used ways to generate an exponential workspace in linear time, used then for solving computationally hard problems in polynomial time.

2.10 P Systems with Objects on Membranes

Consider that objects placed only in membranes cover only part of the biological reality, where most of the reactions taking place in a cell are controlled by proteins bound on membranes. Thus, it is a natural challenge for membrane computing to take care of this situation and, indeed, several types of P systems were investigated where membranes carry objects.

Three main approaches are considered in the literature. The first one is inspired from Cardelli (2005), where six basic operations were introduced. Counterparts of biological operations with membranes and two calculi were defined on this basis: pino/exo/phago calculus and mate/drip/bud calculus. Part of these operations were rephrased in terms of membrane computing and used in defining classes of P systems where objects are placed only on membranes. We call *membrane systems* the obtained P systems. A few ideas from Cardelli and Păun (2006) will be recalled below.

In the above-mentioned systems, we do not have objects also in compartments, as in usual P systems. Two models taking care of this aspect are what we call *the Ruston model* and *the Trento model*. In both cases, objects are placed both on the membranes and in their compartments, with the difference that in the former case the objects do not change their places, those bound on membranes remain there, while in the latter model the objects can move from compartments to membranes and back. The first type of systems were introduced in Păun and Popa (2006) and the second one in Brijder (2007). Several further papers were devoted to these systems, but the reader can refer to Păun et al. (2009) and www.ppage.psystems.eu for details.

Just to have an idea about this direction of research, here the P systems based on mate/drip operations are introduced.

One can consider an alphabet A of special objects called “proteins.” A membrane that has associated a multiset $u \in A^*$ is represented by $[]_u$; we also say that u marks the membrane. When necessary, we can also use labels for identifying membranes, but in most cases the objects marking them are sufficient. The operations considered here are the following:

$$\begin{aligned} \text{mate: } & []_{ua} []_v \rightarrow []_{uxv} \\ \text{drip: } & []_{uav} \rightarrow []_{ux} []_v \end{aligned}$$

where $a \in V$ and $u, v, x \in A^*$. The length of the string uxv (i.e., the total multiplicity of the multiset represented by this string) from each rule is called the *weight* of the rule.

In each case, multisets of objects are transferred from membranes appearing in the left-hand side of rules to membranes appearing in the right-hand side of rules, with protein a evolved into the multiset x . It is important to note that in *drip* rules the multiset uxv is precisely split into ux and v , with these two multisets assigned to the two new membranes.

The rules are applied to given membranes if they are marked with multisets of objects, which include the multisets mentioned in the left-hand side of rules. All objects placed on membranes but not involved in the rules are not affected by the use of rules; in the case of *drip*,

these objects are randomly distributed to the two resulting membranes. In the case of *mate*, the result is uniquely determined.

The contents of membranes involved in these operations (i.e., the membranes possibly placed inside) is transferred from the input membranes to the output membranes in the same way as in brane calculi. Denoting these contents (empty or consisting of other membranes) by \mathbf{P}, \mathbf{Q} , we can indicate the effect of the two operations as follows:

$$\begin{aligned} \text{mate: } & [\mathbf{P}]_{ua}[\mathbf{Q}]_v \rightarrow [\mathbf{P} \ \mathbf{Q}]_{uxv} \\ \text{drip: } & [\mathbf{P}]_{uav} \rightarrow [\]_{ux}[\mathbf{P}]_v \end{aligned}$$

Now, a P system of the following form can be defined

$$\Pi = (A, \mu, u_0, u_1, \dots, u_m, R)$$

where A is an alphabet (its elements are called *objects* or *proteins*), μ is a membrane structure with at least two membranes (hence $m \geq 1$), labeled with $0, 1, \dots, m$, where 0 is the skin membrane, $u_0, u_1, \dots, u_m \in A^*$ are multisets of objects bound to the membranes of μ at the beginning of the computation, with $u_0 = \lambda$, R is a finite set of mate, drip rules, of the forms specified above, using objects from the set A .

As usual in membrane computing, the evolution of the system proceeds through transitions among configurations, based on the nondeterministic maximally parallel use of rules. In each step, each membrane and each object can be involved in only one rule. A membrane remains unchanged if no rule is applied to it. The skin membrane never evolves.

A computation that starts from the initial configuration (the one described by μ and multisets u_0, u_1, \dots, u_m) is successful only if (1) it halts, that is, it reaches a configuration where no rule can be applied, and (2) in the halting configuration there are only two membranes, the skin (marked with the empty multiset) and an inner one. The result of a successful computation is the number of proteins that mark the inner membrane in the halting configuration.

The set of all numbers computed in this way by Π is denoted by $N(\Pi)$. The family of all sets $N(\Pi)$ computed by P systems Π using at any moment during a halting computation at most m membranes, and mate, drip rules of weight at most p, q , respectively, is denoted by $NOP_m(\text{mate}_p, \text{drip}_q)$.

3 Tissue-Like P Systems

We pass now to consider a very important generalization of the membrane structure, passing from the cell-like structure, described by a tree, to a tissue-like structure, with the membranes placed in the nodes of an arbitrary graph (which corresponds to the complex communication networks established among adjacent cells by making their protein channels cooperate, moving molecules directly from one cell to another). Actually, in the basic variant of tissue-like P systems, this graph is virtually a total one; what matters is the communication graph, dynamically defined during computations. In short, several (elementary) membranes – also called cells – are freely placed in a common environment; they can communicate either with each other or with the environment by symport/antiport rules. Specifically, antiport rules of the form $(i, x/y; j)$ are considered, where i, j are labels of cells or at most one is zero, identifying the environment, and x, y are multisets of objects. This means that the multiset x is moved from i to j at the same time as the multiset y is moved from j to i . If one of the multisets x, y is empty, then we have, in fact, a symport rule. Therefore, the communication among cells is done either directly, in one

step, or indirectly, through the environment: One cell throws some objects out and other cells can take these objects in the next step or later. As in symport/antiport P systems, the environment contains a specified set of objects in arbitrarily many copies. A computation develops as standard, starting from the initial configuration and using the rules in the nondeterministic maximally parallel manner. When halting, we count the objects from a specified cell, and this is the result of the computation.

The graph plays a more important role in so-called *tissue-like P systems with channel-states*, whose rules are associated with synapses and are of the form $(s, x/y, s')$, where s, s' are states from a given finite set and x, y are multisets of objects. Such a rule of the form $(s, x/y, s') \in R_{(i,j)}$ is interpreted as an antiport rule $(i, x/y, j)$ as above, acting only if the synapse (i, j) has the state s ; the application of the rule means (1) moving the objects specified by x from cell i (from the environment, if $i = 0$) to cell j , at the same time with the move of the objects specified by y in the opposite direction, and (2) changing the state of the synapse from s to s' . The use of states provides a powerful way to control the application of rules.

A still more elaborated class of systems, called *population P systems*, was introduced in Bernardini and Gheorghe (2004) with motivations related to the dynamics of cells in skin-like tissues, populations of bacteria, and colonies of ants. These systems are highly dynamical: not only the links between cells, corresponding to the channels from the previous model with states assigned to the channels, can change during the evolution of the system, but also the cells can change their names, can disappear (get dissolved), and can divide, thus producing new cells; these new cells inherit, in a well-specified sense, links with the neighboring cells of the parent cell. The generality of this model makes it rather attractive for applications in areas such as those mentioned above, related to tissues, populations of bacteria, etc.

A further interesting version of tissue-like P systems is that of *neural-like P systems*, where we again use a population of cells (each one identified by its label) linked by a specified set of synapses. This time, each cell has at every moment a state from a given finite set of states, its contents in the form of a multiset of objects from a given alphabet of objects, and a set of rules for processing these objects.

The rules are of the form $sw \rightarrow s'(x, here)(y, go)(z, out)$, where s, s' are states and w, x, y, z are multisets of objects; in state s , the cell consumes the multiset w and produces the multisets x, y, z ; the objects from multiset x remain in the cell, those of multiset y have to be communicated to the cells toward which there are synapses starting in the current cell; a multiset z , with the indication *out*, is allowed to appear only in a special cell, designated as the output cell, and for this cell the use of the previous rule entails sending the objects of z to the environment.

The computation starts with all cells in specified initial states, with initially given contents, and proceeds by processing the multisets from all cells, simultaneously, according to the local rules. Then the obtained objects are redistributed along synapses and send a result into the environment through the output cell; a result is accepted only when the computation halts.

Because of the use of states, there are several possibilities for processing the multisets of objects from each cell. In the *minimal* mode, a rule is chosen and applied once to the current pair (state, multiset). In the *parallel* mode, a rule is chosen, for example, $sw \rightarrow s'w'$, and used in the maximally parallel manner: The multiset w is identified in the cell contents in the maximal manner, and the rule is used for processing all these instances of w . Finally, in the *maximal* mode, we apply in the maximally parallel manner all rules of the form $sw \rightarrow s'w'$, that is, with the same states s and s' (note the difference with the parallel mode, where in each step a rule is chosen and only that rule is used as many times as possible).

There are also three ways to move the objects between cells (of course, we only move objects produced by rules in multisets with the indication *go*). Assume that a rule $sw \rightarrow s'(x, \text{here})(y, go)$ is applied in a given cell i . In the *spread* mode, the objects from y are nondeterministically distributed to all cells j such that (i, j) is a synapse of the system. In the *one* mode, all the objects from y are sent to one cell j , provided that the synapse (i, j) exists. Finally, we can also *replicate* the objects of y , and each object from y is sent to all cells j such that (i, j) is an available synapse.

We do not enter into more details; on the one hand, this class of P systems still waits for a systematic investigation; on the other hand, there is another class of P systems inspired from the brain's organization, whose study is much more developed, the *spiking neural P systems*.

4 Spiking Neural P Systems

Spiking neural P systems (SN P systems) were introduced in Ionescu et al. (2006) with the aim of defining P systems based on ideas specific to spiking neurons, recently much investigated in neural computing.

Very shortly, an SN P system consists of a set of *neurons* (cells consisting of only one membrane) placed in the nodes of a directed graph and sending signals (*spikes*, denoted in what follows by the symbol a) along *synapses* (arcs of the graph). Thus, the architecture is that of a tissue-like P system, with only one kind of object present in the cells. The objects evolve by means of *spiking rules*, which are of the form $E/a^c \rightarrow a;d$, where E is a regular expression over $\{a\}$ and c, d are natural numbers, $c \geq 1$, $d \geq 0$. The meaning is that a neuron containing k spikes such that $a^k \in L(E)$, $k \geq c$, can consume c spikes and produce one spike, after a delay of d steps. This spike is sent to all neurons to which a synapse exists outgoing from the neuron where the rule was applied. There also are *forgetting rules*, of the form $a^s \rightarrow \lambda$, with the meaning that $s \geq 1$ spikes are forgotten, provided that the neuron contains exactly s spikes. We say that the rules "cover" the neuron: all spikes are taken into consideration when using a rule. The system works in a synchronized manner; that is, in each time unit, each neuron that can use a rule should do it, but the work of the system is sequential in each neuron: Only (at most) one rule is used in each neuron. One of the neurons is considered to be the *output neuron*, and its spikes are also sent to the environment. The moments of time when a spike is emitted by the output neuron are marked with 1, the other moments are marked with 0. This binary sequence is called the *spike train* of the system – it might be infinite if the computation does not stop.

In the spirit of spiking neurons, the result of a computation is encoded in the distance between consecutive spikes sent into the environment by the (output neuron of the) system. For example, we can consider only the distance between the first two spikes of a spike train, or the distances between the first k spikes, the distances between all consecutive spikes, taking into account all intervals or only intervals that alternate, all computations or only halting computations, etc.

An SN P system can also be used in the accepting mode: A neuron is designated as the *input neuron* and two spikes are introduced in it, at an interval of n steps; the number n is accepted if the computation halts.

Another possibility is to consider the spike train itself as the result of a computation, and then we obtain a (binary) language generating device. We can also consider both input and

output neurons and then an SN P system can work as a transducer. Languages on arbitrary alphabets can be obtained by generalizing the form of rules: Take rules of the form $E/a^c \rightarrow a^p; d$, with the meaning, provided that the neuron is covered by E , c spikes are consumed and p spikes are produced, and sent to all connected neurons after d steps (such rules are called *extended*). Then, with a step when the system sends out i spikes, we associate a symbol b_i , and thus we get a language over an alphabet with as many symbols as the number of spikes simultaneously produced. Another natural extension is to consider several output neurons, thus producing vectors of numbers, not only single numbers.

The technical details are skipped, but a simple example is considered (if a rule $E/a^c \rightarrow a; d$ has $L(E) = \{a\}$, then it can be written in the simplified form $a^c \rightarrow a; d$):

$$\begin{aligned}\Pi_1 &= (O, \sigma_1, \sigma_2, \sigma_s, syn, out), \text{ with} \\ O &= \{a\} \text{ (alphabet, with only one object, the spike)} \\ \sigma_1 &= (2, \{a^2/a \rightarrow a; 0, \quad a \rightarrow \lambda\}) \\ &\quad (\text{first neuron: initial number of spikes, rules}) \\ \sigma_2 &= (1, \{a \rightarrow a; 0, \quad a \rightarrow a; 1\}) \\ &\quad (\text{second neuron: initial number of spikes, rules}) \\ \sigma_3 &= (3, \{a^3 \rightarrow a; 0, \quad a \rightarrow a; 1, \quad a^2 \rightarrow \lambda\}) \\ &\quad (\text{third neuron: initial number of spikes, rules}) \\ syn &= \{(1, 2), (2, 1), (1, 3), (2, 3)\} \text{ (synapses)} \\ out &= 3 \text{ (output neuron).}\end{aligned}$$

This system functions as follows. All neurons can fire in the first step, with neuron σ_2 choosing nondeterministically between its two rules. Note that neuron σ_1 can fire only if it contains two spikes; one spike is consumed, the other remains available for the next step.

Both neurons σ_1 and σ_2 send a spike to the output neuron, σ_3 ; these two spikes are forgotten in the next step. Neurons σ_1 and σ_2 also exchange their spikes; thus, as long as neuron σ_2 uses the rule $a \rightarrow a; 0$, the first neuron receives one spike, thus completing the needed two spikes for firing again.

However, at any moment, starting with the first step of the computation, neuron σ_2 can choose to use the rule $a \rightarrow a; 1$. On the one hand, this means that the spike of neuron σ_1 cannot enter neuron σ_2 and it only goes to neuron σ_3 ; in this way, neuron σ_2 will never work again because it remains empty. On the other hand, in the next step neuron σ_1 has to use its forgetting rule $a \rightarrow \lambda$, while neuron σ_3 fires, using the rule $a \rightarrow a; 1$. Simultaneously, neuron σ_2 emits its spike, but it cannot enter neuron σ_3 (it is closed at this moment); the spike enters neuron σ_1 , but it is forgotten in the next step. In this way, no spike remains in the system. The computation ends with the expelling of the spike from neuron σ_3 . Because of the waiting moment imposed by the rule $a \rightarrow a; 1$ from neuron σ_3 , the two spikes of this neuron cannot be consecutive, but at least two steps must exist in between.

Thus, it can be concluded that Π computes/generates all natural numbers greater than or equal to 2.

There are several classes of SN P systems, for instance, asynchronous SN P systems (no clock is considered, any neuron may use or not a rule), with exhaustive use of rules (when enabled, a rule is used as many times as made possible by the spikes present in a neuron), with certain further conditions imposed on the halting configuration, etc.

5 Computing Power

As previously mentioned, many classes of P systems that combine various ingredients (as described above or similar ones) are able to simulate Turing machines; hence they are *computationally complete*. Always, the proofs of results of this type are constructive, and this has an important consequence from the computability point of view: in this way we obtain *universal* (hence *programmable*) P systems. In short, starting from a universal Turing machine (or an equivalent universal device), we get an equivalent universal P system. Among others, this implies that in the case of Turing complete classes of P systems, the hierarchy on the number of membranes always collapses — at most at the level of the universal P systems. There are, however, classes of nonuniversal P systems for which “the number of membranes matters,” to cite the title of Ibarra (2005), where two classes of P systems were defined for which the hierarchies on the number of membranes are infinite. Actually, the number of membranes in a P system sufficient to characterize the power of Turing machines is always rather small.

Here only a few of the most interesting types of universality results for cell-like P systems are mentioned:

1. P systems with symbol-objects with catalytic rules, using only two catalysts and two membranes, are universal.
2. P systems with symport/antiport rules of a restricted size (e.g., three membranes, symport rules of weight 2, and no antiport rules, or three membranes and minimal symport and antiport rules) are universal.
3. P systems with symport/antiport rules (of arbitrary size), using only three membranes and only three objects, are universal.
4. P systems with active membranes, using rules of types (a), (b), (c) and two polarizations are universal.
5. P systems with string objects and rewriting rules usually generate only matrix languages, but adding priorities, ways to control the permeability of membranes, inhibitors, or using the rules in the leftmost way lead to universality.
6. P systems with *mate*, *drip* rules controlled by at most four objects and using at most five membranes are universal.

Universality can also be obtained for other types of P systems, such as tissue-like systems, SN P systems (provided that the number of spikes present in the neurons is not bounded, otherwise only semilinear sets of numbers can be computed), accepting P systems, etc. However, we do not recall here such results; one can refer to Păun et al. (2009) and www.ppage.psystems.eu for details. In general, around these results there are a series of open problems, mainly related to their optimality. Some of these results are continuously improved, so the interested reader is advised to consult the membrane computing current bibliography for the most recent results.

In the accepting mode, an interesting issue appears, that of using deterministic systems. Most universality results were obtained in the deterministic case, but there also are situations where the deterministic systems are strictly less powerful than the nondeterministic ones. This is proven in Ibarra and Yen (2006) for the accepting catalytic P systems.

It is worth noting that the proofs of computational completeness are based on simulating various types of grammars with restricted derivation (mainly matrix grammars with

appearance checking) or on simulating register machines. In the case of SN P systems, this has an interesting consequence: Starting the proofs from small universal register machines, one can find small universal SN P systems. For instance, as shown in Păun and Păun (2007), there are universal computing SN P systems with 84 neurons using standard rules and with only 49 neurons using extended rules. In the generative case, the best results are 79 and 50 neurons, respectively.

6 Computational Efficiency

The computational power (the “competence”) is only one of the important questions to be dealt with when defining a new (bio-inspired) computing model. The other fundamental question concerns the computing *efficiency*. Because P systems are parallel computing devices, it is expected that they can solve hard problems in an efficient manner – and this expectation is confirmed for systems provided with ways for producing an exponential workspace in a linear time. Mainly, three such biologically inspired possibilities have been considered so far in the literature, and all of them were proven to lead to polynomial solutions to NP-complete problems, by a time–space trade-off.

These three ideas are membrane division, membrane creation, and string replication. The standard problems addressed in this framework were decidability problems, starting with SAT, the Hamiltonian path problem, and the node covering problem. Other types of problems were considered, such as the problem of inverting one-way functions, or the subset-sum and the knapsack problems (note that the last two are numerical problems, where the answer is not of the yes/no type, as in decidability problems). In general, the approach is of a brute force type: All candidate solutions are generated, making use of the massive parallelism (e.g., all truth assignments of the variables appearing in a SAT problem); then they are checked, again in parallel, in search of a solution. The computation may proceed nondeterministically provided that it is *confluent*: Either eventually we reach the same configuration and then we continue deterministically (strong confluence), or all computations halt and each of them provides the same result (weak confluence).

The formal framework for dealing with complexity matters in terms of P systems is provided by the *accepting P systems with input*: A family of P systems of a given type is constructed starting from a given problem, and an instance of the problem is introduced as an input in one such systems; working in a deterministic (or a confluent) mode, the system sends to the environment the answer to the respective instance. The family of systems should be constructed in a uniform mode by a Turing machine, working a polynomial time.

This direction of research has been much investigated. A large number of problems were considered, the membrane computing complexity classes were refined, characterizations of the $P \neq NP$ conjecture were obtained in this framework, several characterizations of the class P, even problems that are PSPACE-complete, were proven to be solvable in polynomial time by means of membrane systems provided with membrane division or membrane creation. An important (and difficult) problem is that of finding the border between efficiency and non-efficiency: which ingredients should be used in order to be able to solve hard problems in a polynomial time? The reader is referred to the complexity chapter from Păun et al. (2006) and to the literature available at www.ppage.psystems.eu for details – including many problems that are still open in this area.

7 Applications

There are many features of membrane computing that make it attractive for applications in several disciplines, especially for biology and biomedicine: *distribution* (with the important system–part interaction, emergent behavior, nonlinearly resulting from the composition of local behaviors), easy programmability, scalability/extensibility, transparency (multiset rewriting rules are nothing other than reaction equations such as those customarily used in chemistry and biochemistry), parallelism, nondeterminism, communication, and so on and so forth.

As expected, most of the applications have been carried out in biology. These applications are usually based on experiments using programs for simulating/implementing P systems on usual computers, and there are already several such programs, more and more elaborated (e.g., with better and better interfaces, which allow for the friendly interaction with the program). An overview of membrane computing software reported in the literature (some programs are available at the webpage www.ppage.psystems.eu together with sample applications) can be found in the volume Ciobanu et al. (2006).

Of course, when using a P system for simulating a biological process, we are no longer interested in its computing behavior (power, efficiency, etc.), but in its evolution in time; the P system is then interpreted as a dynamical system, and its trajectories are of interest, its “life.” Moreover, the ingredients we use are different from those considered in theoretical investigations. For instance, in mathematical terms, we are interested in results obtained with a minimum of premises and with weak prerequisites, while the rules are used in ways inspired from automata and language theory (e.g., in a maximally or minimally parallel way), but when dealing with applications the systems are constructed in such a way that they capture the features of reality (for instance, the rules are of a general form, they are applied according to probabilistic strategies, based on stoichiometric calculations, the systems are not necessarily synchronized, and so on).

The typical applications run as follows. One starts from a biological process described in general in graphical terms (chemicals are related by reactions represented in a graph-like manner, with special conventions for capturing the context-sensitivity of reactions, the existence of promoters or inhibitors, etc.) or already available in data bases in SBML (system biology mark-up language) form; these data are converted into a P system that is introduced in a simulator. The way the evolution rules (reactions) are applied is the key point in constructing this simulator (often, the classical Gillespie algorithm is used in compartments, or multi-compartmental variants of it are considered). As a result, the evolution in time of the multiplicity of certain chemicals is displayed, thus obtaining a graphical representation of the interplay in time of certain chemicals, their growth and decay, and so on. Many illustrations of this scenario can be found in the literature, many times dealing with rather complex processes.

Besides applications in biology, applications were reported in computer graphics, linguistics (both as a representation language for various concepts related to language evolution, dialogue, semantics, and making use of the parallelism, in solving parsing problems in an efficient way), economics (where many biochemical metaphors find a natural counterpart, with the mentioning that the “reactions” that take place in economics, for instance, in market-like frameworks, are not driven only by probabilities/stoichiometric calculations, but also by psychological influences, which makes the modeling still more difficult than in biology), computer science (in devising sorting and ranking algorithms), cryptography, etc.

A very promising direction of research, namely, applying membrane computing in devising approximate algorithms for hard optimization problems, was initiated in Nishida (2004),

who proposed *membrane algorithms* as a new class of distributed evolutionary algorithms. These algorithms can be considered as high-level (distributed and dynamically evolving their structure during the computation) evolutionary algorithms. In short, candidate solutions evolve in compartments of a (dynamical) membrane structure according to local algorithms, with better solutions migrating down in the membrane structure; after a specified halting condition is met, the current best solution is extracted as the result of the algorithm.

This strategy was checked for several optimization problems, and the results were more than encouraging for a series of benchmark problems: The convergence is very fast, the number of membranes is rather influential on the quality of the solution, the method is reliable, both the average quality and the worst solutions were good enough and always better than the average and the worst solutions given by existing optimization procedures.

8 Closing Remarks

This chapter aimed to provide only a quick presentation of membrane computing, of basic ideas and of types of results and applications. Many more things are available in the literature: further classes of P systems (e.g., we said nothing about conformon-P systems, discussed in detail in Frisco (2008)), other problems to investigate (e.g., normal forms, semantics), links with other areas (e.g., X-machines, Petri nets), software (many programs are available, most of them also on www.ppage.psystems.eu, together with examples of applications), and so on. The area also has many unsolved problems and directions for research that wait for further efforts.

We mentioned before the borderline between universality and non-universality and between efficiency and non-efficiency concerning the succinctness of universal P systems or P systems able to solve hard problems in polynomial time. Then, because universality implies undecidability of all nontrivial questions, an important issue is that of finding classes of P systems with decidable properties.

This is also related to the use of membrane computing as a modeling framework: If no insights can be obtained in an analytical manner algorithmically, then what remains is to simulate the system on a computer. To this aim, better programs are still needed, maybe parallel implementations able to handle real-life questions.

Several research topics concern complexity investigations: uniform versus semiuniform solutions, the role of unary encoding, confluent versus deterministic solutions, the possibility of using precomputed resources, activated during the computation (as done for SN P systems in Chen et al. (2006)), etc.

We conclude by stating that membrane computing is a young and well-developed branch of natural computing, rather promising for applications, which still needs considerable research efforts – which might be a challenge for the reader.

References

- Bernardini F, Gheorghe M (2004) Population P systems. *J Univ Comput Sci* 10(5):509–539
- Brijder R, Cavaliere M, Riscos-Núñez A, Rozenberg G, Sburlan D (2007) Membrane systems with marked membranes. *Electron Notes Theor Comput Sci* 171:25–36
- Calude CS, Păun Gh, Rozenberg G, Salomaa A (eds) (2001) Multiset processing. Mathematical,

- computer science, and molecular computing points of view. Lecture notes in computer science, vol. 2235, Springer, Berlin
- Cardelli L (2005) Brane calculi – interactions of biological membranes. Lecture notes in computer science vol. 3082, pp 257–280
- Cardelli L, Păun Gh (2006) An universality result for a (mem)brane calculus based on mate/drip operations. *Intern J Found Comput Sci* 17:49–68
- Chen H, Ionescu M, Ishdorj T-O (2006) On the efficiency of spiking neural P systems. In: Proceedings of 8th international conference on electronics, information, and communication, Ulan Bator, Mongolia, June 2006, pp 49–52
- Ciobanu G, Păun Gh, Pérez-Jiménez MJ (eds) (2006) Applications of membrane computing. Springer, Berlin
- Frisco P (2008) Computing with cells. Advances in membrane computing. Oxford University Press, Oxford, UK
- Ibarra OH (2005) On membrane hierarchy in P systems. *Theor Comput Sci* 334(1–2):115–129
- Ibarra OH, Yen HC (2006) Deterministic catalytic systems are not universal. *Theor Comput Sci* 363(2):149–161
- Ionescu M, Păun Gh, Yokomori T (2006) Spiking neural P systems. *Fundamenta Informaticae* 71(2–3):279–308
- Nishida TY (2004) An application of P systems: a new algorithm for NP-complete optimization problems. In: Callaos N et al. (eds) Proceedings of the 8th world multi-conference on systems, cybernetics and informatics, vol. V, pp 109–112
- Păun A, Păun Gh (2007) Small universal spiking neural P systems. *BioSystems* 10(1):48–60
- Păun A, Popa B (2006) P systems with proteins on membranes. *Fundamenta Informaticae* 72:467–483
- Păun Gh (2001) Computing with membranes. *J Comput Syst Sci* 61(1):108–143 (and Turku Center for Computer Science-TU-CS Report 208, November 1998, <http://www.tucs.fi>)
- Păun Gh (2002) Membrane computing. An introduction. Springer, Berlin
- Păun Gh, Rozenberg G, Salomaa A (eds) (2009) Handbook of membrane computing. Oxford University Press, Oxford, UK
- The P Systems Website: ppage.psystems.eu. Accessed Apr 2010

Section V

Quantum Computation

Mika Hirvensalo

41 Mathematics for Quantum Information Processing

Mika Hirvensalo

Department of Mathematics, University of Turku, Finland
mikhirve@utu.fi

1	<i>History</i>	1382
2	<i>Hilbert Space Formalism</i>	1382
3	<i>Spectral Representation</i>	1389
4	<i>The Meaning of the Quantum States</i>	1392
5	<i>Quantum Bits</i>	1394
6	<i>Compound Systems</i>	1397
7	<i>Subsystem States</i>	1398

Abstract

Information processing in the physical world must be based on a physical system representing the information. If such a system has a quantum physical description, then one talks about *quantum information processing*. The mathematics needed to handle quantum information is somewhat more complicated than that used for classical information. The purpose of this chapter is to introduce the basic mathematical machinery used to describe quantum systems with finitely many potential observable values. A typical example of such a system is the *quantum bit* (or *qubit*, for short), which has at most two potential values for any physical observable. The mathematics for quantum information processing is based on vector spaces over complex numbers, and can be seen as a straightforward generalization of linear algebra of real vector spaces.

1 History

The earliest ideas on *quantum information* theory were introduced by von Neumann very soon after the initial mathematical machinery of quantum mechanics was developed (von Neumann 1927). On the other hand, the upcoming ideas of *quantum information processing* were still decades away. In the 1980s Paul Benioff presented the first ideas of computing in microsystems obeying quantum mechanics (Benioff 1980, 1982).

From the point of view of modern quantum information processing, the most important early idea was that of Richard Feynman. In his seminal article (Feynman 1982), he introduced a most interesting idea: that it may be impossible to simulate quantum mechanical systems with classical computers without an exponential slowdown (when the simulation efficiency is measured with respect to the number of particles in the system to be simulated).

The next important step toward quantum computing was taken by David Deutsch. In (Deutsch 1985), the aim was to address the Church–Turing thesis from the physical point of view. The Church–Turing thesis (see, e.g., Papadimitriou (1994)) says roughly that a Turing machine is the mathematical counterpart of the intuitive notion of an algorithm: For any intuitive algorithm, there exists a Turing machine performing the same task. Briefly, Deutsch’s argument is as follows: Computation of any form is a physical process anyway, and to prove the Church–Turing thesis it is therefore enough to show that there is a physical system capable of simulating all other physical systems. Deutsch did not pay attention to the complexity of computation, and the pillars of quantum complexity theory were later established by Bernstein and Vazirani (1997).

Interesting quantum algorithms were already introduced by Deutsch and subsequently by Simon (1994), but the most famous quantum algorithms, polynomial-time factoring and the discrete logarithm, were designed by Shor (1994).

2 Hilbert Space Formalism

The mathematical objects that establish the basis for Newtonian mechanics are not very complicated: everything can be laid on the notion of vectors. To describe a point-sized body in a three-dimensional space, one can give the *position* $\mathbf{x} \in \mathbb{R}^3$ and *momentum* $\mathbf{p} \in \mathbb{R}^3$, related

via $\mathbf{p} = m \frac{d}{dt} \mathbf{x}$, where m is the mass of the body. The dynamics of this very simple system is described by the Newtonian equation of motion

$$\mathbf{F} = \frac{d}{dt} \mathbf{p}$$

where \mathbf{F} is the force affecting the body.

Toward a more holistic approach to mechanics, consider first a one-dimensional motion of a particle. The *total energy* of the system in this simple case is $H = \frac{1}{2}mv^2 + V(x) = \frac{p^2}{2m} + V(x)$, where $\frac{1}{2}mv^2$ is the *kinetic energy* and $V(x) = -\int_{x_0}^x F(s)ds$ is the *potential energy* of the particle depending only on the position x . Hence

$$\frac{\partial}{\partial x} H = V'(x) = -F(x) = -\frac{d}{dt} p$$

and

$$\frac{\partial}{\partial p} H = \frac{p}{m} = v = \frac{d}{dt} x$$

Differential equations

$$\frac{d}{dt} x = \frac{\partial}{\partial p} H, \quad \frac{d}{dt} p = -\frac{\partial}{\partial x} H$$

thus obtained are an example of *Hamiltonian reformulation* of classical mechanics. A general mechanical system can consist of many bodies, and the bodies themselves need not be point-like, but they may have internal degrees of freedom (such as capability of rotating).

- It is possible to formulate a general mechanical system in terms of Hamiltonian mechanics, and the equation of motion becomes

$$\begin{aligned}\frac{d}{dt} x_i &= \frac{\partial}{\partial p_i} H \\ \frac{d}{dt} p_i &= -\frac{\partial}{\partial x_i} H\end{aligned}$$

where $H = H(x_1, \dots, x_n, p_1, \dots, p_n, t)$ is the *Hamiltonian function* of the system, describing the total energy.

Variables x_1, \dots, x_m are called *generalized coordinates* and may involve, for example, the usual spatial coordinates and angles of rotation, whereas p_1, \dots, p_n are called *generalized momenta* (may include the usual momenta and the angular momenta, for instance).

The general form of Hamiltonian mechanics thus contains $\mathbf{x} = (x_1, \dots, x_n)$, and $\mathbf{p} = (p_1, \dots, p_n)$ (both vectors in \mathbb{R}^n) as a description of the *system state* at a specified moment of time. Collecting these two, one can say that the pair $(\mathbf{x}, \mathbf{p}) \in \mathbb{R}^{2n}$ is the *state vector* of the system. One can also say the system has n *degrees of freedom*.

It should not then come as a surprise that the description of quantum mechanics also needs vector spaces. However, it is not evident that quantum mechanics should be based on *complex* vector spaces. In fact, to introduce complex numbers is not unavoidable (this is trivial, as a complex number can be presented as a pair of real numbers), but mathematically very helpful. In what follows we will introduce the basic ingredients needed for the Hilbert space formalism of quantum mechanics.

The formalism of quantum mechanics was developed from *wave mechanics* (due to Erwin Schrödinger) and *matrix mechanics* (credited to Werner Heisenberg, Max Born, and Pascual Jordan). These mathematically equivalent formalisms were developed most notably by von Neumann (1932) into a Hilbert space formalism, the major topic of this chapter. It is noteworthy that, in general, the most interesting features emerge in *infinite-dimensional* Hilbert spaces, whereas from the quantum computational point of view, usually finite-dimensional Hilbert spaces are satisfactory enough.

To follow this representation, it is assumed that the reader has a basic knowledge of finite-dimensional *real* vector spaces \mathbb{R}^n strong enough to cover objects such as vectors, linear mappings, matrices, bases, eigenvalues, eigenvectors, inner products, orthogonality, and subspaces. These notions can be found in many basic linear algebra textbooks, for instance in Cohn (1994). In the following, these notions are generalized to *complex* vector spaces. The resulting mathematical objects obtained are then used to describe quantum mechanics.

Definition 1 A *Hilbert space* is a vector space over complex numbers, equipped with a Hermitian inner product, complete with respect to the metric induced by the inner product.

To explain the terminology of the above definition, we choose the standpoint of quantum computing, and then the most important examples of Hilbert spaces are the finite-dimensional ones. As a prototype of such, one can introduce the following: Let

$$\mathbb{C}^n = \{(x_1, \dots, x_n) \mid x_i \in \mathbb{C}\}$$

be the set of all n -tuples of complex numbers. When equipped with pointwise scalar multiplication

$$a(x_1, \dots, x_n) = (ax_1, \dots, ax_n)$$

and pointwise vector addition

$$(x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n)$$

set \mathbb{C}^n evidently becomes a vector space. The (Hermitian) inner product in \mathbb{C}^n is defined by

$$\langle x | y \rangle = x_1^* y_1 + \dots + x_n^* y_n$$

where x^* means the complex conjugate of x . Defined as above, the inner product obviously satisfies the following properties.

► Hermitian Inner Product

- $\langle x | y \rangle = \langle y | x \rangle^*$
- $\langle x | x \rangle \geq 0$ and $\langle x | x \rangle = 0$ if and only if $x = 0$
- $\langle x | ay + bz \rangle = a\langle x | y \rangle + b\langle x | z \rangle$

According to our definition, the Hermitian inner product is linear with respect to the second argument, and *antilinear* with respect to the first one (meaning that $\langle ax + by | z \rangle = a^* \langle x | z \rangle + b^* \langle y | z \rangle$). In the mathematics literature, the Hermitian inner product is usually linear with respect to the first argument, but the convention chosen here is more familiar to quantum physics.

The inner product induces a *norm* by $\|x\| = \sqrt{\langle x | x \rangle}$, and the completeness of Definition 1 means completeness with respect to this norm. That is, for each sequence x_n satisfying

$\lim_{n,m \rightarrow \infty} ||\mathbf{x}_n - \mathbf{x}_m|| = 0$, there is a limit $\mathbf{x} = \lim_{n \rightarrow \infty} \mathbf{x}_n$ also belonging to the space. It should be noted that the completeness is self-evident in the finite-dimensional case. In fact, the following theorem is not very difficult to prove.

Theorem 1 *An n-dimensional Hilbert space H_n (n finite) is isomorphic to \mathbb{C}^n .*

Hence the basic structure of quantum computing, a finite-dimensional Hilbert space, can in practice be treated as \mathbb{C}^n , the set of n -tuples of complex numbers. However, it is not only the space H_n on which the formalism of quantum mechanics will be based, but space $L(H_n)$, the set of all linear mappings $H_n \rightarrow H_n$, will be needed for the description. But when restricting to so-called *pure states* (defined later), space H_n is (almost) sufficient to accommodate the mathematical description of quantum mechanics. Especially in the theory of quantum computing, this is frequently the case. In any case, H_n is called the *state space* of an n -level quantum system. Here and hereafter, the dimension n of H_n equals the number of the *perfectly distinguishable states* of the system to be described. This means that there are n states of the system that can be mutually separated by observations with 100% accuracy.

In quantum computing, the spatial location of the system under investigation is not usually the most essential one; more frequently, the system's internal degrees of freedom are those of interest. This can be compared to a classical system, which, in addition to its position, has also its dynamics of rotation. Analogously, in quantum systems there can be internal degrees of freedom, and in current physical realizations of quantum computing, one of the most noteworthy is the *spin* of a particle. (As the name suggests, spin is closely related to the rotation of the particle around its axis, but spin has also nonclassical features.) It should be remembered that the Hamiltonian reformulation of classical mechanics can treat *all degrees of freedom* in an equal way, and the same is true also in quantum mechanics: the *Schrödinger equation of motion* treats all the degrees of freedom in an analogous way; the position of a particle would involve potentially an infinite number of potential values, but there are also internal properties (such as the spin) that can assume only a finite number of measurable values. It is also noteworthy that obviously one can concentrate only on one degree of freedom of a given system, leaving a lot of system information out. This means, for example, that it is evidently possible to create a mathematical description of a particle spin only, leaving out the particle position and other properties.

Before going into the formalism, it is useful to present some notation. So-called Dirac notation will be useful: for a vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{C}^n$, notation

$$|\mathbf{x}\rangle = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad (1)$$

stands for a column vector, and notation

$$\langle \mathbf{x}| = (x_1^*, \dots, x_n^*) \quad (2)$$

for the row vector, the *adjoint* version of [Eq. 1](#). Adjointness for matrices will be treated in more detail later.

For \mathbf{x} and $\mathbf{y} \in \mathbb{C}^n$, notations $|\mathbf{x}\rangle\langle\mathbf{y}|$ and $\langle\mathbf{x}||\mathbf{y}\rangle$ are both interpreted as *tensor products* (alternative term: *Kronecker product*).

Definition 2 The tensor product of matrices A ($k \times l$) and B ($m \times n$) is a $km \times ln$ -matrix defined as (in block form).

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1l}B \\ a_{21}B & a_{22}B & \dots & a_{2l}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1}B & a_{k2}B & \dots & a_{kl}B \end{pmatrix}$$

For vectors $\mathbf{x}=(x_1, \dots, x_n)$ and $\mathbf{y}=(y_1, \dots, y_n)$, we can hence write (identifying 1×1 matrices and real numbers)

$$\langle \mathbf{x} \| \mathbf{y} \rangle = (x_1^*, x_2^*, \dots, x_n^*) \otimes \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = x_1^*y_1 + x_2^*y_2 + \dots + x_n^*y_n$$

which is exactly the same as given by the definition of inner product $\langle \mathbf{x} | \mathbf{y} \rangle$.

The tensor product, on the other hand,

$$| \mathbf{x} \rangle \langle \mathbf{y} | = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \otimes (y_1^*, y_2^*, \dots, y_n^*) = \begin{pmatrix} x_1 y_1^* & x_1 y_2^* & \dots & x_1 y_n^* \\ x_2 y_1^* & x_2 y_2^* & \dots & x_2 y_n^* \\ \vdots & \vdots & \ddots & \vdots \\ x_n y_1^* & x_n y_2^* & \dots & x_n y_n^* \end{pmatrix} \quad (3)$$

is not so evident, but also this matrix has a clear interpretation. Indeed, [Definition 3](#) speaks about the *linear mapping*, whose matrix is in line ([Eq. 3](#)).

Definition 3 The set of linear mappings $H_n \rightarrow H_n$ is denoted by $L(H_n)$. For any pair of vectors $\mathbf{x}, \mathbf{y} \in H_n$ we define $| \mathbf{x} \rangle \langle \mathbf{y} | \in L(H_n)$ by

$$| \mathbf{x} \rangle \langle \mathbf{y} | \mathbf{z} = \langle \mathbf{y} | \mathbf{z} \rangle | \mathbf{x} \rangle$$

As mentioned, the matrix ([Eq. 3](#)) is the matrix of linear mapping $| \mathbf{x} \rangle \langle \mathbf{y} |$, with respect to the natural basis (i.e., the basis $\mathbf{e}_1=(1, 0, \dots, 0), \dots, \mathbf{e}_n=(0, 0, \dots, 1)$). Recalling that $| \mathbf{z} \rangle$ actually represents the column vector consisting of the coordinates of \mathbf{z} (with respect to the natural basis), we can also write

$$| \mathbf{x} \rangle \langle \mathbf{y} | \mathbf{z} = \langle \mathbf{y} | \mathbf{z} \rangle | \mathbf{x} \rangle \quad (4)$$

which differs from Definition 3 only in the following way: in Definition 3, $| \mathbf{x} \rangle \langle \mathbf{y} |$ stands for the *linear mapping*, whereas in ([Eq. 4](#)), the same notation stands for a matrix. While mathematically the linear mapping and its matrix are certainly distinct objects, it is customary to identify them when the basis is fixed and clear from the context. Unless otherwise stated, this identification will be used in what follows.

For later purposes, special attention should be paid to mapping $| \mathbf{x} \rangle \langle \mathbf{x} |$ (assuming $\| \mathbf{x} \| = 1$). As $| \mathbf{x} \rangle \langle \mathbf{x} | \mathbf{y} = \langle \mathbf{x} | \mathbf{y} | \mathbf{x} \rangle$, it is obvious that $| \mathbf{x} \rangle \langle \mathbf{x} |$ is the projection onto the one-dimensional space spanned by \mathbf{x} .

When given a fixed orthonormal basis $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ of H_n , one can represent any mapping $A \in L(H_n)$ uniquely as

$$A = \sum_{i=1}^n \sum_{j=1}^n A_{ij} | \mathbf{e}_i \rangle \langle \mathbf{e}_j | \quad (5)$$

where $A_{ij} = \langle \mathbf{e}_i | A \mathbf{e}_j \rangle$. Equation (2.5) is hence nothing more than the matrix representation of mapping A (with respect to basis $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$). It is evident that also $L(H_n)$ is a vector space, and from representation (2.5) one can directly read its dimension, which is n^2 . The advantage of representation (2.5) lies in the fact that, unlike in the ordinary matrix representation, here we have also the basis \mathbf{e}_i explicitly shown.

Definition 4 The adjoint mapping A^* of A is defined by $\langle \mathbf{x} | A \mathbf{y} \rangle = \langle A^* \mathbf{x} | \mathbf{y} \rangle$ for any vectors $\mathbf{x}, \mathbf{y} \in L(H_n)$.

In the matrix form, the adjoint mapping is easy to recover

$$\left(\sum_{i=1}^n \sum_{j=1}^n A_{ij} |\mathbf{e}_i\rangle \langle \mathbf{e}_j| \right)^* = \sum_{i=1}^n \sum_{j=1}^n A_{ji}^* |\mathbf{e}_i\rangle \langle \mathbf{e}_j|$$

which means that the matrix of an adjoint mapping is obtained by transposing and conjugating the original matrix. It is an easy exercise to verify that taking the adjoint mapping is a linear mapping $L(H_n) \rightarrow L(H_n)$, that is, an element of $L(L(H_n))$. This means that $(A+B)^* = A^* + B^*$, and $(cA)^* = c^* A^*$. In addition to that, it is easy to see that $(AB)^* = B^* A^*$. Moreover, the definitions presented above imply directly that $A|\mathbf{x}\rangle \langle \mathbf{x}|B^* = |A\mathbf{x}\rangle \langle B\mathbf{y}|$, no matter which particular interpretation (matrices of mappings) the mentioned objects possess.

Definition 5 Mapping $A \in L(H_n)$ is self-adjoint, if $A^* = A$.

The following analogy may be useful sometimes: Self-adjoint mappings relate to $L(H_n)$ as real numbers relate to complex numbers. To be more precise, we have the following representation theorem.

Theorem 2 For any $A \in L(H_n)$, there are unique self-adjoint mappings B and C so that

$$A = B + iC.$$

Proof If such a presentation exists, then $A^* = B - iC$, and addition gives $B = \frac{1}{2}(A + A^*)$, whereas subtraction shows $C = \frac{1}{2i}(A - A^*)$. This proves the existence and uniqueness of self-adjoint mappings B and C .

A polar representation analogous to $z = |z| e^{i\theta}$ also exists, but will be represented later. At this moment, it is worth mentioning that the analogy between real numbers and self-adjoint mappings becomes more concrete in the following proposition.

Proposition 1 All eigenvalues of a self-adjoint mapping $A \in L(H_n)$ are real.

Proof Let λ be an eigenvalue of A and \mathbf{x} an eigenvector belonging to λ . Then

$$\lambda \langle \mathbf{x} | \mathbf{x} \rangle = \langle \mathbf{x} | \lambda \mathbf{x} \rangle = \langle \mathbf{x} | A \mathbf{x} \rangle = \langle A^* \mathbf{x} | \mathbf{x} \rangle = \langle A \mathbf{x} | \mathbf{x} \rangle = \langle \lambda \mathbf{x} | \mathbf{x} \rangle = \lambda^* \langle \mathbf{x} | \mathbf{x} \rangle$$

which implies $\lambda = \lambda^*$.

Definition 6 A mapping $A \in L(H_n)$ is positive, if $\langle \mathbf{x} | A \mathbf{x} \rangle \geq 0$ for each $\mathbf{x} \in L(H_n)$.

In the terminology more familiar to mathematicians, “positive” is replaced with “positive semidefinite.” It is easy to see that the converse to the above theorem also holds: if $\langle \mathbf{x} | A \mathbf{x} \rangle \in \mathbb{R}$

for each $\mathbf{x} \in H_n$, then A is necessarily self-adjoint. Hence Definition 6 necessarily speaks only about self-adjoint mappings. The previously mentioned analogy has its extension here: Positive mappings relate to self-adjoint mappings as the nonnegative real numbers relate to all real numbers, since it is easy to see that a mapping is positive if and only if all of its eigenvalues are nonnegative.

One more definition is needed before introducing the quantum mechanical counterpart of the state vector (\mathbf{x}, \mathbf{p}) of classical mechanics.

Definition 7 For any mapping $A \in L(H_n)$, the *trace* is defined as

$$\text{Tr}(A) = \sum_{i=1}^n \langle \mathbf{e}_i | A \mathbf{e}_i \rangle$$

where $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ is an orthonormal basis of H_n .

It is possible to show that the trace is independent of the orthonormal basis chosen, and by the definition the trace equals the sum of the diagonal elements of the matrix. The trace also equals the sum of the eigenvalues (see Axler (1997), for example).

Definition 8 A *state* of an n -level quantum system is a self-adjoint, unit-trace, positive mapping $A \in L(H_n)$. A *density matrix* is a matrix representation of a state.

Recall that in classical mechanics one can represent the state of a physical system as $(\mathbf{x}, \mathbf{p}) \in \mathbb{R}^{2n}$, where \mathbf{x} stands for the (generalized) coordinates and \mathbf{p} for the (generalized) momentum. In quantum mechanics, a state is represented as linear mapping $A \in L(H_n)$ satisfying the conditions of Definition 8.

However, it should be noted carefully that n in the description of classical mechanics has a very different role than n in quantum mechanics. Recall that in classical descriptions n (the number of coordinates of \mathbf{x} and \mathbf{p}) stands for the degrees of freedom of the system under investigation, and in quantum mechanics n , the dimension of H_n , stands for the maximal number of perfectly distinguishable measurement results that can be extracted from the system, for a suitably chosen measurement. The Hilbert space H_n can stand for a *single* degree of freedom: It accommodates the mathematical structure describing how the value of anything observable on this specific degree of freedom is produced.

To describe *various* degrees of freedom in quantum mechanics, one should set up a Hilbert space individually for any such degree. The total description is then in the Hilbert space obtained as a *tensor product* of all “component spaces.” It should be noted that usually observables with an infinite number of potential values (and consequently infinite-dimensional Hilbert spaces) are needed, but in the context of quantum computing physical objects with a finite number of potential values are satisfactory: to describe a *bit*, one needs only a physical system capable of existing in two perfectly distinguishable states.

In the classical description (\mathbf{x}, \mathbf{p}) the value of any observable can be calculated directly from the coordinates of \mathbf{x} and \mathbf{p} . In quantum mechanics, the setup is dramatically different: quantum mechanics is a fundamentally stochastic theory, and for any state of the system there are always observables with values not definitely prescribed, but obtained stochastically. Conversely, for any observable, there are even *pure states* (introduced later) of the system for which the measurement result is not determined uniquely.

A very important structural property of the state set is given in the following proposition, whose proof is actually straightforward from the definitions.

Proposition 2 All states of an n -level quantum system form a convex set, meaning that if S_1 and S_2 are states then their convex combination

$$pS_1 + (1 - p)S_2$$

with $p \in [0,1]$ also is a state.

The above proposition actually expresses the principle that it is possible to obtain new states from previous ones as a *stochastic mixture*: the new state $pS_1 + (1 - p)S_2$ can be seen as a description of the stochastic process, where S_1 occurs with probability p , and S_2 with a complementary probability $1 - p$.

The *extreme points* of the state set deserve special attention.

Definition 9 State S is *pure* if it cannot be presented as a convex combination $S = pS_1 + (1 - p)S_2$, where $S_1 \neq S_2$. A state that is not pure is *mixed*.

There exists a well-known and simple mathematical characterization for pure quantum states, but to introduce that we need to define one more notion.

Definition 10 Mapping $A \in L(H_n)$ is an (*orthogonal*) *projection* if $A^* = A$ and $A^2 = A$.

Hereafter, the attribute “orthogonal” will be ignored as all projections treated in this chapter will be indeed orthogonal. It can be shown that there is one-to-one correspondence between projections and subspaces of H_n . This connection can be briefly explained as follows: for any subspace $W \subseteq H_n$ there exists an *orthogonal complement* W^\perp , also a subspace of H_n , which has the property that any $\mathbf{x} \in W$ and any $\mathbf{y} \in W^\perp$ are orthogonal: $\langle \mathbf{x} | \mathbf{y} \rangle = 0$. Moreover, H_n can be represented as a *direct sum* of the subspaces, $H_n = W \oplus W^\perp$, which means that each $\mathbf{x} \in H_n$ can be uniquely represented as $\mathbf{x} = \mathbf{x}_W + \mathbf{x}_{W^\perp}$, where $\mathbf{x}_W \in W$ and $\mathbf{x}_{W^\perp} \in W^\perp$. Now if A is a projection, then there exists a unique subspace W so that $A\mathbf{x} = \mathbf{x}_W$. Conversely, for any subspace $W \subseteq H_n$ there exists a unique projection A so that $A\mathbf{x} = \mathbf{x}_W$.

We conclude this section with the well-known characterization of pure states.

Proposition 3 A state $A \in L(H_n)$ of a quantum system is pure if $A = |\mathbf{x}\rangle\langle \mathbf{x}|$ is a projection onto a one-dimensional subspace (here $\|\mathbf{x}\| = 1$).

3 Spectral Representation

Spectral representation, also known as *spectral decomposition*, is a very powerful tool of matrix theory. It also provides a representation for an arbitrary quantum state as a convex combination of pure states, as will be shortly seen.

Definition 11 A matrix A is *normal* if it commutes with its adjoint, that is $AA^* = A^*A$.

Especially, self-adjoint mappings are normal, since then $AA^* = A^2 = A^*A$. Normal matrices have the property that they can be diagonalized unitarily, meaning that there exists a possibility to rotate the natural basis of H_n so that the matrix in the new basis is diagonal. Indeed, we understand “rotate” here a bit more generally, meaning that for any normal

matrix A , there is a unitary matrix U (defined later in [Section 7.1.1](#)) so that UAU^* is diagonal. This can also be expressed in the following theorem.

Theorem 3 (Spectral representation) *Let $A \in L(H_n)$ be normal. Then there exists an orthonormal basis $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of H_n so that*

$$A = \lambda_1 |\mathbf{x}_1\rangle\langle\mathbf{x}_1| + \dots + \lambda_n |\mathbf{x}_n\rangle\langle\mathbf{x}_n| \quad (6)$$

It is noteworthy that in the spectral representation numbers λ_i are the eigenvalues of A and \mathbf{x}_i is an eigenvector belonging to λ_i (the set of the eigenvalues is also called the *spectrum* of the matrix or a mapping). Equation [\(6\)](#) hence simply means that the spectral representation corresponds to choosing an orthonormal basis $\mathbf{x}_1, \dots, \mathbf{x}_n$ in which the matrix A is diagonal. The above theorem just states that for normal matrices (or mappings as well), there always exists an orthonormal basis allowing a diagonal presentation. The proof of the spectral representation theorem for *self-adjoint* mappings can be found in Axler ([1997](#)), for instance, and its extension to normal mappings is quite an easy exercise when we know that all mappings can be represented as $C = A + iB$, where A and B are self-adjoint.

Definition 12 Let $f : \mathbb{C} \rightarrow \mathbb{C}$ be an arbitrary function. Then f is defined on normal mappings by

$$f(A) = f(\lambda_1) |\mathbf{x}_1\rangle\langle\mathbf{x}_1| + \dots + f(\lambda_n) |\mathbf{x}_n\rangle\langle\mathbf{x}_n| \quad (7)$$

if A has spectral representation [\(6\)](#).

It is an easy exercise to show that the above definition is independent of the choice of the spectral representation.

Example 1 Matrix $\begin{pmatrix} \frac{\pi}{2} & 0 \\ 0 & -\frac{\pi}{2} \end{pmatrix}$ defines a normal mapping. By fixing the basis of H_2 as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ we can represent this mapping as $A = \frac{\pi}{2} \cdot |0\rangle\langle 0| - \frac{\pi}{2} \cdot |1\rangle\langle 1|$. Now

$$e^{iA} = e^{i\frac{\pi}{2}}|0\rangle\langle 0| + e^{i(-\frac{\pi}{2})}|1\rangle\langle 1| = i|0\rangle\langle 0| - i|1\rangle\langle 1|$$

which can be represented as a matrix $\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$ as well.

As the states of quantum systems are depicted by self-adjoint, positive, unit-trace operators, we will be interested especially in their representations. Since self-adjoint mappings are normal, they admit a spectral representation. Moreover, it should be noted that the eigenvalues λ of self-adjoint operators in presentation [\(6\)](#) are real.

The positivity $\langle \mathbf{x} | Ax \rangle \geq 0$ always implies that for each i , $\lambda_i \geq 0$. As representation [\(6\)](#) corresponds to a diagonal matrix, it is also easy to verify that $\text{Tr}(A) = \lambda_1 + \dots + \lambda_n$. We can conclude this as follows:

- ▶ A state A of an n -level quantum system can be represented as

$$A = \lambda_1 |\mathbf{x}_1\rangle\langle\mathbf{x}_1| + \dots + \lambda_n |\mathbf{x}_n\rangle\langle\mathbf{x}_n| \quad (8)$$

where $\lambda_i \geq 0$, and $\lambda_1 + \dots + \lambda_n = 1$. This is to say that each state $A \in L(H_n)$ can be represented as a convex combination of at most n pure states, and, moreover, the pure states can be found by searching the eigenvectors of A .

Representation (2.8) may recommend that we interpret a general quantum state A as a probability distribution over pure states $|\mathbf{x}_1\rangle\langle\mathbf{x}_1|, \dots, |\mathbf{x}_n\rangle\langle\mathbf{x}_n|$. Unfortunately, representation (2.8) is not always unique.

Example 2 Let $|0'\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1'\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Then

$$\begin{aligned} |0'\rangle\langle 0'| + |1'\rangle\langle 1'| &= \left| \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right\rangle \left\langle \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right| \\ &\quad + \left| \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \right\rangle \left\langle \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \right| \\ &= \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|0\rangle\langle 1| + \frac{1}{2}|1\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \\ &\quad + \frac{1}{2}|0\rangle\langle 0| - \frac{1}{2}|0\rangle\langle 1| - \frac{1}{2}|1\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \\ &= |0\rangle\langle 0| + |1\rangle\langle 1| \end{aligned}$$

Let $\|\mathbf{x}\| = 1$. A simple calculation shows that $|\mathbf{ax}\rangle\langle\mathbf{ax}| = |a|^2|\mathbf{x}\rangle\langle\mathbf{x}|$, which implies that vectors \mathbf{x} and $e^{i\theta}\mathbf{x}$ (with $\theta \in \mathbb{R}$) specify exactly the same projection onto a one-dimensional vector space. Recall that those projections are special cases of quantum states (pure states), and they play an important role in quantum computing and in general quantum theory: most quantum computing models are based on pure states only. Quantum information processing in general will of course require more than pure states.

A pure state $A = |\mathbf{x}\rangle\langle\mathbf{x}|$ is often identified with a unit-length *state vector* or *vector state* \mathbf{x} , but it should be noted that this identification is not unambiguous: Vectors $e^{i\theta}\mathbf{x}$ with $\theta \in \mathbb{R}$ are all vector representations of the same pure state.

A notion constantly present in quantum physics (as well as in classical wave theory) is that of *superposition*.

Definition 13 Let $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be an orthonormal basis of H_n and

$$\mathbf{x} = \alpha_1\mathbf{x}_1 + \dots + \alpha_n\mathbf{x}_n$$

a unit-length vector describing a pure state. We say then that \mathbf{x} is a *superposition* of vector states $\mathbf{x}_1, \dots, \mathbf{x}_n$ with *amplitudes* $\alpha_1, \dots, \alpha_n$.

In classical mechanics, one can always introduce a mixed state when two pure states T_1 and T_2 are given: The new mixed state is a statistical combination $pT_1 + (1-p)T_2$ of the pure ones, meaning that T_1 is present with probability p , and T_2 with the complementary probability $1-p$.

In quantum mechanics, one can also create mixtures from the pure state analogously to the classical case, but a very important nonclassical feature is that from given pure states one can form another pure state by superposition: If \mathbf{x}_1 and \mathbf{x}_2 are perpendicular unit-length vectors representing pure states $|\mathbf{x}_1\rangle\langle\mathbf{x}_1|$ and $|\mathbf{x}_2\rangle\langle\mathbf{x}_2|$, then their linear combination $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2$ (where $|\alpha_1|^2 + |\alpha_2|^2 = 1$) is again a pure (vector) state, the superposition of vector states \mathbf{x}_1 and \mathbf{x}_2 .

A simple calculation shows that

$$\begin{aligned} &|\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2\rangle\langle\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2| \\ &= |\alpha_1|^2|\mathbf{x}_1\rangle\langle\mathbf{x}_1| + \alpha_1\alpha_2^*|\mathbf{x}_1\rangle\langle\mathbf{x}_2| + \alpha_2\alpha_1^*|\mathbf{x}_2\rangle\langle\mathbf{x}_1| + |\alpha_2|^2|\mathbf{x}_2\rangle\langle\mathbf{x}_2| \end{aligned}$$

Term $\alpha_1\alpha_2^*|\mathbf{x}_1\rangle\langle\mathbf{x}_2| + \alpha_2\alpha_1^*|\mathbf{x}_2\rangle\langle\mathbf{x}_1|$ in the above superposition is called then an *interference term*, whereas the remaining term

$$|\alpha_1|^2|\mathbf{x}_1\rangle\langle\mathbf{x}_1| + |\alpha_2|^2|\mathbf{x}_2\rangle\langle\mathbf{x}_2|$$

is indeed a statistical mixture of the original states $|\mathbf{x}_1\rangle\langle\mathbf{x}_1|$ and $|\mathbf{x}_2\rangle\langle\mathbf{x}_2|$.

To create a superposition state, unfortunately the vector representatives of pure states $|\mathbf{x}_1\rangle\langle\mathbf{x}_1|$ and $|\mathbf{x}_2\rangle\langle\mathbf{x}_2|$ are needed. Recall that for example \mathbf{x}_2 and $-\mathbf{x}_2$ both define the same pure state, meaning that $|\mathbf{x}_2\rangle\langle\mathbf{x}_2| = |-\mathbf{x}_2\rangle\langle-\mathbf{x}_2|$, but nevertheless the state vectors $\frac{1}{\sqrt{2}}\mathbf{x}_1 + \frac{1}{\sqrt{2}}\mathbf{x}_2$ and $\frac{1}{\sqrt{2}}\mathbf{x}_1 - \frac{1}{\sqrt{2}}\mathbf{x}_2$ obtained from the different representations for $|\mathbf{x}_2\rangle\langle\mathbf{x}_2|$ are completely different objects.

4 The Meaning of the Quantum States

Quantum physics is structurally a stochastic theory. The stochastic nature comes into the picture when observing a quantum system in a fixed state; there is not usually a single outcome, but only a probability distribution of potential outcomes. By saying that the stochastic nature is structurally built in quantum theory, we mean that the outcome distribution instead of a single outcome does not emerge from the ignorance: even the most accurate knowledge of the system does not admit more than a distribution of the outcomes. The pure states correspond, in a sense, to maximal information about the quantum system, but even in a pure state one must accept a probability distribution of the potential outcomes.

In this section, the *minimal interpretation* of quantum mechanics will be presented, which actually is an axiom describing how the probabilities will be calculated from the mathematical objects describing quantum systems. For that purpose, we need to describe how *observables* are specified in this formalism. This is of course similar to classical physics: For instance, pair (\mathbf{x}, \mathbf{p}) describes the state of a classical system, but the value of the position is revealed from the state by projection $(\mathbf{x}, \mathbf{p}) \mapsto \mathbf{x}$.

Definition 14 An *observable* of a quantum system H_n is a collection P_1, \dots, P_k of projections onto mutually orthogonal subspaces so that $P_1 + \dots + P_k = I$ (the identity mapping), equipped with a collection of real numbers $\lambda_1, \dots, \lambda_k$.

Another equivalent viewpoint is offered in the following definition.

Definition 15 An *observable* of a quantum system H_n is a self-adjoint mapping $A \in L(H_n)$.

It is not difficult to see that the above two definitions speak about the same object. Indeed, if A is a self-adjoint mapping, then it admits a spectral representation

$$A = \sum_{k=1}^n \lambda_k |\mathbf{x}_k\rangle\langle\mathbf{x}_k|$$

which can also be written in the form

$$A = \sum_{k=1}^{n'} \lambda'_k P_k$$

where $\lambda'_1, \dots, \lambda'_{n'}$ are the distinct eigenvalues of A , and

$$P_k = \sum_{l=1}^{d_l} |\mathbf{x}_{k_l}\rangle\langle\mathbf{x}_{k_l}|$$

is the projection onto the eigenspace V_k of λ'_k . As eigenspaces belonging to distinct eigenvalues of a self-adjoint operator A are orthogonal (Axler 1997), the projections satisfy the orthogonality required in the first definition.

On the other hand, a collection P_1, \dots, P_k of projections of H_n onto orthonormal subspaces of H_n together with real numbers $\lambda_1, \dots, \lambda_k$ evidently define a self-adjoint mapping

$$A = \lambda_1 P_1 + \dots + \lambda_k P_k$$

If $P_1 + \dots + P_k \neq I$, then another projection $P_{k+1} = I - (P_1 + \dots + P_k)$ with $\lambda_{k+1} = 0$ is introduced to satisfy the requirement $P_1 + \dots + P_{k+1} = I$.

Remark 1 The notion of “a collection of projections equipped with real numbers” can be expressed by using the notion of *projection-valued measure*. A projection-valued measure on the real line is a mapping $E : \mathbb{R} \rightarrow L(H_n)$ which, in the case of Definition 14, satisfies $E(\lambda_i) = P_i$ and $E(\lambda) = 0$ for $\lambda \notin \{\lambda_1, \dots, \lambda_k\}$. In general, a projection-valued measure must satisfy conditions similar to an ordinary measure on a real line (see Hirvensalo (2004), for instance).

Whichever definition is used for an observable, the intuitive meaning will be that the values $\lambda_1, \dots, \lambda_k$ are the *potential* values of observable A , and probability distributions of the values are expressed in the following definition.

Definition 16 (Minimal interpretation of quantum mechanics)

Let $T \in L(H_n)$ be a state of a quantum system and $A = \lambda_1 P_1 + \dots + \lambda_k P_k$ be an observable, where P_i are mutually orthogonal projections summing up to the identity mapping. Then the probability that λ_i will be observed when the system is in state T is

$$\mathbb{P}_T(\lambda_i) = \text{Tr}(P_i T)$$

The minimal interpretation has its mathematical basis in Gleason’s theorem (Gleason 1957), which shows that for dimension $n \geq 3$ all probability measures in $L(H_n)$ are of the above form.

Example 3 The presentation of an observable as a self-adjoint mapping A is useful when talking about the *expected value* of the observable. Indeed, the expectation (in state T) is given by

$$\mathbb{E}_T(A) = \sum_{i=1}^k \mathbb{P}_T(\lambda_i) \lambda_i = \sum_{i=1}^k \lambda_i \text{Tr}(P_i T) = \text{Tr}(AT)$$

$$\text{since } A = \sum_{i=1}^k \lambda_i P_i.$$

The mathematical formalism presented so far implies rather easily quite deep consequences. One such is the famous *Heisenberg uncertainty relation* (see Hirvensalo (2004) for the proof of the next proposition).

Proposition 4 (Uncertainty relation) Let A and B be any observables. Then the product of variances $\mathbb{V}_T(A)$ and $\mathbb{V}_T(B)$ in a pure state $T = |\mathbf{x}\rangle\langle \mathbf{x}|$ can be bounded below as

$$\mathbb{V}_T(A)\mathbb{V}_T(B) \geq \frac{1}{4}|\langle \mathbf{x}|[A, B]\mathbf{x}\rangle|^2$$

where $[A, B] = AB - BA$ is the commutator of A and B .

The above uncertainty relation, however, has its greatest importance in infinite-dimensional vector spaces: in such cases it is possible that $[A, B] = cI$, a multiple of the identity mapping, and then the lower bound would be state-independent. A traditional example is the commutator of the position and momentum $[X, P] = i\hbar I$.

Let $\mathbf{x} \in H_n$ be a unit-length vector and $T = |\mathbf{x}\rangle\langle \mathbf{x}|$ a pure state determined by \mathbf{x} . Fix also an observable $A = \lambda_1 |\mathbf{x}_1\rangle\langle \mathbf{x}_1| + \dots + \lambda_n |\mathbf{x}_n\rangle\langle \mathbf{x}_n|$ with n distinct eigenvalues (potential values of observable A). As $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is an orthonormal basis of H_n , we can always write

$$\mathbf{x} = \alpha_1 \mathbf{x}_1 + \dots + \alpha_n \mathbf{x}_n$$

and the probability to observe λ_i is given by

$$P(\lambda_i) = \text{Tr}(|\mathbf{x}_i\rangle\langle \mathbf{x}_i||\mathbf{x}\rangle\langle \mathbf{x}|) = |\alpha_i|^2$$

as easily verified. To express this in other words: assume that a quantum system is in a vector state

$$\mathbf{x} = \alpha_1 \mathbf{x}_1 + \dots + \alpha_n \mathbf{x}_n$$

and the observable studied is

$$A = \lambda_1 |\mathbf{x}_1\rangle\langle \mathbf{x}_1| + \dots + \lambda_n |\mathbf{x}_n\rangle\langle \mathbf{x}_n|$$

(distinct eigenvalues λ_i). Then the probability that value λ_i is observed is $|\alpha_i|^2$. For such an observable, it is also typical to omit also the *value* of the observable and merely to refer to the vector \mathbf{x}_i (intuitively one could think about observing \mathbf{x}_i instead of value λ_i) generating a one-dimensional subspace of H_n . The following version of the minimal interpretation is written for pure states, the aforementioned special type of observables, and for ignoring the (eigen)values of the observables.

Definition 17 (Minimal interpretation, special version) Let

$$\mathbf{x} = \alpha_1 \mathbf{x}_1 + \dots + \alpha_n \mathbf{x}_n$$

be a vector representation of a pure quantum state. Then the probability of observing \mathbf{x}_i is $|\alpha_i|^2$.

5 Quantum Bits

When speaking about computation, bits are a most interesting object. Therefore it is worth devoting a section to their quantum physical representations, *quantum bits* or *qubits* for short.

A physical system capable of representing a bit must have two distinguishable states, but there are no further restrictions on the nature of the system. The mathematical description of the system therefore lives in H_2 , two-dimensional Hilbert space.

Definition 18 A *computational basis* of H_2 refers to a fixed orthonormal basis $\{|0\rangle, |1\rangle\}$ of H_2 .

The above definition refers rather to a physical nature and interpretation of the quantum system than to any mathematical properties of H_2 . Indeed, in a physical realization there may be some preferred states that are more natural to interpret as 0 and 1 than any other states. For instance, if representing quantum bits with electron energy states, one may want to associate logical zero with the ground state, and logical one with an excited state.

Mathematically, it is always possible to construct H_2 as the Cartesian product $\mathbb{C} \times \mathbb{C}$ and choose $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Of course, any other choice for the computational basis is equally good, and hence the notion “computational basis” should be understood only contextually. However, in this chapter the aforementioned choice is used for the computational basis.

A *state* of the qubit is, according to the previous definition, a self-adjoint, positive, unit-trace mapping in $L(H_2)$. When thinking about the matrix representation

$$S = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a|0\rangle\langle 0| + b|0\rangle\langle 1| + c|1\rangle\langle 0| + d|1\rangle\langle 1|$$

one sees that the self-adjointness immediately imposes restrictions such as $a, d \in \mathbb{R}$, and that $c = b^*$. Furthermore, from the unit trace condition we see that $a + d = 1$, so a general form of a self-adjoint mapping with unit trace must be of the form

$$S = \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix}$$

where $a \in \mathbb{R}$. Hence there are three real numbers (a and the real and imaginary parts of b) that specify a unit-trace self-adjoint mapping. The positivity obviously imposes extra conditions, and for finding them we choose a slightly different approach.

Definition 19 The *Pauli spin matrices* are defined as follows:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Yet adding the identity matrix $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ we get an interesting set: it is easy to prove that $\{I, \sigma_x, \sigma_y, \sigma_z\}$ is a basis of $L(H_2)$, when regarding $L(H_2)$ as a vector space over \mathbb{C} . Especially this means that any 2×2 complex matrix S can be uniquely represented as

$$S = wI + x\sigma_x + y\sigma_y + z\sigma_z \quad (9)$$

where w, x, y , and $z \in \mathbb{C}$. Furthermore, all matrices in $\{I, \sigma_x, \sigma_y, \sigma_z\}$ are self-adjoint, which implies that

$$S^* = w^*I + x^*\sigma_x + y^*\sigma_y + z^*\sigma_z$$

Now because the representation is unique, we conclude that if S is self-adjoint, then necessarily coefficients w, x, y , and z in [Eq. 9](#) are real. A further observation shows that σ_x, σ_y , and σ_z have zero trace, meaning that the trace of [Eq. 9](#) is $\text{Tr}(wI) = 2w$. This shows that a unit-trace, self-adjoint mapping in $L(H_2)$ has a unique representation

$$S = \frac{1}{2}(I + x\sigma_x + y\sigma_y + z\sigma_z) \quad (10)$$

where x, y , and $z \in \mathbb{R}$. This is perfectly compatible with the aforementioned fact that unit-trace self-adjoint mappings in $L(H_2)$ have three real degrees of freedom: via representation (Eq. 10) we can identify such a mapping with a point in \mathbb{R}^3 , but the restrictions imposed by the positivity condition are still to be discovered.

For that, we notice first that the spectral representation directly implies that a matrix is positive if and only if its eigenvalues are nonnegative. To discover the characteristic polynomial of (Eq. 10), we write explicitly

$$S = \frac{1}{2}(I + x\sigma_x + y\sigma_y + z\sigma_z) = \frac{1}{2} \begin{pmatrix} 1+z & x-iy \\ x+iy & 1-z \end{pmatrix}$$

and find the roots of the characteristic polynomial

$$\lambda = \frac{1}{2} \left(1 \pm \sqrt{x^2 + y^2 + z^2} \right) \quad (11)$$

Both roots are nonnegative if and only if $x^2 + y^2 + z^2 \leq 1$, and hence we see that there is one-to-one correspondence between the points of the unit sphere in \mathbb{R}^3 and pure states of a qubit.

Definition 20 The *Bloch sphere* (also known as the *Poincaré sphere*) is the image of the sphere

$$\{(x, y, z) \mid x^2 + y^2 + z^2 \leq 1\} \subseteq \mathbb{R}^3$$

in $L(H_2)$ under mapping

$$(x, y, z) \mapsto \frac{1}{2}(I + x\sigma_x + y\sigma_y + z\sigma_z) \quad (12)$$

Even more information is easily available: A pure state $|\mathbf{x}\rangle\langle\mathbf{x}|$ is a projection onto a one-dimensional subspace, and the sole eigenvalues of such a mapping are hence 1 and 0. By (Eq. 11) this is possible only if $x^2 + y^2 + z^2 = 1$. This fact, which would as well follow from the convexity-preserving nature of mapping (Eq. 12), shows that the *pure states of a quantum bit lie exactly on the surface of the Bloch sphere*. By using the standard spherical coordinate representation, we can say that the pure states of a quantum bit have two real degrees of freedom; indeed the use of spherical coordinates result in a vector state $\cos\theta|0\rangle + e^{i\phi}\sin\theta|1\rangle$.

Example 4 Pure state $|0\rangle$ corresponds to the projection $|0\rangle\langle 0|$, whose matrix representation is

$$(1, 0) \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}(I + \sigma_z) = \frac{1}{2}(I + 0 \cdot \sigma_x + 0 \cdot \sigma_y + 1 \cdot \sigma_z)$$

so the Bloch sphere point $(0, 0, 1)$ corresponds to the pure state $|0\rangle$. This point is referred to as the *north pole* of the Bloch sphere. Analogously one can see that the *south pole* of the Bloch sphere corresponds to the pure state $|1\rangle$.

Let us find the point corresponding to the pure state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The density matrix of this state is

$$\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \frac{1}{2}(I + \sigma_x) = \frac{1}{2}(I + 1 \cdot \sigma_x + 0 \cdot \sigma_y + 0 \cdot \sigma_z)$$

so the point $(1, 0, 0)$ corresponding to this state is found on the equator of the Bloch sphere.

For finding the point corresponding to $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$, we first find the matrix

$$\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} = \frac{1}{2}I + 0 \cdot \sigma_x + 0 \cdot \sigma_y + 0 \cdot \sigma_z$$

which means that the center $(0,0,0)$ corresponds to the mixed state $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$.

What is then, for example, the pure state corresponding to the equator point $(0,1,0)$? This is easily found, as

$$\frac{1}{2}(I + 0 \cdot \sigma_x + 1 \cdot \sigma_y + 0 \cdot \sigma_z) = \frac{1}{2}\begin{pmatrix} 1 & -i \\ i & 1 \end{pmatrix}$$

This matrix is self-adjoint, so it admits a spectral representation, which can be found by finding its eigenvectors: Vector $\frac{1}{\sqrt{2}}\begin{pmatrix} i \\ 1 \end{pmatrix}$ belongs to the eigenvalue 0, and $\frac{1}{\sqrt{2}}\begin{pmatrix} -i \\ 1 \end{pmatrix}$ to eigenvalue 1. Hence (note that the factor $\frac{1}{2}$ is omitted now)

$$\begin{aligned} \begin{pmatrix} 1 & -i \\ i & 1 \end{pmatrix} &= 0 \cdot \begin{pmatrix} i \\ 1 \end{pmatrix}^* \otimes \begin{pmatrix} i \\ 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} -i \\ 1 \end{pmatrix}^* \otimes \begin{pmatrix} -i \\ 1 \end{pmatrix} \\ &= 0 \cdot (-i, 1) \otimes \begin{pmatrix} i \\ 1 \end{pmatrix} + 1 \cdot (i, 1) \otimes \begin{pmatrix} -i \\ 1 \end{pmatrix} \end{aligned}$$

and we can see that the state vector corresponding to point $(0,1,0)$ can be chosen as $\frac{1}{\sqrt{2}}\begin{pmatrix} -i \\ 1 \end{pmatrix} = \frac{-i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.

6 Compound Systems

The notion of a *joint quantum system* or *compound quantum system* is established via tensor product construction.

Definition 21 Let \mathcal{S}_1 and \mathcal{S}_2 be quantum systems with state spaces H_n and H_m , respectively. Then the state space of the compound system \mathcal{S}_{12} is the mn -dimensional tensor product $H_n \otimes H_m$.

The tensor product is actually a very clever construction in algebra allowing us to represent bilinear mappings as linear ones preceded by a very natural bilinear embedding. However, this sophisticated algebraic definition is not needed to perform the necessary calculations. Hence we take a constructive approach and merely mention the following facts: if $\mathbf{x}_1, \dots, \mathbf{x}_n$ and $\mathbf{y}_1, \dots, \mathbf{y}_m$ are the orthonormal bases of H_n and H_m , respectively, then $\{\mathbf{x}_i \otimes \mathbf{y}_j \mid i \in \{1, \dots, n\}, j \in \{1, \dots, m\}\}$ is an orthonormal basis of $H_n \otimes H_m$ as the inner product takes the form

$$\langle \mathbf{x}_1 \otimes \mathbf{y}_1 \mid \mathbf{x}_2 \otimes \mathbf{y}_2 \rangle = \langle \mathbf{x}_1 \mid \mathbf{x}_2 \rangle \langle \mathbf{y}_1 \mid \mathbf{y}_2 \rangle$$

Any pair of linear mappings $A \in L(H_n)$, $B \in L(H_m)$ together define a linear mapping $A \otimes B \in L(H_n \otimes H_m)$ via

$$(A \otimes B)(\mathbf{x}_i \otimes \mathbf{y}_j) = A\mathbf{x}_i \otimes B\mathbf{y}_j$$

for basis vectors $\mathbf{x}_i \otimes \mathbf{y}_j$, and the extension to the whole space is straightforwardly done by linearity. It should already here be carefully noted that as there are vectors other than $\mathbf{x} \otimes \mathbf{y} \in H_n \otimes H_m$ (so-called *decomposable vectors*), there are also linear mappings other than those of the form $A \otimes B$ in $L(H_n \otimes H_m)$. In the next section, this very evident fact will be examined again, but another identity very useful for the continuation follows easily from the previous ones:

$$\mathrm{Tr}(A \otimes B) = \mathrm{Tr}(A)\mathrm{Tr}(B)$$

It must of course be mentioned here that in the field of quantum computing the tensor sign is very generally omitted. For example, let $\{|0\rangle, |1\rangle\}$ be an orthonormal basis of H_2 . Then the orthonormal basis of $H_4 \simeq H_2 \otimes H_2$ can be constructed as $\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$, but the abbreviations $\{|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle\}$ and $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ are very frequent.

Example 5 Using the notations $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ one can get concrete representations for the aforementioned vectors:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The tensor product of two vector spaces generalizes to that of three or more spaces by defining $H_n \otimes H_m \otimes H_l = H_n \otimes (H_m \otimes H_l)$, which can be shown to be isomorphic to $(H_n \otimes H_m) \otimes H_l$. Subsequently, the notations such as $|0\rangle \otimes |1\rangle \otimes |0\rangle$ are abbreviated in quantum computing as $|0\rangle|1\rangle|0\rangle$, or as $|010\rangle$.

7 Subsystem States

In order to call a system \mathcal{S}_1 a *subsystem* of a quantum system \mathcal{S} , it is necessary that \mathcal{S}_1 must be *identifiable* as a quantum system itself.

Let \mathcal{S}_1 and \mathcal{S}_2 be identifiable quantum systems, and S_1 and S_2 their states, respectively. From the previous section it is known that for example $S_1 \otimes S_2$ is an eligible state of the compound system \mathcal{S}_{12} . State $S_1 \otimes S_2$ is an example of a *decomposable state*.

Example 6 Let H_2 be equipped with an orthonormal basis $(|0\rangle, |1\rangle)$. Then the basis vectors $|0\rangle$ and $|1\rangle$ determine pure states of the system, and states other than these can be obtained for example by superposition: any vector $\alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$ determines a pure state as well.

In the tensor product $H_2 \otimes H_2$, states $|0\rangle \otimes |0\rangle$ (abbreviated as $|00\rangle$) and $|11\rangle$ are pure states, and so is their superposition $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. But unlike original states $|00\rangle$ and

$|11\rangle$, (pure) state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ cannot be represented as a tensor product of pure states in H_2 (which can be verified easily). It even turns out that the pure state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ cannot be represented in the form

$$\sum_i p_i S_i \otimes T_i$$

where S_i and T_i are states of H_2 (pure or mixed) and $\sum_i p_i = 1$. Pure state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

is hence called *entangled*.

- A state $V \in L(H_n \otimes H_m)$ is said to be *decomposable* if V can be expressed as a finite sum

$$V = \sum_i p_i S_i \otimes T_i$$

where $S_i \in L(H_n)$ and $T_i \in L(H_m)$. A state that is not decomposable is *entangled*.

The fact that the trace of a state must be 1 implies that $\sum_i p_i$ must equal to 1 in the above representation, if such a representation is possible. For pure states, the entanglement appears to be a bit simpler than for general states: in fact, a pure state $\mathbf{z} \in H_n \otimes H_m$ is decomposable if and only if \mathbf{z} is of the form $\mathbf{x} \otimes \mathbf{y}$ ($\mathbf{x} \in H_n$, $\mathbf{y} \in H_m$, and entangled otherwise).

Decomposable pure states of compound quantum systems are the easiest ones when determining the subsystem states: if $\mathbf{z} = \mathbf{x} \otimes \mathbf{y}$ is a state of a compound system, then the (pure) subsystem states are (trivially) \mathbf{x} and \mathbf{y} . On the other hand, for an entangled state it is not clear, from an algebraic viewpoint, how to fix the subsystem states, even when the compound system state is pure.

Example 7 Let $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ be a (pure) state of $H_2 \otimes H_2$. What should the subsystem states look like? As mentioned previously, it is not possible to find pure states $\mathbf{x}, \mathbf{y} \in H_2$ to satisfy $\mathbf{x} \otimes \mathbf{y} = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$.

The way to extract the subsystem states from the compound state is based on the statistical interpretation of quantum states. To begin with, consider any observable A on the subsystem \mathcal{S}_1 . Clearly A can be extended into an observable on the whole system \mathcal{S}_{12} as $A \otimes I$, whatever the observable B of the subsystem \mathcal{S}_2 , but choice $B = I$ serves a very special purpose. Indeed, the observable I on the subsystem \mathcal{S}_2 has always value 1 in any state, which means that statistically observable $A \otimes I$ of the compound system does not differ from the observable A of the subsystem \mathcal{S}_1 . With the minimal interpretation of quantum mechanics (Definition 16), this is enough to give the firm mathematical basis for the following definition.

Definition 22 Let \mathcal{S}_1 and \mathcal{S}_2 be quantum systems with state spaces H_n and H_m , respectively. Let also $S \in L(H_n \otimes H_m)$ be the state of a compound quantum system. Then the subsystem \mathcal{S}_1 state is $S_1 \in L(H_n)$, which satisfies

$$\text{Tr}(S_1 A) = \text{Tr}(S(A \otimes I))$$

for all observables $A \in L(H_n)$. We say that the subsystem state S_1 is obtained by *tracing over H_m* and denote

$$S_1 = \text{Tr}_{H_m}(S)$$

The states of the subsystem \mathcal{S}_2 have an analogous definition and notation. The result of tracing over is also known as a *partial trace*.

For the proof of the following proposition, see Hirvensalo (2004), for instance.

Proposition 5 *State S_1 of the previous definition exists and is unique. It can be expressed as*

$$\mathrm{Tr}_{H_m}(S) = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^m \langle \mathbf{x}_i \otimes \mathbf{y}_k | T(\mathbf{x}_j \otimes \mathbf{y}_k) \rangle |\mathbf{x}_i\rangle\langle\mathbf{x}_j|$$

where $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is an orthonormal basis of H_n and $\{\mathbf{y}_1, \dots, \mathbf{y}_m\}$ an orthonormal basis of H_n .

Example 8 It follows directly from the above proposition that $\mathrm{Tr}_{H_m}(S_1 \otimes S_2) = S_1$. Another easy corollary is the linearity of tracing over: $\mathrm{Tr}_{H_m}(p_1 S + p_2 T) = p_1 \mathrm{Tr}_{H_m}(S) + p_2 \mathrm{Tr}_{H_m}(T)$, which subsequently implies that $\mathrm{Tr}_{H_m}(p_1 S_1 \otimes S_2 + p_2 T_1 \otimes S_2) = p_1 S_1 + p_2 T_1$.

By the above proposition, one can compute the subsystem states for a pure state

$$\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \quad (13)$$

They both are

$$\frac{1}{2} |0\rangle\langle 0| + \frac{1}{2} |1\rangle\langle 1| = \frac{1}{2} I_2 \quad (14)$$

a mixed state. On the other hand, a reconstruction of [Eq. 13](#) from subsystem states ([Eq. 14](#)) is doomed to fail:

$$\begin{aligned} & \left(\frac{1}{2} |0\rangle\langle 0| + \frac{1}{2} |1\rangle\langle 1| \right) \otimes \left(\frac{1}{2} |0\rangle\langle 0| + \frac{1}{2} |1\rangle\langle 1| \right) \\ &= \frac{1}{4} |00\rangle\langle 00| + \frac{1}{4} |01\rangle\langle 01| + \frac{1}{4} |10\rangle\langle 10| + \frac{1}{4} |11\rangle\langle 11| = \frac{1}{4} I_4 \end{aligned}$$

which is different from the pure state determined by $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$:

$$\begin{aligned} & \left| \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \right\rangle \langle \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle | \\ &= \frac{1}{2} |00\rangle\langle 00| + \frac{1}{2} |00\rangle\langle 11| + \frac{1}{2} |11\rangle\langle 00| + \frac{1}{2} |11\rangle\langle 11| \end{aligned}$$

The above observations can be expressed as follows: A compound quantum system \mathcal{S}_{12} always determines its subsystem states, but if states of the subsystems \mathcal{S}_1 and \mathcal{S}_2 are known, it is not possible to determine the state of the compound system uniquely without further information. A further analysis in fact shows that the case where both subsystem states are pure is the only one when the (decomposable) compound system state can be determined uniquely.

7.1 Transformations of Quantum States

Since unitary mappings play an important role in quantum systems, we will first devote a section to them.

7.1.1 Unitary Mappings

Definition 23 Mapping $U \in L(H_n)$ is *unitary* if $U^* = U^{-1}$.

Another equivalent form of the definition is as follows.

Definition 24 Mapping U is *unitary* if it preserves the inner products, meaning that $\langle U\mathbf{x} | U\mathbf{y} \rangle = \langle \mathbf{x} | \mathbf{y} \rangle$ for all $\mathbf{x}, \mathbf{y} \in H_n$.

The equivalence of the above two definitions is obvious, since

$$\langle U\mathbf{x} | U\mathbf{y} \rangle = \langle U^*U\mathbf{x} | \mathbf{y} \rangle$$

The latter form of the definition directly implies that unitary mappings also preserve the norms of the vectors: $\|U\mathbf{x}\|^2 = \langle U\mathbf{x} | U\mathbf{x} \rangle = \langle \mathbf{x} | \mathbf{x} \rangle = \|\mathbf{x}\|^2$, hence $\|U\mathbf{x}\| = \|\mathbf{x}\|$. The latter form can also be interpreted so that the unitary mappings preserve the angles between vectors; hence it is justified to say that unitary mappings are *rigid motions* or *rotations* of H_n . What is not so straightforward is that the condition $\|U\mathbf{x}\| = \|\mathbf{x}\|$ for each $\mathbf{x} \in H_n$ is also enough to guarantee the unitarity of mapping U . This follows from the so-called *polarization identity*; for the proof, see Hirvensalo (2004) for instance.

Recall from [Sect. 2](#) that the self-adjoint mappings of H_n are, in a sense, extensions of real numbers. We will shortly see that the unitary mappings relate to the unit circle of complex numbers in the same way as self-adjoint mappings relate to real numbers. The following simple proposition draws the first connection.

Proposition 6 *The eigenvalues of unitary mappings lie in the unit circle.*

Proof If $U\mathbf{x} = \lambda\mathbf{x}$, then by a direct consequence of [Definition 24](#),

$$|\lambda|\|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|U\mathbf{x}\| = \|\mathbf{x}\|$$

hence $|\lambda| = 1$.

Like self-adjoint operators, also the unitary ones are normal: $U^*U = I = UU^*$, hence unitary mappings admit a spectral representation

$$U = \lambda_1 |\mathbf{x}_1\rangle\langle\mathbf{x}_1| + \dots + \lambda_n |\mathbf{x}_n\rangle\langle\mathbf{x}_n| \quad (15)$$

and from Proposition 6 we know that each eigenvalue λ_k lies in the unit circle. Conversely, since any λ in the unit circle satisfies $\lambda^* = \lambda^{-1}$, we see that any mapping of form [Eq. 15](#) with $|\lambda_i| = 1$ is indeed unitary. Since any unit circle number λ_k can be represented as $\lambda_k = e^{i\theta_k}$, where $\theta_k \in \mathbb{R}$, we get the following proposition.

Proposition 7 *If H is a self-adjoint mapping, then e^{iH} is unitary. Conversely, for any unitary mapping $U \in L(H_n)$ there exists a self-adjoint mapping H so that $U = e^{iH}$.*

We will present another analogy between unitary mappings and unit circle complex numbers. For that purpose, the notion of the absolute value defined for the operators will be needed.

Definition 25 Let $A \in L(H_n)$. Then the *absolute value* of A is defined as

$$|A| = \sqrt{A^*A}$$

Notice that this definition makes perfect sense: As $(A^*A)^* = A^*(A^*)^* = A^*A$, operator A^*A is self-adjoint and hence normal. Moreover $\langle \mathbf{x} | A^*A\mathbf{x} \rangle = \langle A\mathbf{x} | A\mathbf{x} \rangle = \|A\mathbf{x}\|^2 \geq 0$, which means that A^*A is a positive mapping; hence its eigenvalues are nonnegative, and the square root can be defined via spectral decomposition: if

$$A^*A = \sum_{k=1}^n \lambda_k |\mathbf{x}_k\rangle\langle\mathbf{x}_k|$$

then

$$\sqrt{A^*A} = \sum_{k=1}^n \sqrt{\lambda_k} |\mathbf{x}_k\rangle\langle\mathbf{x}_k|$$

Numbers $\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}$ (the eigenvalues of $|A|$) are called the *singular values* of matrix A and are frequently denoted by symbols s_1, \dots, s_k .

The next proposition, whose proof will be omitted, establishes another analogue between unitary mappings and unit circle numbers.

Proposition 8 (Polar decomposition) *Each $A \in L(H_n)$ can be represented as $A = U|A|$, where U is a unitary mapping. However, U is unique only if A is invertible.*

In an infinite-dimensional vector space, the above proposition does not necessarily hold, but “unitary” must be replaced with “partial isometry.”

7.1.2 State Transformations

As the state of a quantum system \mathcal{S} with state space H_n corresponds to a linear mapping in $L(H_n)$, it is natural to understand a state transformation (also called *discrete time evolution*) as a mapping $V: L(H_n) \rightarrow L(H_n)$, which takes the state S_1 of system \mathcal{S} at time t_1 into the state S_2 at time t_2 : $V(S_1) = S_2$.

To discover the nature of mapping V , we can think about the obvious constraints. As the most obvious one, $S_2 = V(S_1)$ should be a quantum state if S_1 is. This means that V should at least preserve the positivity and the unit trace. Already these requirements impose some restrictions on the form of mapping V , but the heaviest constraint comes from the linearity assumption. Typically, mapping $V: L(H_n) \rightarrow L(H_n)$ is assumed to be linear, and this assumption is supported, among others, by the idea of mixing states: the statistical mixture $T = pT_1 + (1-p)T_2$ should evolve so that T_1 and T_2 are kept independent, meaning that $V(T) = pV(T_1) + (1-p)V(T_2)$. By questioning the very ontology of the mixed quantum states, one can obviously present arguments against the linearity, too, but such a discussion is not the purpose of this chapter. Instead, this presentation follows the mainstream, accepting the linearity of state transformations. Already these ideas give rise to the following definition.

Definition 26 A linear mapping $V: L(H_n) \rightarrow L(H_n)$ is *positive* if $V(A)$ is a positive mapping whenever A is.

When accepting the aforementioned conditions (positivity and preserving the unit trace, plus the linearity) on the state transformations, we could imagine that no other conditions are needed. However, this is not the case. Indeed, it is possible to imagine a compound system \mathcal{S}_{12} with quantum state transformation $V \otimes I$ that leaves the subsystem \mathcal{S}_2 virtually untouched.

Example 9 Mapping $V: L(H_n) \rightarrow L(H_n)$ defined by $V(A) = A^*$ preserves positivity: Indeed, if A is positive, then $\langle \mathbf{x} | A\mathbf{x} \rangle \geq 0$ for any \mathbf{x} . Then also $\langle \mathbf{x} | V(A)\mathbf{x} \rangle = \langle \mathbf{x} | A^*\mathbf{x} \rangle = \langle A\mathbf{x} | \mathbf{x} \rangle = \langle \mathbf{x} | A\mathbf{x} \rangle^* = \langle \mathbf{x} | A\mathbf{x} \rangle \geq 0$.

Now in the case $n = 2$ V takes the form $V(|0\rangle\langle 0|) = |0\rangle\langle 0|$, $V(|0\rangle\langle 1|) = |1\rangle\langle 0|$, $V(|1\rangle\langle 0|) = |0\rangle\langle 1|$, and $V(|1\rangle\langle 1|) = |1\rangle\langle 1|$. Mapping

$$A = |00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11| \in L(H_2 \otimes H_2)$$

is positive, as easily seen from the spectral representation $A = |00\rangle\langle 00| + |11\rangle\langle 11| + |01\rangle\langle 01| + |10\rangle\langle 10|$. But $V \otimes I_2$ takes $|00\rangle\langle 11| = |0\rangle\langle 1| \otimes |0\rangle\langle 1|$ into $|1\rangle\langle 0| \otimes |0\rangle\langle 1| = |10\rangle\langle 01|$ and $|11\rangle\langle 00| = |1\rangle\langle 0| \otimes |1\rangle\langle 0|$ into $|0\rangle\langle 1| \otimes |1\rangle\langle 0| = |01\rangle\langle 10|$, hence

$$(V \otimes I_2)(A) = |00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11|$$

but the mapping thus obtained is *not positive*, as it has -1 as an eigenvalue:

$$(V \otimes I_2)(A)(|10\rangle - |01\rangle) = |01\rangle - |10\rangle = -(|10\rangle - |01\rangle)$$

The above example shows that we should impose a heavier restriction on mapping V to be an eligible quantum state transform. This stronger condition is expressed in the definition below.

Definition 27 Linear mapping $V: L(H_n) \rightarrow L(H_n)$ is a *completely positive mapping* if $V \otimes I_k$ is a positive mapping for any k (here I_k stands for the identity mapping on k -dimensional Hilbert space).

The above definition completes the characterization of discrete quantum time evolutions (quantum state transformations).

Definition 28 The discrete time evolution operator of a quantum system with state space H_n is the completely positive, trace-preserving linear mapping $L(H_n) \rightarrow L(H_n)$.

The discrete quantum time evolution operators are also known as *superoperators*. There are two well-known characterizations for them, both of which follow from the *Stinespring dilation theorem* (Stinespring 1955). The first characterization is known as the *Kraus representation*.

Theorem 4 A completely positive mapping $V: L(H_n) \rightarrow L(H_n)$ can be represented as

$$V(A) = \sum_{k=1}^m V_k A V_k^* \tag{16}$$

where each $V_k \in L(H_n)$ and $m \leq n^2$. V is trace-preserving if and only if

$$\sum_{k=1}^m V_k^* V_k = I$$

Even though the second characterization also follows from the Stinespring dilation theorem as does the first one, this second one is more frequently known as the *Stinespring representation*. Another name for the representation below is the *Ozawa representation* (Ozawa 1984).

Theorem 5 *For any completely positive, trace-preserving mapping $V : L(H_n) \rightarrow L(H_n)$ there exists a Hilbert space H_m with $m \leq n^2$, pure state $|0\rangle\langle 0| \in H_m$, and a unitary mapping $U \in L(H_n \otimes H_m)$ so that*

$$V(A) = \text{Tr}_{H_m}(U(A \otimes |0\rangle\langle 0|) U^*) \quad (17)$$

Representation (17) offers an interesting interpretation, which will be discussed after introducing a special case in the next section.

7.1.3 Closed Systems

The state transformation as introduced in the previous chapter can be irreversible in general. For instance, $V: L(H_2) \rightarrow L(H_2)$ defined via Kraus representation

$$V(A) = |0\rangle\langle 0|A|0\rangle\langle 0| + |1\rangle\langle 1|A|1\rangle\langle 1|$$

is clearly a completely positive, trace-preserving mapping. The images of the basis elements of $L(H_2)$ under V can be easily computed:

$$V(|0\rangle\langle 0|) = |0\rangle\langle 0|0\rangle\langle 0|0\rangle\langle 0| + |1\rangle\langle 1|0\rangle\langle 0|1\rangle\langle 1| = |0\rangle\langle 0|$$

and similarly $V(|0\rangle\langle 1|) = 0$, $V(|1\rangle\langle 0|) = 0$, and $V(|1\rangle\langle 1|) = |1\rangle\langle 1|$. We can readily observe that V is not injective, hence V^{-1} does not exist. It is also noteworthy that the action of V on a pure (vector) state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ yields

$$\begin{aligned} V\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) &= \frac{1}{\sqrt{2}}|0\rangle\langle 0| + \frac{1}{\sqrt{2}}|1\rangle\langle 1| \\ &= |0\rangle\langle 0|\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)|0\rangle\langle 0| \\ &\quad + |1\rangle\langle 1|\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)|1\rangle\langle 1| \\ &= \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \end{aligned}$$

a mixed state (analogously one can see that V turns vector state $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ into a mixed state $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$).

In general, a quantum system is called *closed* if it does not interact with any other system and if it is not entangled with another system. In this chapter, however, the following definition will be taken for a closed system.

Definition 29 Discrete time evolution in $L(H_n)$ is called *closed* if it is of the form

$$V(A) = UAU^* \quad (18)$$

where $U \in L(H_n)$ is unitary. A quantum system with closed (discrete) time evolution is called *closed*. A system that is not closed is called *open*.

Notice that both the Kraus representation and the Stinespring representation are extensions of [Eq. 18](#). Indeed, [Eq. 18](#) is a Kraus representation with $m=1$ and a Stinespring representation with $m=0$.

Closed discrete time evolution $V(A)=UAU^*$ on a pure (vector) state \mathbf{x} especially means that $|\mathbf{x}\rangle\langle\mathbf{x}|$ is converted into $U|\mathbf{x}\rangle\langle\mathbf{x}|U^* = |U\mathbf{x}\rangle\langle U\mathbf{x}|$, which is another pure state. Moreover, a closed discrete evolution is always invertible: A can be recovered as $A=U^*V(A)U$, which even reveals that the inverse of a closed state transformation is obtained by replacing U with $U^*=U^{-1}$. A closed state transformation means that the state vector turns into another state vector $U\mathbf{x}$.

- ▶ In the theory of quantum computing, it frequently assumed that the computational device has only closed state transformations and handles only pure states. In particular, this implies that one can then remain in the vector state formalism, and any state transformation takes the form

$$\mathbf{x} \mapsto U\mathbf{x}$$

where U is a unitary mapping. This is usually referred to by saying that (discrete) quantum evolution is *unitary*.

It is now interesting to see that the Stinespring representation ([17](#)) can be interpreted as follows: a quantum system with an open state transformation can be augmented with an auxiliary system, and the state transformation in the compound system is closed. Open state transformations of the subsystem hence result from watching the system only partially; the state transformation is closed in a larger system.

7.1.4 Schrödinger Equation

Previously, only the state transformations of quantum systems were considered. In other words, the evolution studied so far has been discrete, meaning that in a fixed time interval state S turns into $V(S)$ when the time has elapsed. There is nothing wrong with such an approach, but evidently it gives only part of the picture. For a refinement, one should learn how to approach *continuous state transformations*. Continuous state transformations are referred to generally as the *time evolution*.

To establish the mathematical machinery for that purpose, we restrict ourselves, for the sake of simplicity, to the closed systems beginning on a pure state. In particular this means that the images of the initial state vector \mathbf{x}_0 are studied under unitary mappings U_t for $t \in \mathbb{R}$; the state at time t is $\mathbf{x}_t = U_t\mathbf{x}_0$. How to continue from this is briefly, but not very thoroughly, described as follows: It is natural to require that $U_{t_1+t_2}\mathbf{x} = U_{t_1}U_{t_2}\mathbf{x}$, which is not far from saying that there is a morphism $t \mapsto U_t$ from real numbers to unitary mappings. Together with a continuity assumption, such a morphism can in fact be represented as $U_t = e^{itH}$, where H is a fixed self-adjoint mapping (Hamiltonian of the quantum system) ([Stone 1932](#)). Consequently,

$$\mathbf{x}_t = U_t\mathbf{x}_0 = e^{itH}\mathbf{x}_0$$

and componentwise differentiation gives

$$\frac{d}{dt}\mathbf{x}_t = iHe^{itH}\mathbf{x}_0 = iHU_t\mathbf{x}_0 = iH\mathbf{x}_t$$

► The equation

$$\frac{d}{dt} \mathbf{x}_t = iH\mathbf{x}_t$$

is called the *Schrödinger* equation.

The Schrödinger equation is a description of how the pure state of a closed system evolves continuously in time. The above form is frequently called the *abstract Schrödinger equation*, but the “concrete” ones can be obtained by substituting “concrete” Hamiltonians for H .

7.2 Models for Quantum Computing

Any classical model of computing also has a quantum version. There is in fact almost a “canonical” way of converting a classical model into a quantum version, namely that of replacing the classical description of the physical objects by the quantum description. In this section, the quantum time evolution means *discrete* time evolution, that is, the state transformation.

7.2.1 Finite Automata

Finite automata are models for real-time computation with a finite memory. The automaton reads the input word one letter at a time. Reading of a letter imposes a change of the internal state of the automaton (there are only finitely many of them), and having read the whole input word, the state of the automaton is observed. If the automaton has then reached a *final state* (can also be called *accepting state*), then the input word is accepted, otherwise rejected. Hence finite automata classify all potential input words into two piles: accepted and rejected. This can be expressed in other words by saying that the finite automata solve computational problems with yes/no answers: The problem instance is encoded into the input word, and the accept/reject behavior of the automaton is interpreted as the yes/no answer.

It turns out that finite automata can solve only a very restricted class of computational problems. Nevertheless, the research community has shown (and still shows) enormous interest in them. This is due to the following facts: the finite automata offer a very simple computational model with a huge number of connections to algebra, logic, combinatorics, logic, etc., not forgetting that the basic model of the finite automaton has beautiful closure properties; for example, constructing new automata of known ones for simultaneous computation or sequential computation is a straightforward task. Moreover, it is easy to present the generalizations and variants of finite automata: nondeterministic, probabilistic, and even quantum, as will be shortly seen. In addition, comparison between the different variants has been relatively successful, a natural measure for the complexity is the number of the states, and, for instance, a classical result shows that nondeterministic finite automata recognizing a language can have exponentially fewer states than the minimal deterministic one (Yu 1997). Even further, almost all classical models of computing (including the Turing machines in the next section) can be obtained by starting with finite automata, then augmenting them with an extra part. A wider description of finite automata can be found in Yu (1997), here we present only the definition.

A finite automaton A is a fivetuple $A=(Q,\Sigma,\delta,q_0,F)$, where Σ is a finite alphabet, Q is the state set, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, $q_0 \in Q$ the initial state, and $F \subseteq Q$ the set of final (accepting) states.

A *probabilistic finite automaton* (see Paz (1971) for more details) is obtained by replacing the transition function $\delta : Q \times \Sigma \rightarrow Q$ by the transition function $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$ whose value at triplet (q, σ, p) is interpreted as the probability of entering state p when being in state q and reading σ as an input letter. Equivalently, one can describe a probabilistic finite automaton as a collection of $|Q| \times |Q|$ probabilistic matrices $(M_a)_{ij} = \delta(q_j, a, q_i)$ together with an initial probability distribution.

To get a *quantum finite automaton*, we assume that the set of states is described by a quantum mechanical systems. Consequently, we can fix a Hilbert space H_n ($n = |Q|$) together with an orthonormal basis $\{|q\rangle |q \in Q\}$. For each $a \in \Sigma$, we define a set of completely positive mappings $V_a : L(H_n) \rightarrow L(H_n)$, fix an *initial state* $S \in L(H_n)$, and a *final projection* $P \in L(H_n)$. The *acceptance probability* $p(w)$ for a word $w = a_1 \dots a_k \in \Sigma^*$ (note that in the previous sections notion A^* was used for the adjoint mapping and here, as in theoretical computer science generally, Σ^* stands for the free monoid generated by Σ) is then computed as

$$p(w) = \text{Tr}(PV_n \dots V_1 S)$$

A *measure once* quantum automaton (introduced in Moore and Crutchfield (2000)) is obtained from the above description by assuming that the initial state is pure, and that each mapping V_a is indeed closed time evolution $V_a(T) = U_a T U_a^*$ (unitary time evolution).

In the model of the *measure many* quantum automaton (introduced in Kondacs and Watrous (1997)), one also assumes a pure initial state and unitary time evolution. The state set is divided into accepting, neutral, and rejecting states, and the subspaces generated by them are denoted by H_A , H_N , and H_R , respectively. For an input letter a , the model includes unitary time evolution V_a followed by a measurement of the observable defined by mutually orthogonal subspaces H_A , H_N , and H_R . If the state observed was accepting (resp. rejecting), then the input word is accepted (resp. rejected), but if the observed state was neutral, then the computation is continued with the next input letter.

It is trivial to note that measure-once quantum automata are subcases of the measure many automata, but it also turns out that both quantum automata models together with the model of probabilistic automata are subcases of the general model with an open time evolution (Hirvensalo (2008)).

7.2.2 Quantum Turing Machines

While finite automata are very restricted in their computational power, *Turing machines* can compute anything one can even imagine to be computed. Or at least this is a very sincere hope expressed in the famous *Church–Turing thesis* (see Papadimitriou (1994) for discussion), which essentially says that for any intuitive algorithm there is a Turing machine doing the same job. Turing machines were introduced by Turing (1936) to satisfy the need for a formal definition of *algorithm*.

Turing machines extend the notion of finite automata by an *infinite memory*, which, in the basic model, is a linear array of memory cells, each capable of containing a symbol from a finite alphabet. This infinite memory of the Turing machine is called the *tape*, and it is accessed via a *read–write head* that can point only at one particular cell at a time, and on a single computational step can move to point at most to the adjacent cells.

Besides the tape, the Turing machines have, as finite automata do, their finite set of internal states. The computation of a Turing machine runs as follows: The input word is initially written on the tape, but all other tape cells are empty. At the initial stage of computation, the

read–write head is set to point at the cell containing the first letter of the input word. A computational step of a Turing machine then can change the internal state of the machine, the content of the tape cell being scanned, and the position of the read–write head. The computational steps follow each other until the machine reaches an internal final state, which can be either of accepting or rejecting type. It may also happen that the final state is never reached, but the computation continues forever.

It can be understood that Turing machines, unlike finite automata, classify the input words into rejected, accepted, and neutral (for which the computation never ends), but also other classifications are possible: we could for example group together rejecting and unending computations to get a yes/no answer when solving computational problems. On the other hand, there is no way to tell in general whether a Turing machine computation is going to halt or not (Papadimitriou 1994).

As a theoretical model of computing, Turing machines do not have intuitively good programming properties. In fact, almost the converse holds true: The program of a Turing machine, which is encoded in the transition function, is usually the most unintuitive one could ever imagine. But, on the other hand, the Turing machine formalism offers the clearest picture on the computational resources one could ever imagine: The time needed for the computation is just the number of the computational steps from the beginning to the end, and the space devoured by the computation is the number of tape cells used during it.

Just as with finite automata, Turing machines also bear notable closure properties: For example, given two Turing machines, it is possible to construct a new machine for parallel computations, as well as for consecutive computation. Moreover, it is possible to construct a single Turing machine acting as a *clock* in the following sense: For an input of length n , the machine performs computations for exactly $f(n)$ steps, and then halts (the construction can be made for a large class of functions f , but of course not for all functions). This offers a potential way of handling nonfinishing computations: Given a Turing machine M , one can construct a time-counting machine T , and join them into a new machine M' , which halts either when the input word is accepted or when the time limit is reached. This construction is particularly good for algorithms with a known time complexity function, as the termination of the computation can then be always guaranteed.

For an extensive exposure on Turing machines, we refer to Papadimitriou (1994), but a brief formal definition follows immediately.

Definition 30 A Turing machine is a sextuple $T = (Q, \Gamma, \delta, q_0, q_a, q_r)$, where Q is the set of the machine’s internal states, $\Gamma = \Sigma \cup \{\square\}$ is the input alphabet Σ augmented with the blank symbol \square , and q_0 , q_a , and q_r are the initial, accepting, and rejecting states, respectively.

The transition function $\delta : Q \times \Sigma \mapsto Q \times \Sigma \times \{L, S, R\}$ specifies the action of the machine: if $\delta(q_1, \sigma_1) = (q_2, \sigma_2, L)$, then the machine being in state q_1 and reading σ_1 writes σ_2 in the memory cell being scanned, enters state q_2 , and moves the read–write head to the left. Correspondingly, values S and R would order the read–write head to stay, and move to the right, respectively.

Now the memory of a Turing machine consists of the finite set of internal states together with the tape content. As the tape is initially assumed to contain only the (finite) input word, there are at any moment of the computation only finitely many non-blank symbols on the tape. This implies that the number of potential *configurations* of the machine is countable. To specify what a configuration of the Turing machine means, there are two traditional ways.

The first one is a triplet (q, i, w) , where $q \in Q$ is the state of the machine, $i \in \mathbb{Z}$ tells the number of the memory cell being scanned by the read–write head (initially assumed 0), and $w \in \Gamma^*$ is the nonblank content of the tape presented in the shortest way. In the second representation, the position of the read–write head is indicated by splitting the content of the tape into two sections: configuration (q, w_1, w_2) then means that the state of the machine is q , the non-blank content of the tape is $w_1 w_2$, and that the read–write head is scanning the first symbol of w_2 .

The quantum version of Turing machines, first introduced by David Deutsch and subsequently analyzed in Bernstein and Vazirani (1997) is obtained by assuming that both the tape and the set of internal states are quantum physical. The internal state set will be obviously described as in the case of finite automata: Hilbert space $H_{|Q|}$ with an orthonormal basis $\{|q\rangle | q \in Q\}$ will serve as a quantum mechanical description of the state set. The tape itself, being an infinite object, is a bit more complicated. A single tape cell can of course be described by using Hilbert space $H_{|\Gamma|}$ with an orthonormal basis $\{|\gamma\rangle | \gamma \in \Gamma\}$, two tape cells as $H_{|\Gamma|} \otimes H_{|\Gamma|}$, three cells as $H_{|\Gamma|} \otimes H_{|\Gamma|} \otimes H_{|\Gamma|}$, etc. The quantum physical description of the tape is hence given by an infinite tensor product of finite dimensional Hilbert spaces $H_{|\Gamma|}$ called T , but the action of the machine is nevertheless defined by a local rule.

Deutsch's model presented the configuration of the machine as (q, i, w) , where i presents the number of the cell being scanned. To get a quantum version of this, one needs to introduce an infinite-dimensional Hilbert space $H_{\mathbb{Z}}$ with orthonormal basis $\{|i\rangle | i \in \mathbb{Z}\}$, and the quantum version of the configuration is simply a state in space $H_{|Q|} \otimes H_{\mathbb{Z}} \otimes T$.

The other model, see Hirvensalo (2004) for instance, uses configurations of the form (q, w_1, w_2) . There are only numerable many of those configurations, and we set up a Hilbert space H_C (the configuration space) with an orthonormal basis $\{(q, w_1, w_2) | q \in Q, w_1, w_2 \in \Gamma^*\}$.

If open time evolution is allowed, then the relationship between classical and quantum Turing machines is not a very problematic one: it is possible to interpret an ordinary Turing machine as a special case of quantum Turing machines merely by defining the time evolution by $V |c\rangle = |c'\rangle$, where c' is the configuration obtained from c by transition function δ .

On the other hand, most papers on quantum computing, including the pioneering articles, assume that the quantum Turing machine model operates with *closed (unitary) time evolution*, V which imposes a heavy restriction on δ . In general it is possible that δ does not specify any reversible time evolution. Fortunately it has been thoroughly examined to what extent the computation allows reversibility. A classical result (Bennett 1973) showed that *all computation* whatsoever can be performed also in a reversible manner, if extra workspace is available. In particular, the result of Bennett (1973) implies that an arbitrary computation can be simulated by a reversible one with the cost of a polynomial space increase.

7.2.3 Quantum Circuits

Quantum Turing machines offer, as their classical counterparts do, a good theoretical model for computation, as they provide a clear idea of the space and time required by computation. However, as in the classical case, it is very difficult to express even simple algorithms by means of a quantum Turing machine. For that purpose, *quantum circuits* are much better. They were first introduced by Deutsch (1989) and provide an easy way of expressing the technical details of quantum algorithms.

Boolean circuits (see Papadimitriou (1994) or Hirvensalo (2004) for instance) can be seen as acyclic graphs with nodes labeled by logical gates. A typical selection of gates includes AND,

NOT, and OR. The idea of the circuit computation is to encode the input into a binary string, and then let the circuit decide whether or not the input is accepted. Hence one needs actually a family C_1, C_2, C_3, \dots , of circuits so that circuit C_n , which takes the inputs of length n , has n input wires and one output wire. C_n then computes the output value 1 or 0 (yes/no) plainly obeying the structure of the Boolean circuit, using the gates as elementary computational devices.

Computation by circuits differs essentially from finite automata and Turing machines. As the latter devices can handle inputs of all lengths by a single machine, the circuit model assumes a particular circuit C_n for each input word length n . This is usually expressed by saying that finite automata and Turing machines are *uniform* computational devices, and the set of circuits is not.

Furthermore, it turns out that for *any* subset $L \subseteq \{0,1\}^*$ there is a family C_1, C_2, C_3, \dots of circuits accepting L , meaning that for a string w of length n , C_n outputs 1 if and only if $w \in L$ (Papadimitriou 1994; Hirvensalo 2004). On the contrary, it is clear that for an arbitrary subset $L \subseteq \{0,1\}^*$ there does not exist a Turing machine accepting L . This does not contradict the Church–Turing thesis claiming that the Turing machines possess the ultimate computational power, as only the *existence* of the circuit family C_1, C_2, C_3, \dots is guaranteed no construction. Without a construction, a circuit family cannot be regarded as a computational device at all, but when the construction is included, circuit formalism can have great advantage over the sole Turing machine formalism, especially in the theory of quantum computing.

The construction of circuit C_n requires an algorithm: given n , the task is to produce the graph of C_n , hopefully in a very efficient manner. As an algorithm is called for, one immediately realizes that we are again speaking of Turing machines (or of some equivalent uniform computational model). One could hence again question the meaningfulness of the circuit model, but it has turned out that splitting the algorithm into two major parts (construction of the circuit and the computation performed by the circuit) can offer understanding, which outperforms that given by a direct uniform computational model.

The aforementioned division (construction of the circuit followed by circuit computation) seems to work especially well in quantum computing. The famous quantum algorithms for integer factoring by Peter W. Shor (Shor 1994) and quadratic speedup for search problems by Lov Grover (Grover 1996) can be very well explained using the quantum circuit model (see Hirvensalo (2004)). Indeed, the construction of the quantum circuit for both tasks turns out to be very easy, and the quantum effects (interference and entanglement) can be explored in the circuit computation.

By allowing open time evolution, one gets the notion of *quantum circuits with mixed states* introduced in Aharonov et al. (1998). Quantum circuits with open time evolution can simulate effectively ordinary Boolean circuits, but when restricted to closed time evolution, the computational power seemingly diminishes. However, such a great variety of research papers use quantum circuits with closed (unitary) time evolution that it is worth concentrating on them in more detail.

A *quantum circuit* with unitary time evolution is a sequence U_1, \dots, U_k of unitary mappings acting on $H_{2^n} = H_2 \otimes \dots \otimes H_2$ (n terms), where n is the number of input qubits. Mappings U_1, \dots, U_k have the following restrictions: there is a finite set $\{G_1, \dots, G_m\}$ of *quantum gates* (unitary mappings), from which mappings U_i are constructed. This means each U_i is one of the mappings G_j operating on some of the n qubits and leaving all others untouched.

The computation of the circuit is then very straightforward: Initially, the qubits contain the input word $\mathbf{x} \in \{0, 1\}^n$, so that the initial state of the circuit is $|\mathbf{x}\rangle$. Then mappings U_1, \dots, U_k are applied to get the final state

$$U_k \dots U_1 |\mathbf{x}\rangle$$

and the output qubits can be observed. Number k is called the *depth* of the circuit.

Note that as the time evolution is reversible, one must have equally many input and output qubits, but of course one can restrict this to read only one of the output bits. In the second place, the finite set $G = \{G_1, \dots, G_m\}$ is the counterpart of classical AND, NOT, and OR gates, and it is natural to assume that G is *universal* in the sense that all unitary mappings $U \in L(H_{2^n})$ can be approximated by gates in set G . The actual choice of G does not matter so much. One can also note that in the classical case one of the gates AND and OR could be omitted, or it could be possible to use only a single gate NAND. In the quantum context, universality must mean approximability: set $L(H_{2^n})$ has continuum cardinality, but mappings obtained from a finite set G in the above fashion form only a numerable set, implying that *most* unitary mappings cannot be represented exactly.

The question of universality has been studied very thoroughly. Already in his seminal paper (Deutsch 1989) Deutsch proved that there is a single three-qubit universal quantum gate. Later, it was shown in Barenco et al. (1995) that unary gates (those acting only on one qubit) together with controlled NOT gate form a universal set even in the strict sense (exact representation). A very simple set of universal gates was given in Shi (2003): the Toffoli gate together with the Hadamard–Walsh gate is a universal set. Another very important point is the approximation efficiency: how many gates from a universal set are needed in general to approximate a given gate with accuracy ε ? The famous Solovay–Kitaev theorem (see Nielsen and Chuang (2000)) implies that only $O(n \log^c \frac{n}{\varepsilon})$ gates from *any* universal gate set are needed to simulate (within accuracy ε) a quantum circuit with n arbitrary gates. (In the Solovay–Kitaev theorem, $c \in [1, 2]$ is a constant whose optimal value is not known.)

To conclude this section, we discuss briefly the relationship between quantum circuits with mixed states (open time evolution) and quantum circuits with closed (unitary) time evolution. As mentioned previously, quantum circuits with unitary time evolution are apparently more restricted than those with open time evolution, but there is a standard way of circumventing this restriction, namely the *ancilla bits*. This means that in addition to the n input qubits, there will be a number, say $a(n)$, of ancilla qubits initially all set to state $|0\rangle$, so that the initial state of the circuit is $|\mathbf{x}\rangle |0\rangle$. Then an extended circuit operates on $n + a(n)$ qubits to get the desired result, and it can be shown that an arbitrary quantum circuit with open time evolution can be simulated in this manner, using $a(n)$ (where a is a polynomial) ancilla qubits (Aharonov et al. 1998).

References

-
- Aharonov A, Kitaev A, Nisan N (1998) Quantum circuits with mixed states. In: proceedings of the 30th annual ACM symposium on theory of computation, Dallas, TX, May 1998, pp 20–30
- Axler S (1997) Linear algebra done right. Springer, New York, <http://www.springer.com/mathematics/algebra/book/978-0-387-98259-5>
- Barenco A, Bennett CH, Cleve R et al. (1995) Elementary gates for quantum computation. Phys Rev A 52(5): 3457–3467
- Benioff PA (1980) The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. J Stat Phys 22(5):563–591

- Benioff PA (1982) Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: application to Turing machines. *Int J Theor Phys* 21(3/4):177–202
- Bennett CH (1973) Logical reversibility of computation. *IBM J Res Dev* 17:525–532
- Bernstein E, Vazirani U (1997) Quantum complexity theory. *SIAM J Comput* 26(5):1411–1473
- Cohn PM (1994) Elements of linear algebra. Chapman & Hall, Boca Raton, FL. CRC Press reprint (1999). <http://www.amazon.com/Elements-Linear-Algebra-Chapman-Mathematics/dp/0412552809>
- Feynman RP (1982) Simulating physics with computers. *Int J Theor Phys* 21(6/7):467–488
- Deutsch D (1985) Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc R Soc Lond Ser A Math Phys Sci* 400:97–117
- Deutsch D (1989) Quantum computational networks. *Proc R Soc Lond A* 425:73–90
- Gleason AM (1957) Measures on the closed subspaces of a Hilbert space. *J Math Mech* 6:885–893
- Grover LK (1996) A fast quantum-mechanical algorithm for database search. In: Proceedings of the 28th annual ACM symposium on the theory of computing, Philadelphia, PA, May 1996, pp 212–219
- Hirvensalo M (2004) Quantum computing, 2nd edn. Springer, Heidelberg
- Hirvensalo M (2008) Various aspects of finite quantum automata. In: Proceedings of the 12th international conference on developments in language theory, Kyoto, Japan, September 2008. Lecture notes in computer science, vol 5257. Springer, Berlin, pp 21–33
- Kondacs A, Watrous J (1997) On the power of quantum finite state automata. In: Proceedings of the 38th annual symposium on foundations of computer science, Miami Beach, FL, October 1997, pp 66–75
- Moore C, Crutchfield JP (2000) Quantum automata and quantum grammars. *Theor Comput Sci* 237(1–2): 275–306
- Nielsen MA, Chuang IL (2000) Quantum computation and quantum information. Cambridge University Press
- Ozawa M (1984) Quantum measuring processes of continuous observables. *J Math Phys* 25:79–87
- Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Reading, MA
- Paz A (1971) Introduction to probabilistic automata. Academic, New York
- Shi Y (2003) Both Toffoli and controlled-NOT need little help to do universal quantum computation. *Quantum Info Comput* 3(1):84–92
- Shor PW (1994) Algorithms for quantum computation: discrete log and factoring. In: Proceedings of the 35th annual IEEE symposium on foundations of computer science, Santa Fe, NM, November 1994, pp 20–22
- Simon DR (1994) On the power of quantum computation. In: Proceedings of the 35th annual IEEE symposium on foundations of computer science, Santa Fe, NM, November 1994, pp 116–123
- Stinespring WF (1955) Positive functions on C^* -algebras. *Proc Am Math Soc* 6:211–216
- Stone MH (1932) On one-parameter unitary groups in Hilbert space. *Ann Math* 33:643–648
- Turing AM (1936) On computable numbers, with an application to the Entscheidungsproblem. *Proc Lond Math Soc* 2(42):230–265
- von Neumann J (1927) Thermodynamik quantummechanischer Gesamtheiten. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen* 1:273–291
- von Neumann J (1932) Mathematische Grundlagen der Quantenmechanik. Springer, Berlin
- Yu S (1997) Regular languages. In: Rozenberg G, Salomaa A (eds) *Handbook of formal languages*. Springer, Berlin

42 Bell's Inequalities — Foundations and Quantum Communication

Časlav Brukner¹ · Marek Żukowski²

¹Faculty of Physics, University of Vienna, Vienna, Austria
caslav.brukner@univie.ac.at

²Institute of Theoretical Physics and Astrophysics, University of Gdańsk,
Poland
marek.zukowski@univie.ac.at
fizmz@univ.gda.pl

1	<i>Introduction</i>	1414
2	<i>Quantum Predictions for Two-Qubit Systems</i>	1415
3	<i>Einstein–Podolsky–Rosen Experiment</i>	1418
4	<i>Bell's Theorem</i>	1420
5	<i>All Bell's Inequalities for Two Possible Settings on Each Side</i>	1427
6	<i>Quantum Reduction of Communication Complexity</i>	1433
7	<i>The Kochen–Specker Theorem</i>	1441
8	<i>Temporal Bell's Inequalities (Leggett–Garg Inequalities)</i>	1444

Abstract

For individual events, quantum mechanics makes only probabilistic predictions. Can one go beyond quantum mechanics in this respect? This question has been a subject of debate and research since the early days of the theory. Efforts to construct a deeper, realistic level of physical description, in which individual systems have, like in classical physics, preexisting properties revealed by measurements are known as hidden-variable programs. Demonstrations that a hidden-variable program necessarily requires outcomes of certain experiments to disagree with the predictions of quantum theory are called “no-go theorems.” The Bell theorem excludes local hidden variable theories. The Kochen–Specker theorem excludes non-contextual hidden variable theories. In local hidden-variable theories faster-than-light-influences are forbidden, thus the results for a given measurement (actual, or just potentially possible) are independent of the settings of other measurement devices which are at space-like separation. In non-contextual hidden-variable theories, the predetermined results of a (degenerate) observable are independent of any other observables that are measured jointly with it.

It is a fundamental doctrine of quantum information science that quantum communication and quantum computation outperform their classical counterparts. If this is to be true, some fundamental quantum characteristics must be behind the better-than-classical performance of information-processing tasks. This chapter aims at establishing connections between certain quantum information protocols and foundational issues in quantum theory. After a brief discussion of the most common misinterpretations of Bell’s theorem and a discussion of what its real meaning is, we will demonstrate *how quantum contextuality and violation of local realism can be used as useful resources* in quantum information applications. In any case, the readers should bear in mind that this chapter is not a review of the literature of the subject, but rather a quick introduction to it.

1 Introduction

Which quantum states are useful for quantum information processing? All non-separable states? Only distillable non-separable states? Only those which violate constraints imposed by local realism? Entanglement is the most distinct feature of quantum physics with respect to the classical world (Schrödinger 1935). On the one hand, entangled states violate Bell’s inequalities, and thus rule out a local realistic explanation of quantum mechanics. On the other hand, they enable certain communication and computation tasks to have an efficiency not achievable by the laws of classical physics. Intuition suggests that these two aspects, the fundamental one, and the one associated with applications, are intimately linked. It is natural to assume that the quantum states, which allow the no-go theorems of quantum theory, such as Kochen–Specker, Bell’s or the Greenberger–Horne–Zeilinger theorem should also be useful for quantum information processing. If this were not true, one might expect that the efficiency of quantum information protocols could be simulatable by classical, local realistic or non-contextual models, and thus achievable already via classical means. This intuitive reasoning is supported by the results of, for example, Ekert (1991), Scarani and Gisin (2001) and Acin et al. (2006): violation of a Bell’s inequality is a criterion for the security of quantum key distribution protocols. Also, it was shown that violation of Bell’s inequalities by a quantum state implies that pure-state entanglement can be distilled from it (Acin et al. 2002) and

that Bell's inequalities are related to optimal solutions of quantum state targeting (Bechmann-Pasquinucci 2005). This overview will give other examples that demonstrate the strong link between fundamental features of quantum states and their applicabilities in quantum information protocols, such as in quantum communication complexity problems, quantum random access coding, or certain quantum games.

2 Quantum Predictions for Two-Qubit Systems

To set the stage for the story first, a description of two-qubit systems is given in full detail.

Predictions shall be presented for all possible local yes–no experiments on two spin-1/2 systems (in modern terminology, qubits) for all possible quantum states, that is, from the pure maximally entangled singlet state (or the Bohm-EPR state), via factorizable (i.e., non-entangled) states, up to any mixed state. This will enable us to reveal the distinguishing traits of the quantum predictions for entangled states of the simplest possible compound quantum system. The formalism can be applied to any system consisting of two subsystems, such that each of them is described by a two-dimensional Hilbert space. The spin-1/2 convention is chosen to simplify the description.

2.1 Pure States

An important tool simplifying the analysis of the pure states of two subsystems is the so-called Schmidt decomposition.

2.1.1 Schmidt Decomposition

For any nonfactorizable (i.e., entangled) pure state, $|\psi\rangle$ of a pair of quantum subsystems, one described by a Hilbert space of dimension N , the other by space of dimension M , $N \leq M$, it is always possible to find preferred bases, one basis for the first system, another one for the second, such that the state becomes a sum of bi-orthogonal terms, that is

$$|\psi\rangle = \sum_{i=1}^N c_i |a_i\rangle_1 |b_i\rangle_2 \quad (1)$$

with $n\langle x_i|x_j\rangle_n = \delta_{ij}$ for $x = a, b$ and $n = 1, 2$. It is important to stress that the appropriate single subsystem bases, here $|a_i\rangle_1$ and $|b_j\rangle_2$, depend upon the state that we want to Schmidt decompose.

The ability to Schmidt decompose the state is equivalent to a well-known fact from matrix algebra, that any $N \times M$ matrix \hat{A} can be always put into a diagonal form \hat{D} , by applying a pair of unitary transformations, U and V such that: $\sum_{j=1}^N \sum_{k=1}^M U_{ij} A_{jk} V_{kl} = D_l \delta_{il}$.

The interpretation of the above formula could be put as follows. If the quantum pure state of two systems is non-factorizable, then there exists a pair of local observables (for system 1 with eigenstates $|a_i\rangle$, and for system 2 with eigenstates $|b_i\rangle$) such that the results of their measurements are perfectly correlated.

The method of the Schmidt decomposition allows us to put every pure normalized state of two spins into

$$|\psi\rangle = \cos\alpha/2|+\rangle_1|+\rangle_2 + \sin\alpha/2|-\rangle_1|-\rangle_2 \quad (2)$$

for some angle α and local bases spent by states $|+\rangle_i$ and $|-\rangle_i$, $i=1,2$. The Schmidt decomposition generally allows the coefficients to be real. This is achievable via trivial phase transformations of the preferred bases.

2.2 Arbitrary States

Systems can be in mixed states. Such states describe situations in which there does not exist any *nondegenerate* observable for which the measurement result is deterministic. This is the case when the system can be with various probabilities $P(x) \geq 0$ in some nonequivalent states $|\psi(x)\rangle$, with $\sum_x P(x) = 1$. Mixed states are represented by self-adjoint nonnegative (density) operators $\rho = \sum_x P(x)|\psi(x)\rangle\langle\psi(x)|$. As $\text{Tr}|\psi(x)\rangle\langle\psi(x)| = 1$ one has $\text{Tr}\rho = 1$.

The properties of mixed states of the two spin-1/2 systems will now be presented in detail. Any self-adjoint operator for one spin-1/2 particle is a linear combination of the Pauli matrices σ_i , $i = 1, 2, 3$, and the identity operator, $\sigma_0 = \mathbf{1}$, with *real* coefficients. Thus, any self-adjoint operator in the tensor product of the two spin-1/2 Hilbert spaces must be a real linear combination of all possible products of the operators $\sigma_\mu^1 \sigma_\nu^2$, where the Greek indices run from 0 to 3, and the superscripts denote the particle. As the trace of σ_i is zero, we arrive at the following form of the general density operator for two spin 1/2 systems:

$$\rho = \frac{1}{4} \left(\sigma_0^{(1)} \sigma_0^{(2)} + \mathbf{r} \cdot \boldsymbol{\sigma}^{(1)} \sigma_0^{(2)} + \sigma_0^{(1)} \mathbf{s} \cdot \boldsymbol{\sigma}^{(2)} + \sum_{m,n=1}^3 T_{nm} \sigma_n^{(1)} \sigma_m^{(2)} \right) \quad (3)$$

where \mathbf{r} and \mathbf{s} are real three-dimensional (Bloch) vectors and $\mathbf{r} \cdot \boldsymbol{\sigma} \equiv \sum_{i=1}^3 r_i \sigma_i$ is the scalar product between vectors \mathbf{r} and $\boldsymbol{\sigma}$. The tensor product symbol \otimes shall be used only sparingly, only whenever it is deemed necessary. The condition $\text{Tr}\rho = 1$ is satisfied, thanks to the first term.

Since the average of any real variable which can have only two values +1 and -1 cannot be larger than 1 and less than -1, the real coefficients T_{mn} satisfy relations

$$-1 \leq T_{mn} = \text{Tr}\rho\sigma_n^{(1)}\sigma_m^{(2)} \leq 1 \quad (4)$$

and they form a matrix (also called a correlation tensor), which will be denoted by $\hat{\mathbf{T}}$. One also has

$$-1 \leq r_n = \text{Tr}\rho\sigma_n^{(1)} \leq 1 \quad (5)$$

and

$$-1 \leq s_m = \text{Tr}\rho\sigma_m^{(2)} \leq 1 \quad (6)$$

2.2.1 Reduced Density Matrices for Subsystems

A reduced density matrix represents the local state of a compound system. If there are two subsystems, then the average of any observable, which pertains to the first system only, that is, of the form $A \otimes \mathbf{1}$, where $\mathbf{1}$ is the identity operation for system 2, can be expressed as follows

$\text{Tr}_{12}(A \otimes 1\rho) = \text{Tr}_1[A(\text{Tr}_2\rho)]$. Here, Tr_i represents a trace with respect to system i . As trace is a basis independent notion, one can always choose a factorizable basis, and therefore split the trace calculation into two stages.

The reduced one particle matrices for spins 1/2, are of the following form:

$$\rho_1 \equiv \text{Tr}_2\rho = \frac{1}{2}(\mathbf{1} + \mathbf{r} \cdot \boldsymbol{\sigma}^{(1)}) \quad (7)$$

$$\rho_2 \equiv \text{Tr}_1\rho = \frac{1}{2}(\mathbf{1} + \mathbf{s} \cdot \boldsymbol{\sigma}^{(2)}) \quad (8)$$

with \mathbf{r} and \mathbf{s} the two local Bloch vectors of the spins.

Let us denote the eigenvectors of the spin projection along direction \mathbf{a} of the first spin as: $|\psi(\pm 1, \mathbf{a})\rangle_1$. They are defined by the relation

$$\mathbf{a} \cdot \boldsymbol{\sigma}^{(1)} |\psi(\pm 1, \mathbf{a})\rangle_1 = \pm 1 |\psi(\pm 1, \mathbf{a})\rangle_1 \quad (9)$$

where \mathbf{a} is a real vector of unit length (i.e., $\mathbf{a} \cdot \boldsymbol{\sigma}^{(1)}$ is a Pauli operator in the direction of \mathbf{a}). The probability of a measurement of this Pauli observable to give a result ± 1 is given by

$$P(\pm 1|\mathbf{a})_1 = \text{Tr}_1 \rho_1 \pi_{(\mathbf{a}, \pm 1)}^{(1)} = \frac{1}{2}(1 \pm \mathbf{a} \cdot \mathbf{r}) \quad (10)$$

and it is positive for arbitrary \mathbf{a} , if and only if, the norm of \mathbf{r} satisfies

$$|\mathbf{r}| \leq 1 \quad (11)$$

Here $\pi_{(\mathbf{a}, \pm 1)}^{(1)}$ is the projector $|\psi(\pm 1, \mathbf{a})\rangle_{11} \langle \psi(\pm 1, \mathbf{a})|$

2.3 Local Measurements on Two Qubits

The probabilities for local measurements to give the result $l = \pm 1$ for particle 1 and the result $m = \pm 1$ for particle 2, under specified local settings, \mathbf{a} and \mathbf{b} respectively, are given by:

$$P(l, m|\mathbf{a}, \mathbf{b})_{1,2} = \text{Tr} \rho \pi_{(\mathbf{a}, l)}^{(1)} \pi_{(\mathbf{b}, m)}^{(2)} = \frac{1}{4}(1 + l\mathbf{a} \cdot \mathbf{r} + m\mathbf{b} \cdot \mathbf{s} + lm\mathbf{a} \cdot \hat{\mathbf{T}}\mathbf{b}) \quad (12)$$

where $\hat{\mathbf{T}}\mathbf{b}$ denotes the transformation of the column vector \mathbf{b} by the matrix $\hat{\mathbf{T}}$ (Euclidean vectors are treated here as column matrices).

One can simplify all these relations by performing suitable local unitary transformations upon each of the subsystems, that is, via factorizable unitary operators $U^{(1)}U^{(2)}$. It is well known that any unitary operation upon a spin 1/2 is equivalent to a three-dimensional rotation in the space of Bloch vectors. In other words, for any real vector \mathbf{w}

$$U(\hat{\mathbf{O}})\mathbf{w} \cdot \boldsymbol{\sigma} U(\hat{\mathbf{O}})^\dagger = (\hat{\mathbf{O}}\mathbf{w}) \cdot \boldsymbol{\sigma} \quad (13)$$

where $\hat{\mathbf{O}}$ is the orthogonal matrix of the rotation. If the density matrix is subjected to such a transformation on either spins subsystem, that is, to the $U^1(\hat{\mathbf{O}}_1)U^2(\hat{\mathbf{O}}_2)$ transformation, the parameters \mathbf{r} , \mathbf{s} , and $\hat{\mathbf{T}}$ transform themselves as follows

$$\begin{aligned} \mathbf{r}' &= \hat{\mathbf{O}}_1 \mathbf{r} \\ \mathbf{s}' &= \hat{\mathbf{O}}_2 \mathbf{s} \\ \hat{\mathbf{T}}' &= \hat{\mathbf{O}}_1 \hat{\mathbf{T}} \hat{\mathbf{O}}_2^T \end{aligned} \quad (14)$$

Thus, for an arbitrary state, we can always choose such factorizable unitary transformation that the corresponding rotations (i.e., orthogonal transformations) will diagonalize the correlation tensor (matrix) \hat{T} . This can be seen as another application of Schmidt's decomposition, this time in the case of the second rank tensors.

The physical interpretation of the above is that one can always choose two (local) systems of coordinates, one for the first particle, the other for the second particle, in such a way that the \hat{T} matrix will be diagonal.

Let us note that one can decompose the two spin density matrices into:

$$\rho = \rho_1 \otimes \rho_2 + \frac{1}{4} \sum_{m,n=1}^3 C_{nm} \sigma_n^1 \otimes \sigma_m^2 \quad (15)$$

that is, it is a sum of the product of the two reduced density matrices and a term $\hat{C} = \hat{T} - \mathbf{r}\mathbf{s}^T$, which is responsible for the correlation effects.

Any density operator satisfies the inequality $\frac{1}{d} < \text{Tr}\rho^2 \leq 1$, where d is the dimension of the Hilbert space in which it acts, that is, of the system it describes. The value of $\text{Tr}\rho^2$ is a measure of the purity of the quantum state. It is equal to 1 only for single dimensional projectors, that is, the pure states. In the studied case of two qubits one must have

$$|\mathbf{r}|^2 + |\mathbf{s}|^2 + ||\hat{\mathbf{T}}||^2 \leq 3 \quad (16)$$

For pure states, represented by Schmidt decomposition (🔗 2), \hat{T} is diagonal with entries $T_{xx} = -\sin\alpha$, $T_{yy} = \sin\alpha$, and $T_{zz} = 1$, whereas $\mathbf{r} = \mathbf{s}$, and their z component is nonzero: $s_z = m_z = \cos\alpha$. Thus, in the case of a maximally entangled state, \hat{T} has only diagonal entries equal to +1 and -1. In the case of the singlet state,

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|+\rangle_1 |-\rangle_2 - |-\rangle_1 |+\rangle_2) \quad (17)$$

which can be obtained from (🔗 Eq. 2), by putting $\alpha = -\frac{\pi}{2}$ and rotating one of the subsystems such that $|+\rangle$ and $|-\rangle$ interchange (this is equivalent to a 180° rotation with respect to the axis x where $|+\rangle$ and $|-\rangle$ are eigenstates of spin along z ; see above (🔗 Eq. 14)), the diagonal elements of the correlation tensor are all -1.

3 Einstein–Podolsky–Rosen Experiment

In their seminal 1935 paper (Einstein et al. 1935) entitled “*Can quantum-mechanical description of physical reality be considered complete?*” Einstein, Podolsky, and Rosen (EPR) consider quantum systems consisting of two particles such that, while neither position nor momentum of either particle is well-defined, the difference of their positions and the sum of their momenta are both precisely defined. It then follows that measurement of either position or momentum performed on, say, particle 1 immediately implies, for particle 2, a precise position or momentum, respectively, even when the two particles are separated by arbitrary distances without any actual interaction between them.

We shall present the EPR argumentation for incompleteness of quantum mechanics in the language of spins 1/2. This was done by Bohm (1952) in 1952. A two-qubit example of an EPR state is the singlet state (🔗 Eq. 17). The properties of a singlet can be inferred without mathematical considerations given above. This is a state of zero total spin. Thus, measurements of the same component of the two spins must always give opposite values – this is

simply the conservation of angular momentum at work. In terms of the language of Pauli matrices the product of the local results is then always -1 . We have (infinitely many) *perfect (anti-)correlations* for every possible choice of spin component. We assume that the two spins are very far away, but nevertheless in the singlet state.

After the translation into Bohm's example, the EPR argument runs as follows. Here are their premises:

1. *Perfect correlations*: If whatever spin components of particles 1 and 2 are measured, and the settings are identical for both particles, then with certainty the outcomes will be found to be perfectly anti-correlated.
2. *Locality*: "Since at the time of measurements the two systems no longer interact, no real change can take place in the second system in consequence of anything that may be done to the first system."
3. *Reality*: "If, without in any way disturbing a system, we can predict with certainty (i.e., with probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity."
4. *Completeness*: "Every element of the physical reality must have a counterpart in the [complete] physical theory."

In contrast to the last three premises, which though they are quite plausible are still indications of a certain philosophical viewpoint, the first premise is a statement about a well-established property of a singlet state.

The EPR argument is as follows. Because of the perfect anti-correlations (1), we can predict with certainty the result of measuring either the x component or the y component of the spin of particle 2 by previously choosing to measure the same quantity of particle 1. By locality (2), the measurement of particle 1 cannot cause any real change in particle 2. This implies that by the premise (3) both the x and the y components of the spin of particle 2 are elements of reality. This is also the case for particle 1 by a parallel argument where particle 1 and 2 interchange their roles. Yet, (according to Heisenberg's uncertainty principle) there is no quantum state of a single spin in which both x and y spin components have definite values. Therefore, by premise (4), quantum mechanics cannot be a complete theory.

In his answer (Bohr 1935), published in the same year and under the same title as the EPR paper, Bohr criticized the EPR concept of "reality" as assuming the systems have intrinsic properties independently of whether they are observed or not and he argued for "the necessity of a final renunciation of the classical ideal of causality and a radical revision of one's attitude toward the problem of physical reality." Bohr pointed out that the wording of the criterion of physical reality (3) proposed by EPR contains an ambiguity with respect to the expression, "without in any way disturbing the system." And, while, as Bohr wrote, there is "no question of mechanical disturbance of the system," there is "the question of *an influence on the very conditions, which define the possible types of predictions regarding the future behavior of the system*." Bohr thus pointed out that the results of quantum measurements, in contrast to those of classical measurements, depend on the complete experimental arrangement (*context*), which can even be nonlocal as in the EPR case. Before any measurement is performed, only the correlations between the spin components of two particles, but not spin components of individual particles, are defined. The x or y component (but never both) of an individual particle becomes defined only when the respective observable of the distant particle is measured.

Perhaps the clearest way to see how strongly the philosophical viewpoints of EPR and Bohr differ is in their visions of the future development of quantum physics. While EPR wrote: "We

believe that such a [complete] theory is possible,” Bohr’s opinion is that (his) complementarity “provides room for new physical law, the coexistence of which might at first sight appear irreconcilable with the basic principles of science.”

4 Bell's Theorem

Bell's theorem can be thought of as a disproof of the validity of the EPR ideas. The elements of physical reality cannot be an internally consistent notion. A broader interpretation of this result is that a local and realistic description of nature, at the fundamental level, is untenable. Further consequences are that there exist quantum processes, which cannot be modeled by any classical ones, not necessarily physical processes, but also some classical computer simulations with a communication constraint. This opened up the possibility of the development of quantum communication.

We shall now present a derivation of Bell's inequalities. The stress will be put on clarification of the underlying assumptions. These will be presented in the most reduced form.

4.1 Thought Experiment

At two measuring stations A and B , which are far away from each other, two characters Alice and Bob observe simultaneous flashy appearances of numbers $+1$ or -1 at the displays of their local devices (or the monitoring computers). The flashes appear in perfect coincidence (with respect to a certain reference frame). In the middle, between the stations, is something that they call “source.” When it is absent, or switched off, the numbers ± 1 's do not appear at the displays. The activated source always causes two flashes, one at A , and one at B . They appear slightly after a relativistic retardation time with respect to the activation of the source, never before. Thus there is enough “evidence” for Alice and Bob that the source causes the flashes. The devices at the stations have a knob, which can be put in two positions: $m = 1$ or 2 at A station, and $n = 1$ or 2 at B . Local procedures used to generate random choices of local knob positions are equivalent to *independent, fair coin tosses*. Thus, each of the four possible values of the pair n, m are equally likely, that is, the probability $P(n, m) = P(n)P(m) = \frac{1}{4}$. The “tosses,” and knob settings, are made at random times, and often enough, so that the information on these is never available at the source during its activation periods (the tosses and settings cannot have a causal influence on the workings of source). The local measurement data (setting, result, and moment of measurement) are stored and very many runs of the experiment are performed (☞ [Fig. 1](#)).

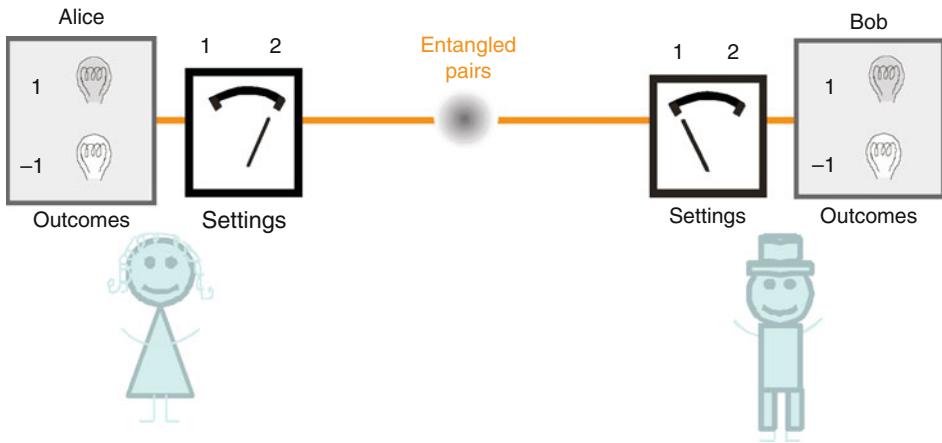
4.1.1 Assumptions Leading to Bell's Inequalities

A concise *local realistic* description of such an experiment would use the following assumptions (Gill et al. 2002, 2003):

1. We assume *realism*, which is any logically self-consistent model that allows one to use *eight* variables in the theoretical description of the experiment: $A_{m,n}, B_{n,m}$, where $n, m = 1, 2$. The variable $A_{m,n}$ gives the value, ± 1 , which could be obtained at station A , if the knob settings, at A and B , were at positions n, m , respectively. Similarly, $B_{n,m}$ plays the same role for station B , under the same settings. This is equivalent to the assumption that a joint (nonnegative, properly normalized) probability distribution of these variables,

Fig. 1

Test of Bell's inequalities. Alice and Bob are two separated parties who share entangled particles. Each of them is free to choose two measurement settings 1 and 2 and they observe flashes in their detection stations, which indicate one of the two possible measurement outcomes +1 or -1.



$P(A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}; B_{1,1}, B_{1,2}, B_{2,1}, B_{2,2}, n, m)$, is always allowed to exist. (Note, that no hidden variables appear, beyond these ten. However, given a (possibly stochastic) hidden variables theory, one will be able to define the ten variables as (possibly random) functions of the variables in that theory.)

2. The assumption of *locality* does not allow influences to go outside the light cone.
3. Alice and Bob are free to choose their settings “at the whim.” This is the *freedom*, or “*free will*,” often only a tacit assumption (Bell 1985, Kofler et al. 2006). A less provocative version of this assumption: *There exist stochastic processes, which could be used to choose the values of the local settings of the devices which are independent of the workings of the source, that is, they neither influence it nor are influenced by it.* By the previous assumptions, the events of activation of the source and of the choice and fixing of the local settings must be space-like separated and they furthermore cannot have common cause in the past.

Note that when setting labels m, n are sent to the measurement devices, they are likely to cause some unintended disturbance: by these assumptions, *any disturbance at A, as far as it influences the outcome at A, is not related to the coin toss nor to the potential outcomes at B, and vice versa.*

Note further, that $A_{n,m}$ and $B_{n,m}$ are not necessarily actual properties of the systems. The only thing that is assumed is that there is a theoretical description, which allows one to use all these *eight* values.

4.1.2 First Consequences

Let us write down the immediate consequences of these assumptions:

- By *locality*: for all n, m :

$$A_{m,n} = A_m, \quad B_{n,m} = B_n \quad (18)$$

That is, the outcome which would appear at A does not depend on which setting might be chosen at B , and vice versa. Thus $P(A_{1,1}, \dots, B_{2,2}, n, m)$ can be reduced to $P(A_1, A_2, B_1, B_2, n, m)$.

- By freedom

$$(n, m) \text{ is statistically independent of } (A_1, A_2, B_1, B_2) \quad (19)$$

Thus, the *overall* probability distributions for potential settings and potential outcomes satisfy

$$P(n, m, A_1, A_2, B_1, B_2) = P(n, m)P(A_1, A_2, B_1, B_2) \quad (20)$$

The choice of settings in the two randomizes, A and B , is causally separated from the local realistic mechanism, which produces the potential outcomes.

4.1.3 Lemma: Bell's Inequality

The probabilities, \Pr , of the four logical propositions, $A_n = B_m$, satisfy

$$\Pr\{A_1 = B_2\} - \Pr\{A_1 = B_1\} - \Pr\{A_2 = B_1\} - \Pr\{A_2 = B_2\} \leq 0 \quad (21)$$

Proof: Only four, or two, or none of the propositions in the left-hand side of the inequality can be true, thus (21). QED.

Now, if the observation settings are totally random (dictated by “coin tosses”), $P(n, m) = \frac{1}{4}$. Then, according to all the assumptions

$$P(A_n = B_m | n, m) = P(n, m) \Pr\{A_n = B_m\} = \frac{1}{4} \Pr\{A_n = B_m\} \quad (22)$$

Therefore, we have a Bell's inequality: under the *conjunction* of the assumptions for the *experimentally accessible* probabilities one has

$$P(A_1 = B_2 | 1, 2) - P(A_1 = B_1 | 1, 1) - P(A_2 = B_1 | 2, 1) - P(A_2 = B_2 | 2, 2) \leq 0 \quad (23)$$

This is the well-known Clauser–Horne–Shimony–Holt (CHSH) inequality (Clauser et al. 1969).

4.2 Bell's Theorem

Quantum mechanics predicts for experiments satisfying all the features of the thought experiment the left-hand side of inequality (23) to be as high as $\sqrt{2} - 1$, which is larger than the local realistic bound 0. Hence, one has Bell's theorem (Bell 1964): if quantum mechanics holds, local realism, defined by the full set of the above assumptions, is untenable. But, how does nature behave – according to local realism or quantum mechanics? It seems that we are approaching the moment in which one could have, as perfect as possible, a laboratory realization of the thought experiment (the locality loophole was closed in Weihs et al. (1998) and Aspect et al. (1982), the detection loophole was closed in Rowe et al. (2001), and in recent experiments measurement settings were space-like separated from the photon pair emission (Scheidl et al. 2008)). Hence, the local realistic approach to the description of physical phenomena is close to being shown untenable, too.

4.2.1 The Assumptions as a Communication Complexity Problem

Assume that we have two programmers P_k , where $k = 1, 2$, each possessing an enormously powerful computer. They share certain joint classically correlated strings of arbitrary lengths and/or some computer programs. All these will be collectively denoted as λ . But, once they both possess λ , no communication whatsoever between them is allowed. After this initial stage, each one of them gets from a referee a one-bit random number $x_k \in \{0, 1\}$, known only to him/her (P_1 knows only x_1 , P_2 knows only x_2). The *individual* task of each of them is to produce, via whatever computational program, a one-bit number $I_k(x_k, \lambda)$, and communicate only this one bit to a Referee, who just compares the received bits. There is no restriction on the form and complication of the *possibly stochastic* functions I_k , or any actions taken to define the values, but any communication between the partners is absolutely not allowed. The *joint* task of the partners is to devise a computer code which, under the constraints listed above and without any cheating, allows them to have, after very many repetitions of the procedures (each starting with establishing a new shared λ), the following functional dependence of the probability that their bits sent back to the Referee are equal:

$$P\{I_1(x_1) = I_2(x_2)\} = \frac{1}{2} + \frac{1}{2} \cos [-\pi/4 + (\pi/2)(x_1 + x_2)] \quad (24)$$

This is a variant of communication complexity problems. The current task is absolutely impossible to achieve with the classical means at their disposal, and without communication. Simply because whatever the protocol

$$\Pr\{I_1(1) = I_2(1)\} - \Pr\{I_1(0) = I_2(0)\} - \Pr\{I_1(1) = I_2(0)\} - \Pr\{I_1(0) = I_2(1)\} \leq 0 \quad (25)$$

whereas the value of this expression in the quantum strategy can be as high as $\sqrt{2} - 1$. This value can be obtained on average if the programmers use entanglement as a resource and receive their respective qubits from an entangled pair (e.g., singlet) during the communication stages (when λ is established). Instead of computing, the partners make a local measurement on their qubits. They measure Pauli observables $\mathbf{n} \cdot \boldsymbol{\sigma}$, where $\|\mathbf{n}\| = 1$. Since the probability for them to get identical results, r_1, r_2 , for observation directions $\mathbf{n}_1, \mathbf{n}_2$ is

$$P_Q\{r_1 = r_2 | \mathbf{n}_1, \mathbf{n}_2\} = \frac{1}{2} - \frac{1}{2} \mathbf{n}_1 \cdot \mathbf{n}_2 \quad (26)$$

for suitably chosen $\mathbf{n}_1(x_1), \mathbf{n}_2(x_2)$, they get values of P_Q equal to those in (Eq. 24). The messages sent back to the Referee encode the local results of measurements of $\mathbf{n}_1 \cdot \boldsymbol{\sigma} \otimes \mathbf{n}_2 \cdot \boldsymbol{\sigma}$, and the local measurement directions are suitably chosen as functions of x_1 and x_2 . The relation between Bell's inequalities and quantum communication complexity problems will be reviewed in more detail in (Sect. 6).

4.2.2 Philosophy or Physics? Which Assumptions?

The assumptions behind Bell's inequalities are often criticized as being “philosophical.” If we remind ourselves about Mach's influence on Einstein, philosophical discussions related to physics may be very fruitful.

For those who are, however, still skeptical one can argue as follows. The whole (relativistic) classical theory of physics is realistic (and local). Thus, we have an important exemplary realization of the postulates of local realism. Philosophical propositions could be defined as those which *are not* observationally or experimentally falsifiable at the given moment of the development of human knowledge or, in pure mathematical theory, are not logically derivable. Therefore, the *conjunction* of all assumptions of Bell's inequalities is not a philosophical statement, as it is *testable* both experimentally and logically (within the known current mathematical formulation of the fundamental laws of physics). Thus, Bell's theorem removed the question of the possibility of a local realistic description from the realm of philosophy. Now, this is just a question of a good experiment.

The other criticism is formulated in the following way. Bell's inequalities can be derived using a single assumption of the existence of a joint probability distribution for the observables involved in them, or that the probability calculus of the experimental propositions involved in the inequalities is of Kolmogorovian nature, and nothing more. But if we want to understand the violation of Bell's inequalities, we stumble on the following question: *Does the joint probability take into account the full experimental context or not?* The experimental context is in the present case (at least) the full state of the settings (m, n). Thus, if we use the same notation as above for the realistic values, this time applied to the possible results of measurements of observables, initially we can assume the existence of only $p(A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}; B_{1,1}, B_{1,2}, B_{2,1}, B_{2,2})$. Note that such a probability could be, for example, factorizable into $\prod_{n,m} P(A_{n,m} | B_{n,m})$. That is, one could, in such a case, have different probability distributions pertaining to different experimental contexts (which can even be defined through the choice of measurement settings in space-like separated laboratories!).

Let the discussion of this be from the quantum mechanical point of view, only because such considerations have a nice formal description within this theory, familiar to all physicists. Two observables, say $\hat{A}_1 \otimes \hat{B}_1$ and $\hat{A}_2 \otimes \hat{B}_2$, as well as other possible pairs are functions of two different *maximal* observables for the whole system (which are nondegenerate by definition). If one denotes such a maximal observable linked with $\hat{A}_m \otimes \hat{B}_n$ by $\hat{M}_{m,n}$ and its eigenvalues by $M_{m,n}$ the existence of the aforementioned joint probability is equivalent to the existence of a $p(M_{1,1}, M_{1,2}, M_{2,1}, M_{2,2})$ in the form of a proper probability distribution. Only if one assumes, additionally, context independence, can this be reduced to the question of the existence of (nonnegative) probabilities $P(A_1, A_2, B_1, B_2)$, where A_m and B_n are eigenvalues of $\hat{A}_m \otimes \mathbf{1}$ and $\mathbf{1} \otimes \hat{B}_n$, where, in turn, $\mathbf{1}$ is the unit operator for the given subsystem. While context independence is physically doubtful when the measurements are not spatially separated, and thus one can have mutual causal dependence, it is well justified for spatially separated measurements. That is, *locality* enters one's reasoning, whether we like it or not. Of course one cannot derive any Bell's inequality of the usual type if the random choice of settings is not independent of the distribution of A_1, A_2, B_1, B_2 , that is without (☞ Eq. 20).

There is yet another challenge to the set of assumptions presented above. It is often claimed that realism can be derived, once one considers the fact that maximally entangled quantum systems reveal perfect correlations, and one additionally assumes locality. Therefore, it would seem that the only basic assumption behind Bell's inequalities is locality, with the other auxiliary one of freedom. Such a claim is based on the ideas of EPR, who conjectured that one can introduce "elements of reality" of a remote system, provided this system is perfectly correlated with another system. To show the fallacy of such a hope, three particle correlations are now discussed, in the case of which consideration of just a few "elements of

“reality” reveals that they are a logically inconsistent notion. Therefore, they cannot be a starting point for deriving a self-consistent realistic theory. The three-particle reasoning is used here because of its beauty and simplicity, not because one cannot reach a similar conclusion for two-particle correlations.

4.3 Bell's Theorem Without Inequalities: Three Entangled Particles or More

As the simplest example, take a Greenberger–Horne–Zeilinger (Greenberger et al. 1989, 1990) (GHZ) state of $N = 3$ particles (Fig. 2):

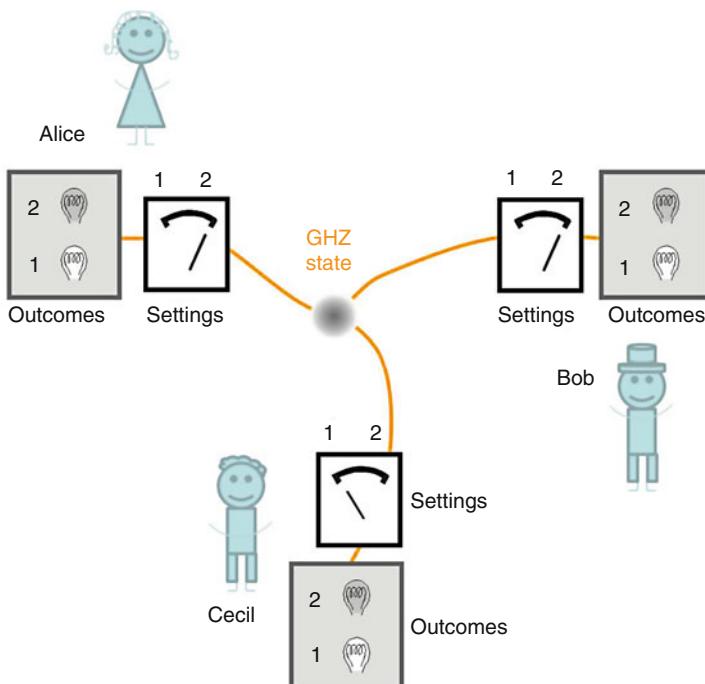
$$|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|a\rangle|b\rangle|c\rangle + |a'\rangle|b'\rangle|c'\rangle) \quad (27)$$

where $\langle x|x' \rangle = 0$ ($x = a, b, c$, and kets denoted by one letter pertain to one of the particles). The observers, Alice, Bob, and Cecil measure the observables: $\hat{A}(\phi_A)$, $\hat{B}(\phi_B)$, $\hat{C}(\phi_C)$, defined by

$$\hat{X}(\phi_X) = |+, \phi_X\rangle\langle +, \phi_X| - |-, \phi_X\rangle\langle -, \phi_X| \quad (28)$$

Fig. 2

Test of the GHZ theorem. Alice, Bob, and Cecil are three separated parties who share three entangled particles in the GHZ state. Each of them is free to choose between two measurement settings, 1 and 2, and they observe flashes in their detection stations, which indicate one of the two possible measurement outcomes +1 or -1.



where

$$|\pm, \phi_X\rangle = \frac{1}{\sqrt{2}} (\pm i|x'\rangle + \exp(i\phi_X)|x\rangle) \quad (29)$$

and $\hat{X} = \hat{A}, \hat{B}, \hat{C}$. The quantum prediction for the expectation value of the product of the three local observables is given by

$$E(\phi_A, \phi_B, \phi_C) = \langle \text{GHZ} | \hat{A}(\phi_A) \hat{B}(\phi_B) \hat{C}(\phi_C) | \text{GHZ} \rangle = \sin(\phi_A + \phi_B + \phi_C) \quad (30)$$

Therefore, if $\phi_A + \phi_B + \phi_C = \pi/2 + k\pi$, quantum mechanics predicts perfect (anti)correlations. For example, for $\phi_A = \pi/2$, $\phi_B = 0$ and $\phi_C = 0$, whatever the results of local measurements of the observables, for, say, the particles belonging to the i th triple of the measured ensemble represented by the quantum state $|\text{GHZ}\rangle$, their product must be unity. In a local realistic theory, one would have

$$A^i(\pi/2)B^i(0)C^i(0) = 1 \quad (31)$$

where $X^i(\phi)$, $X = A, B$, or C is the local realistic value of a local measurement of the observable $\hat{X}(\phi)$ that *would have been obtained* for the i th particle triple, if the setting of the measuring device were ϕ . By locality $X^i(\phi)$ depends solely on the local parameter. The \bullet Eq. 31 indicates that we can predict, with certainty, the result of measuring the observable, pertaining to one of the particles (say c), by choosing to measure suitable observables for the other two. Hence, the values $X^i(\phi)$ are EPR elements of reality.

However, if the local apparatus settings are different, one *would have had*, for example,

$$A^i(0)B^i(0)C^i(\pi/2) = 1 \quad (32)$$

$$A^i(0)B^i(\pi/2)C^i(0) = 1 \quad (33)$$

$$A^i(\pi/2)B^i(\pi/2)C^i(\pi/2) = -1 \quad (34)$$

Yet, the four statements (31–34) are inconsistent within local realism. Since $X^i(\phi) = \pm 1$, if one multiplies side by side \bullet Eqs. 31–33, the result is

$$1 = -1 \quad (35)$$

This shows that the mere concept of the existence of “elements of physical reality” as introduced by EPR is in contradiction with quantum mechanical predictions. We have a “Bell’s theorem without inequalities” (Greenberger et al. 1989, 1990).

Some people still claim that EPR correlations together with the assumption of locality allow us to derive realism. The above example clearly shows that such a realism would allow us to infer that $1 = -1$.

4.4 Implications of Bell’s Theorem

Violations of Bell’s inequalities imply that the underlying *conjunction of assumptions of realism, locality, and “free will”* is not valid, and *nothing more*.

It is often said that the violations indicate “(quantum) nonlocality.” However, if one wants *nonlocality* to be *the implication*, one has to assume “free will” and realism. But this is only at the moment a philosophical choice (it seems that there is no way to falsify it). *It is not a necessary condition for violations of Bell’s inequalities.*

Bell's theorem shows that even a local inherently probabilistic hidden-variable theory cannot agree with all the predictions of the quantum theory (considerations are based on $p(A_1, A_2, B_1, B_2)$ without assuming its actual structure, or whether the distribution for a single run is essentially deterministic, all we require is a joint “coexistence” of the variables A_1, \dots, B_2 in a *theoretical* description). Therefore, the above statements cover theories that treat probabilities as irreducible, and for which one can define $p(A_1, A_2, B_1, B_2)$. Such theories contradict quantum predictions. This, for some authors, indicates that nature is nonlocal. While the mere existence of Bohm's model (Bohm 1952) demonstrates that nonlocal hidden-variables are a logically valid option, it is now known that there are plausible models, such as Leggett's crypto-nonlocal hidden-variable model (Leggett 2003), that are in disagreement with both quantum predictions and experiment (Gröblacher et al. 2007; Branciard et al. 2007). But, perhaps more importantly, if one is ready to consider inherently probabilistic theories, then there is no immediate reason to require the existence of (nonnegative and normalized) probabilities $p(A_{1,1}, \dots, B_{2,2})$. Violation of this condition on realism, together with locality, which allows us to reduce the distribution to $p(A_1, \dots, B_2)$, is not in a *direct* conflict with the theory of relativity, as it does not necessarily imply the possibility of signaling superluminally. To the contrary, quantum correlations cannot be used for direct communication between Alice to Bob, but still violate Bell's inequalities. It is therefore legitimate to consider quantum theory as a probability theory subject to, or even derivable from, more general principles, such as the non-signaling condition (Popescu and Rohrlich 1994; Barrett 2007) or information-theoretical principles (von Weizsäcker 1985; Zeilinger 1999; Pawłowski et al. 2009; Dakic and Brukner 2009).

Note that complementarity, inherent in the quantum formalism (which can be mathematically expressed as the nonexistence of joint probabilities for non-commuting, i.e., non-commensurable, observables), completely contradicts the form of realism defined above. So why quantum-nonlocality?

To put it short, Bell's theorem does not imply *any* property of quantum mechanics. It just tells us what it is not.

5 All Bell's Inequalities for Two Possible Settings on Each Side

A general method of deriving *all* standard Bell's inequalities shall now be presented (that is, Bell's inequalities involving two-outcome measurements and with two settings per observer). Although these will not be spelt out explicitly, all the assumptions discussed above are behind the algebraic manipulations leading to the inequalities. A derivation for the two-observer problem is presented in detail, because the generalization to more observers is, surprisingly, obvious.

Consider pairs of particles (say, photons) simultaneously emitted in well-defined opposite directions. After some time, the photons arrive at two very distant measuring devices A and B operated by Alice and Bob. Alice, chooses to measure either observable \hat{A}_1 or \hat{A}_2 , and Bob either \hat{B}_1 or \hat{B}_2 . The hypothetical results that they may get for the j th pair of photons are A_1^j and A_2^j , for Alice's two possible choices, and B_1^j and B_2^j , for Bob's. The numerical values of these results (+1 or -1) are defined by the two eigenvalues of the observables.

Since always either $|B_1^j - B_2^j| = 2$ and $|B_1^j + B_2^j| = 0$, or $|B_1^j - B_2^j| = 0$ and $|B_1^j + B_2^j| = 2$, with a similar property of Alice's hypothetical results, the following relation holds

$$|A_1^j \pm A_2^j| \cdot |B_1^j \pm B_2^j| = 0 \quad (36)$$

for all possible sign choices within $\textcircled{2}$ Eq. 36 except one, for which we have 4. Therefore

$$\sum_{k,l=0}^1 |(A_1^j + (-1)^k A_2^j)(B_1^j + (-1)^l B_2^j)| = 4 \quad (37)$$

or equivalently we have the set of identities

$$\sum_{s_1,s_2=-1}^1 S(s_1,s_2)[(A_1^j + s_1 A_2^j)(B_1^j + s_2 B_2^j)] = \pm 4 \quad (38)$$

with any $S(s_1,s_2) = \pm 1$. There are $2^2 = 16$ such S functions.

Imagine now that N pairs of photons are emitted, pair by pair (N is sufficiently large, such that $\sqrt{1/N} \ll 1$). The average value of the products of the local values is given by

$$E(A_n, B_m) = \frac{1}{N} \sum_{j=1}^N A_n^j B_m^j \quad (39)$$

where $n, m = 1, 2$.

Therefore, after averaging, the following single Bell-type inequality emerges:

$$\sum_{k,l=0}^1 |E(A_1, B_1) + (-1)^l E(A_1, B_2) + (-1)^k E(A_2, B_1) + (-1)^{k+l} E(A_2, B_2)| \leq 4 \quad (40)$$

or, equivalently, a series of inequalities:

$$-4 \leq \sum_{s_1,s_2=-1}^1 S(s_1,s_2)[E(A_1, B_1) + s_2 E(A_1, B_2) + s_1 E(A_2, B_1) + s_1 s_2 E(A_2, B_2)] \leq 4 \quad (41)$$

As the choice of the measurement settings is assumed to be statistically independent of the working of the source, that is, of the distribution of A_1 's, A_2 's, B_1 's, and B_2 's, the averages $E(A_n, B_m)$ cannot differ much, for high N , from the *actually observed* ones in the subsets of runs for which the given pair of settings was selected.

5.1 Completeness of the Inequalities

The inequalities form a complete set. That is, they define the faces of the convex polytope formed out of all possible local realistic models for the given set of measurements. Whenever the local realistic model exists, inequality $(\textcircled{40})$ is satisfied by its predictions. To prove the sufficiency of condition $(\textcircled{40})$, we construct a local realistic model for any correlation functions that satisfy it, that is, we are interested in the local realistic models for $E_{k_1 k_2}^{LR}$, such that they fully agree with the measured correlations $E(k_1, k_2)$ for all possible observables $k_1, k_2 = 1, 2$.

One can introduce \hat{E} which is a “tensor” or matrix built out of E_{ij} with $i, j = 1, 2$. If all its components can be derived from local realism, one must have

$$\hat{E}_{LR} = \sum_{\mathbf{A}, \mathbf{B}} P(\mathbf{A}, \mathbf{B}) \mathbf{A} \otimes \mathbf{B} \quad (42)$$

with $\mathbf{A} = (A_1, A_2)$, $\mathbf{B} = (B_1, B_2)$ and nonnegative probabilities $P(\mathbf{A}, \mathbf{B})$. The summation is over all possible values of \mathbf{A} and \mathbf{B} .

Let us ascribe for fixed s_1, s_2 , a hidden probability that $A_1 = s_1 A_2$ and $B_1 = s_2 B_2$ in the form familiar from [Eq. 40](#):

$$P(s_1, s_2) = \frac{1}{4} \left| \sum_{k_1, k_2=1}^2 s_1^{k_1-1} s_2^{k_2-1} E(k_1, k_2) \right| \quad (43)$$

Obviously these probabilities are positive. However, they sum up to identity only if inequality ([Eq. 40](#)) is saturated, otherwise there is a “probability deficit,” ΔP . This deficit can be compensated without affecting the correlation functions, see below.

First we construct the following structure, which is indeed the local realistic model of the set of correlation functions if the inequality is saturated:

$$\sum_{s_1, s_2=-1}^1 \Sigma(s_1, s_2) P(s_1, s_2) (1, s_1) \otimes (1, s_2) \quad (44)$$

where $\Sigma(s_1, s_2)$ is the sign of the expression within the modulus in [Eq. 43](#).

Now, if $\Delta P > 0$, a “tail” is added to this expression given by:

$$\frac{\Delta P}{16} \sum_{A_1=-1}^1 \sum_{A_2=-1}^1 \sum_{B_1=-1}^1 \sum_{B_2=-1}^1 (A_1, A_2) \otimes (B_1, B_2) \quad (45)$$

This “tail” does not contribute to the values of the correlation functions, because it represents the fully random noise. The sum of [Eq. 44](#) is a valid local realistic model for $\hat{E} = (E(1, 1), E(1, 2), E(2, 1), E(2, 2))$. The sole role of the “tail” is to make all hidden probabilities add up to 1.

To give the reader some intuitive grounds for the actual form and the completeness of the derived inequalities, some remarks shall now be given. The gist is that the consecutive terms in the inequalities are just expansion coefficients of the tensor \hat{E} in terms of a complete orthogonal sequence of basis tensors. Thus, the expansion coefficients represent the tensors in a one-to-one way.

In the four-dimensional real space, where both \hat{E}_{LR} and \hat{E} are defined, one can find an orthonormal basis set $\hat{S}_{s_1 s_2} = \frac{1}{2}(1, s_1) \otimes (1, s_2)$. Within these definitions the hidden probabilities acquire a simple form:

$$P(s_1, s_2) = \frac{1}{2} |\hat{S}_{s_1 s_2} \cdot \hat{E}| \quad (46)$$

where the dot denotes the scalar product in \mathbf{R}^4 . Now the local realistic correlations, \hat{E}_{LR} , can be expressed as:

$$\hat{E}_{LR} = \sum_{s_1, s_2=-1}^1 |\hat{S}_{s_1 s_2} \cdot \hat{E}| \Sigma(s_1, s_2) \hat{S}_{s_1 s_2} \quad (47)$$

The modulus of any number $|x|$ can be split into $|x| = x \operatorname{sign}(x)$, and we can always demand that the product $A_1 B_1$ has the same sign as the expression inside the modulus. Thus, we have

$$\hat{E} = \sum_{s_1, s_2=-1}^1 (\hat{S}_{s_1 s_2} \cdot \hat{E}) \hat{S}_{s_1 s_2} \quad (48)$$

The expression in the bracket is the coefficient of tensor \hat{E} in the basis $\hat{S}_{s_1 s_2}$. These coefficients are then summed over the same basis vectors, therefore the last equality appears.

5.2 Two-Qubit States That Violate the Inequalities

A general two-qubit state can be expressed in the following concise form

$$\hat{\rho} = \frac{1}{4} \sum_{\mu,\nu=0}^3 T_{\mu\nu} (\hat{\sigma}_\mu^1 \otimes \hat{\sigma}_\nu^2) \quad (49)$$

The two-qubit correlation function for measurements of spin 1 along direction $\mathbf{n}(1)$ and of spin 2 along $\mathbf{n}(2)$ is given by

$$E_{QM}(\mathbf{n}(1), \mathbf{n}(2)) = \text{Tr}[\hat{\rho}(\mathbf{n}(1) \cdot \hat{\mathbf{o}}^1 \otimes \mathbf{n}(2) \cdot \hat{\mathbf{o}}^2)] \quad (50)$$

and it reads

$$E_{QM}(\mathbf{n}(1), \mathbf{n}(2)) = \sum_{i,j=1}^3 T_{ij} n(1)_i n(2)_j \quad (51)$$

Two-particle correlations are fully defined once we know the components of T_{ij} , $i, j = 1, 2, 3$, of the tensor $\hat{\mathbf{T}}$. \bullet [Equation 51](#) can be put into a more convenient form:

$$E_{QM}(\mathbf{n}(1), \mathbf{n}(2)) = \hat{\mathbf{T}} \cdot \mathbf{n}(1) \otimes \mathbf{n}(2) \quad (52)$$

where “.” is the scalar product in the space of tensors, which in turn is isomorphic with $\mathbf{R}^3 \otimes \mathbf{R}^3$.

According to (40) the quantum correlation $E_{QM}(\mathbf{n}(1), \mathbf{n}(2))$ can be described by a local realistic model if and only if, for any choice of the settings $\mathbf{n}(1)^{k_1}$ and $\mathbf{n}(2)^{k_2}$, where $k_1, k_2 = 1, 2$, we have

$$\frac{1}{4} \sum_{k,l=1}^2 |\hat{\mathbf{T}} \cdot [\mathbf{n}(1)^1 + (-1)^k \mathbf{n}(1)^2] \otimes [\mathbf{n}(2)^1 + (-1)^l \mathbf{n}(2)^2]| \leq 1 \quad (53)$$

Since there always exist two mutually orthogonal unit vectors $\mathbf{a}(x)^1$ and $\mathbf{a}(x)^2$ such that

$$\mathbf{n}(x)^1 + (-1)^k \mathbf{n}(x)^2 = 2\alpha(x)_k \mathbf{a}(x)^k \text{ with } k = 1, 2 \quad (54)$$

and with $\alpha(x)_1 = \cos \theta(x)$, $\alpha(x)_2 = \sin \theta(x)$, we obtain

$$\sum_{k,l=1}^2 |\alpha(1)_k \alpha(2)_l \hat{\mathbf{T}} \cdot \mathbf{a}(1)^k \otimes \mathbf{a}(2)^l| \leq 1 \quad (55)$$

Note that $\hat{\mathbf{T}} \cdot \mathbf{a}(1)^k \otimes \mathbf{a}(2)^l$ is a component of the tensor $\hat{\mathbf{T}}$ after a transformation of the local coordinate systems of each of the particles into those where the two first basis vectors are $\mathbf{a}(x)^1$ and $\mathbf{a}(x)^2$. We shall denote such transformed components again by T_{kl} .

The necessary and sufficient condition for a two-qubit correlation to be described, within a local realistic model, is that in any plane of observation for each particle (defined by the two observation directions) we must have

$$\sum_{k,l=1}^2 |\alpha(1)_k \alpha(2)_l T_{kl}| \leq 1 \quad (56)$$

for arbitrary $\alpha(1)_k$, $\alpha(2)_l$.

Using the Cauchy inequality, we obtain

$$\sum_{k,l=1}^2 |\alpha(1)_k \alpha(2)_l T_{kl}| \leq \sqrt{\sum_{k,l=1}^2 T_{kl}^2} \quad (57)$$

Therefore, if

$$\sum_{k,l=1}^2 T_{kl}^2 \leq 1 \quad (58)$$

for any set of local coordinate systems, the two-particle correlation functions of the form of \bullet Eq. 51 can be understood within the local realism (in a two-settings-per-observer experiment).

As will be shown below, this condition is both necessary and sufficient.

5.2.1 Sufficient Condition for Violation of the Inequality

The full set of inequalities is derivable from the identity (\bullet 38) where we put non-factorable sign function $S(s_1, s_2) = \frac{1}{2}(1 + s_1) + (1 - s_1)s_2$. In this case, we obtain the CHSH inequality in its standard form

$$\left| \langle (A_1 + A_2)B_1 + (A_1 - A_2)B_2 \rangle_{\text{avg}} \right| \leq 2 \quad (59)$$

where $\langle \dots \rangle_{\text{avg}}$ denotes the average over an ensemble of particles. All other nontrivial inequalities are obtainable by all possible sign changes $X_k \rightarrow -X_k$ (with $k = 1, 2$ and $X = A, B$). It is easy to see that factorizable sign functions, such as, for example, $S(s_1, s_2) = s_1 s_2$, lead to trivial inequalities $|E(A_m, B_m)| \leq 1$. As noted above, the quantum correlation function, $E_Q(\mathbf{a}_k, \mathbf{b}_l)$, is given by the scalar product of the correlation tensor $\hat{\mathbf{T}}$ with the tensor product of the local measurement settings represented by unit vectors $\mathbf{a}_k \otimes \mathbf{b}_l$, that is, $E_Q(\mathbf{a}_k, \mathbf{b}_l) = (\mathbf{a}_k \otimes \mathbf{b}_l) \cdot \hat{\mathbf{T}}$. Thus, the condition for a quantum state endowed with the correlation tensor $\hat{\mathbf{T}}$ to satisfy the inequality (\bullet 59), is that for all directions $\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2$ we have

$$\left| \left[\left(\frac{\mathbf{a}_1 + \mathbf{a}_2}{2} \right) \otimes \mathbf{b}_1 + \left(\frac{\mathbf{a}_1 - \mathbf{a}_2}{2} \right) \otimes \mathbf{b}_2 \right] \cdot \hat{\mathbf{T}} \right| \leq 1 \quad (60)$$

where both sides of \bullet Eq. 59 were divided by 2.

Next notice that $\mathbf{A}_{\pm} = \frac{1}{2}(\mathbf{a}_1 \pm \mathbf{a}_2)$ satisfy the following relations: $\mathbf{A}_+ \cdot \mathbf{A}_- = 0$ and $\|\mathbf{A}_+\|^2 + \|\mathbf{A}_-\|^2 = 1$. Thus $\mathbf{A}_+ + \mathbf{A}_-$ is a unit vector, and \mathbf{A}_{\pm} represent its decomposition into two orthogonal vectors. If one introduces unit vectors \mathbf{a}_{\pm} such that $\mathbf{A}_{\pm} = a_{\pm}\mathbf{a}_{\pm}$, we have $a_+^2 + a_-^2 = 1$. Thus, we can put inequality (\bullet 60) into the following form:

$$|\hat{\mathbf{S}} \cdot \hat{\mathbf{T}}| \leq 1 \quad (61)$$

where $\hat{\mathbf{S}} = a_+ \mathbf{a}_+ \otimes \mathbf{b}_1 + a_- \mathbf{a}_- \otimes \mathbf{b}_2$. Note that since $\mathbf{a}_+ \cdot \mathbf{a}_- = 0$, we have $\hat{\mathbf{S}} \cdot \hat{\mathbf{S}} = 1$, that is, $\hat{\mathbf{S}}$ is a tensor of unit norm. Any tensor of unit norm, $\hat{\mathbf{U}}$, has the following Schmidt decomposition $\hat{\mathbf{U}} = \lambda_1 \mathbf{v}_1 \otimes \mathbf{w}_1 + \lambda_2 \mathbf{v}_2 \otimes \mathbf{w}_2$, where $\mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$, $\mathbf{w}_i \cdot \mathbf{w}_j = \delta_{ij}$ and $\lambda_1^2 + \lambda_2^2 = 1$. The (complete) freedom of the choice of the measurement directions \mathbf{b}_1 and \mathbf{b}_2 allows us by choosing \mathbf{b}_2 orthogonal to \mathbf{b}_1 to put $\hat{\mathbf{S}}$ in the form isomorphic with $\hat{\mathbf{U}}$, and the freedom of choice of \mathbf{a}_1 and \mathbf{a}_2 allows \mathbf{A}_+ and \mathbf{A}_- to be arbitrary orthogonal unit vectors, and \mathbf{a}_+ and \mathbf{a}_- to also be arbitrary. Thus $\hat{\mathbf{S}}$ can be equal to any unit tensor. To get the maximum of the left-hand side of \bullet Eq. 60, we Schmidt decompose the correlation tensor and take two terms of the decomposition which have the largest coefficients. In this way, we get a tensor $\hat{\mathbf{T}}'$, of Schmidt rank two.

We put $\hat{\mathbf{S}} = \frac{1}{\|\hat{\mathbf{T}}'\|} \hat{\mathbf{T}}'$, and the maximum is $\|\hat{\mathbf{T}}'\| = \sqrt{\hat{\mathbf{T}}' \cdot \hat{\mathbf{T}}'}$. Thus, in other words,

$$\max \left[\sum_{k,l=1}^2 T_{kl}^2 \right] \leq 1 \quad (62)$$

is the necessary and sufficient condition for the inequality (❷ 40) to hold, provided the maximization is taken over all local coordinate systems of two observers. The condition is equivalent to the necessary and sufficient condition of the Horodecki family (Horodecki et al. 1995) for violation of the CHSH inequality.

5.3 Bell's Inequalities for N Particles

Let us consider a Bell's inequality test with N observers. Each of them chooses between two possible observables, determined by local parameters $\mathbf{n}_1(j)$ and $\mathbf{n}_2(j)$, where $j = 1, \dots, N$. Local realism implies the existence of two numbers A_1^j and A_2^j , each taking values +1 or -1, which describe the predetermined result of a measurement by the j th observer for the two observables. The following algebraic identity holds:

$$\sum_{s_1, \dots, s_N=-1}^1 S(s_1, \dots, s_N) \prod_{j=1}^N [A_1^j + s_j A_2^j] = \pm 2^N \quad (63)$$

where $S(s_1, \dots, s_N)$ is an arbitrary “sign” function, that is, $S(s_1, \dots, s_N) = \pm 1$. It is a straightforward generalization of that for two observers as given in ❷ Eq. 38. The correlation function is the average over many runs of the experiment $E_{k_1, \dots, k_N} = \left\langle \prod_{j=1}^N A_{k_j}^j \right\rangle_{\text{avg}}$ with $k_1, \dots, k_N \in \{1, 2\}$. After averaging ❷ Eq. 63 over the ensemble of the runs, we obtain the Bell's inequalities. (This set of inequalities is a sufficient and necessary condition for the correlation functions entering them to have a local realistic model; compare it to the two-particle case in 5.1.)

$$\left| \sum_{s_1, \dots, s_N=-1}^1 S(s_1, \dots, s_N) \sum_{k_1, \dots, k_N=1}^2 s_1^{k_1-1} \dots s_N^{k_N-1} E_{k_1, \dots, k_N} \right| \leq 2^N \quad (64)$$

Since there are 2^{2^N} different functions S , the above inequality represents a set of 2^{2^N} Bell's inequalities.

All these boil down to just one inequality (!):

$$\sum_{s_1, \dots, s_N=-1}^1 \left| \sum_{k_1, \dots, k_N=1}^2 s_1^{k_1-1} \dots s_N^{k_N-1} E_{k_1, \dots, k_N} \right| \leq 2^N \quad (65)$$

The proof of this fact is a trivial exercise with the use of the property that either $|X| = X$ or $|X| = -X$, where X is a real number. An inequality equivalent to (65) was derived independently in Weinfurter and Żukowski (2001) and Werner and Wolf (2001). The presented derivation follows Żukowski and Brukner (2002) where the necessary and sufficient condition for the violation of the inequality was obtained.

5.4 N -Qubit Correlations

A general N -qubit state can be put in the form

$$\hat{\rho} = \frac{1}{2^N} \sum_{\mu_1, \dots, \mu_N=0}^3 T_{\mu_1 \dots \mu_N} (\otimes_{k=1}^N \hat{\sigma}_{\mu_k}^k) \quad (66)$$

Thus, the N qubit correlation function has the following structure

$$E_{QM}(\mathbf{n}(1), \mathbf{n}(2), \dots, \mathbf{n}(N)) = \hat{\mathbf{T}} \cdot \mathbf{n}(1) \otimes \mathbf{n}(2) \dots \otimes \mathbf{n}(N) \quad (67)$$

This expression when inserted into the Bell inequality gives

$$1/2^N \sum_{k_1, \dots, k_N=1} |\hat{\mathbf{T}} \cdot \prod_{x=1}^N (\mathbf{n}^1(x) + (-1)^x \mathbf{n}^2(x))| \leq 1$$

By introducing for every observer $x = 1, \dots, N$ two mutually orthogonal unit vectors $\mathbf{a}^1(x)$ and $\mathbf{a}^2(x)$, just as in (54), and denoting $\hat{\mathbf{T}} \cdot \mathbf{a}^{k_1}(1) \otimes \dots \otimes \mathbf{a}^{k_N}(N) = T_{k_1 \dots k_N}$, where $k_i = 1, 2, 3$, we obtain the necessary and sufficient condition for a description of the correlation function within local realism:

$$\sum_{k_1, k_2, \dots, k_N=1}^2 |\alpha(1)_{k_1} \alpha(2)_{k_2} \dots \alpha(N)_{k_N} T_{k_1 k_2 \dots k_N}| \leq 1 \quad (68)$$

for any possible choice of local coordinate systems for individual particles. Again if

$$\sum_{k_1, \dots, k_N=1}^2 T_{k_1 \dots k_N}^2 \leq 1 \quad (69)$$

for any set of local coordinate systems, the N -qubit correlation function can be described by a local realistic model. The proofs of these facts are generalizations of those presented earlier, pertaining to two particles. The sufficient condition for violation of the general Bell's inequality for N particles by a general state of N qubits can be found in Źukowski and Brukner (2002).

5.5 Concluding Remarks

The inequalities presented above represent the full set of standard “tight” Bell’s inequalities for an arbitrary number of parties. Any non-tight inequality is weaker than tight ones. Such Bell’s inequalities can be used to detect entanglement, however not as efficiently as entanglement witnesses. Nevertheless, they have the advantage over the witnesses, as they are system independent. They detect entanglement no matter the actual Hilbert space which describes the subsystems.

As will be shown below, the entanglement violating Bell’s inequalities analyzed above is directly applicable in some quantum informational protocols that beat any classical ones of the same kind. This will be shown via an explicit construction of such protocols.

6 Quantum Reduction of Communication Complexity

In his review paper entitled “Quantum Communication Complexity (A Survey)” Brassard (2003) posed a question: “*Can entanglement be used to save on classical communication?*” He continued that there are good reasons to believe at first that the answer to the question is negative. Holevo’s theorem (Holevo 1973) states that no more than n bits of classical information can be communicated between parties by the transmission of n qubits regardless of the coding scheme as long as no entanglement is shared between parties. If the

communicating parties share prior entanglement, twice as much classical information can be transmitted (this is the so-called “superdense coding” (Bennett and Wiesner 1992)), but no more. It is thus reasonable to expect that even if the parties share entanglement, no savings in communication can be achieved beyond that of the superdense coding ($2n$ bits per n qubits transmitted).

It is also well known that entanglement alone cannot be used for communication. Local operations performed on any subsystem of an entangled composite system cannot have any observable effect on any other subsystem; otherwise it could be exploited to communicate faster than light. One would thus intuitively conclude that entanglement is useless for saving communication. Brassard, however, concluded “... *all the intuition in this paragraph is wrong.*”

The topic of classical communication complexity was introduced and first studied by Yao (1979). A typical communication complexity problem can be formulated as follows. Let Alice and Bob be two separated parties who receive some input data of which they know only their own data and not the data of the partner. Alice receives an input string x and Bob an input string y and the goal for both of them is to determine the value of a certain function $f(x, y)$. Before they start the protocol, Alice and Bob are even *allowed to share (classically correlated) random strings* or any other data, which might improve the success of the protocols. They are allowed to process their data locally in whatever way. The obvious method to achieve the goal is for Alice to communicate x to Bob, which allows him to compute $f(x, y)$. Once obtained, Bob can then communicate the value $f(x, y)$ back to Alice. It is the topic of communication complexity to address the questions: *Could there be more efficient solutions for some functions $f(x, y)$? What are these functions?*

A trivial example that there could be more efficient solutions than the obvious one given above is a constant function $f(x, y) = c$, where c is a constant. Obviously here Alice and Bob do not need to communicate at all, as they can simply take c for the value of the function. However, there are functions for which the only obvious solution is optimal, that is, only the transmission of x to Bob warrants that he reaches the correct result. For instance, it is shown that n bits of communication are necessary and sufficient for Bob to decide whether or not Alice’s n -bit input is the same as his (Brassard 2003; Kushilevitz and Nisan 1997).

Generally, one might distinguish the following two types of communication complexity problems:

1. What is the minimal amount of communication (minimal number of bits) required for the parties to determine the value of the function with certainty?
2. What is the highest possible probability for the parties to arrive at the correct value for the function if only a *restricted* amount of communication is allowed?

Here, only the second class of problems will be considered. Note that in this case one does not insist on the correct value of the function to be obtained with certainty. While an error in computing the function is allowed, the parties try to compute it correctly with as high probability as possible.

From the perspective of the physics of quantum information processing, the natural question is: *Are there communication complexity tasks for which the parties could increase the success in solving the problem if they share prior entanglement?* In their original paper Cleve and Buhrman (1997) showed that entanglement can indeed be used to save classical communication. They showed that, to solve a certain three-party problem with certainty, the parties need to broadcast at least 4 bits of information in a classical protocol, whereas in the quantum protocol (with entanglement shared), it is sufficient for them to broadcast only 3 bits of

information. This was the first example of a communication complexity problem that could be solved with higher success than would be possible with any classical protocol. Subsequently, Buhrman et al. (1997) found a two-party problem that could be solved with a probability of success exceeding 85% and 2 bits of information communicated if prior shared entanglement is available, whereas the probability of success in a classical protocol could not exceed 75% with the same amount of communication.

The first problem whose quantum solution requires a significantly smaller amount of communication compared to classical solutions was discovered by Buhrman et al. (1999). They considered a k -party task, which requires roughly $k \ln k$ bits of communication in a classical protocol, and exactly k bits of classical communication if the parties are allowed to share prior entanglement. The quantum protocol of Buhrman et al. (1997) is based on the violation of the CHSH inequality by a two-qubit maximally entangled state. Similarly, the quantum protocols for multiparty problems (Buhrman et al. 1997, 1999; Cleve and Buhrman 1997) are based on an application of the GHZ-type argument against local realism for multi-qubit maximally entangled states. Galvao (2001) has shown an equivalence between the CHSH and GHZ tests for three particles and the two- and three-party quantum protocols of Buhrman et al. (1997), respectively. In a series of papers (Brukner et al. 2002, 2003, 2004b; Trojek et al. 2005), it was shown that entanglement violating Bell's inequality can always be exploited to find a better-than-any-classical solution to some communication complexity problems. In this brief overview, the approach mainly followed is the one introduced in these papers. This approach was further developed and applied in Augusiak and Horodecki (2006) and Tamir (2007) (see also Marcovitch and Reznik 2008).

6.1 The Problem and Its Optimal Classical Solution

Imagine several spatially separated partners, P_1 to P_N , each of which has some data known to him/her only, denoted here as X_i , with $i = 1, \dots, N$. They face a joint task: to compute the value of a function $T(X_1, \dots, X_N)$. This function depends on all data. Obviously, they can get the value of T by sending all their data to partner P_N , who does the calculation and announces the result. But are there ways to reduce the amount of communicated bits, that is, to reduce the communication complexity of the problem?

Assume that every partner P_k receives a two-bit string $X_k = (z_k, x_k)$ where $z_k, x_k \in \{0, 1\}$. We shall consider specific binary task functions that have the following form

$$T = f(x_1, \dots, x_N)(-1)^{\sum_{k=1}^N z_k}$$

where $f \in \{-1, 1\}$. The partners also know the probability distribution (“promise”) of the bit strings (“inputs”). There are two constraints on the problem. First, only distributions shall be considered that are completely random with respect to z_k 's, that is a class of the form $p(X_1, \dots, X_N) = 2^{-N} p'(x_1, \dots, x_N)$. Second, communication between the partners is restricted to $N - 1$ bits. Assume that we ask the last partner to give his/her answer $A(X_1, \dots, X_N)$, equal to ± 1 , to the question of what is the functional value of $T(X_1, \dots, X_N)$ in each run for the given set of inputs X_1, \dots, X_N .

For simplicity, we introduce $y_k = (-1)^{z_k}$, $y_k \in \{-1, 1\}$. We shall use y_k as a synonym of z_k . Since T is proportional to $\prod_k y_k$ the final answer A is completely random if it does not depend on *every* y_k . Thus, information on z_k 's from all $N - 1$ partners must somehow reach P_N .

Therefore, the only communication “trees” which might lead to success are those in which each P_k sends only a one-bit message $m_k \in \{0, 1\}$. Again we introduce $e_k = (-1)^{m_k}$ and e_k will be treated as a synonym of m_k .

The average success of a communication protocol can be measured with the following fidelity function

$$F = \sum_{X_1, \dots, X_N} p(X_1, \dots, X_N) T(X_1, \dots, X_N) A(X_1, \dots, X_N) \quad (70)$$

or equivalently

$$F = \frac{1}{2^N} \sum_{x_1, \dots, x_N=0}^1 p'(x_1, \dots, x_N) f(x_1, \dots, x_N) \sum_{y_1, \dots, y_N=-1}^1 \prod_{k=1}^N y_k A(x_1, \dots, x_N; y_1, \dots, y_N) \quad (71)$$

The probability of success is $P = (1 + F)/2$.

The first steps of a derivation of the reduced form of the fidelity function for an optimal classical protocol will now be presented (the reader may reconstruct the other steps or consult references Brukner et al. (2004b) and Trojek et al. (2005)). In a classical protocol, the answer A of the partner P_N can depend on the local input y_N , x_N , and messages, e_{i_1}, \dots, e_{i_l} , received directly from a subset of l partners P_{i_1}, \dots, P_{i_l} :

$$A = A(x_N, y_N, e_{i_1}, \dots, e_{i_l}) \quad (72)$$

Let us fix x_N and treat A as a function A_{x_N} of the remaining $l + 1$ dichotomic variables

$$y_N, e_{i_1}, \dots, e_{i_l}$$

That is, we now treat x_N as a fixed index. All such functions can be thought of as 2^{l+1} -dimensional vectors, because the values of each such function form a sequence of length equal to the number of elements in the domain. In the 2^{l+1} -dimensional space containing such functions, we have an orthogonal basis given by

$$V_{j_1 \dots j_l}(y_N, e_{i_1}, \dots, e_{i_l}) = y_N^j \prod_{k=1}^l e_{i_k}^{j_k} \quad (73)$$

where $j, j_1, \dots, j_l \in \{0, 1\}$. Thus, we can expand $A(x_N, y_N, e_{i_1}, \dots, e_{i_l})$ with respect to this basis and the expansion coefficients read

$$c_{j_1 \dots j_l}(x_N) = \frac{1}{2^{l+1}} \sum_{y_N, e_{i_1}, \dots, e_{i_l}=-1}^1 A(x_N, y_N, e_{i_1}, \dots, e_{i_l}) V_{j_1 \dots j_l}(y_N, e_{i_1}, \dots, e_{i_l}) \quad (74)$$

Since $|A| = |V_{j_1 \dots j_l}| = 1$, we have $|c_{j_1 \dots j_l}(x_N)| \leq 1$. The expansion is put into the expression (71) for F and

$$F = \frac{1}{2^N} \sum_{x_1, \dots, x_N=0}^1 g(x_1, \dots, x_N) \sum_{y_1, \dots, y_N=-1}^1 \prod_{h=1}^N y_h \left[\sum_{j_1, \dots, j_l=0}^1 c_{j_1 \dots j_l}(x_N) y_N^j \prod_{k=1}^l e_{i_k}^{j_k} \right] \quad (75)$$

is obtained, where $g(x_1, \dots, x_N) \equiv f(x_1, \dots, x_N) p'(x_1, \dots, x_N)$. Because $\sum_{y_N=-1}^1 y_N y_N^0 = 0$, and $\sum_{y_k=-1}^1 y_k e_k^0 = 0$, only the term with all j, j_1, \dots, j_l equal to unity can give a nonzero contribution to F . Thus, A in F can be replaced by

$$A' = y_N c_N(x_N) \prod_{k=1}^l e_{i_k} \quad (76)$$

where $c_N(x_N)$ stands for $c_{11\dots 1}(x_N)$. Next, notice that, for example, e_{i_1} , can depend only on local data x_{i_1}, y_{i_1} and the messages obtained by P_{i_1} from a subset of partners: e_{p_1}, \dots, e_{p_m} . This set does not contain any of the e_{i_k} 's of the formula (Eq. 76) above. In analogy with A , the function e_{i_1} , for a fixed x_{i_1} , can be treated as a vector, and thus can be expanded in terms of orthogonal basis functions (of a similar nature as Eq. 73), etc. Again, the expansion coefficients satisfy $|c'_{j_1\dots j_m}(x_{i_1})| \leq 1$. If we put this into A' , we obtain a new form of F , which after a trivial summation over y_N and y_{i_1} depends on $c_N(x_N)c_{i_1}(x_i)\prod_{k=2}^l e_{i_k}$, where $c_{i_1}(x_i)$ stands for $c_{11\dots 1}(x_{i_1})$, and its modulus is again bounded by 1. Note that y_N and y_{i_1} disappear, as $y_k^2 = 1$.

As each message appears in the product only once, we continue this procedure of expanding those messages which depend on earlier messages, till it halts. The final reduced form of the formula for the fidelity of an optimal protocol reads

$$F = \sum_{x_1, \dots, x_N=0}^1 g(x_1, \dots, x_N) \prod_{n=1}^N c_n(x_n) \quad (77)$$

with $|c_n(x_n)| \leq 1$. Since F in Eq. 77 is linear in every $c_n(x_n)$, its extrema are at the limiting values $c_n(x_n) = \pm 1$. In other words, a Bell-like inequality $|F| \leq \text{Max}(F) \equiv B(N)$ gives the upper fidelity bound. Note that the above derivation shows that optimal classical protocols include one in which partners P_1 to P_{N-1} send to P_N one-bit messages which encode the value of $e_k = y_k c(x_k)$, where $k = 1, 2, \dots, N - 1$.

6.2 Quantum Solutions

The inequality for F suggests that some problems may have quantum solutions, which surpass any classical ones in their fidelity. Simply, one may use an entangled state $|\psi\rangle$ of N qubits that violates the inequality. Send to each of the partners one of the qubits. In a protocol run all N partners make measurements on the local qubits, the settings of which are determined by x_k . They measure a certain qubit observable $\mathbf{n}_k(x_k) \cdot \boldsymbol{\sigma}$. The measurement results $\gamma_k = \pm 1$ are multiplied by y_k and the partner P_k for $1 \leq k \leq N - 1$, sends a bit message to P_N encoding the value of $m_k = y_k \gamma_k$. The last partner calculates $y_N \gamma_N \prod_{k=1}^{N-1} m_k$, and announces this as A . The average fidelity of such a process is

$$F = \sum_{x_1, \dots, x_N=0}^1 g(x_1, \dots, x_N) \langle \psi | \otimes_{k=1}^N (\mathbf{n}_k(x_k) \cdot \boldsymbol{\sigma}_k) | \psi \rangle \quad (78)$$

and in certain problems can even reach *unity*.

For some tasks, the quantum versus classical fidelity ratio grows *exponentially* with N . This is the case, for example, for the so-called *modulo-4 sum* problem. Each partner receives a two-bit input string ($X_k = 0, 1, 2, 3$; $k = 1, \dots, N$). The promise is that X_k s are distributed such that $(\sum_{k=1}^N X_k) \bmod 2 = 0$. The task is: P_N must tell whether the sum modulo-4 of all inputs is 0 or 2. (It can be formulated in terms of a task function $T = 1 - (\sum_{k=1}^N X_k) \bmod 4$. An alternative formulation of the problem reads $f = \cos(\frac{\pi}{2} \sum_{k=1}^N X_k)$ with $p' = 2^{-N+1} |\cos(\frac{\pi}{2} \sum_{k=1}^N X_k)|$.)

For this problem, the classical fidelity bounds decrease exponentially with N , that is $B(F) \leq 2^{-K+1}$, where $K = N/2$ for even and $K = (N+1)/2$ for odd number of parties.

If we use the N qubit GHZ states: $|GHZ\rangle = \frac{1}{\sqrt{2}}(|z+, \dots, z+\rangle + |z-, \dots, z-\rangle)$, where $|z\pm\rangle$ is the state of spin ± 1 along the z -axis, and suitable pairs of local settings, the associated Bell's inequality can be violated maximally. Thus, we have a quantum protocol that always gives the correct answer.

In all quantum protocols considered here, entanglement that leads to a violation of Bell's inequality is a resource that allows for better-than-classical efficiency of the protocol. Surprisingly, we can also show a version of a quantum protocol without entanglement (Galvão 2001; Trojek et al. 2005). The partners exchange a single qubit, P_k to P_{k+1} and so on, and each of them makes a suitable unitary transformation on it (which depends on z_k and x_k). The partner P_N , who receives the qubit as the last one, additionally performs a dichotomic measurement. The result we get is equal to T . For details, including an experimental realization, see Trojek et al. (2005). The obvious conceptual advantage of such a procedure is that the partners exchange a single qubit, from which, due to the Holevo bound (Holevo 1973), we can read out at most one bit of information. In contrast with the protocol involving entanglement, no classical transfer of any information is required, except from the announcement by P_N of his measurement result!

In summary, if we have a pure entangled state of many qubits (this can be generalized to higher-dimensional systems and Bell's inequalities involving more than two measurement settings per observer), there exists a Bell's inequality that is violated by this state. This inequality has some coefficients $g(x_1, \dots, x_n)$, in front of correlation functions, that can always be renormalized in such a way that

$$\sum_{x_1, \dots, x_n=0}^1 |g(x_1, \dots, x_n)| = 1$$

The function g can always be interpreted as a product of the dichotomic function $f(x_1, \dots, x_n) = \frac{g(x_1, \dots, x_n)}{|g(x_1, \dots, x_n)|} = \pm 1$ and a probability distribution $p'(x_1, \dots, x_n) = |g(x_1, \dots, x_n)|$. Thus, we can construct a communication complexity problem that is tailored to a given Bell's inequality, with task function $T = \prod_i^N y_i f$. All this can be extended beyond qubits, see Brukner et al. (2002, 2003).

As was shown, for three or more parties, $N \geq 3$, quantum solutions for certain communication complexity problems can achieve probabilities of success of unity. This is not the case for $N = 2$ and the problem based on the CHSH inequality. The maximum quantum value for the left-hand side of the CHSH inequality (► 25) is just $\sqrt{2} - 1$. This is much bigger than the Bell bound of 0, but still not the largest possible value, for an arbitrary theory that is not following local realism, which equals 1. Because the maximum possible violation of the inequality is not attainable by quantum mechanics, several questions arise. Is this limit forced by the theory of probability, or by physical laws? We will address this question in the next section and we will look at the consequences of the maximal logically possible violation of the CHSH inequality.

6.3 Stronger-Than-Quantum Correlations

The Clauser–Horne–Shimony–Holt (CHSH) inequality (Clauser et al. 1969) for local realistic theories gives the upper bound on a certain combination of correlations between two space-like separated experiments. Consider Alice and Bob who independently perform one out of two

measurements on their part of the system, such that in total there are four experimental setups: $(x, y) = (0, 0), (0, 1), (1, 0)$, or $(1, 1)$. For any local hidden variable theory, the CHSH inequality must hold. We put it in the following form:

$$\begin{aligned} p(a = b|x = 0, y = 0) + p(a = b|x = 0, y = 1) + p(a = b|x = 1, y = 0) \\ + p(a = -b|x = 1, y = 1) \leq 3 \end{aligned} \quad (79)$$

or equivalently,

$$\sum_{x,y=0,1} p(a \oplus b = x \cdot y) \leq 3 \quad (80)$$

In the latter form, we interpret the dichotomic measurement results as binary values, 0 or 1, and their relations are “modulo 2 sums,” denoted here by \oplus . We have $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1$. For example, $p(a = b|x = 0, y = 0)$ is the probability that Alice’s and Bob’s outcomes are the same when she chooses setting $x = 0$ and he setting $y = 0$.

As discussed in previous sections, quantum mechanical correlations can violate the local realistic bound of inequality (• 80) and the limit was proven by Cirel’son (1980) to be $2 + \sqrt{2}$. Popescu and Rohrlich (1994) asked why quantum mechanics allows a violation of the CHSH inequality with a value of $2 + \sqrt{2}$, but not more, though the maximal logically possible value is 4. Would a violation with a value larger than $2 + \sqrt{2}$ lead to (superluminal) signaling? If this were true, then quantum correlations could be understood as maximal allowed correlations respecting non-signaling requirement. This could give us an insight into the origin of quantum correlations, without any use of the Hilbert space formalism.

The non-signaling condition is equivalent to the requirement that the marginals are independent of the partner’s choice of setting

$$p(a|x, y) \equiv \sum_{b=0,1} p(a, b|x, y) = p(a|x) \quad (81)$$

$$p(b|x, y) \equiv \sum_{a=0,1} p(a, b|x, y) = p(b|y) \quad (82)$$

where $p(a, b|x, y)$ is the joint probability for outcomes a and b to occur, given x and y are the choices of measurement settings respectively, $p(a|x)$ is the probability for outcome a , given x is the choice of measurement setting and $p(b|y)$ similarly. Popescu and Rohrlich constructed a toy-theory where the correlations reach the maximal algebraic value of 4 for left-hand expression of the inequality (• 79), but are nevertheless not in contradiction with no-signaling. The probabilities in the toy model are given by

$$\left. \begin{array}{l} p(a = 0, b = 0|x, y) = \frac{1}{2} \\ p(a = 1, b = 1|x, y) = \frac{1}{2} \\ p(a = 1, b = 0|x, y) = \frac{1}{2} \\ p(a = 0, b = 1|x, y) = \frac{1}{2} \end{array} \right\} \begin{array}{l} \text{if } xy \in \{00, 01, 10\} \\ \text{if } xy = 11 \end{array} \quad (83)$$

Indeed we have

$$\sum_{x,y=0,1} p(a \oplus b = x \cdot y) = 4 \quad (84)$$

Van Dam (2000) and, independently, Cleve considered how plausible are stronger-than-quantum correlations from the point of view of communication complexity, which describes how much communication is needed to evaluate a function with distributed inputs. It was shown that the existence of correlations that maximally violate the CHSH inequality would allow us to perform all distributed computations (between two parties) of dichotomic functions with a communication constraint of just one bit. If one is ready to believe that nature should not allow for “easy life” concerning communication problems, this could be a reason why superstrong correlations are indeed not possible.

Instead of superstrong correlations, one usually speaks about a “nonlocal box” (NLB), an imaginary device that takes as inputs x at Alice’s and y at Bob’s side, and outputs a and b at respective sides, such that $a \oplus b = x \cdot y$. Quantum mechanical measurements on a maximally entangled state allow for a success probability of $p = \cos^2 \frac{\pi}{8} = \frac{2+\sqrt{2}}{4} \approx 0.854$ at the game of simulating NLBs. Recently, it was shown that in any “world” in which it is possible to implement an approximation to the NLB, that works correctly with probability greater than $\frac{3+\sqrt{6}}{6} = 90.8\%$, for all distributed computations of dichotomic functions with a one-bit communication constraint, one can find a protocol that always gives the correct values (Brassard et al. 2006). This bound is an improvement over van Dam’s one, but still has a gap with respect to the bound imposed by quantum mechanics.

6.3.1 Superstrong Correlations Trivialize Communication Complexity

We shall present a proof that availability of a perfect NLB would allow for a solution of a general communication complexity problem for a binary function, with an exchange of a single bit of information. The proof is due to van Dam (2000).

Consider a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, which has as inputs two n -bit strings $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$. Suppose that Alice receives the \mathbf{x} string and Bob, who is separated from Alice, the \mathbf{y} -string, and they are to determine the function value $f(\mathbf{x}, \mathbf{y})$ by communicating as little as possible. They have, however, NLBs as resources.

First, let us notice that any dichotomic function $f(\mathbf{x}, \mathbf{y})$ can be rewritten as a finite summation:

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{2^n} P_i(\mathbf{x}) Q_i(\mathbf{y}) \quad (85)$$

where $P(\mathbf{x})$ are polynomials in $\mathbf{x} \in \{0, 1\}$ and $Q_i(\mathbf{y}) = y_1^{i_1} \cdot \dots \cdot y_n^{i_n}$ are monomials in $y_i \in \{0, 1\}$ with $i_1, \dots, i_n \in \{0, 1\}$. Note that the latter ones constitute an orthogonal basis in a 2^n -dimensional space of polynomials with no higher power of any variable than 1. The decomposed function f is treated as a function of y ’s, while the inputs x_1, \dots, x_n are considered as indices numbering functions f . Note that there are 2^n different monomials. Alice can locally compute all the P_i values by herself and likewise Bob can compute all Q_i by himself. These values determine the settings of Alice and Bob that will be chosen in the i th run of the experiment. Note that to this end they need, in general, exponentially many NLBs. Alice and Bob perform for every $i \in \{1, \dots, 2^n\}$ a measurement on the i th NLB in order to obtain without any communication a collection of bit values a_i and b_i , with the property $a_i \oplus b_i = P_i(\mathbf{x}) Q_i(\mathbf{y})$. Bob can add all his b_i to $\sum_{i=1}^{2^n} b_i$ values without requiring any information from Alice, and he can broadcast this single bit to Alice. She, on her part, computes the sum of her a_i to $\sum_{i=1}^{2^n} a_i$ and adds Bob’s bit to it. The final result

$$\sum_{i=1}^{2^n} (a_i \oplus b_i) = \sum_{i=1}^{2^n} P_i(\mathbf{x}) Q_i(\mathbf{y}) = f(\mathbf{x}, \mathbf{y}) \quad (86)$$

is the function value. Thus, *superstrong correlations trivialize every communication complexity problem*.

7 The Kochen–Specker Theorem

In the previous sections, we saw that tests of Bell's inequalities are not only theory-independent tests of nonclassicality, but also have applications in quantum information protocols. Examples are communication complexity problems (Brassard 2003, Brukner et al. 2004b), entanglement detection (Hyllus et al. 2005), security of key distribution (Ekert 1991; Scarani and Gisin 2001; Acín et al. 2006), and quantum state discrimination (Schmid et al. 2008). Thus, entanglement which violates local realism can be seen as a resource for efficient information processing. Can quantum contextuality – the fact that quantum predictions disagree from those of non-contextual hidden-variable theories – also be seen as such a resource? An affirmative answer shall be given to this question by considering an explicit example of a quantum game.

The Kochen–Specker theorem is a “no go” theorem that proves a contradiction between predictions of quantum theory and those of *non-contextual* hidden variable theories. It was proved by Bell (1966), and independently by Kochen and Specker (1967). The non-contextual hidden-variable theories are based on the conjecture of the following three assumptions:

1. *Realism*: It is a model that allows us to use all variables $A_m(n)$ in the theoretical description of the experiment, where $A_m(n)$ gives the value of some observable A_m which *could* be obtained if the knob setting were at positions m . The index n describes the entire experimental “context” in which A_m is measured and is operationally defined through the positions of all other knob settings in the experiment, which are used to measure other observables jointly with A_m . All $A_m(n)$'s are treated as, perhaps, unknown, but still fixed, (real) numbers, or variables for which a proper joint probability distribution can be defined.
2. *Non-contextuality*: The value assigned to an observable $A_m(n)$ of an individual system is independent of the experimental context n in which it is measured, in particular of any properties that are measured *jointly* with that property. This implies that $A_m(n) = A_m$ for all contexts n .
3. “Free will.” The experimenter is free to choose the observable and its context. The choices are independent of the actual hidden values of A_m 's, etc.

Note that “non-contextuality” implies locality (i.e., non-contextually with respect to a remote context), but there is no implication the other way round. One might have theories which are local, but locally contextual.

It should be stressed that the local realistic and non-contextual theories provide us with predictions, which can be tested experimentally, and which can be derived *without making any reference to quantum mechanics* (though many derivations in the literature give exactly the opposite impression). In order to achieve this, it is important to realize that predictions for non-contextual realistic theories can be derived in a completely operational way (Simon et al. 2001). For concreteness, imagine that an observer wants to perform a measurement of an

observable, say the square, $S_{\mathbf{n}}^2$, of a spin component of a spin-1 particle along a certain direction \mathbf{n} . There will be an experimental procedure for trying to do this as accurately as possible. This procedure will be referred to by saying that one sets the “control switch” of one’s apparatus to the position \mathbf{n} . In all the experiments that will be discussed, only a finite number of different switch positions is required. By definition, different switch positions are clearly distinguishable for the observer, and the switch position is all he knows about. Therefore, in an operational sense, the measured physical observable is entirely defined by the switch position. From the above definition, it is clear that the same switch position can be chosen again and again in the course of an experiment. Notice that in such an approach as described above, it does not matter which observable is “really” measured and to what precision. One just derives general predictions, provided that certain switch positions are chosen.

In the original Kochen–Specker proof (Kochen and Specker 1968), the observables that are considered are squares of components of spin 1 along various directions. Such observables have values 1 or 0, as the components themselves have values 1, 0, or -1 . The squares of spin components $\hat{S}_{\mathbf{n}_1}^2$, $\hat{S}_{\mathbf{n}_2}^2$, and $\hat{S}_{\mathbf{n}_3}^2$ along any three orthogonal directions \mathbf{n}_1 , \mathbf{n}_2 , and \mathbf{n}_3 can be measured jointly. Simply, the corresponding quantum operators commute with each other. In the framework of a hidden-variable theory, one assigns to an individual system a set of numerical values, say $+1, 0, +1, \dots$ for the square of spin component along each direction $S_{\mathbf{n}_1}^2, S_{\mathbf{n}_2}^2, S_{\mathbf{n}_3}^2, \dots$ that can be measured on the system. If any of the observables is chosen to be measured on the individual system, the result of the measurement would be the corresponding value. In a non-contextual hidden variable theory, one has to assign to an observable, say $S_{\mathbf{n}_1}^2$, the *same* value independently of whether it is measured in an experimental procedure jointly as a part of some set $\{S_{\mathbf{n}_1}^2, S_{\mathbf{n}_2}^2, S_{\mathbf{n}_3}^2\}$ or of some other set $\{S_{\mathbf{n}_1}^2, S_{\mathbf{n}_4}^2, S_{\mathbf{n}_5}^2\}$ of physical observables, where $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$ and $\{\mathbf{n}_1, \mathbf{n}_4, \mathbf{n}_5\}$ are triads of orthogonal directions. Note that within quantum theory, some of the operators corresponding to the observables from the first set may *not commute* with some corresponding to the observables from the second set.

The squares of spin components along orthogonal directions satisfy

$$\hat{S}_{\mathbf{n}_1}^2 + \hat{S}_{\mathbf{n}_2}^2 + \hat{S}_{\mathbf{n}_3}^2 = s(s+1) = 2 \quad (87)$$

This is *always* so for a particle of spin 1 ($s = 1$). This implies that for every measurement of three squares of mutually orthogonal spin components, two of the results will be equal to one, and one of them will be equal to zero. The Kochen–Specker theorem considers a set of triads of orthogonal directions $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}, \{\mathbf{n}_1, \mathbf{n}_4, \mathbf{n}_5\}, \dots$, for which at least some of the directions have to appear in several of the triads. The statement of the theorem is that there are sets of directions for which it is not possible to give any assignment of 1’s and 0’s to the directions consistent with the constraint (87). The original theorem in Kochen and Specker (1968) used 117 vectors, but this was subsequently reduced to 33 vectors (Peres 1991) and 18 vectors (Cabello et al. 1996). Mathematically, the contradiction with quantum predictions has its origin in the fact that the classical structure of non-contextual hidden variable theories is represented by commutative algebra, whereas quantum mechanical observables need not be commutative, making it impossible to embed the algebra of these observables in a commutative algebra.

The disproof of non-contextually relies on the assumption that the same value is assigned to a given physical observable, $\hat{S}_{\mathbf{n}}^2$, regardless with which two other observables the experimenter chooses to measure it. In quantum theory, the additional observables from one of those sets correspond to operators that do not commute with the operators corresponding to additional observables from the other set. As was stressed in a masterly review on hidden

variable theories by Mermin (1993), Bell wrote (Bell 1966): “These different possibilities require different experimental arrangements; there is no *a priori* reason to believe that the results . . . should be the same. The result of observation may reasonably depend not only on the state of the system (including hidden variables) but also on the complete disposition apparatus.” Nevertheless, as Bell himself showed, the disagreement between predictions of quantum mechanics and of the hidden-variables theories can be strengthened if non-contextuality is replaced by a much more compelling assumption of locality. Note that in Bohr’s doctrine of the inseparability of the object and the measuring instrument, an observable *is* defined through the entire measurement procedure applied to measure it. Within this doctrine one would not speak about measuring the same observable in different contexts, but rather about measuring entirely different maximal observables, and deriving from it the value of a degenerate observable. Note that the Kochen–Specker argument necessarily involves degenerate observables. This is why it does not apply to single qubits.

7.1 A Kochen–Specker Game

A quantum game will now be considered which is based on the Kochen–Specker argument strengthened by the locality condition (see Svozil (2004)). A pair of entangled spin 1 particles is considered which forms a singlet state with total spin 0. A formal description of this state is given by

$$|\Psi\rangle = \frac{1}{\sqrt{3}} (|1\rangle_n | -1\rangle_n + | -1\rangle_n | 1\rangle_n - | 0\rangle_n | 0\rangle_n) \quad (88)$$

where, for example, $|1\rangle_n | -1\rangle_n$ is the state of the two particles with spin projection +1 for the first particle and spin projection −1 for the second particle 1 along the same direction \mathbf{n} . It is important to note that this state is invariant under a change of the direction \mathbf{n} . This implies that if the spin components for the two particles are measured along an arbitrary direction, however the same on both sides, the sum of the two local results is always zero. This is a direct consequence of the conservation of angular momentum.

Let us present the quantum game introduced in Cleve et al. (2004). The requirement in the proof of the Kochen–Specker theorem can be formulated as the following problem in geometry. There exists an explicit set of vectors $\{\mathbf{n}_1, \dots, \mathbf{n}_m\}$ in \mathbb{R}^3 that cannot be colored in red (i.e., assign the value 1 to the spin squared component along that direction) or blue (i.e., assign the value 0) such that both of the following conditions hold:

1. For every orthogonal pair of vectors \mathbf{n}_i and \mathbf{n}_j , they are not both colored blue.
2. For every mutually orthogonal triple of vectors \mathbf{n}_i , \mathbf{n}_j , and \mathbf{n}_k , at least one of them is colored blue.

For example, the set of vectors can consist of 117 vectors from the original Kochen–Specker proof (Kochen and Specker 1968), 33 vectors from Peres’ proof or 18 vectors from Cabello’s proof (Cabello et al. 1996).

The Kochen–Specker game employs the above sets of vectors. Consider two separated parties, Alice and Bob. Alice receives a random triple of orthogonal vectors as her input and Bob receives a single vector randomly chosen from the triple as his input. Alice is asked to give a trit indicating which of her three vectors is assigned color red or 1 (implicitly, the other two vectors are assigned color blue or 0). Bob outputs a bit assigning a color to his vector. The

requirement in the game is that Alice and Bob assign the same color to the vector that they receive in common. It is straightforward to show that the existence of a perfect classical strategy in which Alice and Bob can share classically correlated strings for this game would violate the reasoning used in the Kochen–Specker theorem. On the other hand, there is a perfect quantum strategy using the entangled state (☞ Eq. 88). If Alice and Bob share two particles in this state, Alice can perform a measurement of squared spin components pertaining to directions $\{\mathbf{n}_i, \mathbf{n}_j, \mathbf{n}_k\}$, which are equal to those of the three input vectors, and Bob measures squared spin component in direction \mathbf{n}_l for his input. Then Bob's measurement will necessarily yield the same answer as the measurement by Alice along the same direction.

Concluding this section, we note that quantum contextuality is also closely related to quantum error correction (DiVincenzo and Peres 1997), quantum key distribution (Nagata 2005), one-location quantum games (Aharon and Vaidman 2008), and entanglement detection between internal degrees of freedom.

8 Temporal Bell's Inequalities (Leggett–Garg Inequalities)

In the last section, one more basic information-processing task, the random access code problem, will be considered. It can be solved with a quantum setup with a higher efficiency than is classically possible. It will be shown that the resource for better-than-classical efficiency is a violation of “temporal Bell's inequalities” – the inequalities that are satisfied by temporal correlations of certain classes of hidden-variable theories. Instead of considering correlations between measurement results on distantly located physical systems, here the focus is on one and the same physical system and there is an analysis of the correlations between measurement outcomes at different times. The inequalities were first introduced by Leggett and Garg (1985), Leggett (2002) in the context of testing superpositions of macroscopically distinct quantum states. Since the aim here is different, we shall discuss general assumptions which allow us to derive temporal Bell's inequalities irrespective of whether the object under consideration is macroscopic or not. This is why the assumptions differ from the original ones of Leggett and Garg (1985), and Leggett (2002). Compare also to Paz (1993), Shafee and Golshani (2003), and Brukner et al. (2004a).

Theories which are based on the conjunction of the following four assumptions are now considered. (There is one more difference between the present approach and that of Leggett and Garg (1985) and Leggett (2002). While there the observer measures a single observable having a choice between different times of measurement, here, at any given time, the observer has a choice between two (or more) different measurement settings. One can use both approaches to derive temporal Bell's inequalities.)

1. *Realism:* It is a model that allows us to use all variables $A_m(t)$ $m = 1, 2, \dots$ in the theoretical description of the experiment performed at time t , where $A_m(t)$ gives the value of some observable, which *could* be obtained if it were measured at time t . All $A_m(t)$'s are treated as perhaps unknown, but still fixed numbers, or variables for which a proper joint probability distribution can be defined.
2. *Non-invasiveness:* The value assigned to an observable $A_m(t_1)$ at time t_1 is independent whether or not a measurement was performed at some earlier time t_0 or which observable $A_n(t_0)$ $n = 1, 2, \dots$ at that time was measured. In other words, (actual or potential) measurement values $A_m(t_1)$ at time t_1 are *independent* of the measurement settings chosen at earlier times t_0 .

3. *Induction*: The standard arrow of time is assumed. In particular, the values $A_m(t_0)$ at earlier times t_0 do not depend on the choices of measurement settings at later times t_1 . (Note that this already follows from the “noninvasiveness” when applied symmetrically to both arrows of time.)
4. “*Free will*”: The experimenter is free to choose the observable. The choices are independent of the actual hidden values of A_m 's, etc.

Consider an observer and allow her to choose at time t_0 and at some later time t_1 to measure one of two dichotomic observables $A_1(t_i)$ and $A_2(t_i)$, $i \in \{0, 1\}$. The assumptions given above imply the existence of numbers for $A_1(t_i)$ and $A_2(t_i)$, each taking values either +1 or -1, which describe the (potential or actual) predetermined result of the measurement. For the temporal correlations in an individual experimental run, the following identity holds: $A_1(t_0)[A_1(t_1) - A_2(t_1)] + A_2(t_0)[A_1(t_1) + A_2(t_1)] = \pm 2$. With similar steps as in the derivation of the standard Bell's inequalities, one easily obtains:

$$p(A_1A_1 = 1) + p(A_1A_2 = -1) + p(A_2A_1 = 1) + p(A_2A_2 = -1) \leq 3 \quad (89)$$

where we omit the dependence on time.

An important difference between tests of non-contextuality and temporal Bell's inequalities is that the latter are also applicable to single qubits or *two-dimensional* quantum systems. The temporal correlation function will now be calculated for consecutive measurements of a single qubit. Take an arbitrary mixed state of a qubit, written as $\rho = \frac{1}{2}(\mathbf{1} + \mathbf{r} \cdot \boldsymbol{\sigma})$, where $\mathbf{1}$ is the identity operator, $\boldsymbol{\sigma} \equiv (\sigma_x, \sigma_y, \sigma_z)$ are the Pauli operators for three orthogonal directions x, y , and z , and $\mathbf{r} \equiv (r_x, r_y, r_z)$ is the Bloch vector with the components $r_i = \text{Tr}(\rho\sigma_i)$.

Suppose that the measurement of the observable $\boldsymbol{\sigma} \cdot \mathbf{a}$ is performed at time t_0 , followed by the measurement of $\boldsymbol{\sigma} \cdot \mathbf{b}$ at t_1 , where \mathbf{a} and \mathbf{b} are directions at which the spin is measured. The quantum correlation function is given by $E_{QM}(\mathbf{a}, \mathbf{b}) = \sum_{k,l=\pm 1} k l \text{Tr}(\rho\pi_{a,k}) \text{Tr}(\pi_{a,k}\pi_{b,l})$, where, for example, $\pi_{a,k}$ is the projector onto the subspace corresponding to the eigenvalue $k = \pm 1$ of the spin along \mathbf{a} . Here, we use the fact that after the first measurement the state is projected on the new state $\pi_{a,k}$. Therefore, the probability to obtain the result k in the first measurement and l in the second one is given by $\text{Tr}(\rho\pi_{a,k})\text{Tr}(\pi_{a,k}\pi_{b,l})$. Using $\pi_{a,k} = \frac{1}{2}(\mathbf{1} + k\boldsymbol{\sigma} \cdot \mathbf{a})$ and $\frac{1}{2}\text{Tr}[(\boldsymbol{\sigma} \cdot \mathbf{a})(\boldsymbol{\sigma} \cdot \mathbf{b})] = \mathbf{a} \cdot \mathbf{b}$ one can easily show that the quantum correlation function can simply be written as

$$E_{QM}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \mathbf{b} \quad (90)$$

Note that, in contrast to the usual (spatial) correlation function, the temporal one (Eq. 90) does not depend on the initial state ρ . Note also that a slight modification of the derivation of Eq. 90 can also apply to the cases in which the system evolves between the two measurements following an arbitrary unitary transformation.

The scalar product form of quantum correlations (Eq. 90) allows for the violation of the temporal Bell's inequality and the maximal value of the left-hand side of (Eq. 89) is achieved for the choice of the measurement settings: $\mathbf{a}_1 = \frac{1}{\sqrt{2}}(\mathbf{b}_1 - \mathbf{b}_2)$, $\mathbf{a}_2 = \frac{1}{\sqrt{2}}(\mathbf{b}_1 + \mathbf{b}_2)$ and is equal to $2 + \sqrt{2}$.

8.1 Quantum Random Access Codes

Random access code is a communication task for two parties, say again Alice and Bob. Alice receives some classical n -bit string known only to her (her local input). She is allowed

to send just a one-bit message, m , to Bob. Bob is asked to tell the value of the b th bit of Alice, $b = 1, 2, \dots, n$. However b is known only to him (this is his local input data). The goal is to construct a protocol enabling Bob to tell the value of the b th bit of Alice, with as high average probability of success as possible, for a uniformly random distribution of Alice's bit-strings, and a uniform distribution of b 's. Note that Alice does not know in advance which bit Bob is to recover. Thus, she has no option to send just this required bit.

If they share a quantum channel, then one speaks about a quantum version of the previous problem. Alice is asked to encode her classical n -bit message into 1 qubit (quantum bit) and send it to Bob. He performs some measurement on the received qubit to extract the required bit. In general, the measurement that he uses will depend on which bit he wants to reveal. The idea behind these so-called quantum random access codes already appeared in a paper written circa 1970, and published in 1983, by Wiesner (1983).

We illustrate the concept of random access code with the simplest scheme, in which in a classical framework Alice needs to encode a two-bit string $b_0 b_1$ into a single bit, or into a single qubit in a quantum framework.

In the classical case, Alice and Bob need to decide on a protocol defining which bit-valued message is to be sent by Alice, for each of the four possible values of her two-bit string $b_0 b_1$. There are only $2^4 = 16$ different deterministic protocols, thus the probability of success can be evaluated in a straightforward way. The optimal deterministic classical protocols can then be shown to have a probability of success $P_C = 3/4$. For example, if Alice sends one of the two bits, then Bob will reveal this bit with certainty and have probability of 1/2 to reveal the other one. Since any probabilistic protocol can be represented as a convex combination of the 16 deterministic protocols, the corresponding probability of success for any such probabilistic protocol will be given by the weighted sum of the probabilities of success of the individual deterministic protocols. This implies that the optimal probabilistic protocols can at best be as efficient as the optimal deterministic protocol, which is 3/4.

Ambainis et al. (1999) showed that there is a quantum solution of the random access code with probability of success $P_Q = \cos^2(\pi/8) \approx 0.85$. It is realized as follows: depending on her two-bit string $b_0 b_1$, Alice prepares one of the four states $|\psi_{b_0 b_1}\rangle$. These states are chosen to be on the equator of the Bloch sphere, separated by equal angles of $\pi/2$ radians (see Fig. 3). Using the Bloch sphere parametrization $|\psi(\theta, \phi)\rangle = \cos(\theta/2)|0\rangle + \exp(i\phi)\sin(\theta/2)|1\rangle$, the four encoding states are represented as:

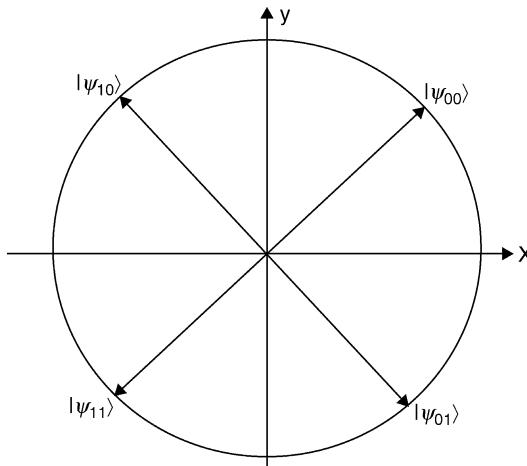
$$\begin{aligned} |\psi_{00}\rangle &= |\psi(\pi/2, \pi/4)\rangle \\ |\psi_{01}\rangle &= |\psi(\pi/2, 7\pi/4)\rangle \\ |\psi_{10}\rangle &= |\psi(\pi/2, 3\pi/4)\rangle \\ |\psi_{11}\rangle &= |\psi(\pi/2, 5\pi/4)\rangle \end{aligned} \tag{91}$$

Bob's measurements, which he uses to guess the bits, will depend on which bit he wants to obtain. To guess b_0 , he projects the qubit along the x -axis in the Bloch sphere, and to decode b_1 he projects it along the y -axis. He then estimates the bit value to be 0 if the measurement outcome was along the positive direction of the axis and 1 if it was along the negative axis. It can easily be calculated that the probability of successful retrieving of the correct bit value is the same in all cases: $P_Q = \cos^2(\pi/8) \approx 0.85$, which is higher than the optimal probability of success $P_C = 0.75$ of the classical random access code using one bit of communication.

We will now introduce a hidden variable model of the quantum solution to see that the key resource in its efficiency lies in violation of temporal Bell's inequalities. Galvao (2002) was

Fig. 3

The set of encoding states and decoding measurements in quantum random access code represented in the $x - y$ plane of the Bloch sphere. Alice prepares one of the four quantum states $|\psi_{b_0 b_1}\rangle$ to encode two bits $b_0, b_1 \in \{0, 1\}$. Depending on which bit Bob wants to reveal he performs either a measurement along the x (to reveal b_0) or along the y axis (to reveal b_1).



the first to point to the relation between violation of Bell's type inequalities and quantum random access codes. See also Spekkens et al. (2009) for a relation with the parity-oblivious multiplexing.

A hidden-variable model equivalent to the quantum protocol, which best fits the temporal Bell's inequalities can be considered as a description of the following modification of the original quantum protocol. Alice prepares the initial state of her qubit as a completely random state, described by a density matrix proportional to the unit operator, σ_0 . Her parity of bit values $b_0 \oplus b_1$ defines a measurement basis, which is used by her to prepare the state to be sent to Bob. Note that the result of the dichotomic measurement in the basis defined by $b_0 \oplus b_1$ is due to the nature of the initial state, completely random and totally uncontrollable by Alice. To fix the bit value b_1 (and thus also the value b_0 , since the parity is defined by the choice of the measurement basis) on her wish, Alice either leaves the state unchanged, if the result of measurement corresponds to her wish of b_1 , or she rotates the state in the $x - y$ plane at 180° to obtain the orthogonal state, if the result corresponds to $b_1 \oplus 1$. Just a glance at the states involved in the standard quantum protocol shows the two complementary (unbiased) bases which define her measurement settings, and which resulting states are linked with which values of $b_0 b_1$. After the measurement, the resulting state is sent to Bob, while Alice is in possession of a bit pair $b_0 b_1$, which is perfectly correlated with the qubit state on the way to Bob. That is, we have exactly the same starting point as in the original quantum protocol.

Now, it is obvious that the quantum protocol violates the temporal inequalities, while any hidden variable model of the above procedure using the four assumptions (1–4) behind the temporal inequalities is not violating them. What is important is the saturation of the temporal inequalities which is equivalent to a probability of success of $3/4$.

The link with temporal Bell's inequalities points to another advantage of quantum over classical random access codes. Usually, one considers the advantage to be *resource dependent*.

With this, we mean that there is an advantage as far as one compares one classical bit with one qubit. Yet the proof given above shows that the quantum strategy has an advantage over *all* hidden variable models respecting (1–4), that is, also those where Alice and Bob use temporal classical correlations between systems of arbitrarily large number of degrees of freedom.

Let us conclude this chapter by pointing out an interesting research avenue. Here, a brief review was given on the results demonstrating that “no go theorems” for various hidden variable classes of theories are behind better-than-classical efficiency in many quantum communication protocols. It would be interesting to investigate the link between fundamental features of quantum mechanics and the power of quantum computation. We showed that temporal Bell's inequalities distinguish between classical and quantum search (Grover) algorithms (Morikoshi 2006). Also cluster states – entangled states on which the logic gates are applied as sequences of only single-qubit measurements – are known to violate Bell's inequalities (Scarani et al. 2005; Tóth et al. 2006). These results point to the aforementioned link but we are still far away from understanding what the key non-classical ingredients are that give rise to the enhanced quantum computational power. The question gets even more fascinating after realizing that not only too little (Van den Nest et al. 2006; Brennen and Miyake 2008; Anders and Browne 2009; Vidal 2007; Markov and Shi 2008; Fannes et al. 1992) but also too much entanglement does not allow powerful quantum computation (Gross et al. 2008; Bremner et al. 2008).

Acknowledgments

Support from the Austrian Science Foundation FWF within Project No. P19570-N16, SFB FoQuS and CoQuS No. W1210-N16, and the European Commission, Projects QAP and QESSENCE, is acknowledged. MZ is supported by MNiSW grant N202 208538. The collaboration is a part of an ÖAD/MNiSW program.

References

- Acín A, Scarani V, Wolf MM (2002) Bell inequalities and distillability in N -quantum-bit systems. *Phys Rev A* 66:042323
- Acín A, Gisin N, Masanes L (2006) From Bell's theorem to secure quantum key distribution. *Phys Rev Lett* 97:120405
- Aharon N, Vaidman L (2008) Quantum advantages in classically defined tasks. *Phys Rev A* 77:052310
- Ambainis A, Nayak A, Ta-Shma A, Vazirani U (1999) Dense quantum coding and a lower bound for 1-way quantum automata. Proceedings of the 31st annual ACM symposium on the theory of computing, New York, pp 376–383
- Anders J, Browne DE (2009) Computational power of correlations. *Phys Rev Lett* 102:050502
- Aspect A, Dalibard J, Roger G (1982) Experimental test of Bell's inequalities using time-varying analyzers. *Phys Rev Lett* 49:1804–1807
- Augusiak R, Horodecki P (2006) Bound entanglement maximally violating Bell inequalities: quantum entanglement is not fully equivalent to cryptographic security. *Phys Rev A* 74:010305
- Barrett J (2007) Information processing in generalized probabilistic theories. *Phys Rev A* 75:032304
- Bechmann-Pasquinucci H (2005) From quantum state targeting to Bell inequalities. *Found Phys* 35:1787–1804
- Bell JS (1964) On the Einstein-Podolsky-Rosen paradox. *Physics* 1:195–200; reprinted in Bell JS (1987) Speakable and unspeakable in quantum mechanics. Cambridge University Press, Cambridge
- Bell JS (1966) On the problem of hidden variables in quantum mechanics. *Rev Mod Phys* 38:447–452
- Bell JS (1985) Free variables and local causality. *Dialectica* 39:103–106
- Bennett CH, Wiesner SJ (1992) Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Phys Rev Lett* 69:2881–2884
- Bohm D (1952) A suggested interpretation of the quantum theory in terms of “hidden” variables, I and II. *Phys Rev* 85:166–193

- Bohr N (1935) Can quantum-mechanical description of physical reality be considered complete? *Phys Rev* 48:696–702
- Branciard C, Ling A, Gisin N et al. (2007) Experimental falsification of Leggett's nonlocal variable model. *Phys Rev Lett* 99:210407
- Brassard G (2003) Quantum communication complexity (a survey). *Found Phys* 33:1593–1616. Available via <http://arxiv.org/abs/quant-ph/0101005>
- Brassard G, Buhrman H, Linden N et al. (2006) Limit on nonlocality in any world in which communication complexity is not trivial. *Phys Rev Lett* 96:250401
- Bremner M J, Mora C, Winter A (2009) Are random pure states useful for quantum computation? *Phys Rev Lett* 102:190502
- Brennen GK, Miyake A (2008) Measurement-based quantum computer in the gapped ground state of a two-body Hamiltonian. *Phys Rev Lett* 101:010502
- Brukner Č, Paterek T, Žukowski M (2003) Quantum communication complexity protocols based on higher-dimensional entangled systems. *Int J Quantum Inf* 1:519–525
- Brukner Č, Žukowski M, Zeilinger A (2002) Quantum communication complexity protocol with two entangled qutrits. *Phys Rev Lett* 89:197901
- Brukner Č, Taylor S, Cheung S, Vedral V (2004a) Quantum entanglement in time. Available via arXiv: quant-ph/0402127
- Brukner Č, Žukowski M, Pan J-W, Zeilinger A (2004b) Bell's inequality and quantum communication complexity. *Phys Rev Lett* 92:127901
- Buhrman H, Cleve R, van Dam W (1997) Quantum entanglement and communication complexity. Available via <http://arxiv.org/abs/quant-ph/9705033>
- Buhrman H, van Dam W, Hoyer P, Tapp A (1999) Multiparty quantum communication complexity. *Phys Rev A* 60:2737–2741
- Cabello A, Estebaranz JM, García-Alcaine G (1996) Bell-Kochen-Specker theorem: a proof with 18 vectors. *Phys Lett A* 212:183–187
- Cirel'son BS (1980) Quantum generalizations of Bell's inequality. *Lett Math Phys* 4:93–100
- Clauser JF, Horne MA, Shimony A, Holt RA (1969) Proposed experiment to test local hidden-variable theories. *Phys Rev Lett* 23:880–884
- Cleve R, Buhrman H (1997) Substituting quantum entanglement for communication. *Phys Rev A* 56:1201–1204
- Cleve R, Hoyer P, Toner B, Watrous J (2004) Consequences and limits of nonlocal strategies. Proceedings of the 19th IEEE annual conference on computational complexity, Amherst, MA, pp 236–249
- Dakic B, Brukner C (2009) Quantum theory and beyond: is entanglement special? arXiv:0911.0695
- DiVincenzo DP, Peres A (1997) Quantum code words contradict local realism. *Phys Rev A* 55: 4089–4092
- Einstein A, Podolsky B, Rosen N (1935) Can quantum-mechanical description of physical reality be considered complete? *Phys Rev* 47:777–780
- Ekert A (1991) Quantum cryptography based on Bell's theorem. *Phys Rev Lett* 67:661–663
- Fannes M, Nachtergaelie B, Werner RF (1992) Finitely correlated states on quantum spin chains. *Commun Math Phys* 144:443–490
- Gálvão EF (2001) Feasible quantum communication complexity protocol. *Phys Rev A* 65:012318
- Gálvão EF (2002) Foundations of quantum theory and quantum information applications. Ph.D. thesis, University of Oxford, Oxford. Available via arXiv: quant-ph/0212124
- Gill RD, Weihs G, Zeilinger A, Žukowski M (2002) No time loophole in Bell's theorem: the Hess-Philipp model is nonlocal. *Proc Nat Acad Sci USA* 99:14632–14635
- Gill RD, Weihs G, Zeilinger A, Žukowski M (2003) Comment on “Exclusion of time in the theorem of Bell” by Hess K and Philipp W. *Europhys Lett* 61:282–283
- Greenberger DM, Horne MA, Zeilinger A (1989) Going beyond Bell's theorem. In: Kafatos M (ed) Bell's theorem, quantum theory, and conceptions of the universe. Kluwer, Dordrecht, pp 73–76
- Greenberger DM, Horne MA, Shimony A, Zeilinger A (1990) Bell's theorem without inequalities. *Am J Phys* 58:1131–1143
- Gröblacher S, Paterek T, Kaltenbaek R et al. (2007) An experimental test of non-local realism. *Nature* 446:871–875
- Gross D, Flammia S, Eisert J (2009) Most quantum states are too entangled to be useful as computational resources. *Phys Rev Lett* 102:190501
- Holevo AS (1973) Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii* 9:3–11. English translation in *Probl Inf Transm* 9:177–183
- Horodecki R, Horodecki P, Horodecki M (1995) Violating Bell inequality by mixed spin- $\frac{1}{2}$ states: necessary and sufficient condition. *Phys Lett A* 200:340–344
- Hyllus P, Gühne O, Bruß D, Lewenstein M (2005) Relations between entanglement witnesses and Bell inequalities. *Phys Rev A* 72:012321
- Kochen S, Specker E (1968) The problem of hidden variables in quantum mechanics. *J Math Mech* 17:59–87
- Kofler J, Paterek T, Brukner C (2006) Experimenter's freedom in Bell's theorem and quantum cryptography, *Phys Rev A* 73:022104
- Kushilevitz E, Nisan N (1997) Communication complexity. Cambridge University Press, Cambridge
- Leggett AJ (2002) Testing the limits of quantum mechanics: motivation, state of play, prospects. *J Phys Condensed Matter* 14:15 R415

- Leggett AJ (2003) Nonlocal hidden-variable theories and quantum mechanics: an incompatibility theorem. *Found Phys* 33:1469–1493
- Leggett AJ, Garg A (1985) Quantum mechanics versus macroscopic realism: is the flux there when nobody looks? *Phys Rev Lett* 54:857–860
- Marcovitch S, Reznik B (2008) Implications of communication complexity in multipartite systems. *Phys Rev A* 77:032120
- Markov IL, Shi Y (2008) Simulating quantum computation by contracting tensor networks. *SIAM J Comput* 38(3):963–981
- Mermin ND (1993) Hidden variables and the two theorems of John Bell. *Rev Mod Phys* 65:803–815
- Morikoshi F (2006) Information-theoretic temporal Bell inequality and quantum computation. *Phys Rev A* 73:052308
- Nagata K (2005) Kochen-Specker theorem as a precondition for secure quantum key distribution. *Phys Rev A* 72:012325
- Pawlowski M, Paterek T, Kaszlikowski D, Scarani V, Winter A and Zukowski M (2009) Information causality as a physical principle. *Nature (London)* 461(7267):1101–1104
- Paz JP, Mahler G (1993) Proposed test for temporal Bell inequalities. *Phys Rev Lett* 71:3235–3239
- Peres A (1991) Two simple proofs of the Kochen-Specker theorem. *J Phys A* 24:L175–L178
- Peres A (1994) Quantum theory: concepts and methods. Kluwer, Boston
- Popescu S, Rohrlich D (1994) Quantum nonlocality as an axiom. *Found Phys* 24:379–385
- Rowe MA, Kielpinski D, Meyer V, Sackett CA, Itano WM, Monroe C, Wineland DJ (2001) Experimental violation of a Bell's inequality with efficient detection. *Nature* 409:791–794
- Scarani V, Gisin N (2001) Quantum communication between N partners and Bell's inequalities. *Phys Rev Lett* 87:117901
- Scarani V, Acín A, Schenck E, Aspelmeyer M (2005) Nonlocality of cluster states of qubits. *Phys Rev A* 71:042325
- Schmid C, Kiesel N, Laskowski W et al. (2008) Discriminating multipartite entangled states. *Phys Rev Lett* 100:200407
- Scheidl T, Ursin R, Kofler J et al. (2008) Violation of local realism with freedom of choice. Available via <http://arxiv.org/abs/0811.3129>
- Schrödinger E (1935) Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften* 23:807–812; 823–828; 844–849. Translation published in *Proc Am Philos Soc* 124:323–338 and in Wheeler JA, Zurek WH (eds) (1983) *Quantum theory and measurement*. Princeton University Press, Princeton, NJ, pp 152–167
- Shafiee A, Golshani M (2003) Single-particle Bell-type inequality. *Annales de la Fondation de Broglie* 28:105–118
- Simon C, Brukner Č, Zeilinger A (2001) Hidden-variable theorems for real experiments. *Phys Rev Lett* 86:4427–4430
- Spekkens RW, Buzacott DH, Keehn AJ et al. (2009) Preparation contextuality powers parity-oblivious multiplexing. *Phys Rev Lett* 102:010401
- Svozil K (2004) Quantum mechanics is noncontextual. Available via arXiv:quant-ph/0401112
- Tamir B (2007) Communication complexity protocols for qutrits. *Phys Rev A* 75:032344
- Tóth G, Gühne O, Briegel HJ (2006) Two-setting Bell inequalities for graph states. *Phys Rev A* 73:022303
- Trojek P, Schmid C, Bourennane M et al. (2005) Experimental quantum communication complexity. *Phys Rev A* 72:050305
- van Dam W (2000) Implausible consequences of super-strong nonlocality. Chapter 9 in van Dam W, *Nonlocality & communication complexity*. Ph.D. Thesis, University of Oxford, Department of Physics. Available via arXiv:quant-ph/0501159
- Van den Nest M, Miyake A, Dür W, Briegel HJ (2006) Universal resources for measurement-based quantum computation. *Phys Rev Lett* 97:150504
- Vidal G (2007) Classical simulation of infinite-size quantum lattice systems in one spatial dimension. *Phys Rev Lett* 98:070201
- Weihl G, Jennewein T, Simon C, Weinfurter H, Zeilinger A (1998) Violation of Bell's inequality under strict Einstein locality conditions. *Phys Rev Lett* 81:5039–5043
- Weinfurter H, Žukowski M (2001) Four-photon entanglement from down-conversion. *Phys Rev A* 64:010102
- Werner RF, Wolf MM (2001) All multipartite Bell correlation inequalities for two dichotomic observables per site. *Phys Rev A* 64:032112
- Wiesner S (1983) Conjugate coding. *ACM Sigact News* 15:78–88
- von Weizsäcker CF (1985) *Aufbau der Physik*. Hanser, Munich
- Yao AC (1979) Some complexity questions related to distributed computing. Proceedings of the 11th Annual ACM Symposium on Theory of Computing, pp 209–213
- Zeilinger A (1999) A foundational principle for quantum mechanics. *Found Phys* 29:631–643
- Žukowski M, Brukner Č (2002) Bell's theorem for general N-qubit states. *Phys Rev Lett* 88:210401

43 Algorithms for Quantum Computers

Jamie Smith¹ · Michele Mosca²

¹Institute for Quantum Computing and Department of Combinatorics & Optimization, University of Waterloo, Canada

ja5smith@iqc.ca

²Institute for Quantum Computing and Department of Combinatorics & Optimization, University of Waterloo and St. Jerome's University and Perimeter Institute for Theoretical Physics, Waterloo, Canada

mmosca@iqc.ca

1	<i>Introduction</i>	1452
2	<i>Algorithms Based on the Quantum Fourier Transform</i>	1453
3	<i>Amplitude Amplification and Estimation</i>	1457
4	<i>Quantum Walks</i>	1461
5	<i>Tensor Networks and Their Applications</i>	1477
6	<i>Conclusion</i>	1489

Abstract

This chapter surveys the field of quantum computer algorithms. It gives a taste of both the breadth and the depth of the known algorithms for quantum computers, focusing on some of the more recent results. It begins with a brief review of quantum Fourier transform-based algorithms, followed by quantum searching and some of its early generalizations. It continues with a more in-depth description of two more recent developments: algorithms developed in the quantum walk paradigm, followed by tensor network evaluation algorithms (which include approximating the Tutte polynomial).

1 Introduction

Quantum computing is a new computational paradigm created by reformulating information and computation in a quantum mechanical framework (Feynman 1982; Deutsch 1985). Since the laws of physics appear to be quantum mechanical, this is the most relevant framework to consider when considering the fundamental limitations of information processing. Furthermore, in recent decades a major shift from just observing quantum phenomena to actually controlling quantum mechanical systems has been seen. The communication of quantum information over long distances, the “teleportation” of quantum information, and the encoding and manipulation of quantum information in many different physical media were also observed. We still appear to be a long way from the implementation of a large-scale quantum computer; however, it is a serious goal of many of the world’s leading physicists, and progress continues at a fast pace.

In parallel with the broad and aggressive program to control quantum mechanical systems with increased precision, and to control and interact a larger number of subsystems, researchers have also been aggressively pushing the boundaries of what useful tasks one could perform with quantum mechanical devices. These include improved metrology, quantum communication and cryptography, and the implementation of large-scale quantum algorithms.

It was known very early on (Deutsch 1985) that quantum algorithms cannot compute functions that are not computable by classical computers, however they might be able to efficiently compute functions that are not efficiently computable on a classical computer. Or, at the very least, quantum algorithms might be able to provide some sort of speed-up over the best possible or best-known classical algorithms for a specific problem.

The purpose of this chapter is to survey the field of quantum algorithms, which has grown tremendously since Shor’s breakthrough algorithms (Shor 1994, 1997) over 15 years ago. Much of the work in quantum algorithms is now textbook material (e.g., Nielsen and Chuang (2000); Hirvensalo (2001); Kitaev et al. (2002); Kaye et al. (2007); Mermin (2007)), and these examples will only be briefly mentioned in order to provide a broad overview. Other parts of this chapter, in particular, [Sects. 4](#) and [5](#), give a more detailed description of some more recent work.

We organized this chapter according to underlying tools or approaches taken, and we include some basic applications and specific examples, and relevant comparisons with classical algorithms.

In [Sect. 2](#) we begin with algorithms one can naturally consider to be based on a quantum Fourier transform, which includes the famous factoring and discrete logarithm algorithms of Peter Shor (1994, 1997). Since this topic is covered in several textbooks and recent surveys, this topic will only be briefly surveyed. Several other sections could have been added on algorithms

for generalizations of these problems, including several cases of the non-Abelian hidden subgroup problem, and hidden lattice problems over the reals (which have important applications in number theory); however, these are covered in a recent survey (Mosca 2009) and in substantial detail in Childs and van Dam (2010).

☞ [Section 3](#) continues with a brief review of classic results on quantum searching and counting, and more generally amplitude amplification and amplitude estimation.

In ☞ [Sect. 4](#), algorithms based on quantum walks are discussed. The related topic of adiabatic algorithms, which was briefly summarized in Mosca (2009) will not be covered here; a broader survey of this and related techniques (“quantum annealing”) can be found in Das and Chakrabarti (2008).

☞ [Section 5](#) concludes with algorithms based on the evaluation of the trace of an operator, also referred to as the evaluation of a tensor network, and which has applications such as the approximation of the Tutte polynomial.

The field of quantum algorithms has grown tremendously since the seminal work in the mid-1990s, and a full detailed survey would simply be infeasible for one chapter. Several other sections could have been added. One major omission is the development of algorithms for simulating quantum mechanical systems, which was Feynman’s original motivation for proposing a quantum computer. This field was briefly surveyed in Mosca (2009), with emphasis on the recent results in Berry et al. (2007) (more recent developments can be found in Wiebe et al. (2008)). This remains an area worthy of a comprehensive survey; like many other areas that researchers have tried to survey, it is difficult because it is still an active area of research. It is also an especially important area because these algorithms tend to offer a fully exponential speed-up over classical algorithms, and thus are likely to be among the first quantum algorithms to be implemented that will offer a speed-up over the fastest-available classical computers.

Finally, one could also write a survey of quantum algorithms for intrinsically quantum information problems, such as entanglement concentration, or quantum data compression. This topic is not covered in this chapter, though there is a very brief survey in Mosca (2009).

One can find a rather comprehensive list of the known quantum algorithms (up to mid-2008) in Stephen Jordan’s PhD thesis (Jordan 2008). It is hoped that this chapter complements some of the other recent surveys in providing a reasonably detailed overview of the state of the art in quantum algorithms.

2 Algorithms Based on the Quantum Fourier Transform

The early line of quantum algorithms was developed in the “black-box” or “oracle” framework. In this framework, part of the input is a black box that implements a function $f(x)$, and the only way to extract information about f is to evaluate it on inputs x . These early algorithms used a special case of quantum Fourier transform, the Hadamard gate, in order to solve the given problem with fewer black-box evaluations of f than a classical algorithm would require.

Deutsch (1985) formulated the problem of deciding whether a function $f: \{0, 1\} \rightarrow \{0, 1\}$ was constant or not. Suppose one has access to a black box that implements f reversibly by mapping $x, 0 \mapsto x, f(x)$; one can further assume that the black box in fact implements a unitary transformation U_f that maps $|x\rangle |0\rangle \mapsto |x\rangle |f(x)\rangle$. Deutsch’s problem is to output “constant” if $f(0) = f(1)$ and to output “balanced” if $f(0) \neq f(1)$, given a black box for evaluating f . In other words determine $f(0) \oplus f(1)$ (where \oplus denotes addition modulo 2). Outcome “0” means f is constant and “1” means f is not constant.

A classical algorithm would need to evaluate f twice in order to solve this problem. A quantum algorithm can apply U_f only once to create

$$\frac{1}{\sqrt{2}}|0\rangle|f(0)\rangle + \frac{1}{\sqrt{2}}|1\rangle|f(1)\rangle$$

Note that if $f(0) = f(1)$, then applying the Hadamard gate to the first register yields $|0\rangle$ with probability 1, and if $f(0) \neq f(1)$, then applying the Hadamard gate to the first register and ignoring the second register leaves the first register in the state $|1\rangle$ with probability $\frac{1}{2}$; thus a result of $|1\rangle$ could only occur if $f(0) \neq f(1)$.

As an aside, one can note that in general, given

$$\frac{1}{\sqrt{2}}|0\rangle|\psi_0\rangle + \frac{1}{\sqrt{2}}|1\rangle|\psi_1\rangle$$

applying the Hadamard gate to the first qubit and measuring it will yield “0” with probability $\frac{1}{2} + \text{Re}(\langle\psi_0|\psi_1\rangle)$; this “Hadamard test” is discussed in more detail in [Sect. 5](#).

In Deutsch's case, measuring a “1” meant $f(0) \neq f(1)$ with certainty, and a “0” was an inconclusive result. Even though it was not perfect, it still was something that could not be done with a classical algorithm. The algorithm can be made exact (Cleve et al. 1998) (i.e., one that outputs the correct answer with probability 1) if one assumes further that U_f maps $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$, for $b \in \{0, 1\}$, and one sets the second qubit to $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Then U_f maps

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \mapsto (-1)^{f(0)} \left(\frac{|0\rangle + (-1)^{f(0) \oplus f(1)}|1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Thus, a Hadamard gate on the first qubit yields the result

$$(-1)^{f(0)}|f(0) \oplus f(1)\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

and measuring the first register yields the correct answer with certainty.

The general idea behind the early quantum algorithms was to compute a black-box function f on a superposition of inputs, and then extract a global property of f by applying a quantum transformation to the input register before measuring it. It is usually assumed that we have access to a black box that implements

$$U_f : |\mathbf{x}\rangle|\mathbf{b}\rangle \mapsto |\mathbf{x}\rangle|\mathbf{b} \oplus f(\mathbf{x})\rangle$$

or in some other form where the input value \mathbf{x} is kept intact and the second register is shifted by $f(\mathbf{x})$ in some reversible way.

Deutsch and Jozsa (1992) used this approach to get an exact algorithm that decides whether $f : \{0, 1\}^n \mapsto \{0, 1\}$ is constant or “balanced” (i.e., $|f^{-1}(0)| = |f^{-1}(1)|$), with a promise that one of these two cases holds. Their algorithm evaluated f only twice, while classically any exact algorithm would require $2^{n-1} + 1$ queries in the worst case. Bernstein and Vazirani (1997) defined a specific class of such functions $f_{\mathbf{a}}$: $\mathbf{x} \mapsto \mathbf{a} \cdot \mathbf{x}$, for any $\mathbf{a} \in \{0, 1\}^n$, and showed how the same algorithm that solves the Deutsch–Jozsa problem allows one to determine \mathbf{a} with two evaluations of $f_{\mathbf{a}}$ while a classical algorithm requires n evaluations. (Both of these algorithms can be done with one query if we have $|\mathbf{x}\rangle|b\rangle \mapsto |\mathbf{x}\rangle|b \oplus f(\mathbf{x})\rangle$.) They further showed how a related “recursive Fourier sampling” problem could be solved super-polynomially faster on a quantum computer. Simon (1994) later built on these tools to

develop a black-box quantum algorithm that was exponentially faster than any classical algorithm for finding a hidden string $\mathbf{s} \in \{0, 1\}^n$ that is encoded in a function $f: \{0, 1\}^n \rightarrow \{0, 1\}^n$ with the property that $f(\mathbf{x}) = f(\mathbf{y})$ if and only if $\mathbf{x} = \mathbf{y} \oplus \mathbf{s}$.

Shor (1994, 1997) built on these black-box results to find an efficient algorithm for finding the order of an element in the multiplicative group of integers modulo N (which implies an efficient classical algorithm for factoring N) and for solving the discrete logarithm problem in the multiplicative group of integers modulo a large prime p . Since the most widely used public key cryptography schemes at the time relied on the difficulty of integer factorization, and others relied on the difficulty of the discrete logarithm problem, these results had very serious practical implications. Shor's algorithms straightforwardly apply to black-box groups, and thus permit finding orders and discrete logarithms in any group that is reasonably presented, including the additive group of points on elliptic curves, which is currently one of the most widely used public key cryptography schemes (see e.g., Menezes et al. (1996)).

Researchers tried to understand the full implications and applications of Shor's technique, and a number of generalizations were soon formulated (e.g., Boneh and Lipton 1995; Grigoriev 1997). One can phrase Simon's algorithm, Shor's algorithm, and the various generalizations that soon followed as special cases of the *hidden subgroup problem*. Consider a finitely generated Abelian group G , and a hidden subgroup K that is defined by a function $f: G \rightarrow X$ (for some finite set X) with the property that $f(x) = f(y)$ if and only if $x - y \in K$ (we use additive notation, without loss of generality). In other words, f is constant on cosets of K and distinct on different cosets of G . In the case of Simon's algorithm, $G = \mathbb{Z}_2^n$ and $K = \{\mathbf{0}, \mathbf{s}\}$. In the case of Shor's order-finding algorithm, $G = \mathbb{Z}$ and $K = r\mathbb{Z}$ where r is the unknown order of the element. Other examples and how they fit in the hidden subgroup paradigm are given in Mosca (2008).

Soon after, Kitaev (Kitaev 1995) solved a problem he called the *Abelian stabilizer problem* using an approach that seemed different from Shor's algorithm, one based in eigenvalue estimation. Eigenvalue estimation is in fact an algorithm of independent interest for the purpose of studying quantum mechanical systems. The Abelian stabilizer problem is also a special case of the hidden subgroup problem. Kitaev's idea was to turn the problem into one of estimating eigenvalues of unitary operators. In the language of the hidden subgroup problem, the unitary operators were shift operators of the form $f(x) \mapsto f(x + y)$. By encoding the eigenvalues as relative phase shifts, he turned the problem into a phase estimation problem.

The Simon/Shor approach for solving the hidden subgroup problem is to first compute $\sum_x |x\rangle |f(x)\rangle$. In the case of finite groups G , one can sum over all the elements of G , otherwise one can sum over a sufficiently large subset of G . For example, if $G = \mathbb{Z}$, and $f(x) = a^x \bmod N$ for some large integer N , we first compute $\sum_{x=0}^{2^n-1} |x\rangle |a^x\rangle$, where $2^n > N^2$ (we omit the “mod N ” for simplicity). If r is the order of a (i.e., r is the smallest positive integer such that $a^r \equiv 1$) then every value x of the form $x = y + kr$ gets mapped to a^y . Thus, we can rewrite the above state as

$$\sum_{x=0}^{2^n-1} |x\rangle |a^x\rangle = \sum_{y=0}^{r-1} \left(\sum_j |y + jr\rangle \right) |a^y\rangle \quad (1)$$

where each value of a^y in this range is distinct. Tracing out the second register one is thus left with a state of the form

$$\sum_j |y + jr\rangle$$

for a random y and where j goes from 0 to $\lfloor(2^n - 1)/r\rfloor$. We loosely refer to this state as a “periodic” state with period r . We can use the inverse of the quantum Fourier transform (or the quantum Fourier transform) to map this state to a state of the form $\sum_x \alpha_x |x\rangle$, where the amplitudes are biased toward values of x such that $x/2^n \approx k/r$. With probability at least $4/\pi^2$ we obtain an x such that $|x/2^n - k/r| \leq 1/2^{n+1}$. One can then use the continued fractions algorithm to find k/r (in lowest terms) and thus find r with high probability. It is important to note that the continued fractions algorithm is not needed for many of the other cases of the Abelian hidden subgroup considered, such as Simon’s algorithm or the discrete logarithm algorithm when the order of the group is already known.

In contrast, Kitaev’s approach for this special case was to consider the map $U_a: |b\rangle \mapsto |ba^x\rangle$. It has eigenvalues of the form $e^{2\pi i k/r}$, and the state $|1\rangle$ satisfies

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle$$

where $|\psi_k\rangle$ is an eigenvector with eigenvalue $e^{2\pi i k/r}$:

$$U_a: |\psi_k\rangle \mapsto e^{2\pi i k x/r} |\psi_k\rangle$$

If one considers the controlled- U_a , denoted $c - U_a$, which maps $|0\rangle |b\rangle \mapsto |0\rangle |b\rangle$ and $|1\rangle |b\rangle \mapsto |1\rangle |ba\rangle$, and if one applies it to the state $(|0\rangle + |1\rangle)|\psi_k\rangle$ one gets

$$(|0\rangle + |1\rangle)|\psi_k\rangle \mapsto (|0\rangle + e^{2\pi i k/r}|1\rangle)|\psi_k\rangle$$

In other words, the eigenvalue becomes a relative phase, and thus one can reduce eigenvalue estimation to phase estimation. Furthermore, since one can efficiently compute a^j by performing j multiplications modulo N , one can also efficiently implement $c - U_{a^j}$ and thus easily obtain the qubit $|0\rangle + e^{2\pi i 2^j(k/r)}|1\rangle$ for integer values of j without performing $c - U_a$ a total of 2^j times. Kitaev developed an efficient ad hoc phase estimation scheme in order to estimate k/r to high precision, and this phase estimation scheme could be optimized further (Cleve et al. 1998). In particular, one can create the state

$$(|0\rangle + |1\rangle)^n |1\rangle = \sum_{x=0}^{2^n-1} |x\rangle |1\rangle = \sum_{k=0}^{r-1} \sum_{x=0}^{2^n-1} |x\rangle |\psi_k\rangle$$

(we use the standard binary encoding of the integers $x \in \{0, 1, \dots, 2^n - 1\}$ as bit strings of length n) to apply the $c - U_{a^j}$ using the $(n-j)$ th bit as the control bit, and using the second register (initialized in $|1\rangle = \sum_k |\psi_k\rangle$) as the target register, for $j = 0, 1, 2, \dots, n-1$, to create

$$\sum_k (|0\rangle + e^{2\pi i 2^{n-1} k} |1\rangle) \cdots (|0\rangle + e^{2\pi i 2^1 k} |1\rangle) (|0\rangle + e^{2\pi i k} |1\rangle) |\psi_k\rangle \quad (2)$$

$$= \sum_{x=0}^{2^n-1} |x\rangle |a^x\rangle = \sum_{k=0}^{r-1} \sum_{x=0}^{2^n-1} e^{2\pi i x k} |x\rangle |\psi_k\rangle \quad (3)$$

If the second register is ignored or discarded, one is left with a state of the form $\sum_{x=0}^{2^n-1} e^{2\pi i x k/r} |x\rangle$ for a random value of $k \in \{0, 1, \dots, r-1\}$. The inverse quantum Fourier transformation maps this state to a state $\sum_y \alpha_y |y\rangle$ where most of the weight of the amplitudes is near values of y such that $y/2^n \approx k/r$ for some integer k . More specifically $|y/2^n - k/r| \leq 1/2^{n+1}$ with probability at least $4/\pi^2$; furthermore $|y/2^n - k/r| \leq 1/2^n$ with probability at least $8/\pi^2$. As in the case of Shor’s algorithm, one can use the continued fractions algorithm to determine k/r (in lowest terms) and thus determine r with high probability.

It was noted (Cleve et al. 1998) that this modified eigenvalue estimation algorithm for order-finding was essentially equivalent to Shor’s period-finding algorithm for order-finding. This can be seen by noting that we have the same state in [Eqs. 1](#) and [2](#), and in both cases, we discard the second register and apply an inverse Fourier transform to the first register. The only difference is the basis in which the second register is mathematically analyzed.

The most obvious direction in which to try to generalize the Abelian hidden subgroup algorithm is to solve instances of the hidden subgroup problem for non-Abelian groups. This includes, for example, the graph automorphism problem (which corresponds to finding a hidden subgroup of the symmetric group). There has been nontrivial, but limited, progress in this direction, using a variety of algorithmic tools, such as sieving, “pretty good measurements,” and other group theoretic approaches. Other generalizations include the hidden shift problem and its generalizations, hidden lattice problems on real lattices (which has important applications in computational number theory and computationally secure cryptography), and hidden nonlinear structures. These topics and techniques would take several dozen pages just to summarize, so the reader can refer to Mosca (2009) or Childs and van Dam (2010) so that more room is left to summarize other important topics.

3 Amplitude Amplification and Estimation

A very general problem for which quantum algorithms offer an improvement is that of searching for a solution to a computational problem in the case that a solution x can be easily verified. One can phrase such a general problem as finding a solution x to the equation $f(x) = 1$, given a means for evaluating the function f . One can further assume that $f: \{0, 1\}^n \mapsto \{0, 1\}$.

The problems from [Sect. 2](#) can be rephrased in this form (or a small number of instances of problems of this form), and the quantum algorithms for these problems exploit some nontrivial algebraic structure in the function f in order to solve the problems superpolynomially faster than the best possible or best-known classical algorithms. Quantum computers also allow a more modest speed-up (up to quadratic) for searching for a solution to a function f without any particular structure. This includes, for example, searching for solutions to NP -complete problems.

Note that classically, given a means for guessing a solution x with probability p , one could amplify this success probability by repeating many times, and after a number of guesses in $O(1/p)$, the probability of finding a solution is in $\Omega(1)$. Note that the quantum implementation of an algorithm that produces a solution with probability p will produce a solution with probability amplitude \sqrt{p} . The idea behind quantum searching is to somehow amplify this probability to be close to 1 using only $O(1/\sqrt{p})$ guesses and other steps.

Lov Grover (1996) found precisely such an algorithm, and this algorithm was analyzed in detail and generalized (Boyer et al. 1998; Brassard and Høyer 1997; Grover 1998; Brassard et al. 2002) to what is known as “amplitude amplification.” Any procedure for guessing a solution with probability p can be (with modest overhead) turned into a unitary operator A that maps $|0\rangle \mapsto \sqrt{p}|\psi_1\rangle + \sqrt{1-p}|\psi_0\rangle$, where $0 = 00\dots 0$, $|\psi_1\rangle$ is a superposition of states encoding solutions x to $f(x) = 1$ (the states could in general encode x followed by other “junk” information) and $|\psi_0\rangle$ is a superposition of states encoding values of x that are not solutions.

One can then define the quantum search iterate (or “Grover iterate”) to be

$$Q = -AU_0A^{-1}U_f$$

where $U_f: |\mathbf{x}\rangle \mapsto (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle$, and $U_0 = I - 2|\mathbf{0}\rangle\langle\mathbf{0}|$ (in other words, maps $|\mathbf{0}\rangle \mapsto -|\mathbf{0}\rangle$ and $|\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$ for any $\mathbf{x} \neq \mathbf{0}$). Here, for simplicity, it is assumed that there are no “junk” bits in the unitary computation of f by A . Any such junk information can either be “uncomputed” and reset to all 0s, or even ignored (letting U_f act only on the bits encoding \mathbf{x} and applying the identity to the junk bits).

This algorithm is analyzed thoroughly in the literature and in textbooks, so the main results and ideas are only summarized here to help the reader understand the later sections on searching via quantum walk.

If one applies Q a total of k times to the input state $A|00\dots0\rangle = \sin(\theta)|\psi_1\rangle + \cos(\theta)|\psi_0\rangle$, where $\sqrt{p} = \sin(\theta)$, then one obtains

$$Q^k A|00\dots0\rangle = \sin((2k+1)\theta)|\psi_1\rangle + \cos((2k+1)\theta)|\psi_0\rangle$$

This implies that with $k \approx \frac{\pi}{4\sqrt{p}}$ one obtains $|\psi_1\rangle$ with probability amplitude close to 1, and thus measuring the register will yield a solution to $f(\mathbf{x}) = 1$ with high probability. This is quadratically better than what could be achieved by preparing and measuring $A|\mathbf{0}\rangle$ until a solution is found.

One application of such a generic amplitude amplification method is for searching. One can also apply this technique to approximately count (Brassard et al. 1997) the number of solutions to $f(\mathbf{x}) = 1$, and more generally to estimate the amplitude (Brassard et al. 2002) with which a general operator A produces a solution to $f(\mathbf{x}) = 1$ (in other words, the transition amplitude from one recognizable subspace to another).

There are a variety of other applications of amplitude amplification that cleverly incorporate amplitude amplification in an unobvious way into a larger algorithm. Some of these examples are discussed in Mosca (2009) and most are summarized in Jordan (2008).

Since there are some connections to some of the more recent tools developed in quantum algorithms, we will briefly explain how amplitude estimation works.

Consider any unitary operator A that maps some known input state, say $|\mathbf{0}\rangle = |00\dots0\rangle$, to a superposition $\sin(\theta)|\psi_1\rangle + \cos(\theta)|\psi_0\rangle$, $0 \leq \theta \leq \pi/2$, where $|\psi_1\rangle$ is a normalized superposition of “good” states \mathbf{x} satisfying $f(\mathbf{x}) = 1$ and $|\psi_0\rangle$ is a normalized superposition of “bad” states \mathbf{x} satisfying $f(\mathbf{x}) = 0$ (again, for simplicity, one can ignore extra junk bits). If we measure $A|\mathbf{0}\rangle$, we would measure a “good” \mathbf{x} with probability $\sin^2(\theta)$. The goal of amplitude estimation is to approximate the amplitude $\sin(\theta)$. One can assume for convenience that there are $t > 0$ good states and $n - t > 0$ bad states, and that $0 < \sin(\theta) < \pi/2$.

It can be noted that the quantum search iterate $Q = -AU_0A^{-1}U_f$ has eigenvalues $|\psi_+\rangle = \frac{1}{\sqrt{2}}(|\psi_0\rangle + i|\psi_1\rangle)$ and $|\psi_-\rangle = \frac{1}{\sqrt{2}}(|\psi_0\rangle - i|\psi_1\rangle)$ with respective eigenvalues $e^{i2\theta}$ and $e^{-i2\theta}$. It also has $2^n - 2$ other eigenvectors; $t - 1$ of them have eigenvalue +1 and are the states orthogonal to $|\psi_1\rangle$ that have support on the states $|\mathbf{x}\rangle$ where $f(\mathbf{x}) = 1$, and $n - t - 1$ of them have eigenvalue -1 and are the states orthogonal to $|\psi_0\rangle$ that have support on the states $|\mathbf{x}\rangle$ where $f(\mathbf{x}) = 0$. It is important to note that $A|\psi\rangle = \frac{e^{i\theta}}{\sqrt{2}}|\psi_+\rangle + \frac{e^{-i\theta}}{\sqrt{2}}|\psi_-\rangle$ has its full support on the two-dimensional subspace spanned by $|\psi_+\rangle$ and $|\psi_-\rangle$.

It is worth noting that the quantum search iterate $Q = AU_0A^{-1}U_f$ can also be thought of as two reflections

$$-U_{A|\mathbf{0}\rangle}U_f = (2A|\mathbf{0}\rangle\langle\mathbf{0}|A^\dagger - I)(I - 2\sum_{\mathbf{x}|f(\mathbf{x})=1}|\mathbf{x}\rangle\langle\mathbf{x}|)$$

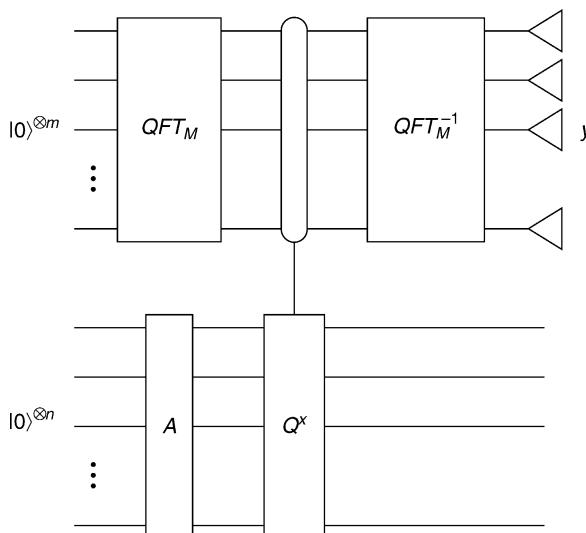
one that flags the “good” subspace with a -1 phase shift, and then one that flags the subspace orthogonal to $A|0\rangle$ with a -1 phase shift. In the two-dimensional subspace spanned by $A|0\rangle$ and $U_f A|0\rangle$, these two reflections correspond to a rotation by angle 2θ . Thus, it should not be surprising to find eigenvalues $e^{\pm 2\pi i \theta}$ for states in this two-dimensional subspace. (In [Sect. 4](#), a more general situation where we have an operator Q with many nontrivial eigenspaces and eigenvalues will be considered.)

Thus, one can approximate $\sin(\theta)$ by estimating the eigenvalue of either $|\psi_+\rangle$ or $|\psi_-\rangle$. Performing the standard eigenvalue estimation algorithm on Q with input $A|0\rangle$ (as illustrated in [Fig. 1](#)) gives a quadratic speed-up for estimating $\sin^2(\theta)$ versus simply repeatedly measuring $A|0\rangle$ and counting the frequency of 1s. In particular, we can obtain an estimate $\tilde{\theta}$ of θ such that $|\tilde{\theta} - \theta| < \varepsilon$ with high (constant) probability using $O(1/\varepsilon)$ repetitions of Q , and thus $O(1/\varepsilon)$ repetitions of $U_f A$ and A^{-1} . For fixed p , this implies that $\tilde{p} = \sin^2(\tilde{\theta})$ satisfies $|\tilde{p} - p| \in O(\varepsilon)$ with high probability. Classical sampling requires $O(1/\varepsilon^2)$ samples for the same precision. One application is speeding up the efficiency of the “Hadamard” test mentioned in [Sect. 5.2.3](#).

Another interesting observation is that increasingly precise eigenvalue estimation of Q on input $A|0\rangle$ leaves the eigenvector register in a state that gets closer and closer to the mixture

Fig. 1

The circuit estimates the amplitude with which the unitary operator A maps $|0\rangle$ to the subspace of solutions to $f(x) = 1$. One uses $m \in \Omega(\log(1/\varepsilon))$ qubits in the top register, and prepares it in a uniform superposition of the strings y representing the integers $0, 1, 2, \dots, 2^m - 1$ (one can in fact optimize the amplitudes of each y to achieve a slightly better estimate (D’Ariano et al. 2007)). The controlled- Q^y circuit applies Q^y to the bottom register when the value y is encoded in the top qubits. If $A|0\rangle = \sin(\theta)|\psi_1\rangle + \cos(\theta)|\psi_0\rangle$, where $|\psi_1\rangle$ is a superposition of solutions x to $f(x) = 1$, and $|\psi_0\rangle$ is a superposition of values x with $f(x) = 0$, then the value $y = y_1 y_2 \dots y_m$ measured in the top register corresponds to the phase estimate $2\pi y / 2^m$, which is likely to be within $\frac{2\pi}{2^m}$ (modulo 2π) of either 2θ or -2θ . Thus, the value of $\sin^2 \pi y / 2^m$ is likely to satisfy $|\sin^2 \pi y / 2^m - \sin^2 \theta| \in O(\varepsilon)$.

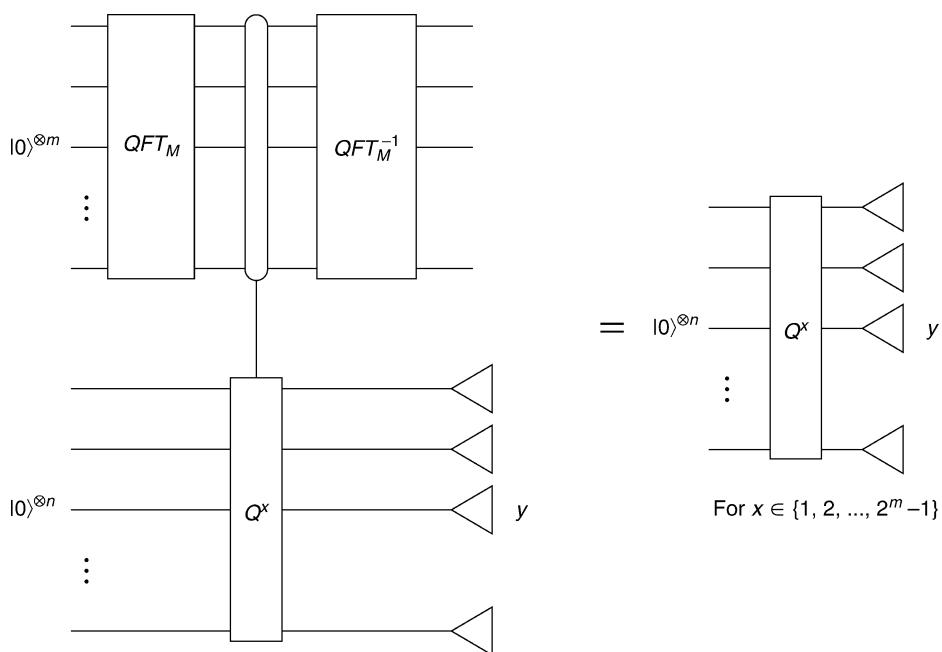


$\frac{1}{2}|\psi_+\rangle\langle\psi_+| + \frac{1}{2}|\psi_-\rangle\langle\psi_-|$ that equals $\frac{1}{2}|\psi_1\rangle\langle\psi_1| + \frac{1}{2}|\psi_0\rangle\langle\psi_0|$. Thus, eigenvalue estimation will leave the eigenvector register in a state that contains a solution to $f(\mathbf{x}) = 1$ with probability approaching $\frac{1}{2}$. One can in fact, using this entire algorithm as subroutine in another quantum search algorithm, obtain an algorithm with success probability approaching 1 (Mosca 2001; Kaye et al. 2007), and the convergence rate can be improved further (Tulsi et al. 2006). Another important observation is that for the purpose of searching, the eigenvalue estimate register is never used except in order to determine a random number of times in which to apply Q to the second register. This in fact gives the quantum searching algorithm of Boyer et al. (1998). In other words, applying Q a random number of times decoheres or approximately projects the eigenvector register in the eigenbasis, which gives a solution with high probability (Fig. 2). The method of approximating projections by using randomized evolutions was recently refined, generalized, and applied in Boixo et al. (2009).

Quantum searching as discussed in this section has been further generalized in the quantum walk paradigm, which is the topic of the next section.

Fig. 2

The amplitude estimation circuit can also be used for searching for a solution to $f(\mathbf{x}) = 1$, as illustrated in the figure on the left. The second register starts off in the state $|0\rangle^{\otimes m}$, and after a sufficiently precise eigenvalue estimation of the quantum search iterate Q , the second register is approximately projected in the eigenbasis. The idea projection would yield the mixed state $\frac{1}{2}|\psi_+\rangle\langle\psi_+| + \frac{1}{2}|\psi_-\rangle\langle\psi_-| = \frac{1}{2}|\psi_1\rangle\langle\psi_1| + \frac{1}{2}|\psi_0\rangle\langle\psi_0|$, and thus yields a solution to $f(\mathbf{x}) = 1$ with probability approaching $\frac{1}{2}$ as m gets large (the probability is in $\Omega(1)$ once $m \in \Omega(1/\sqrt{p})$). The same algorithm can in fact be achieved by discarding the top register and instead randomly picking a value $y \in \{0, 1, \dots, 2^m - 1\}$ and applying Q^y to $|0\rangle^{\otimes n}$, as illustrated on the right.



4 Quantum Walks

Random walks are a common tool throughout classical computer science. Their applications include the simulation of biological, physical, and social systems, as well as probabilistic algorithms such as Monte Carlo methods and the Metropolis algorithm. A classical random walk is described by a $n \times n$ matrix P , where the entry $P_{u,v}$ is the probability of a transition from a vertex v to an adjacent vertex u in a graph G . In order to preserve normalization, it is required that P is *stochastic* – that is, the entries in each column must sum to 1. We denote the initial probability distribution on the vertices of G by the column vector q . After n steps of the random walk, the distribution is given by $P^n q$.

Quantum walks were developed in analogy to classical random walks, but it was not initially obvious how to do this. Most generally, a quantum walk could begin in an initial state $\rho_0 = |\psi_0\rangle\langle\psi_0|$ and evolve according to any completely positive map \mathcal{E} such that, after t time steps, the system is in state $\rho(t) = \mathcal{E}^t(\rho_0)$. Such a quantum walk is simultaneously using classical randomness and quantum superposition. We can focus on the power of quantum mechanics by restricting to unitary walk operations, which maintain the system in a coherent quantum state. So, the state at time t can be described by $|\psi(t)\rangle = U^t|\psi_0\rangle$, for some unitary operator U . However, it is not initially obvious how to define such a unitary operation. A natural idea is to define the state space A with basis $\{|v\rangle : v \in V(G)\}$ and walk operator \tilde{P} defined by $\tilde{P}_{u,v} = \sqrt{P_{u,v}}$. However, this will not generally yield a unitary operator \tilde{P} , and a more complex approach is required. Some of the earliest formulations of unitary quantum walks appear in papers by Nayak and Vishwanath (2000), Ambainis et al. (2001), Kempe (2003), and Aharonov et al. (2001). These early works focused mainly on quantum walks on the line or a cycle. In order to allow unitary evolution, the state space consisted of the vertex set of the graph, along with an extra “coin register.” The state of the coin register is a superposition of $|\text{LEFT}\rangle$ and $|\text{RIGHT}\rangle$. The walk then proceeds by alternately taking a step in the direction dictated by the coin register and applying a unitary “coin tossing operator” to the coin register. The coin tossing operator is often chosen to be the Hadamard gate. It was shown in Aharonov et al. (2001), Ambainis (2003), Ambainis et al. (2001), Kempe (2003), and Nayak and Vishwanath (2000) that the mixing and propagation behavior of these quantum walks was significantly different from their classical counterparts. These early constructions developed into the more general concept of a discrete time quantum walk, which will be defined in detail.

Two methods will be described for defining a unitary walk operator. In a *discrete time* quantum walk, the state space has basis vectors $\{|u\rangle \otimes |v\rangle : u, v \in V\}$. Roughly speaking, the walk operator alternately takes steps in the first and second registers. This is often described as a walk on the edges of the graph. In a *continuous time* quantum walk, the attention will be restricted to symmetric transition matrices P . We take P to be the Hamiltonian for the system. Applying Schrödinger’s equation, this will define continuous time unitary evolution in the state space spanned by $\{|u\rangle : u \in V\}$. Interestingly, these two types of walk are not known to be equivalent. An overview of both types of walk as well as some of the algorithms that apply them will be given.

4.1 Discrete Time Quantum Walks

Let P be a stochastic matrix describing a classical random walk on a graph G . We would like the quantum walk to respect the structure of the graph G , and take into account the

transition probabilities $P_{u,v}$. The quantum walk should be governed by a unitary operation, and is therefore reversible. However, a classical random walk is not, in general, a reversible process. Therefore, the quantum walk will necessarily behave differently than the classical walk. While the state space of the classical walk is V , the state quantum walk takes place in the space spanned by $\{|u, v\rangle : u, v \in V\}$. One can think of the first register as the current location of the walk, and the second register as a record of the previous location. To facilitate the quantum walk from a state $|u, v\rangle$, one can first mix the second register over the neighbors of u , and then swap the two registers. The method by which one mixes over the neighbors of u must be chosen carefully to ensure that it is unitary. To describe this formally, one can define the following states for each $u \in V$:

$$|\psi_u\rangle = |u\rangle \otimes \sum_{v \in V} \sqrt{P_{vu}} |v\rangle \quad (4)$$

$$|\psi_u^*\rangle = \sum_{v \in V} \sqrt{P_{vu}} |v\rangle \otimes |u\rangle \quad (5)$$

Furthermore, define the projections onto the space spanned by these states:

$$\Pi = \sum_{u \in V} |\psi_u\rangle \langle \psi_u| \quad (6)$$

$$\Pi^* = \sum_{u \in V} |\psi_u^*\rangle \langle \psi_u^*| \quad (7)$$

In order to mix the second register, we perform the reflection $(2\Pi - I)$. Letting S denote the swap operation, this process can be written as

$$S(2\Pi - I) \quad (8)$$

It turns out that we will get a more elegant expression for a single step of the quantum walk if we define the walk operator W to be two iterations of this process:

$$W = S(2\Pi - I)S(2\Pi - I) \quad (9)$$

$$= (2(S\Pi S) - I)(2\Pi - I) \quad (10)$$

$$= (2\Pi^* - I)(2\Pi - I) \quad (11)$$

So, the walk operator W is equivalent to performing two reflections.

Many of the useful properties of quantum walks can be understood in terms of the spectrum of the operator W . We define D , the $n \times n$ matrix with entries $D_{u,v} = \sqrt{P_{u,v}P_{v,u}}$. This is called the *discriminant matrix*, and has eigenvalues in the interval $[0, 1]$. In the theorem that follows, the eigenvalues of D that lie in the interval $(0, 1)$ will be expressed as $\cos(\theta_1), \dots, \cos(\theta_k)$. Let $|\theta_1\rangle, \dots, |\theta_k\rangle$ be the corresponding eigenvectors of D . Now, define the subspaces

$$A = \text{span}\{|\psi_u\rangle\} \quad (12)$$

$$B = \text{span}\{|\psi_u^*\rangle\} \quad (13)$$

Finally, define the operator

$$Q = \sum_{v \in V} |\psi_v\rangle \langle v| \quad (14)$$

and

$$|\phi_j\rangle = Q|\theta_j\rangle \quad (15)$$

The following spectral theorem for quantum walks can be now stated:

Theorem 1 (Szegedy 2004) *The eigenvalues of W acting on the space $A + B$ can be described as follows:*

1. *The eigenvalues of W with non-zero imaginary part are $e^{\pm 2i\theta_1}, e^{\pm 2i\theta_2}, \dots, e^{\pm 2i\theta_k}$ where $\cos(\theta_1), \dots, \cos(\theta_k)$ are the eigenvalues of D in the interval $(0, 1)$. The corresponding (un-normalized) eigenvectors of $W(P)$ can be written as $|\phi_j\rangle - e^{\pm 2i\theta_j} S |\phi_j\rangle$ for $j = 1, \dots, k$.*
2. *$A \cap B$ and $A^\perp \cap B^\perp$ span the $+1$ eigenspace of W . There is a direct correspondence between this space and the $+1$ eigenspace of D . In particular, the $+1$ eigenspace of W has the same degeneracy as the $+1$ eigenspace of D .*
3. *$A \cap B^\perp$ and $A^\perp \cap B$ span the -1 eigenspace of W .*

We say that P is *symmetric* if $P^T = P$ and *ergodic* if it is aperiodic. Note that if P is symmetric, then the eigenvalues of D are just the absolute values of the eigenvalues of P . It is well-known that if P is ergodic, then it has exactly one stationary distribution (i.e., a unique $+1$ eigenvalue). Combining this fact with theorem (1) gives the following corollary.

Corollary 1 *If P is ergodic and symmetric, then the corresponding walk operator W has a unique $+1$ eigenvector in $\text{span}(A, B)$:*

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{v \in V} |\psi_v\rangle = \frac{1}{\sqrt{n}} \sum_{v \in V} |\psi_v^*\rangle \quad (16)$$

Moreover, if we measure the first register of $|\psi\rangle$, we get a state corresponding to vertex u with probability

$$\Pr(u) = \frac{1}{n} \sum_{v \in V} P_{u,v} = \frac{1}{n} \quad (17)$$

This is the uniform distribution, which is the unique stationary distribution for the classical random walk.

4.1.1 The Phase Gap and the Detection Problem

In this section, an example of a quadratic speedup will be given for the problem of detecting whether there are any “marked” vertices in the graph G . First, the following are defined:

Definition 1 The *phase gap* of a quantum walk is defined as the smallest positive value 2θ such that $e^{\pm 2i\theta}$ are eigenvalues of the quantum walk operator. It is denoted by $\Delta(P)$.

Definition 2 Let $M \subseteq V$ be a set of marked vertices. In the *detection problem*, one is asked to decide whether M is empty.

Let P be ergodic and reversible, with positive eigenvalues. We define the following modified walk P' :

$$P'_{uv} = \begin{cases} P_{uv} & v \notin M \\ 0 & u \neq v, v \in M \\ 1 & u = v, v \in M \end{cases} \quad (18)$$

This walk resembles P , except that it acts as the identity on the set M . That is, if the walk reaches a marked vertex, it stays there. Let P_M denote the operator P' restricted to $V \setminus M$. Then, arranging the rows and columns of P' , we can write

$$P' = \begin{pmatrix} P_M & 0 \\ P'' & I \end{pmatrix} \quad (19)$$

By Theorem 1, if $M = \emptyset$, then $P_M = P' = P$ and $\|P_M\| = 1$. Otherwise, we have the strict inequality $\|P_M\| < 1$. The following theorem bounds $\|P_M\|$ away from 1:

Theorem 2 *If $(1 - \delta)$ is the absolute value of the eigenvalue of P with second largest magnitude, and $|M| \geq |\varepsilon|V$, then $\|P_M\| \leq 1 - \frac{\delta\varepsilon}{2}$.*

It will be now shown that the detection problem can be solved using eigenvalue estimation. \blacklozenge Theorem 2 will allow us to bound the running time of this method. First, we describe the discriminant matrix for P' :

$$D(P')_{uv} = \begin{cases} P_{uv} & u, v \notin M \\ 1 & u = v, v \in M \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

Now, beginning with the state

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{v \in V} |\psi_v\rangle = \frac{1}{\sqrt{n}} \sum_{v \in V} \sqrt{P_{v,u}} |u\rangle |\nu\rangle \quad (21)$$

we measure whether or not we have a marked vertex; if so, we are done. Otherwise, we have the state

$$|\psi_M\rangle = \frac{1}{\sqrt{|V \setminus M|}} \sum_{u, v \in V \setminus M} \sqrt{P_{vu}} |u\rangle |\nu\rangle \quad (22)$$

If $M = \emptyset$, then this is the state $|\psi\rangle$ defined in \blacklozenge Eq. 16, and is the +1 eigenvector of $W(P)$. Otherwise, by Theorem 1, this state lies entirely in the space spanned by eigenvectors with values of the form $e^{\pm 2i\theta_j}$, where θ_j is an eigenvalue of P_M . Applying \blacklozenge Theorem 2, we know that

$$\theta \geq \cos^{-1} \left(1 - \frac{\delta\varepsilon}{2} \right) \geq \sqrt{\frac{\delta\varepsilon}{2}} \quad (23)$$

So, the task of distinguishing between M being empty or nonempty is equivalent to that of distinguishing between a phase parameter of 0 and a phase parameter of at least $\sqrt{\frac{\delta\varepsilon}{2}}$. Therefore, applying phase estimation to $W(P')$ on state $|\psi_M\rangle$ with precision $O(\sqrt{\delta\varepsilon})$ will decide whether M is empty with constant probability. This requires time $O\left(\frac{1}{\sqrt{\delta\varepsilon}}\right)$.

By considering the modified walk operator P' , it can be shown that the detection problem requires $O\left(\frac{1}{\delta\varepsilon}\right)$ time in the classical setting. Therefore, the quantum algorithm provides a quadratic speedup over the classical one for the detection problem.

4.1.2 Quantum Hitting Time

Classically, the first hitting time is denoted $H(\rho, M)$. For a walk defined by P , starting from the probability distribution ρ on V , $H(\rho, M)$ is the smallest value n such that the walk reaches a marked vertex $v \in M$ at some time $t \in \{0, \dots, n\}$ with constant probability. This idea is captured by applying the modified operator P' and some n times, and then considering the probability that the walk is in some marked state $v \in M$. Let ρ_M be any initial distribution restricted to the vertices $V \setminus M$. Then, at time t , the probability that the walk is in an unmarked state is $\|P_M^t \rho_M\|_1$, where $\|\cdot\|_1$ denotes the L_1 norm. Assuming that M is nonempty, we can see that $\|P_M\| < 1$. So, as $t \rightarrow \infty$, we have $\|P_M^t \rho_M\|_1 \rightarrow 0$. So, as $t \rightarrow \infty$, the walk defined by P' is in a marked state with probability 1. As a result, if we begin in the uniform distribution π on V , and run the walk for some time t , we will “skew” the distribution toward M , and thus away from the unmarked vertices. So, we define the classical hitting time to be the minimum t such that

$$\|P_M^t \rho_M\|_1 < \varepsilon \quad (24)$$

for any constant ε of our choosing. Since the quantum walk is governed by a unitary operator, it does not converge to a particular distribution the way that the classical walk does. We cannot simply wait an adequate number of time steps and then measure the state of the walk; the walk might have already been in a state with high overlap on the marked vertices and then evolved away from this state! Quantum searching has the same problem when the number of solutions is unknown. We can get around this in a similar way by considering an expected value over a sufficiently long period of time. This will form the basis of the definition of hitting time. Define

$$|\pi\rangle = \frac{1}{\sqrt{|V \setminus M|}} \sum_{v \in V \setminus M} |\psi_v\rangle \quad (25)$$

Then, if M is empty, $|\pi\rangle$ is a $+1$ eigenvector of W , and $W^t|\pi\rangle = |\pi\rangle$ for all t . However, if M is nonempty, then the spectral theorem tells that $|\pi\rangle$ lies in the space spanned by eigenvectors with eigenvalues $e^{\pm 2i\theta_j}$ for nonzero θ_j . As a result, it can be shown that, for some values of t , the state $W^t|\pi\rangle$ is “far” from the initial distribution $|\pi\rangle$. The quantum hitting is defined in the same way as Szegedy (2004). The hitting time $H_Q(W)$ as the minimum value T such that

$$\frac{1}{T+1} \sum_{t=0}^T \|W^t|\pi\rangle - |\pi\rangle\|^2 \geq 1 - \frac{|M|}{|V|} \quad (26)$$

This leads to Szegedy’s hitting time theorem (2004).

Theorem 3 *The quantum hitting time $H_Q(W)$ is*

$$O\left(\frac{1}{\sqrt{1 - \|P_M\|}}\right)$$

Corollary 2 *Applying \bullet Theorem 2, if the second largest eigenvalue of P has magnitude $(1 - \delta)$ and $|M|/|V| \geq \varepsilon$, then $H_Q(W) \in O\left(\frac{1}{\sqrt{\delta\varepsilon}}\right)$.*

Notice that this corresponds to the running time for the algorithm for the detection problem, as described above. Similarly, the classical hitting time is in $O\left(\frac{1}{\delta\varepsilon}\right)$, corresponding to the best classical algorithm for the detection problem.

It is also worth noting that, if there are no marked elements, then the system remains in the state $|\pi\rangle$, and the algorithm never “hits.” This gives an alternative way to approach the detection problem. We run the algorithm for a randomly selected number of steps $t \in \{0, \dots, T\}$ with T of size $O(\sqrt{1/\delta\varepsilon})$, and then measure whether the system is still in the state $|\pi\rangle$; if there are any marked elements, then we can expect to find some other state with constant probability.

4.1.3 The Element Distinctness Problem

In the element distinctness problem, a black box is given that computes the function

$$f: \{1, \dots, n\} \rightarrow S \quad (27)$$

and we are asked to determine whether there exist $x, y \in \{1, \dots, n\}$ with $x \neq y$ and $f(x) = f(y)$. We would like to minimize the number of queries made to the black box. There is a lower bound of $\Omega(n^{2/3})$ on the number of queries, due indirectly to Aaronson and Shi (2004). The algorithm of Ambainis (2004) proves that this bound is tight. The algorithm uses a quantum walk on the Johnson graph $J(n, p)$ has vertex set consisting of all subsets of $\{1, \dots, n\}$ of size p . Let S_1 and S_2 be p -subsets of $\{1, \dots, n\}$. Then, S_1 is adjacent to S_2 if and only if $|S_1 \cap S_2| = p - 1$. The Johnson graph $J(n, p)$ therefore has $\binom{n}{p}$ vertices, each with degree $p(n - p)$. The state corresponding to a vertex S of the Johnson graph will not only record which subset $\{s_1, \dots, s_p\} \subseteq \{1, \dots, n\}$ it represents, but the function values of those elements. That is,

$$|V\rangle = |s_1, s_2, \dots, s_p; f(s_1), f(s_2), \dots, f(s_p)\rangle \quad (28)$$

Setting up such a state requires p queries to the black box.

The walk then proceeds for t iteration, where t is chosen from the uniform distribution on $\{0, \dots, T\}$ and $T \in O(1/\sqrt{\delta\varepsilon})$. Each iteration has two parts to it. First, one needs to check if there are distinct s_i, s_j with $f(s_i) = f(s_j)$ – that is, whether the vertex S is marked. This requires no calls to the black box, since the function values are stored in the state itself. Second, if the state is unmarked, one needs to take a step of the walk. This involves replacing, say, s_i with s'_i , requiring one query to erase $f(s_i)$ and another to insert the value $f(s'_i)$. So, each iteration requires a total of 2 queries. ε and δ will be bounded now. If only one pair x, y exists with $f(x) = f(y)$, then there are $\binom{n-2}{p-2}$ marked vertices. This tells that, if there are any such pairs at all, epsilon is $\Omega(p^2/n^2)$. Johnson graphs are very well understood in graph theory. It is a well-known result that the eigenvalues for the associated walk operator are given by

$$\lambda_j = 1 - \frac{j(n+1-j)}{p(n-p)} \quad (29)$$

for $0 \leq j \leq p$. For a proof, see Brouwer (1989). This gives $\delta = \frac{n}{p(n-p)}$. Putting this together, we find that $\sqrt{1/\delta\varepsilon}$ is $O\left(\frac{n}{\sqrt{p}}\right)$. So, the number of queries required is $O\left(p + \frac{n}{\sqrt{p}}\right)$. In order to minimize this quantity, we choose p to be of size $\Theta(n^{2/3})$. The query complexity of this algorithm is $O(n^{2/3})$, matching the lower bound of Aaronson and Shi (2004).

4.1.4 Unstructured Search as a Discrete Time Quantum Walk

Unstructured search in terms of quantum walks will be considered now. For unstructured search, we are required to identify a marked element from a set of size n . Let M denote the set

of marked elements, and U denote the set of unmarked elements. Furthermore, let $m = |M|$ and $q = |U|$. It is assumed that the number of marked elements, m , is very small in relation to the total number of elements n . If this were not the case, we could efficiently find a marked element with high probability by simply checking a randomly chosen set of vertices. Since the set lacks any structure or ordering, the corresponding walk takes place on the complete graph on n vertices. One can define the following three states:

$$|UU\rangle = \frac{1}{q} \sum_{u,v \in U} |u, v\rangle \quad (30)$$

$$|UM\rangle = \frac{1}{\sqrt{nq}} \sum_{\substack{u \in U \\ v \in M}} |u, v\rangle \quad (31)$$

$$|MU\rangle = \frac{1}{\sqrt{nq}} \sum_{\substack{u \in M \\ v \in U}} |u, v\rangle \quad (32)$$

Noting that $\{|UU\rangle, |UM\rangle, |MU\rangle\}$ is an orthonormal set, the action of the walk operator on the three-dimensional space will be considered

$$\Gamma = \text{span}\{|UU\rangle, |UM\rangle, |MU\rangle\} \quad (33)$$

In order to do this, we will express the spaces A and B , defined in [Eqs. 12](#) and [13](#) in terms of a different basis. First we can label the unmarked vertices:

$$U = \{u_0, \dots, u_{q-1}\} \quad (34)$$

Then, we can define

$$|\gamma_k\rangle = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} e^{\frac{2\pi i j k}{q}} |\psi_{u_j}\rangle \quad (35)$$

and

$$|\gamma_k^*\rangle = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} e^{-\frac{2\pi i j k}{q}} |\psi_{u_j}^*\rangle \quad (36)$$

with k ranging from 0 to $q - 1$. Note that $|\gamma_0\rangle$ corresponds to the definition of $|\pi\rangle$ in [Eq. 25](#). We can then rewrite A and B :

$$A = \text{span} \{|\gamma_k\rangle\} \quad (37)$$

$$B = \text{span} \{|\gamma_k^*\rangle\} \quad (38)$$

Now, note that for $k \neq 0$, the space Γ is orthogonal to $|\gamma_k\rangle$ and $|\gamma_k^*\rangle$. Furthermore,

$$|\gamma_0\rangle = \frac{1}{\sqrt{n}} (\sqrt{m}|UM\rangle + \sqrt{q}|UU\rangle) \quad (39)$$

and

$$|\gamma_0^*\rangle = \frac{1}{\sqrt{n}} (\sqrt{m}|MU\rangle + \sqrt{q}|UU\rangle) \quad (40)$$

Therefore, the walk operator

$$W = (2\Pi^* - I)(2\Pi - I) \quad (41)$$

when restricted to Γ is simply

$$W' = \left(2|\gamma_0^*\rangle\langle\gamma_0^*| - I\right)\left(2|\gamma_0\rangle\langle\gamma_0| - I\right) \quad (42)$$

► *Figure 3* illustrates the space Γ that contains the vectors $|\gamma_0\rangle$ and $|\gamma_k^*\rangle$.

At this point, it is interesting to compare this algorithm to Grover's search algorithm. It can be defined that

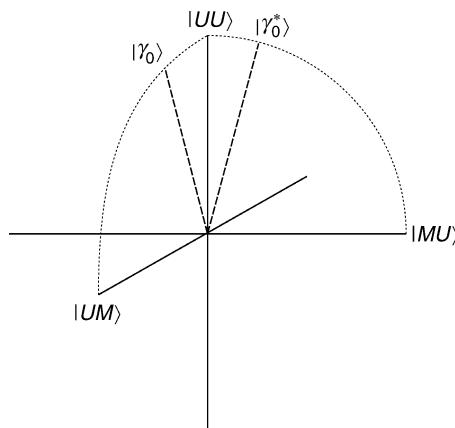
$$|\rho_t\rangle = W^t|UU\rangle. \quad (43)$$

Now, define $|\rho_t^1\rangle$ and $|\rho_t^2\rangle$ to be the projection of $|\rho_t\rangle$ onto $\text{span}\{|UU\rangle, |UM\rangle\}$ and $\text{span}\{|UU\rangle, |MU\rangle\}$, respectively. We can think of these as the “shadow” cast by the vector $|\rho_t\rangle$ on two-dimensional planes within the space Γ . Note that $|\gamma_0\rangle$ lies in the $\text{span}\{|UU\rangle, |UM\rangle\}$ and its projection onto $\text{span}\{|UU\rangle, |MU\rangle\}$ is $\sqrt{q/n}|UU\rangle$. So, the walk operator acts on $|\rho_t\rangle$ by reflecting it around the vector γ_0 and then around $\sqrt{q/n}|UU\rangle$. This is very similar to Grover search, except for the fact that the walk operator in this case does not preserve the magnitude of $|\rho_t^1\rangle$. So, with each application of the walk operator, $|\rho_t^1\rangle$ is rotated by 2θ , where $\theta = \tan^{-1}(\sqrt{m/q})$ is the angle between $|UU\rangle$ and $|\gamma_0\rangle$. The case for $|\rho_t^2\rangle$ is exactly analogous, with the rotation taking place in the plane $\text{span}\{|UU\rangle, |MU\rangle\}$. So, we can think of the quantum walk as Grover search taking place simultaneously in two intersecting planes. It should not come as a surprise, then, that we can achieve a quadratic speedup similar to that of Grover search.

A slightly modified definition of hitting time will be now used to show that the walk can indeed be used to find a marked vertex in $O(\sqrt{n})$ time. Rather than using the hitting time as defined in ◉ Sect. 4.1.2, the state $|\pi\rangle = |\gamma_0\rangle$ will be replaced with $|UU\rangle$. Note that we can create $|UU\rangle$ from $|\pi\rangle$ by simply measuring whether the second register contains a marked vertex. This is only a small adjustment, since $|\gamma_0\rangle$ is close to $|UU\rangle$. Furthermore, both lie in the space Γ , and the action of the walk operator is essentially identical for both starting states.

■ **Fig. 3**

The three-dimensional space Γ . The walk operator acts on this space by alternately performing reflections in the vector $|\gamma_0\rangle$ and $|\gamma_k^*\rangle$.



It should not be surprising to the reader that the results of [Sect. 4.1.2](#) apply to this modified definition as well.

In this case, the operator P is defined by

$$P_{uv} = \begin{cases} 0 & \text{if } u = v \\ \frac{1}{n-1} & \text{if } u \neq v \end{cases} \quad (44)$$

Let v_0, \dots, v_{n-1} be a labeling of the vertices of G . Let x^0, \dots, x^{n-1} denote the eigenvectors of P , with $x_{v_j}^k$ denoting the amplitude on v_j in x^k . Then, the eigenvalues of P are as follows:

$$x_{v_j}^k = \frac{1}{n} \cdot e^{\frac{2\pi i j k}{n}} \quad (45)$$

Then, x^0 has eigenvalue 1, and is the stationary distribution. All the other x^k have eigenvalue $-1/(n-1)$, giving a spectral gap of $\delta = \frac{n-2}{n-1}$. Applying corollary 2 gives the quantum hitting time for the corresponding quantum walk operator W

$$H_Q(W) \leq \sqrt{\frac{(n-1)n}{n-2}} = O(\sqrt{n}) \quad (46)$$

So, we run the walk for some randomly selected time $t \in \{0, \dots, T-1\}$ with T of size $O(\sqrt{n})$, then measure whether either the first or second register contains a marked vertex. Applying Theorem 3, the probability that neither contains a marked vertex is

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} |\langle \rho_t | UU \rangle|^2 &\leq \frac{1}{T} \sum_{t=0}^{T-1} |\langle \rho_t | UU \rangle| \\ &= 1 - \frac{1}{2T} \sum_{t=0}^{T-1} \| |\rho_t\rangle - |UU\rangle \|^2 \\ &\leq 1 - \frac{|M|}{2|V|} \end{aligned}$$

Assuming that $|M|$ is small compared to $|V|$, we find a marked vertex in either the first or second register with high probability. Repeating this procedure a constant number of times, the success probability can be forced arbitrarily close to 1.

4.1.5 The MNRS Algorithm

Magniez et al. (2007) developed an algorithm that generalizes the search algorithm described above to any graph. A brief overview of this algorithm and others is also given in the survey paper by Santha (2008). The MNRS algorithm employs similar principles to Grover's algorithm; we apply a reflection in M , the space of unmarked states, followed by a reflection in $|\pi\rangle$, a superposition of marked and unmarked states. This facilitates a rotation through an angle related to the number of marked states. In the general case considered by Magniez et al., $|\pi\rangle$ is the stationary distribution of the walk operator. It turns out to be quite difficult to implement a reflection in $|\pi\rangle$ exactly. Rather, the MNRS algorithm employs an approximate version of this reflection. This algorithm requires $O\left(\frac{1}{\sqrt{\delta\epsilon}}\right)$ applications of the walk operator, where δ is the eigenvalue gap of the operator P , and ϵ is the proportion vertices that are marked. In his survey paper, Santha (2008) outlines some applications of the MNRS algorithm, including a version of the element distinctness problem where we are asked to find elements x and y such that $f(x) = f(y)$.

4.2 Continuous Time Quantum Walks

To define a classical continuous time random walk for a graph with no loops, we define a matrix similar to the adjacency matrix of G , called the *Laplacian*:

$$L_{uv} = \begin{cases} 0 & u \neq v, uv \notin E \\ 1 & u \neq v, uv \in E \\ -\deg(v) & u = v \end{cases} \quad (47)$$

Then, given a probability distribution $p(t)$ on the vertices of G , the walk is defined by

$$\frac{d}{dt} p(t) = Lp(t) \quad (48)$$

Using the Laplacian rather than the adjacency matrix ensures that $p(t)$ remains normalized. A continuous time quantum walk is defined in a similar way. For simplicity, it will be assumed that the Laplacian is symmetric, although it is still possible to define the walk in the asymmetric case. Then, since the Laplacian is Hermitian, we can simply take it to be the Hamiltonian of our system. Letting $|\rho(t)\rangle$ be a normalized vector in \mathbb{C}^V , Schrödinger's equation gives

$$i \frac{d}{dt} |\rho(t)\rangle = L|\rho(t)\rangle \quad (49)$$

Solving this equation, we get an explicit expression for $\rho(t)$:

$$|\rho(t)\rangle = e^{-itL} |\rho(0)\rangle \quad (50)$$

Let $\{|\lambda_1\rangle \dots |\lambda_n\rangle\}$ be the eigenvectors of L with corresponding values $\{\lambda_1, \dots, \lambda_n\}$. We can rewrite the expression for $\rho(t)$ in terms of this basis as follows:

$$|\rho(t)\rangle = \sum_{j=1}^n e^{-i\lambda_j t} \langle \lambda_j | \rho_0 \rangle |\lambda_j\rangle \quad (51)$$

Clearly, the behavior of a continuous time quantum walk is very closely related to the eigenvectors and spectrum of the Laplacian.

Note that we are not required to take the Laplacian as the Hamiltonian for the walk. We could take any Hermitian matrix we like including the adjacency matrix, or the transition matrix of a (symmetric) Markov chain.

4.2.1 A Continuous Time Walk for Unstructured Search

For unstructured search, the walk takes place on a complete graph. First, the following two states are defined:

$$|V\rangle = \frac{1}{\sqrt{|V|}} \sum_{v \in V} |v\rangle \quad (52)$$

$$|M\rangle = \frac{1}{\sqrt{|M|}} \sum_{v \in M} |v\rangle \quad (53)$$

The Hamiltonian that will be used is a slightly modified version of the Laplacian for the complete graph, with an extra “marking term” added:

$$H = |V\rangle\langle V| + |M\rangle\langle M| \quad (54)$$

It is convenient to consider the action of this Hamiltonian in terms of the vectors $|M\rangle$ and $|M^\perp\rangle$, where

$$|M^\perp\rangle = \frac{1}{\sqrt{|V \setminus M|}} \sum_{v \in V \setminus M} |v\rangle = \frac{1}{\sqrt{1 - \langle M|V\rangle^2}} (|V\rangle - |M\rangle\langle M|V\rangle) \quad (55)$$

As outlined in Childs (2008), we let $\alpha = \langle M|V\rangle$. We can then rewrite the Hamiltonian in terms of the basis $\{|M\rangle, |M^\perp\rangle\}$:

$$H = \begin{pmatrix} \alpha^2 & \alpha\sqrt{1-\alpha^2} \\ \alpha\sqrt{1-\alpha^2} & 1-\alpha^2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad (56)$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \alpha^2 \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \alpha\sqrt{1-\alpha^2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (57)$$

$$= I + \alpha^2 \sigma_Z + \alpha\sqrt{1-\alpha^2} \sigma_X \quad (58)$$

$$= I + \alpha(\alpha\sigma_Z + \sqrt{1-\alpha^2}\sigma_X) \quad (59)$$

where σ_X and σ_Z are the Pauli X and Z operators. Note that the identity term in the sum simply introduces a global phase, and can be ignored. Note that the operator

$$A = \alpha\sigma_Z + \sqrt{1-\alpha^2}\sigma_X \quad (60)$$

has eigenvalues ± 1 , and is therefore Hermitian and unitary. Therefore, we can write

$$e^{-iHt} = e^{-iAt} = \cos(\alpha t)I - i\sin(\alpha t)A \quad (61)$$

Also note that

$$A|V\rangle = (\alpha\sigma_Z + \sqrt{1-\alpha^2}\sigma_X)(\alpha|M\rangle + \sqrt{1-\alpha^2}|M^\perp\rangle) \quad (62)$$

$$= |M\rangle \quad (63)$$

If we start with the state $|V\rangle$, we can now calculate the state of the system at time t :

$$|\rho(t)\rangle = \cos(\alpha t)|V\rangle - i\sin(\alpha t)A|V\rangle \quad (64)$$

$$= \cos(\alpha t)|V\rangle - i\sin(\alpha t)|M\rangle \quad (65)$$

So, at time t , the probability of finding the system in a state corresponding to a marked vertex is

$$\sum_{v \in M} |\langle v|\rho(t)\rangle|^2 = |M| \left(\frac{\cos^2(\alpha t)}{|V|} + \frac{\sin^2(\alpha t)}{|M|} \right) \quad (66)$$

$$= \alpha^2 \cos^2(\alpha t) + \sin^2(\alpha t) \quad (67)$$

At time $t = 0$, this is the same as sampling from the uniform distribution, as we would expect. However, at time $t = \pi/2\alpha = \pi/2 \cdot \sqrt{N/M}$, we observe a marked vertex with probability 1. Therefore, this search algorithm runs in time $O(\sqrt{N/M})$, which coincides with the running time of Grover's search algorithm and the related discrete time quantum walk algorithm.

4.2.2 Mixing in Quantum Walks and the Glued Tree Traversal Problem

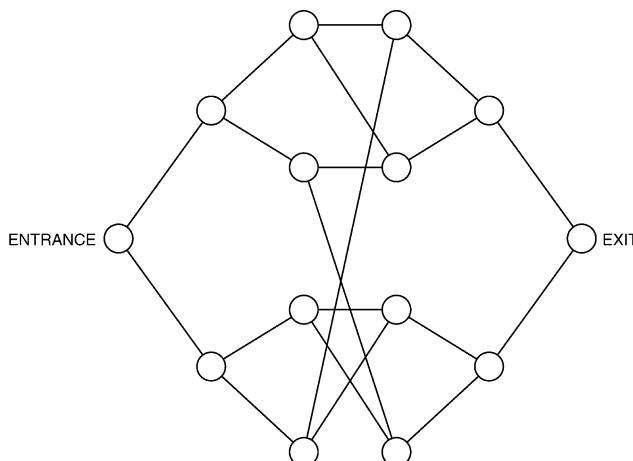
While continuous time quantum walks give a generic quadratic speedup over their classical counterparts, there are some graphs on which the quantum walk gives exponential speedup. One such example is the “glued trees” of Childs et al. (2003). In this example, the walk takes place on the following graph, obtained by taking two identical binary trees of depth d and joining their leaves using a random cycle, as illustrated in Fig. 4. Beginning at the vertex ENTRANCE, we would like to know how long we need to run the algorithm to reach EXIT. It is straightforward to show that classically this will take time exponential in d , the depth of the tree. Intuitively, this is because there are so many vertices and edges in the “middle” of the graph, that there is small probability of a classical walk “escaping” to the exit. It will then be proved that a continuous time quantum walk can achieve an overlap of $\Omega(P(d))$ with the EXIT vertex in time $O\left(\frac{1}{Q(d)}\right)$, where Q and P are polynomials.

Let V_s denote the set of vertices at depth s , so that $V_0 = \{\text{ENTRANCE}\}$ and $V_{2d+1} = \{\text{EXIT}\}$. Taking the adjacency matrix to be the Hamiltonian, the operator $U(t) = e^{-iHt}$ acts identically on the vertices in V_s for any s . Therefore, the states

$$|s\rangle = \frac{1}{\sqrt{|V_s|}} \sum_{v \in V_s} |v\rangle \quad (68)$$

Fig. 4

A small glued tree graph.



form a convenient basis. We can therefore think of the walk on G as a walk on the states $\{|s\rangle : 0 \leq s \leq 2d + 1\}$. We also note that, if A is the adjacency matrix of G ,

$$A|s\rangle = \begin{cases} \sqrt{2}|s+1\rangle & s = 0 \\ \sqrt{2}|s-1\rangle & s = 2d + 1 \\ \sqrt{2}|s-1\rangle + 2|s+1\rangle & s = d \\ \sqrt{2}|s+1\rangle + 2|s-1\rangle & s = d + 1 \\ \sqrt{2}|s+1\rangle + \sqrt{2}|s-1\rangle & \text{otherwise} \end{cases} \quad (69)$$

Aside from the exceptions at ENTRANCE, EXIT, and the vertices in the center, this walk looks exactly like the walk on a line with uniform transition probabilities.

Continuous time classical random walks will eventually converge to a limiting distribution. The time that it takes for the classical random walk to get “close” to this distribution is called the *mixing time*. More formally, we can take some small constant γ , and take the mixing time to be the amount of time it takes to come within γ of the stationary distribution, according to some metric. In general, we express the mixing time in terms of $1/\gamma$ and n , the number of vertices in the graph.

Since quantum walks are governed by a unitary operator, the same convergent behavior cannot be expected. However, the limiting behavior of quantum walks can be defined by taking an average over time. In order to do this, we define $\Pr(u,v,T)$. If we select $t \in [0, T]$ uniformly at random and run the walk for time t , beginning at vertex u , then $\Pr(u,v,T)$ is the probability that we find the system in state v . Formally, this can be written as

$$\Pr(u,v,T) = \frac{1}{T} \int_0^T |\langle v | e^{-iAt} | u \rangle|^2 dt \quad (70)$$

where A is the adjacency matrix of the graph. Now, if $\{|\lambda\rangle\}$ is taken to be the set of eigenvectors of A with corresponding eigenvalues $\{\lambda\}$, then $\Pr(u,v,T)$ can be rewritten as follows:

$$\Pr(u,v,T) = \frac{1}{T} \int_0^T \left| \sum_{\lambda} e^{-i\lambda t} \langle v | \lambda \rangle \langle \lambda | u \rangle \right|^2 dt \quad (71)$$

$$= \frac{1}{T} \int_0^T \sum_{\lambda, \lambda'} \left(e^{-i(\lambda - \lambda')t} \langle v | \lambda \rangle \langle \lambda | u \rangle \langle v | \lambda' \rangle \langle \lambda' | u \rangle \right) dt \quad (72)$$

$$= \sum_{\lambda} |\langle v | \lambda \rangle \langle \lambda | u \rangle|^2 \quad (73)$$

$$+ \frac{1}{T} \sum_{\lambda \neq \lambda'} \langle v | \lambda \rangle \langle \lambda | u \rangle \langle v | \lambda' \rangle \langle \lambda' | u \rangle \int_0^T e^{-i(\lambda - \lambda')t} dt \quad (74)$$

$$= \sum_{\lambda} |\langle v | \lambda \rangle \langle \lambda | u \rangle|^2 \quad (75)$$

$$+ \sum_{\lambda \neq \lambda'} \langle v | \lambda \rangle \langle \lambda | u \rangle \langle v | \lambda' \rangle \langle \lambda' | u \rangle \frac{1 - e^{-i(\lambda - \lambda')T}}{i(\lambda - \lambda')T} \quad (76)$$

In particular, we have

$$\lim_{T \rightarrow \infty} \Pr(u, v, T) = \sum_{\lambda} |\langle v | \lambda \rangle \langle \lambda | u \rangle|^2 \quad (77)$$

This value will be denoted by $\Pr(u, v, \infty)$. This is the quantum analogue of the limiting distribution for a classical random walk. We would now like to apply this to the specific case of the glued tree traversal problem. First, we will lower bound $\Pr(\text{ENTRANCE}, \text{EXIT}, \infty)$. It will be then shown that $\Pr(\text{ENTRANCE}, \text{EXIT}, T)$ approaches this value rapidly as T is increased, implying that we can traverse the glued tree structure efficiently using a quantum walk.

Define the reflection operator Θ by

$$\Theta|j\rangle = |2d - 1 - j\rangle \quad (78)$$

This operator commutes with the adjacency matrix, and hence the walk operator because of the symmetry of the glued trees. This implies that Θ can be diagonalized in the eigenbasis of the walk operator e^{-iAt} for any t . What is more, the eigenvalues of Θ are ± 1 . As a result, if $|\lambda\rangle$ is an eigenvalue of e^{-iAt} , then

$$\langle \lambda | \text{ENTRANCE} \rangle = \pm \langle \lambda | \text{EXIT} \rangle \quad (79)$$

This can be applied to [Eq. 77](#), yielding

$$\Pr(\text{ENTRANCE}, \text{EXIT}, \infty) = \sum_{\lambda} |\langle \text{ENTRANCE} | \lambda \rangle|^4 \quad (80)$$

$$\geq \frac{1}{2d+2} \sum_{\lambda} |\langle \text{ENTRANCE} | \lambda \rangle|^2 \quad (81)$$

$$= \frac{1}{2d+2} \quad (82)$$

Now, we need to determine how quickly $\Pr(\text{ENTRANCE}, \text{EXIT}, T)$ approaches $\Pr(\text{ENTRANCE}, \text{EXIT}, \infty)$ as we increase T :

$$|\Pr(\text{ENTRANCE}, \text{EXIT}, T) - \Pr(\text{ENTRANCE}, \text{EXIT}, \infty)| \quad (83)$$

$$= \left| \sum_{\lambda \neq \lambda'} \langle \text{EXIT} | \lambda \rangle \langle \lambda | \text{ENTRANCE} \rangle \langle \text{EXIT} | \lambda' \rangle \langle \lambda' | \text{ENTRANCE} \rangle \frac{1 - e^{-i(\lambda-\lambda')T}}{i(\lambda - \lambda')T} \right| \quad (84)$$

$$\leq \sum_{\lambda \neq \lambda'} |\langle \lambda | \text{ENTRANCE} \rangle|^2 |\langle \lambda' | \text{ENTRANCE} \rangle|^2 \frac{1 - e^{-i(\lambda-\lambda')T}}{i(\lambda - \lambda')T} \quad (85)$$

$$\leq \frac{2}{T\delta} \quad (86)$$

where δ is the difference between the smallest gap between any two distinct eigenvalues of A . As a result, we get

$$\Pr(\text{ENTRANCE}, \text{EXIT}, T) \geq \frac{1}{2d-1} - \frac{2}{T\delta} \quad (87)$$

Childs et al. (2003) show that δ is $\Omega(1/d^3)$. Therefore, if we take T of size $O(d^4)$, we get success probability $O(1/d)$. Repeating this process, we can achieve an arbitrarily high probability of success in time polynomial in d – an exponential speedup over the classical random walk.

4.2.3 AND-OR Tree Evaluation

AND-OR trees arise naturally when evaluating the value of a two player combinatorial game. The players will be called P_0 and P_1 . The positions in this game are represented by nodes on a tree. The game begins at the root, and players alternate, moving from the root toward the leaves of the tree. For simplicity, we assume that a binary tree is being dealt with; that is, for each move, the players have exactly two moves. While the algorithm can be generalized to any approximately balanced tree, but the binary case is considered for simplicity. We also assume that every game lasts some fixed number of turns d , where a turn consists of one player making a move. The total number of leaf nodes is denoted by $n = 2^d$. We can label the leaf nodes according to which player wins if the game reaches each node; they are labeled with a 0 if P_0 wins, and a 1 if P_1 wins. We can then label each node in the graph by considering its children. If a node corresponds to P_0 's turn, we take the AND of the children. For P_1 's move, we take the OR of the children. The value at the root node tells which player has a winning strategy, assuming perfect play.

Now, since

$$\text{AND}(x_1, \dots, x_k) = \text{NAND}(\text{NAND}(x_1, \dots, x_k)) \quad (88)$$

$$\text{OR}(x_1, \dots, x_k) = \text{NAND}(\text{NAND}(x_1), \dots, \text{NAND}(x_k)) \quad (89)$$

we can rewrite the AND-OR tree using only NAND operations. Furthermore, since

$$\text{NOT}(x_1) = \text{NAND}(x_1) \quad (90)$$

rather than label leaves with a 1, we can insert an extra node below the leaf and label it 0. Also, consecutive NAND operations cancel out, and will be removed. This is illustrated in [Fig. 5](#). Note that the top-most node will be omitted from now on, and the shaded node will be referred to as the ROOT node. Now, the entire AND-OR tree is encoded in the structure of the NAND tree. We will write $\text{NAND}(v)$ to denote the NAND of the value obtained by evaluating the tree up to a vertex v . The value of the tree is therefore $\text{NAND}(\text{ROOT})$.

The idea of the algorithm is to use the adjacency matrix of the NAND tree as the Hamiltonian for a continuous time quantum walk. It will be shown that the eigenspectrum of this walk operator is related to the value of $\text{NAND}(\text{ROOT})$. This idea first appeared in a paper of Farhi et al. (2007), and was later refined by Ambainis et al. (2007). We will begin with the following lemma:

Lemma 1 *Let $|\lambda\rangle$ be an eigenvector of A with value λ . Then, if v is a node in the NAND tree with parent node p and children C ,*

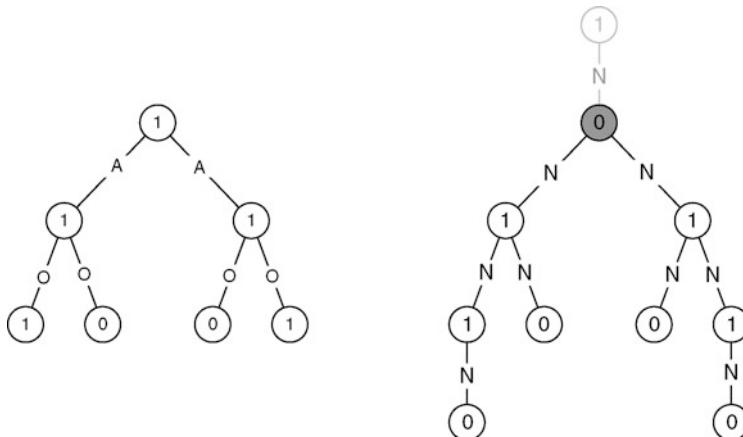
$$\langle p|\lambda\rangle = -\lambda\langle v|\lambda\rangle + \sum_{c \in C} \langle c|\lambda\rangle \quad (91)$$

Therefore, if A has an eigenvector with value 0, then

$$\langle p|\lambda_0\rangle = \sum_{c \in C} \langle c|\lambda_0\rangle \quad (92)$$

Fig. 5

An AND-OR tree and the equivalent NAND tree, with example values. Note that the top-most node will be omitted from now on, and the shaded node in the NAND tree will be referred to as the **ROOT node**.



Using this fact and some inductive arguments, Ambainis et al. (2007) prove the following theorem:

Theorem 4 *If $\text{NAND}(\text{ROOT}) = 0$, then there exists an eigenvector $|\lambda_0\rangle$ of A with eigenvalue 0 such that $|\langle \text{ROOT} | \lambda_0 \rangle| \geq \frac{1}{\sqrt{2}}$. Otherwise, if $\text{NAND}(\text{ROOT}) = 1$, then for any eigenvector $|\lambda\rangle$ of A with $\langle \lambda | A | \lambda \rangle < \frac{1}{2\sqrt{n}}$, we have $\langle \text{ROOT} | \lambda_0 \rangle = 0$.*

This result immediately leads to an algorithm. We perform phase estimation with precision $O(1/\sqrt{n})$ on the quantum walk, beginning in the state $|\text{ROOT}\rangle$. If $\text{NAND}(\text{ROOT}) = 0$, get a phase of 0 with probability $\geq \frac{1}{2}$. If $\text{NAND}(\text{ROOT}) = 1$, we will never get a phase of 0. This gives a running time of $O(\sqrt{n})$. While attention has been restricted to binary trees here, this result can be generalized to any m -ary tree.

It is worth noting that unstructured search is equivalent to taking the OR of n variables. This is an AND-OR tree of depth 1, and n leaves. The $O(\sqrt{n})$ running time of Grover's algorithm corresponds to the running time for the quantum walk algorithm.

Classically, the running time depends not just on the number of leaves, but on the structure of the tree. If it is a balanced m -ary tree of depth d , then the running time is

$$O\left(\left(\frac{(m-1 + \sqrt{m^2 + 14m + 1})}{4}\right)^d\right) \quad (93)$$

In fact, the quantum speedup is maximal when we have an n -ary tree of depth 1. This is just unstructured search, and requires $\Omega(n)$ time classically.

Reichardt and Spalek (2008) generalize the AND-OR tree problem to the evaluation of a broader class of logical formulas. Their approach uses *span programs*. A span program P consists of a set of target vector t , and a set of input vectors $\{v_1^0, v_1^1, \dots, v_n^0, v_n^1\}$ corresponding

to logical literals $\{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$. The program corresponds to a Boolean function $f_P: \{0, 1\}^n \rightarrow \{0, 1\}$ such that, for $\sigma \in \{0, 1\}^n$, $f(\sigma) = 1$ if and only if (Karchmer and Wigderson 1993)

$$\sum_{j=1}^n v_j^{\sigma_j} = t$$

Reichardt and Spalek outline the connection between span programs and the evaluation of logical formulas. They show that finding $\sigma \in f^{-1}(1)$ is equivalent to finding a zero eigenvector for a graph G_P corresponding to the span program P . In this sense, the span program approach is similar to the quantum walk approach of Childs et al. – both methods evaluate a formula by finding a zero eigenvector of a corresponding graph.

5 Tensor Networks and Their Applications

A tensor network consists of an underlying graph G , with an algebraic object called a *tensor* assigned to each vertex of G . The value of the tensor network is calculated by performing a series of operations on the associated tensors. The nature of these operations is dictated by the structure of G . At their simplest, tensor networks capture basic algebraic operations, such as matrix multiplication and the scalar product of vectors. However, their underlying graph structure makes them powerful tools for describing combinatorial problems as well. Two such examples will be explored – the Tutte polynomial of a planar graph, and the partition function of a statistical mechanical model defined on a graph. As a result, the approximation algorithm that will be described below for the value of a tensor network is implicitly an algorithm for approximating the Tutte polynomial, as well as the partition function for these statistical mechanics models. We begin by defining the notion of a tensor. We then outline how these tensors and tensor operations are associated with an underlying graph structure. For a more detailed account of this algorithm, the reader is referred to Arad and Landau (2008). Finally, we will describe the quantum approximation algorithm for the value of a tensor network, as well as the applications mentioned above.

5.1 Tensors: Basic Definitions

Tensors are formally defined as follows:

Definition 3 A tensor M of rank m and dimension q is an element of \mathbf{C}^{q^m} . Its entries are denoted by M_{j_1, j_2, \dots, j_m} , where $0 \leq j_k \leq q - 1$ for all j_k .

Based on this definition, a vector is simply a tensor of rank 1, while a square matrix is a tensor of rank 2. Several operations on tensors, which will generalize many familiar operations from linear algebra, will be now defined.

Definition 4 Let M and N be two tensors of dimension q and rank m and n , respectively. Then, their product, denoted $M \otimes N$, is a rank $m + n$ tensor with entries

$$(M \otimes N)_{j_1, \dots, j_m, k_1, \dots, k_n} = M_{j_1, \dots, j_m} \cdot N_{k_1, \dots, k_n} \quad (94)$$

This operation is simply the familiar tensor product. While the way that the entries are indexed is different, the resulting entries are the same.

Definition 5 Let M be a tensor of rank m and dimension q . Now, take a and b with $1 \leq a < b \leq m$. The contraction of M with respect to a and b is a rank $m - 2$ tensor N defined as follows:

$$N_{j_1, \dots, j_{a-1}, j_a+1, \dots, j_{b-1}, j_b+1, \dots, j_m} = \sum_{k=0}^{q-1} M_{j_1, \dots, j_{a-1}, k, j_{a+1}, \dots, j_{b-1}, k, j_{b+1}, \dots, j_m} \quad (95)$$

One way of describing this operation is that each entry in the contracted tensor is given by summing along the “diagonal” defined by a and b . This operation generalizes the partial trace of a density operator. The density operator of two-qubit system can be thought of as a rank 4 tensor of dimension two. Tracing out the second qubit is then just taking a contraction with respect to 3 and 4. It is also useful to consider the combination of these two operations.

Definition 6 If M and N are two tensors of dimension q and rank m and n , then for a and b with $1 \leq a \leq m$ and $1 \leq b \leq n$, the contraction of M and N is the result of contracting the product $M \otimes N$ with respect to a and $m + b$.

We now have the tools to describe a number of familiar operations in terms of tensor operations. For example, the inner product of two vectors can be expressed as the contraction of two rank 1 tensors. Matrix multiplication is just the contraction of 2 rank 2 tensors M and N with respect to the second index of M and the first index of N . Finally, if a Hilbert space $H = \mathbb{C}^q$ is taken, then we can identify a tensor M of dimension q and rank m with a linear operator $M^{s,t} : H^{\otimes t} \rightarrow H^{\otimes s}$ where $s + t = m$:

$$M^{s,t} = \sum_{j_1, \dots, j_m} M_{j_1, \dots, j_m} |j_1\rangle \otimes \dots \otimes |j_s\rangle \langle j_{s+1}| \otimes \dots \otimes \langle j_m| \quad (96)$$

This correspondence with linear operators is essential to understanding tensor networks and their evaluation.

5.2 The Tensor Network Representation

A tensor network $T(G, \mathbf{M})$ consists of a graph $G = (V, E)$ and a set of tensors $\mathbf{M} = \{M[v] : v \in V\}$. A tensor is assigned to each vertex, and the structure of the graph G encodes the operations to be performed on the tensors. It can be said that the *value* of a tensor network is the tensor that results from applying the operations encoded by G to the set of tensors \mathbf{M} . When the context is clear, we will let $T(G, \mathbf{M})$ denote the value of the network. In addition to the typical features of a graph, G is allowed to have *free edges* – edges that are incident with a vertex on one end, but are unattached at the other. For simplicity, it will be assumed that all of the tensors in \mathbf{M} have the same dimension q . It is also required that $\deg(v)$, the degree of vertex v , is equal to the rank of the associated tensor $M[v]$. If G consists of a single vertex v with m free edges, then $T(G, \mathbf{M})$ represents a single tensor of rank m . Each index of M is associated with a particular edge. It will often be convenient to refer to the edge associated with an index a as e_a . A set of rules that will

allow us to construct a tensor network corresponding to any sequence of tensor operations will be defined now:

- If M and N are the value of two tensor networks $T(G, \mathbf{M})$ and $T(H, \mathbf{N})$, then taking the product $M \otimes N$ is equivalent to taking the disjoint union of the two tensor networks, $T(G \cup H, \mathbf{M} \cup \mathbf{N})$.
- If M is the value of a tensor network $T(G, \mathbf{M})$, then contracting M with respect to a and b is equivalent to joining the free edges e_a and e_b in G .
- As a result, taking the contraction of M and N with respect to a and b corresponds to joining the associated edge e_a from $T(G, \mathbf{M})$ and e_b from $T(H, \mathbf{N})$.

Some examples are illustrated in Fig. 6. Applying these simple operations iteratively allows one to construct a network corresponding to any series of products and contractions of tensors. Note that the number of free edges in $T(G, \mathbf{M})$ is always equal to the rank of the corresponding tensor M . Tensor networks that have no free edges, and whose value is therefore a scalar, will be of particular interest.

The tensor network interpretation of a linear operator $M^{s,t}$ defined in Eq. 96 can now be considered. $M^{s,t}$ is associated with a rank $m = s + t$ tensor M . An equivalent tensor network could consist of a single vertex with m free edges (e_1, \dots, e_m) . The operator $M^{s,t}$ acts on elements of $(\mathbb{C}^q)^{\otimes t}$, which are rank t tensors, and can therefore be represented by a single vertex with t free edges (e'_1, \dots, e'_t) . The action of $M^{s,t}$ on an element $N \in (\mathbb{C}^q)^{\otimes t}$ is therefore represented by connecting e'_k to e_{s+k} for $k = 1, \dots, t$. Fig. 7 illustrates the operator $M^{2,2}$ acting on a rank 2 tensor. Note that the resulting network has two free edges, corresponding to the rank of the resulting tensor.

Fig. 6

The network representation of a tensor M of rank 4, and the contraction of two rank 4 tensors, M and N .

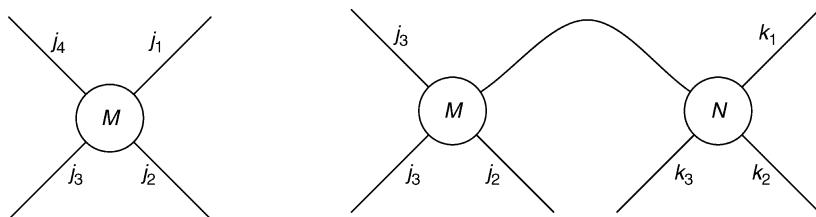
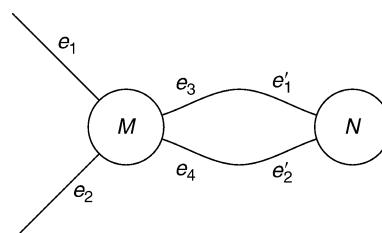


Fig. 7

The network corresponding to the operator $M^{2,2}$ acting on a rank 2 tensor N .



Now, the tensor networks that are of most interest – those with no free edges – will be reconsidered. We first consider the example of an inner product of two rank 1 tensors. This corresponds to a graph consisting of two vertices u and v joined by a single edge. The value of the tensor is then

$$T(G, \mathbf{M}) = \sum_{j=0}^q (M[u])_j \cdot (M[v])_j \quad (97)$$

That is, it is a sum of q terms, each of which corresponds to the assignment of an element of $\{0, \dots, q-1\}$ to the edge (u, v) . We can refer to this assignment as an q -edge coloring of the graph. In the same way, the value of a more complex tensor network is given by a sum whose terms correspond to q -edge colorings of G . Given a coloring of G , let $M[v]^\gamma = (M[v])_{i_1, \dots, i_m}$ where i_1, \dots, i_m are the values assigned by γ to the edges incident at v . That is, a q -edge coloring specifies a particular entry of each tensor in the network. Then, we can rewrite the value of $T(G, \mathbf{M})$ as follows:

$$T(G, \mathbf{M}) = \sum_{\gamma} \left(\prod_{v \in V} M[v]^\gamma \right) \quad (98)$$

where γ runs over all q -edge colorings of G . Thinking of the value of a tensor network as a sum over all q -edge colorings of G will prove useful when we consider applications of tensor networks to statistical mechanics models.

5.3 Evaluating Tensor Networks: the Algorithm

In this section, we will first show how to interpret a tensor network as a series of linear operators. We will then show how to apply these operators using a quantum circuit. Finally, we will see how to approximate the value of the tensor network using the Hadamard test.

5.3.1 Overview

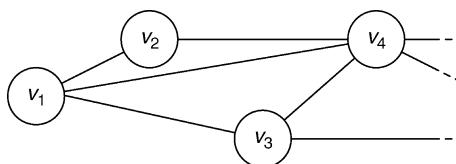
In order to evaluate a tensor network $T(G, \mathbf{M})$, some structure is first given to it, by imposing an ordering on the vertices of G ,

$$V(G) = \{v_1, \dots, v_n\}$$

The sets $S_j = \{v_1, \dots, v_j\}$ are further defined. This gives a natural way to depict the tensor network, which is shown in Fig. 8. The evaluation algorithm will “move” from left to right, applying linear operators corresponding to the tensors M_{v_i} . One can think of the edges in

Fig. 8

Part of the graph G , with an ordering on $V(G)$.



analogy to the “wires” of a quantum circuit diagram. To make this more precise, let v_i be a vertex of degree d . Let t be the number of vertices connecting v_i to S_{i-1} and s be the number of edges connecting v_i to $V(G) \setminus S_i$. Then, the operator that is applied at v_i is $M_{v_i}^{s,t}$. Letting j be the number of edges from S_{i-1} to $V(G) \setminus S_i$, we apply the identity operator I_j on the j indices that correspond to edges not incident at v_i . Combining these, we get an operator

$$N_i = M_{v_i}^{s,t} \otimes I_j \quad (99)$$

Note that, taking the product of these operators, $\prod_{i=1}^n N_i$, gives the value of $T(G, \mathbf{M})$. However, there are a few significant details remaining. Most notably, the operators N_i are not, in general, unitary. In fact, they may not even correspond to square matrices. A method for converting the N_i into equivalent unitary operators will be now outlined.

5.3.2 Creating Unitary Operators

The first task is to convert the operators N_i into “square” operators – that is, operators whose domain and range have the same dimension. Referring to [Eq. 99](#), the identity I_j is already square, so we need only modify $M_{v_i}^{s,t}$. In order to do this, some extra qubits will be added in the $|0\rangle$ state. We consider three cases:

1. If $s = t$, we define $\widetilde{M}_{v_i}^{s,t} = M_{v_i}^{s,t}$
2. If $s > t$, then we add $s - t$ extra qubits in the $|0\rangle$ state, and define $\widetilde{M}_{v_i}^{s,t}$ such that

$$\widetilde{M}_{v_i}^{s,t}(|\psi\rangle \otimes |0\rangle^{\otimes(s-t)}) = M_{v_i}^{s,t}|\psi\rangle \quad (100)$$

For completeness, we say that if the last $s - t$ qubits are not in the $|0\rangle$ state, $\widetilde{M}_{v_i}^{s,t}$ takes them to the 0 vector.

3. If $s < t$, then we define

$$\widetilde{M}_{v_i}^{s,t}|\psi\rangle = (M_{v_i}^{s,t}|\psi\rangle) \otimes |0\rangle^{\otimes(t-s)} \quad (101)$$

Finally, we define

$$\widetilde{N}_i = \widetilde{M}_{v_i}^{s,t} \otimes I_j \quad (102)$$

which is a square operator. Now, to derive corresponding unitary operators, we make use of the following lemma:

Lemma 2 *Let M be a linear map $A: H^{\otimes t} \rightarrow H^{\otimes t}$, where H is a Hilbert space $H = \mathbb{C}^q$. Furthermore, let $G = \mathbb{C}^2$ be a spanned by $\{|0\rangle, |1\rangle\}$. Then, there exists a unitary operator $U_M: (H^{\otimes t} \otimes G) \rightarrow (H^{\otimes t} \otimes G)$ such that*

$$U_M(|\psi\rangle \otimes |0\rangle) = \frac{1}{\|M\|} M|\psi\rangle \otimes |0\rangle + |\phi\rangle \otimes |1\rangle \quad (103)$$

U_M can be implemented on a quantum computer in $\text{poly}(q^t)$ time.

A proof can be found in Arad and Landau ([2008](#)) as well as Aharonov et al. ([2007](#)). Applying this lemma, we can create n unitary operators $U_{\widetilde{N}_j}$ for $1 \leq j \leq n$, and define

$$U = \prod_{j=1}^n U_{\widetilde{N}_j} \quad (104)$$

It is easily verified that

$$\langle 0|^{\otimes r} U | 0 \rangle^{\otimes r} = \frac{T(G, \mathbf{M})}{\prod_j \| M_{v_j}^{s,t} \|} \quad (105)$$

where r is dependent on the number of vertices n as well as the structure of G and the ordering of the vertices v_1, \dots, v_n .

5.3.3 The Hadamard Test

In order to approximate $\langle 0|^{\otimes r} U | 0 \rangle^{\otimes r}$, most authors suggest the *Hadamard test* – a well-known method for approximating the weighted trace of a unitary operator. First, we add an ancillary qubit that will act as a control register. We then apply the circuit outlined below, and measure the ancillary qubit in the computational basis.

It is not difficult to show that we measure

$$\begin{aligned} |0\rangle &\text{ with probability } \frac{1}{2}(1 + \operatorname{Re}\langle \psi | U | \psi \rangle) \\ |1\rangle &\text{ with probability } \frac{1}{2}(1 - \operatorname{Re}\langle \psi | U | \psi \rangle) \end{aligned}$$

So, if one assigns a random variable X such that $X = 1$ when one measures $|0\rangle$ and $X = -1$ when one measures $|1\rangle$, then X has expected value $\operatorname{Re}\langle \psi | U | \psi \rangle$. So, in order to approximate $\operatorname{Re}\langle \psi | U | \psi \rangle$ to a precision of ε with constant probability $p > 1/2$, one requires $O(\varepsilon^{-2})$ repetitions of this protocol. In order to calculate the imaginary portion, we apply the gate

$$R = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$$

to the ancillary qubit, right after the first Hadamard gate. It is important to note that the Hadamard test gives an *additive approximation* of $\langle 0|^{\otimes r} U | 0 \rangle^{\otimes r}$. Additive approximations will be examined in [Sect. 5.3.5](#).

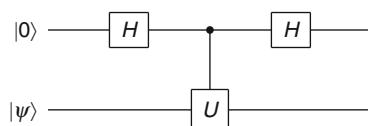
5.3.4 Approximating $\langle 0|^{\otimes r} U | 0 \rangle^{\otimes r}$ Using Amplitude Estimation

In order to estimate $\langle 0|^{\otimes r} U | 0 \rangle^{\otimes r}$, most of the literature applies the Hadamard test ([Fig. 9](#)). However, it is worth noting that we can improve the running time by using the process of *amplitude estimation*, as outlined in [Sect. 3](#). Using the notation from [Sect. 3](#), we have $U_f = U_0$ and $A = U$, the unitary corresponding to the tensor network $T(G, \mathbf{M})$. We begin in the state $U|0\rangle^{\otimes r}$, and the search iterate is

$$Q = -UU_0U^{-1}U_0 \quad (106)$$

Fig. 9

The quantum circuit for the Hadamard test.



In order to approximate $\langle 0 |^{\otimes r} U | 0 \rangle^{\otimes r}$ to a precision of ε , the running time is in $O(1/\varepsilon)$, a quadratic improvement over the Hadamard test.

5.3.5 Additive Approximation

Let $f: S \rightarrow \mathbb{C}$ be a function that we would like to evaluate. Then, an *additive approximation* A with approximation scale $\Delta: S \rightarrow \mathbb{R}$ is an algorithm that, on input $x \in S$ and parameter $\varepsilon > 0$, outputs $A(x)$ such that

$$\Pr(|A(x) - f(x)| \geq \varepsilon \Delta(x)) \leq c \quad (107)$$

for some constant c with $0 \leq c < 1/2$. If $\Delta(x)$ is $O(f(x))$, then A is a *fully polynomial randomized approximation scheme*, or FPRAS.

The approximation scale for the algorithm outlined above will be now determined. Using amplitude estimation, we estimate

$$\langle 0 |^{\otimes r} U | 0 \rangle^{\otimes r} = \frac{T(G, \mathbf{M})}{\prod_j \|M_{v_j}^{s,t}\|} \quad (108)$$

to within ε with time requirement in $O(1/\varepsilon)$. However, the quantity we actually want to evaluate is $T(G, \mathbf{M})$, so we must multiply by $\prod_j \|M_{v_j}^{s,t}\|$. Therefore, the algorithm has approximation scale

$$\Delta(G, \mathbf{M}) = \prod_j \|M_{v_j}^{s,t}\| \quad (109)$$

We apply [Lemma 2](#) to each vertex in G , and require $O(1/\varepsilon)$ repetitions of the algorithm to approximate $T(G, \mathbf{M})$ with the desired accuracy using amplitude estimation. This gives an overall running time

$$O\left(\text{poly}(q^{D(G)}) \cdot |V(G)| \cdot \varepsilon^{-1}\right) \quad (110)$$

where $D(G)$ is the maximum degree of any vertex in G . The algorithm is, therefore, an additive approximation of $T(G, \mathbf{M})$.

5.4 Approximating the Tutte Polynomial for Planar Graphs Using Tensor Networks

In 2000, Kitaev, Freedman, Larsen, and Wang (Freedman et al. 2000, 2001) demonstrated an efficient quantum simulation for topological quantum field theories. In doing so, they implied that the Jones polynomial can be efficiently approximated at $e^{2\pi i/5}$. This suggests that efficient quantum approximation algorithms might exist for a wider range of knot invariants and values. Aharonov et al. (2008) developed such a Jones polynomial for any complex value. Yard and Wocjan (2006) developed a related algorithm for approximating the HOMFLYPT polynomial of a braid closure. Both of these knot invariants are special cases of the Tutte polynomial. While the tensor network algorithm in [Sect. 3.2](#) follows directly from a later paper of Aharonov et al. (2007), it owes a good deal to these earlier results for knot invariants.

A definition of the Tutte polynomial, as well as an overview of its relationship to tensor networks and the resulting approximation algorithm, will be given.

5.4.1 The Tutte Polynomial

The multivariate Tutte Polynomial is defined for a graph $G = (V, E)$ with edge weights $\mathbf{w} = \{w_e\}$ and variable q as follows:

$$Z_G(q, \mathbf{w}) = \sum_{A \subseteq E} q^{k(A)} \prod_{e \in A} w_e \quad (111)$$

where $k(A)$ denotes the number of connected components in the graph induced by A . The power of the Tutte polynomial arises from the fact that it captures nearly all functions on graphs defined by a *skein relation*. A skein relation is of the form

$$f(G) = x \cdot f(G/e) + y \cdot f(G \setminus e) \quad (112)$$

where G/e is created by contracting an edge e and $G \setminus e$ is created by deleting e . Oxley and Welsh (Welsh 1993) show that, with a few additional restrictions on f , if f is defined by a skein relation, then computing f can be reduced to computing Z_G . It turns out that many functions can be defined in terms of a skein relation, including

1. The partition functions of the Ising and Potts models from statistical physics
2. The chromatic and flow polynomials of a graph G
3. The Jones Polynomial of an alternating link

The exact evaluation of the Tutte polynomial, even when restricted to planar graphs, turns out to be $\#P$ -hard for all but a handful of values of q and \mathbf{w} . While FPRAS seems very unlikely for the Tutte polynomial of a general graph and any parameters q and \mathbf{w} . The interesting question seems to be *which* graphs and parameters admit an efficient and accurate approximation. By describing the Tutte polynomial as the evaluation of a Tensor network, an additive approximation algorithm for the Tutte polynomial is immediately produced. However, we do not have a complete characterization of the graphs and parameters for which this approximation is nontrivial.

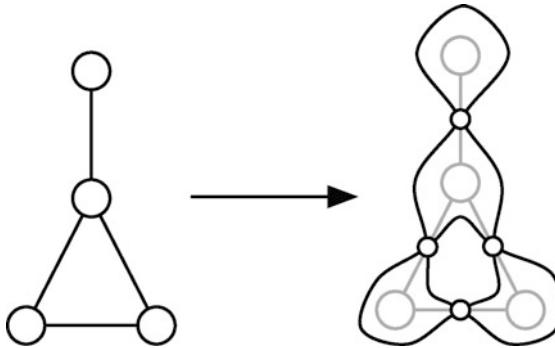
5.4.2 The Tutte Polynomial as a Tensor Network

Given a planar graph G , and an embedding of G in the plane, we define the *medial graph* L_G . The vertices of L_G are placed on the edges of G . Two vertices of L_G are joined by an edge if they are adjacent on the boundary of a face of G . If G contains vertices of degree 2, then L_G will have multiple edges between some pair of vertices. Also note that L_G is a regular graph with valency 4. An example of a graph and its associated medial graph is depicted in [Fig. 10](#).

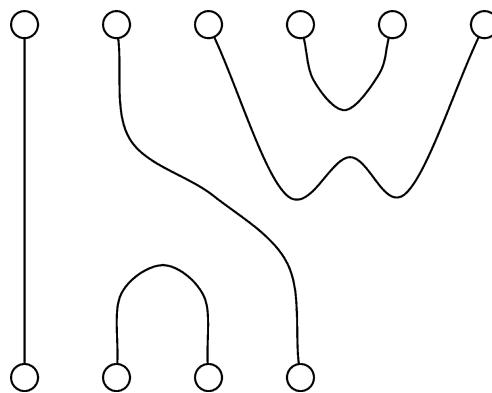
In order to describe the Tutte polynomial in terms of a tensor network, we make use of the *generalized Temperley–Lieb algebra*, as defined in Aharonov et al. (2007). The basis elements of the Temperley–Lieb algebra $GTL(d)$ can be thought of as diagrams in which m upper pegs are joined to n lower pegs by a series of strands. The diagram must contain no crossings or loops. See [Fig. 11](#) for an example of such an element. Two basis elements are considered equivalent if they are isotopic to each other or if one can be obtained from the other by “padding” it on the right with a series of vertical strands. The algebra consists of all complex weighted sums of these basis elements. Given two elements of the algebra T_1 and T_2 , we can take their product $T_2 \cdot T_1$ by simply placing T_2 on top of T_1 and joining the strands. Note that if the number of strands do not match, we can simply pad the appropriate element

Fig. 10

A graph G and the resulting medial graph L_G .

**Fig. 11**

An element of the Temperley–Lieb algebra with six upper pegs and four lower pegs.



with some number of vertical strands. As a consequence, the identity element consists of any number of vertical strands. In composing two elements in this way, a loop may be created. Let us say that $T_1 \cdot T_2$ contains one loop. Then, if T_3 is the element created by removing the loop, we define $T_1 \cdot T_2 = dT_3$, where d is the complex parameter of $GTL(d)$.

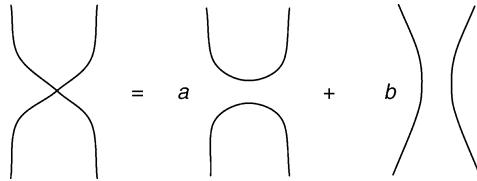
We would also like this algebra to accommodate drawings with crossings. Let T_1 be such a diagram, and T_2 and T_3 be the diagrams resulting from “opening” the crossing in the two ways indicated in Fig. 12. Then, we define $T_1 = aT_2 + bT_3$, for appropriately defined a and b .

If we think of L_G as the projection of a knot onto the plane, where the vertices of L_G are the crossings of the knot, we see that L_G can be expressed in terms of the generalized Temperley–Lieb algebra. We would like to take advantage of this in order to map L_G to a series of tensors that will allow us to approximate the Tutte polynomial of G . Aharonov et al. define such a representation ρ of the Temperley–Lieb algebra. If $T \in GTL(d)$ is a basis element with m lower pegs and n upper pegs, then ρ is a linear operator such that

$$\rho(T) : H^{\otimes m+1} \rightarrow H^{\otimes n+1} \quad (113)$$

Fig. 12

An element that contains a crossing is equated to a weighted sum of the elements obtained by “opening” the crossing.



where $H = \mathbf{C}^k$ for some k , which depends on the particular embedding of G , such that

1. ρ preserves multiplicative structure. That is, if T_1 has m upper pegs and T_2 has m lower pegs, then

$$\rho(T_2 \cdot T_1) = \rho(T_2) \cdot \rho(T_1) \quad (114)$$

2. ρ is linear. That is,

$$\rho(\alpha T_1 + \beta T_2) = \alpha \rho(T_1) + \beta \rho(T_2) \quad (115)$$

For these purposes, k can be upper bounded by the number of edges $|E(L_G)| = 2|E(G)|$, and corresponds to the value r in [Eq. 105](#). To represent L_G , each crossing (vertex) of L_G is associated with a weighted sum of two basis elements, and therefore is represented by a weighted sum of the corresponding linear operators, $\rho(T_1) = a\rho(T_2) + b\rho(T_3)$. The minima and maxima of L_G are also basis elements of $GTL(d)$, and can be represented as well. Finally, since L_G is a closed loop, and has no “loose ends”, $\rho(L_G)$ must be a scalar multiple of the identity operator on $H = \mathbf{C}^k$. This relates well to our notion that the value of a tensor network with no loose ends is just a scalar.

We would like this scalar to be the Tutte polynomial of the graph G . In order to do this, we need to choose the correct values for a , b , and d . Aharonov et al. show how to choose these values so that the scalar is a graph invariant called the Kauffman bracket, from which we can calculate the Tutte polynomial for planar graphs. So, we can construct a tensor network whose value is the Kauffman bracket of L_G , and this gives the Tutte polynomial of G by the following procedure:

1. Construct the medial graph L_G and embed it in the plane.
2. Let L'_G be constructed by adding a vertex at each local minimum and maximum of L_G . These vertices are required in order to assign the linear operator associated with each minimum/maximum with a vertex in the tensor network.
3. Each vertex v of L'_G has a Temperley–Lieb element T_v associated with it. For some vertices, this is a crossing; for others, it is a local minimum or maximum. Assign the tensor $M[v] = \rho(T_v)$ to the vertex v .

The tensor network we are interested in is then $T(L'_G, \mathbf{M})$, where $\mathbf{M} = \{M[v] : v \in V(L'_G)\}$. Approximating the value of this tensor network gives an approximation of the Kauffman bracket of L_G , and therefore of the Tutte polynomial of G . Furthermore, it is known that L'_G has maximum degree 4. Finally, we will assume that $|E(G)| \geq |V(G)|$ giving a running time of

$$O(\text{poly}(q^4) \cdot |E(G)| \cdot \varepsilon^{-1}) \quad (116)$$

In this case, q depends on the particular embedding $f: L_G$, but can be upper bounded by $|E|$. For more details on the representation ρ and quantum approximations of the Tutte polynomial as well as the related Jones polynomial, the reader is referred to the work of Aharonov et al., in particular Aharonov and Arad (2006) and Aharonov et al. (2007).

5.5 Tensor Networks and Statistical Mechanics Models

Many models from statistical physics describe how simple local interactions between microscopic particles give rise to macroscopic behavior. See Beaudin et al. (2008) for an excellent introduction to the combinatorial aspects of these models. The models that we are concerned with here are described by a weighted graph $G = (V, E)$. In this graph, the vertices represent particles, while edges represent an interaction between particles. A configuration σ of a q -state model is an assignment of a value from the set $\{0, \dots, q - 1\}$ to each vertex of G . We denote the value assigned to v by σ_v . For each edge $e = (u, v)$, we define a local Hamiltonian $h_e(\sigma_u, \sigma_v) \in \mathbb{C}$. The Hamiltonian for the entire system is then given taking the sum:

$$H(\sigma) = \sum_{e=(u,v)} h_e(\sigma_u, \sigma_v) \quad (117)$$

The sum runs over all the edges of G . The *partition function* is then defined as

$$Z_G(\beta) = \sum_{\sigma} e^{-\beta H(\sigma)} \quad (118)$$

where $\beta = 1/kT$ is referred to as the *inverse temperature* and k is Boltzmann's constant. The partition function is critical to understanding the behavior of the system. First, the partition function allows one to determine the probability of finding the system in a particular configuration, given an inverse temperature β :

$$\Pr(\sigma') = \frac{e^{-\beta H(\sigma')}}{Z_G(\beta)} \quad (119)$$

This probability distribution is known as the Boltzmann distribution. Calculating the partition function also allows us to derive other properties of the system, such as entropy and free energy. An excellent discussion of the partition function from a combinatorial perspective can be found in Welsh (1993).

A tensor network whose value is the partition function Z_G will be now constructed. Given the graph G , we define the vertices of $G' = (V', E')$ as follows:

$$V' = V \cup \{v_e : e \in E\} \quad (120)$$

We are simply adding a vertex for each edge e of G , and identifying it by v_e . We then define the edge set of G' :

$$E' = \{(x, v_e), (v_e, y) : (x, y) \in E\} \quad (121)$$

So, the end product G' resembles the original graph; vertices in the middle of each edge of the original graph have been simply added. The tensors that will be identified with each vertex will be defined separately for the vertex set V of the original graph and the set $V_E = \{v_e\}$ of vertices that are added to define G' . In each case, they will be dimension q tensors for a q -state model. For $v \in V$, $M[v]$ is an “identity” operator; that is, it takes on the value 1 when all indices are

equal, and 0 otherwise:

$$(M[v])_{i_1, \dots, i_m} = \begin{cases} 1 & \text{if } i_1 = i_2 = \dots = i_m \\ 0 & \text{otherwise} \end{cases} \quad (122)$$

The vertices in V_E encode the actual interactions between neighboring particles. The tensor associated with v_e is

$$(M_{v_e})_{s,t} = e^{-\beta h_e(s,t)} \quad (123)$$

Now, one can consider the value of the tensor network in terms of q -edge colorings of G :

$$T(G', \mathbf{M}) = \sum_{\gamma} \left(\prod_{v \in V'} M[v]^{\gamma} \right) \quad (124)$$

where γ runs over all q -edge colorings of G' . Based on the definitions of the tensors $M[v]$ for $v \in V$, we see that $\prod_{v \in V'} M[v]^{\gamma}$ is nonzero only when the edges incident at each vertex are all colored identically. This restriction ensures that each nonzero term of the sum (❷ Eq. 124) corresponds to a configuration σ of the q -state model. That is, a configuration σ corresponds to a q -edge coloring γ of G' where each $\gamma(e) = \sigma_v$ whenever e is incident with $v \in V$. This gives the following equality:

$$\begin{aligned} T(G', \mathbf{M}) &= \sum_{\gamma} \left(\prod_{v \in V'} M[v]^{\gamma} \right) \\ &= \sum_{\sigma} \prod_{e=(u,v)} e^{-\beta h_e(\sigma_u, \sigma_v)} \\ &= \sum_{\sigma} e^{-\beta H(\sigma)} \\ &= Z_G(\beta) \end{aligned}$$

The time required to approximate the value of this tensor network will be now considered. Recall that the running time is given by

$$O\left(\text{poly}(q^{D(G')}) \cdot |V(G')| \cdot \varepsilon^{-1}\right) \quad (125)$$

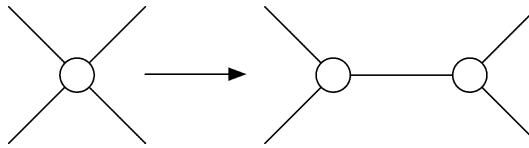
First, it is observed that, if one assumes that G is connected, then $|V(G')|$ is $O(|E|)$. Restrictions are not placed on the maximum degree $D(G')$. The vertices of G' of high degree must be in V , since all vertices in V_E have degree 2. Now, the tensors assigned to vertices of V are just identity operators; they ensure that the terms of the sum from ❷ Eq. 124 are nonzero only when all the edges incident at v are colored identically. As a result, one can replace each vertex $v \in V$ with $\deg(v) > 3$ by $(\deg(v) - 2)$ vertices, each of degree 3. See ❷ Fig. 13 for an example. The tensor assigned to each of these new vertices is the identity operator of rank 3. It is not difficult to show that this replacement does not affect the value of the tensor network. It does, however, affect the number of vertices in the graph, adding a multiplicative factor to the running time, which can now be written

$$O\left(\text{poly}(q) \cdot d(G) \cdot |E| \cdot \varepsilon^{-1}\right) \quad (126)$$

For more details on the approximation scale of this algorithm, the reader is referred to Arad and Landau (2008).

Fig. 13

Replacing a vertex v by $(\deg(v) - 2)$ vertices.



Arad and Landau also discuss a useful restriction of q -state models called *difference models*. In these models, the local Hamiltonians h_e depend only on the difference between the states of the vertices incident with e . That is, $h_e(\sigma_u, \sigma_v)$ is replaced by $h_e(|\sigma_u - \sigma_v|)$, where $|\sigma_u - \sigma_v|$ is calculated modulo q . Difference models include the well-known Potts, Ising and Clock models. Arad and Landau show that the approximation scale can be improved when attention is restricted to difference models.

Van den Nest et al. (Cuevas et al. 2008; Hübener et al. 2008) show that the partition function for difference models can be described as the overlap between two quantum states

$$Z_G = \langle \psi_G | \left(\bigotimes_{e \in E(G)} |\alpha_e\rangle \right) \quad (127)$$

Van den Nest uses the state $|\psi_G\rangle$ to encode the structure of the graph, while each state $|\alpha_e\rangle$ encodes the strength of the local interaction at e . The actual computation of this overlap is best described as a tensor network. While this description does not yield a computational speedup over the direct method described above, it is an instructive way to deconstruct the problem. In the case of planar graphs, it also relates the partition function of a graph G and its planar dual.

Geraci and Lidar (Geraci 2008; Geraci and Lidar 2008) show that the Potts partition may be efficiently and exactly evaluated for a class of graphs related to irreducible cyclic cocycle codes. While their algorithm does not employ tensor networks, it is interesting to consider the implications of their results in the context of tensor networks. Is there a class of tensor networks whose value can be efficiently and exactly calculated using similar techniques?

6 Conclusion

In this chapter, we reviewed quantum algorithms of several types: those based on the quantum Fourier transform, amplitude amplification, quantum walks, and evaluating tensor networks. This is by no means a complete survey of quantum algorithms; most notably absent are algorithms for simulating quantum systems. This is a particularly natural application of quantum computers, and these algorithms often achieve an exponential speedup over their classical counterparts.

Algorithms based on the quantum Fourier transform, as reviewed in [Sect. 2](#), include some of the earliest quantum algorithms that give a demonstrable speedup over classical algorithms, such as the algorithms for Deutsch's problem and Simon's problem. Many problems in this area can be described as hidden subgroup problems. Although problems of this type have been well-studied, many questions remain open. For example, there is no efficient quantum algorithm known for most examples of non-Abelian groups. In particular, the quantum complexity of the graph isomorphism problem remains unknown.

Similarly, the family of algorithms based on amplitude amplification and estimation has a relatively long history. While Grover's original searching algorithm is the best-known example, this approach has been greatly generalized. For example, in Brassard et al. (2002) it is shown how the same principles applied by Grover to searching can be used to perform amplitude estimation and quantum counting. Amplitude amplification and estimation have become ubiquitous tools in quantum information processing, and are often employed as critical subroutines in other quantum algorithms.

Quantum walks provide an interesting analogue to classical random walks. While they are inspired by classical walks, quantum walks have many properties that set them apart. Some applications of quantum walks are natural and somewhat unsurprising; using a walk to search for a marked element is an intuitive idea. Others, such as evaluating AND-OR trees are much more surprising. Quantum walks remain an active area of research, with many open questions. For example, the relative computational capabilities of discrete and continuous time quantum walks are still not fully understood.

The approximation algorithm for tensor networks as described in this chapter (Arad and Landau 2008) is relatively new. However, it is inspired by the earlier work of Aharonov et al. (2008), as well as Wocjan and Yard (2006). The tensor network framework allows us to capture a wide range of problems, particularly problems of a combinatorial nature. Indeed, many $\#P$ -hard problems, such as evaluating the Tutte polynomial, can be captured as tensor networks. The difficulty arises from the additive nature of the approximation. One of the most important questions in this area is when this approximation is useful, and when the size of the approximation window renders the approximation trivial.

Numerous other families of new quantum algorithms that are not elaborated in this chapter have also been briefly mentioned, and many others unfortunately go unmentioned. We tried to balance an overview of some of the classic techniques with a more in-depth treatment of some of the more recent and novel approaches to finding new quantum algorithms.

We expect and hope to see many more exciting developments in the coming years: novel applications of existing tools and techniques, new tools and techniques in the current paradigms, as well as new algorithmic paradigms.

References

- Aaronson S, Shi Y (2004) Quantum lower bounds for the collision and the element distinctness problems. J ACM 51(4):595–605. doi: <http://doi.acm.org/10.1145/1008731.1008735>
- Aharonov D, Ambainis A, Kempe J, Vazirani U (2001) Quantum walks on graphs. In: STOC'01: proceedings of the 33rd annual ACM symposium on theory of computing. ACM Press, New York, pp 50–59. doi: <http://doi.acm.org/10.1145/380752.380758>
- Aharonov D, Arad I (2006) The BQP-hardness of approximating the Jones polynomial. <http://arxiv.org/abs/quant-ph/0605181>
- Aharonov D, Arad I, Eban E, Landau Z (2007) Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane. <http://arxiv.org/abs/quant-ph/0702008>
- Aharonov D, Jones V, Landau Z (2008) A polynomial quantum algorithm for approximating the Jones polynomial. Algorithmica 55(3):395–421
- Ambainis A (2003) Quantum walks and their algorithmic applications. Int J Quantum Inform 1:507–518
- Ambainis A (2004) Quantum walk algorithm for element distinctness. In: Proceedings of the 45th annual IEEE symposium on foundations of computer science, pp 22–31. doi: 10.1109/FOCS.2004.54
- Ambainis A, Bach E, Nayak A, Vishwanath A, Watrous J (2001) One-dimensional quantum walks. In: STOC'01: proceedings of the 33rd annual ACM

- symposium on theory of computing. ACM Press, New York, pp 37–49. doi: <http://doi.acm.org/10.1145/380752.380757>
- Ambainis A, Childs A, Reichardt B, Spalek R, Zhang S (2007) Any and-or formula of size n can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer. In: Proceedings of the 48th annual IEEE symposium on foundations of computer science, pp 363–372. doi: 10.1109/FOCS.2007.57
- Arad I, Landau Z (2008) Quantum computation and the evaluation of tensor networks. <http://arxiv.org/abs/0805.0040>
- Beaudin L, Ellis-Monaghan J, Pangborn G, Shrock R (2008) A little statistical mechanics for the graph theorist. <http://arxiv.org/abs/0804.2468>
- Bernstein BK, Vazirani U (1997) Quantum complexity theory. *SIAM J Comput* 26:1411–1473
- Berry DW, Ahokas G, Cleve R, Sanders BC (2007) Efficient quantum algorithms for simulating sparse Hamiltonians. *Commun Math Phys* 270:359
- Boixo S, Knill E, Somma R (2009) Quantum state preparation by phase randomization. <http://arxiv.org/abs/0903.1652>
- Boneh D, Lipton R (1995) Quantum cryptanalysis of hidden linear functions (extended abstract). In: Proceedings of the 15th annual international cryptology conference on advances in cryptology. Lecture notes in computer science, vol. 963. Springer, London, UK, pp 424–437
- Boyer M, Brassard G, Høyer P, Tapp A (1998) Tight bounds on quantum searching. *Fortschritte der Physik* 56(5–5):493–505
- Brassard G, Høyer P (1997) An exact quantum polynomial-time algorithm for Simon’s problem. In: Proceedings of the fifth Israeli symposium on theory of computing and systems (ISTCS’97). IEEE Press, Piscataway, pp 12–23
- Brassard G, Høyer P, Mosca M, Tapp A (2002) Quantum amplitude amplification and estimation. *Quantum Computation & Information, AMS Contemporary Math Series* 305:53–74
- Brassard G, Høyer P, Tapp A (1997) Cryptology column — quantum algorithm for the collision problem. *ACM SIGACT News* 28:14–19
- Brouwer AE (1989) Distance-regular graphs. Springer, New York
- Childs A (2008) CO781 Topics in quantum information: quantum algorithms. Lecture notes on quantum algorithms. <http://www.math.uwaterloo.ca/~amchilds/teaching/w08/co781.html>
- Childs AM, Cleve R, Deotto E, Farhi E, Gutmann S, Spielman DA (2003) Exponential algorithmic speedup by a quantum walk. In: STOC ’03: proceedings of the 35th annual ACM symposium on theory of computing. ACM Press, New York, pp 59–68. doi: <http://doi.acm.org/10.1145/780542.780552>
- Childs A, van Dam W (2010) Quantum algorithms for algebraic problems. *Rev Mod Phys* 82(1):1–52
- Cleve R, Ekert A, Macchiavello C, Mosca M (1998) Quantum algorithms revisited. *Proc Roy Soc Lond A* 454:339–354
- D’Ariano GM, van Dam W, Ekert E, Macchiavello C, Mosca M (2007) General optimized schemes for phase estimation. *Phys Rev Lett* 98(9):090,501
- Das A, Chakrabarti BK (2008) Quantum annealing and analog quantum computation. *Rev Mod Phys* 80:1061
- De las Cuevas G, Dür W, Van den Nest M, Briegel HJ (2008) Completeness of classical spin models and universal quantum computation. <http://arxiv.org/abs/0812.2368>
- Deutsch D (1985) Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc Roy Soc Lond A* 400:97–117
- Deutsch D, Jozsa R (1992) Rapid solutions of problems by quantum computation. *Proc Roy Soc Lond, A* 439:553–558
- Farhi E, Goldstone J, Gutmann S (2007) A quantum algorithm for the Hamiltonian NAND tree. <http://arxiv.org/abs/quant-ph/0702144>
- Feynman R (1982) Simulating physics with computers. *Int J Theor Phys* 21(6,7):467–488
- Freedman MH, Kitaev A, Larsen MJ, Wang Z (2001) Topological quantum computation. <http://arxiv.org/abs/quant-ph/0101025>
- Freedman MH, Kitaev A, Wang Z (2000) Simulation of topological field theories by quantum computers. <http://arxiv.org/abs/quant-ph/0001071>
- Geraci J (2008) A BQP-complete problem related to the Ising model partition function via a new connection between quantum circuits and graphs. <http://arxiv.org/abs/0801.4833>
- Geraci J, Lidar DA (2008) On the exact evaluation of certain instances of the Potts partition function by quantum computers. *Commun Math Phys* 279(3): 735–768
- Grigoriev D (1997) Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. *Theor Comput Sci* 180:217–228
- Grover L (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th annual ACM symposium on the theory of computing (STOC ’96). ACM Press, New York, pp 212–219
- Grover L (1998) A framework for fast quantum mechanical algorithms. In: Proceedings of the 13th annual ACM symposium on theory of computing (STOC ’98). ACM Press, New York, pp 53–62
- Hirvensalo M (2001) Quantum computing. Series: Natural Computing Series. Springer
- Hübener R, Van den Nest M, Dür W, Briegel HJ (2008) Classical spin systems and the quantum stabilizer formalism: general mappings and applications. <http://arxiv.org/abs/0812.2127>

- Jordan S (2008) Quantum computation beyond the circuit model. PhD thesis, MIT University, Cambridge
- Karchmer M, Wigderson A (1993) On span programs. In: Proceedings of the 8th IEEE structures in complexity conference. IEEE Press, Piscataway, pp 102–111
- Kaye P, Laflamme R, Mosca M (2007) An introduction to quantum computation. Oxford University Press, Oxford, UK
- Kempe J (2003) Quantum random walks - an introductory overview. *Contemp Phys* 44(4):307–327
- Kitaev AY (1995) Quantum measurements and the Abelian stabilizer problem. <http://arxiv.org/abs/quant-ph/9511026>
- Kitaev A, Shen A, Vyalyi M (2002) Classical and quantum computation. American Mathematical Society, Providence, RI
- Magniez F, Nayak A, Roland J, Santha M (2007) Search via quantum walk. In: STOC '07: proceedings of the 39th annual ACM symposium on theory of computing. ACM, New York, pp 575–584. doi: <http://doi.acm.org/10.1145/1250790.1250874>
- Menezes A, van Oorschot P, Vanstone S (1996) Handbook of applied cryptography. CRC Press, Boca Raton
- Mermin ND (2007) Quantum computer science: an introduction. Cambridge University Press, Cambridge
- Mosca M (2001) Counting by quantum eigenvalue estimation. *Theor Comput Sci* 264:139–153
- Mosca M (2008) Abelian hidden subgroup problem. In: Kao M-Y (ed) Encyclopedia of algorithms. Springer, Berlin
- Mosca M (2009) Quantum algorithms. In: Meyers R (ed) Encyclopedia of complexity and systems science. Springer
- Nayak A, Vishwanath A (2000) Quantum walk on the line. <http://arxiv.org/abs/quant-ph/0010117>
- Nielsen M, Chuang I (2000) Quantum computation and quantum information. Cambridge University Press, Cambridge, UK
- Reichardt BW, Spalek R (2008) Span-program-based quantum algorithm for evaluating formulas. In: STOC '08: proceedings of the 40th annual ACM symposium on theory of computing. ACM Press, New York, pp 103–112. doi: <http://doi.acm.org/10.1145/1374376.1374394>
- Santha M (2008) Quantum walk based search algorithms. In: Agrawal M, Du D-Z, Duan Z, Li A (eds) Theory and applications of models of computation. Lecture notes in computer science, vol 4978. Springer, Berlin, Heidelberg, pp 31–46. doi: 10.1007/978-3-540-79228-4_3
- Shor P (1994) Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings of the 35th annual symposium on foundations of computer science. IEEE Computer Society, Washington, DC, pp 124–134
- Shor P (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput* 26:1484–1509
- Simon D (1994) On the power of quantum computation. In: Proceedings of the 35th IEEE symposium on the foundations of computer science (FOCS). IEEE Computer Society, Washington, DC, pp 116–123
- Szegedy M (2004) Quantum speed-up of Markov chain based algorithms. In: Proceedings of the 45th annual IEEE symposium on foundations of computer science (FOCS). IEEE Computer Society, Washington, DC, pp 32–41. doi: <http://dx.doi.org/10.1109/FOCS.2004.53>
- Tulsi T, Grover L, Patel A (2006) A new algorithm for fixed point quantum search. *Quant Inform Comput* 6(6):483–494
- Welsh D (1993) Complexity: knots, colourings and countings. Cambridge University Press, Cambridge, UK
- Wiebe N, Berry DW, Høyer P, Sanders BC (2008) Higher order decompositions of ordered operator exponentials. <http://arxiv.org/abs/0812.0562>
- Wocjan P, Yard J (2006) The Jones polynomial: quantum algorithms and applications in quantum complexity theory. <http://arxiv.org/abs/quant-ph/0603069>

44 Physical Implementation of Large-Scale Quantum Computation

Kalle-Antti Suominen

Department of Physics and Astronomy, University of Turku, Finland
kalle-antti.suominen@utu.fi

1	<i>Introduction</i>	1494
2	<i>DiVincenzo Criteria</i>	1495
3	<i>Elements of Quantum Computation</i>	1496
4	<i>Cavity QED</i>	1503
5	<i>Physical Realization with Electromagnetically Trapped Particles</i>	1504
6	<i>Nuclear Spin Computers</i>	1507
7	<i>Solid-State Qubits with Quantum Dots</i>	1512
8	<i>Superconducting Qubits</i>	1513
9	<i>All-Optical Quantum Computing</i>	1516
10	<i>Closing Remarks</i>	1517

Abstract

The development of large-scale quantum computing started rapidly in 1994 after the presentation of the factoring algorithm by Peter Shor. In this review, the basic requirements for the successful implementation of quantum algorithms on physical systems are first discussed and then a few basic concepts in actual information processing are presented. After that, the current situation is evaluated, concentrating on the most promising methods for which actual experimental progress has taken place. Among these are trapped ions, nuclear spins, and various solid-state structures such as quantum dots and Josephson junctions.

1 Introduction

The term “quantum computing” was perhaps first expressed by Richard Feynman (1982), when he suggested that one might optimize the simulation of quantum systems with computers that are based on the principles of quantum mechanics. Such quantum simulators are now studied both theoretically and experimentally. A good example is provided by cold neutral atoms that are trapped into optical lattices formed by laser beams (Bloch 2005). As the atomic interactions can be tuned with magnetic fields, and quantum statistics plays a definite role in the system dynamics, one can hope to simulate nontrivial models for many-body dynamics in lattices, including quantum phase transitions (Sachdev 1999).

The modern view on quantum computing, however, leans toward an analogy with digital computers, in the sense that the framework consists of binary numbers (registers), logical operations (gates), and networks (circuits) (Steane 1998; Nielsen and Chuang 2000; Stenholm and Suominen 2005; Mermin 2007). Rather than solving problems in physics, the aim is to solve mathematical problems. Appropriate examples are the famous factoring algorithm of Shor (1994), or the search algorithm of Grover (1997). These algorithms use some of the special properties of quantum mechanics such as superpositions and entanglement, to achieve a computational speedup that has a more fortuitous scaling with the proportion of the problem than the corresponding classical approaches. Part of the wide attention given to quantum computing is based on the special role that the factoring of large integers has in the security of the public key cryptosystems (Mollin 2002). Apart from these two algorithms, the apparent benefits of quantum computers are still unclear.

The basic element in quantum computing is the physical representation of the quantum bit, also called a qubit. For quantum communication and quantum measurement purposes, it is quite meaningful to study systems consisting only of a few qubits; this is the limit that some of the experimentally studied systems have already reached. The number of qubits needed for solving nontrivial mathematical problems, however, is on the order of a hundred in the ideal case, closer to a thousand or more with error correction included (currently the best figure is on the order of ten qubits, with liquid nuclear magnetic resonance (NMR) systems (Negrevergne et al. 2006)). This defines the concept of large-scale quantum computing, which in practice usually means that one needs to construct a mesoscopic or even a macroscopic quantum system. There are many detailed studies of physical implementation of quantum computing, such as Chen et al. (2007); Nakahara and Ohmi (2008); Stolze et al. (2008). The purpose of this review is to provide a short overview of the topic.

2 DiVincenzo Criteria

The general conditions for building a successfully working quantum computer are rather obvious, and the best known formulation for them was given by DiVincenzo (2000) (he was also the first to make careful comparison of various physical systems suitable for quantum computing; see, for example, DiVincenzo (1995)). The five items in his famous “checklist” can be formulated roughly as

1. *A scalable set of well-defined and individually identifiable two-state quantum systems that can act as qubits and which relate as a whole to a well-defined and finite state space.*

To represent binary numbers, we need to give each qubit a label. Scalability means that we should be able to increase the number of qubits without limit (at least in theory). Combined with the other items below, this can prove to be a hard demand.

2. *Physical preparation of single qubits or groups of qubits into pure states in the state space.*

This often reduces to the possibility of preparing the fiducial state where all qubits are firmly set to the state representing the number 0. Any other state can be then obtained if the next item on the checklist is available.

3. *Implementation of unitary operations that involve one or two qubits.*

Since it is expected that all algorithms can be mapped into networks of single qubit operations and two-qubit conditional logic, this is sufficient. Furthermore, any network can be realized with single qubit operations and control-NOT operations, and the latter in turn can be constructed from simple conditional phase flips. This last stage is important because it opens the possibility for adiabatic operations and the use of geometric phases.

Physically, the requirement for conditional logic maps usually into controlling the interaction between two arbitrary qubits. In experimental realizations we can use the fact that as the computational power scales exponentially with the number of qubits, we nevertheless need to implement only a polynomially increasing number of SWAP gates to reduce the condition of interactions between arbitrary two qubits into nearest-neighbor operations only. This greatly helps in finding eligible physical implementations.

It should be noted that in optimizing networks, it may be helpful to allow other operations such as the three-qubit control-NOT operation either with two control bits and one target bit (Toffoli gate) or one control bit and two target bits with conditional mutual swapping (Fredkin gate).

4. *Performance of all required unitary operations at a time scale that is faster than any physical mechanism that causes the quantum system to decohere.*

Unfortunately, this requirement clashes often with the scalability requirement in item 1, and it is sometimes considered as a fundamental obstacle to quantum computing (Haroche and Raimond 1996). At worst, the decoherence times for quantum registers scale exponentially with the number of qubits (Unruh 1995). In fact, decoherence times for registers made of qubits can be much faster than the decoherence time for a single qubit times the number of qubits (Palma et al. 1996). In addition, for typical algorithms, the number of necessary basic operations can also increase rapidly with the number of qubits (at least polynomially as in the quantum Fourier transform (Coppersmith 1994; Barenco et al. 1996) needed in Shor’s factoring algorithm).

Although this requirement sets in principle the upper bound to the number of available qubits, it does not necessarily prevent one from reaching computationally useful numbers of qubits. Also, error correction (usually by redundancy) or the use of decoherence-free subspaces are among the tools used to further evade the effects of decoherence.

5. *The possibility to physically read the state of each qubit once the computation has been performed.*

This requirement is often fulfilled if one can satisfy items 1–3 as well and it could be combined with item 2. In actual proof-of-principle demonstrations, one often performs full quantum tomography on the final state, and uses the fidelity of the experimental state with the expected ideal result as the figure of merit.

For quantum communication purposes, these five requirements have been augmented by two additional ones (DiVincenzo 2000):

6. *The ability to interconvert stationary and flying qubits.*
7. *The ability to faithfully transmit qubits between specified locations.*

These additions also apply to quantum computing if we consider grid-like quantum computing, in which single qubits or collections of qubits form the nodes of a metacomputer; see, for example, the ion trap proposal in Kielpinski et al. (2000). If realizable, such an approach provides an option to circumvent the scalability requirement in item 1. Recently quantum teleportation has become a viable tool for satisfying items 6 and 7, and a suitable method to access quantum memories, for example.

These stringent requirements usually mean that we need to have a mesoscopic quantum system and the ability to control its dynamics with high accuracy. Several suggestions for physical implementation of quantum computing have appeared since 1994. Although many of them might have a chance of experimental realization at least on the level of a few qubits, very few have been selected as targets for serious studies. For few-qubit applications, their intended use in, for example, quantum communication sets other requirements such as the possibility of transmission or interaction with information carriers such as photons(items 6 and 7 in the list above). The purpose of this review is to discuss those realizations that are now studied by several groups of experimentalists, with some initial success, for the purpose of large-scale quantum computing (at least in principle). Before that, it is advantageous to provide a short representation of some of the concepts that were briefly mentioned above.

3 Elements of Quantum Computation

The purpose of a quantum computer is to solve problems following an algorithm. The implementation of an algorithm must be mapped into a unitary operation, and for practical purposes this operation must be factored into a series of few-qubit operations known as gates. In this sense, the quantum computer resembles its classical counterpart. For the more mathematical aspects of quantum algorithms and quantum computing, the book by Hirvensalo (2004) is suggested.

3.1 Quantum Algorithms

As mentioned above, the main existing algorithms aim at either factoring large integers (Shor) or searching databases (Grover). The factoring algorithm relies on calculating a modular function $f(x) = a^x \bmod M$, where a is an almost freely chosen integer and M is the number that we want to factor. The periodicity of this function is the key single number that one seeks.

The quantum parallelism means that for a combined set of input and output registers (both with N qubits and $2^N \simeq M^2$), the initial state can be written as

$$|\Psi_0\rangle_{2N} = |\Psi_0^{\text{input}}\rangle_N \otimes |\Psi_0^{\text{output}}\rangle_N = \left(\frac{1}{2^{N/2}} \sum_{x=0}^{2^N-1} |x\rangle_N \right) \otimes |0\rangle_N \quad (1)$$

where the input state is an equal superposition of all possible numbers and in the output register all qubits are in the 0 state.

At this point, it is convenient to introduce the standard single-qubit Hadamard operation H_{Had} , that is of the form

$$U_{\text{Had}} = \frac{1}{\sqrt{2}} [(|0\rangle + |1\rangle)\langle 0| + (|0\rangle - |1\rangle)\langle 1|] \quad (2)$$

and which clearly puts any qubit from a state 0 or 1 to their equal superposition. By applying U_{Had} to each input qubit, the above initial state for the input register is easily produced (assuming all qubits can be set first to state 0; even this is not always trivial in actual physical implementations because such resetting is not necessarily a unitary operation). An alternative representation of operations is the matrix form, where the qubit states are given as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (3)$$

The Hadamard operation is a 2×2 matrix

$$U_{\text{Had}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (4)$$

and by applying it twice we get the identity $U_{\text{Had}}^2 = \mathbf{1}$.

Next, one applies the unitary transformation U_f corresponding to calculating $f(x)$ so that the result is stored into the output register while the input is left intact. Due to the linearity of quantum mechanics, this leads to the new two-register state

$$|\Psi_f\rangle_{2N} = \frac{1}{2^{N/2}} \sum_{x=0}^{2^N-1} |x\rangle_N |f(x)\rangle_N \quad (5)$$

This is a remarkable state as it represents entanglement between the register states so that each outcome can exist only with its corresponding input, in a total quantum correlation. As one obtains the result by applying U_f only once, this is often called quantum parallelism. As one can extract by measurements only one $(x, f(x))$ pair at a time, this approach is not useful for just calculating values of $f(x)$.

Since $f(x)$ is periodic, the amount of possible numbers in $|f(x)\rangle$ is limited, and for each result there is a periodic subset of numbers in the input register. If one performs a quantum Fourier transform (QFT) on the input register, one obtains an integer multiple of the inverse of the period. This, together with inaccuracies due to the finiteness of the registers, means that the total running time for the algorithm with the standard QFT network scales as N^3 , where N^2 is for the QFT, and additional N repetitions are needed to fix the period with sufficient accuracy. As the best known classical approach scales as $\exp(N^{1/3})$, the advantage increases nearly exponentially with increasing N .

The Shor algorithm is genuinely quantum in the sense that it utilizes the entanglement between the registers. Whether its speed is due to its quantum nature, or due to the inherent

efficiency of QFT, is an open question. Another quantum property is the superposition and, together with infinitesimal projections, it is the basic ingredient in the Grover algorithm.

For the database search, we need only one quantum register. Assuming that there is a mechanism that allows the computer to identify the desired item with a number stored in a register of N qubits, one can take as the initial state

$$|\Psi_0\rangle_N = \frac{1}{2^{N/2}} \sum_{x=0}^{2^N-1} |x\rangle_N \quad (6)$$

The search operation then recognizes a special value x_0 related to the wanted item. As a unitary operation, it changes the superposition into

$$|\Psi_1\rangle_N = \frac{1}{2^{N/2}} \left(\sum_{x=0, x \neq x_0}^{2^N-1} |x\rangle_N - |x_0\rangle \right) \quad (7)$$

The next step is to perform a specific single-qubit operation on each qubit, and then repeat the above. It can be shown that after $N_0 \simeq (\pi/4)\sqrt{N}$ repetitions of this two-step process, one gets

$$|\Psi_{N_0}\rangle_N = |x_0\rangle \quad (8)$$

and the item is then identified through x_0 . Compared to the Shor algorithm, the speed-up is not huge (classical search scales as $N/2$). For a nice description of the Grover algorithm, the book by Vedral (2006) is referred to.

A third algorithm that is often mentioned especially in few-qubit demonstrations is the Deutsch–Jozsa algorithm (Cleve et al. 1998) although in practice it serves only demonstration purposes. We assume an a priori unknown one-qubit function $f(x)$ that maps a one-digit binary number to a one-digit binary number. The question is whether the function is constant ($f(0) = f(1)$) or balanced ($f(0) \neq f(1)$). This is the Deutsch problem, and classically one solves it by calculating the function twice with different arguments (0 and 1), and comparing the result. The Deutsch–Jozsa algorithm is a generalization of this, so that the argument can be a number between 0 and 2^N-1 , while the values of the function are still limited to 0 or 1.

In brief, the operation starts by setting the state (input register with N qubits and output with 1 qubit)

$$|\Psi_0\rangle_{N+1} = \frac{1}{2^{(N+1)/2}} \sum_{x=0}^{2^N-1} |x\rangle_N \otimes (|0\rangle_1 - |1\rangle_1) \quad (9)$$

and then calculates $f(x)$ by a unitary transformation U_f , so that $f(x)$ is added to the existing value of the output bit (binary addition, so $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$):

$$|\Psi_f\rangle_{N+1} = \frac{1}{2^{(N+1)/2}} \sum_{x=0}^{2^N-1} |x\rangle_N \otimes (|f(x)\rangle_1 - |f(x) + 1\rangle_1) \quad (10)$$

This is equivalent to

$$|\Psi_f\rangle_{N+1} = \frac{1}{2^{(N+1)/2}} \sum_{x=0}^{2^N-1} (-1)^{f(x)} |x\rangle_N \otimes (|0\rangle_1 - |1\rangle_1) \quad (11)$$

Now the output register can be discarded, and one performs a Hadamard operation to each qubit of the input register. If projected to the all-zero state $|0\rangle_N$, one gets that the projection is successful with probability 1 for constant case, and with probability 0 for the balanced case.

3.2 Quantum Gates and Quantum Networks

It is tempting to consider performing simple unitary operations on large registers without problems, but in practice that involves too many degrees of freedom for simultaneous control. A general operation on an N -qubit register would correspond to a $2^N \times 2^N$ matrix. Clearly, one loses the quantum advantage if one thinks of the register as a 2^N -dimensional quantum state instead of working with N two-state systems. In practice, all computations are made in networks of quantum gates that act typically on single qubits, or on two qubits (for conditional logic). Such operations are unitary and thus also reversible. The Hadamard gate above is a typical single-qubit operation; most of these can be described as rotations in the single-qubit Hilbert space. These operations set qubits to specific superposition states and can also affect the phase relation between the probability amplitudes of single qubit states, and they were studied extensively in the early days of quantum computing, and reviewed in many textbooks, see, for example, Nielsen and Chuang (2000); Stenholm and Suominen (2005); Mermin (2007).

Of the two-qubit gates, we concentrate here only on a few essential ones, and their relationship to each other. The controlled-NOT (CNOT) gate is a standard element in conditional logic. In short, it flips the state of the target qubit if the control qubit is in state 1. Specifically, if the control qubit is in a superposition state, CNOT entangles it with the target qubit (in case they were initially in uncorrelated states). An alternative gate is the controlled phase gate, which alters the global phase of the two qubits by ϕ if they are both in state 1. Especially, if one can effect a gate with conditional phase change of π , that is, $|11\rangle \rightarrow -|11\rangle$, this conditional phase flip gate can be used to construct the CNOT gate. This is important for geometric quantum computing discussed later. CNOT gates are also needed in error correction methods for quantum computers. For demonstration purposes and for quantum communication one wants to either create highly entangled states of qubits, or to detect entangled states such as the Bell states. The CNOT gate is a very convenient tool to produce such states, or in mapping them to easily distinguishable qubit states.

The two-qubit gates are usually described as 4×4 matrices in the joint basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. The representations for the CNOT gate, phase gate, and the phase flip gate are, respectively,

$$U_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad U_\phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}, \quad U_{\text{ph-flip}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (12)$$

Another two-qubit gate is the SWAP gate that interchanges the state of two qubits:

$$U_{\text{SWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad U_{\text{iSWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (13)$$

Here, we introduce also the iSWAP gate discussed later. Swapping is not conditional logic, but it is important because instead of moving qubits physically, one can alter their labeling instead. Thus, any two qubits can be made nearest neighbors, which enlarges the number of those physical systems that are realistic for quantum computing purposes. The swapping can be realized with a simple combination of three CNOT gates.

In general, the CNOT gate combined with a set of single-qubit operations form a universal set: all possible two-qubit operations can be realized with them. In the same manner as the CNOT can be constructed with conditional phase flips, we can find other forms of conditional operations that can be taken as a starting point for realizing actual computation (Bremner et al. 2002). Often the experimental realization of two-qubit operations determines which kind of conditional logic gate is the basic element. Earlier, the demonstration of the CNOT gate was considered as a necessary step for successful computation, but lately it has been replaced by many other alternatives such as CROT, $\sqrt{\text{SWAP}}$, and iSWAP (given above). The first two are represented by the following unitary transformations:

$$U_{\text{CROT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad U_{\sqrt{\text{SWAP}}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\pi/4} & ie^{i\pi/4} & 0 \\ 0 & ie^{i\pi/4} & e^{i\pi/4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (14)$$

The second definition is obviously not a unique one for $\sqrt{\text{SWAP}}$ (e.g., clearly $\pi/4$ can be replaced with $3\pi/4$). The variation $\sqrt{i\text{SWAP}}$ has also been used. We can see that the variations of the SWAP gate are suitable for situations where only the states $|01\rangle$ and $|10\rangle$ can be coupled.

A quantum network consists of the gates and the information to which qubits and in which order one applies the gates. The quantum Fourier transform network for N qubits, for example, consists of N single-qubit Hadamard gates, and $N(N-1)/2$ conditional phase shifts (with qubit index-dependent values of phase shifts) (Coppersmith 2002; Barenco et al. 1996). In some special cases, one can utilize gates that operate on three qubits or more. The Toffoli gate swaps the state of the target qubit, if the two control qubits are both in state 1. This can be generalized into n control qubits, naturally. The Fredkin gate performs a SWAP gate on two target qubits if the control qubit is in state 1.

3.3 Error Correction

The sensitivity of quantum mechanical superpositions to disturbances is perhaps the most serious problem for actually realizing quantum computing. In addition to decoherence and dissipation, due to the coupling of quantum systems to their environment, small errors due to imperfections in gate operations interfere with the computing process. Although small, the cumulative effect of small errors can be devastating if they are allowed to propagate freely in the system through conditional logic. The errors can be suppressed by applying error correction. However, it requires operations that are typically subjected to the same inaccuracies as the actual computational operations. If the error correction during computation is effective, that is, it reduces errors more than it adds them, computation is regarded as *fault-tolerant*.

The standard quantum error correction relies on redundancy. For example, for one qubit in state

$$\alpha|0\rangle + \beta|1\rangle \quad (15)$$

we write the highly entangled three-qubit state

$$\alpha|000\rangle + \beta|111\rangle \quad (16)$$

This is achieved by starting with $(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle \otimes |0\rangle$ and applying the CNOT on the last two zero qubits with the first one as the control. The use of three qubits increases the size of the qubit Hilbert space from 2 to 2^3 . If, at most, one of the spins flips, the location of the flip can be identified by projecting the state on four orthogonal states, which are

$$\begin{aligned} P_0 &= |000\rangle\langle 000| + |111\rangle\langle 111| \\ P_1 &= |100\rangle\langle 100| + |011\rangle\langle 011| \\ P_2 &= |010\rangle\langle 010| + |101\rangle\langle 101| \\ P_3 &= |001\rangle\langle 001| + |110\rangle\langle 110| \end{aligned} \quad (17)$$

The result identifies which qubit flipped (or that none flipped), and one can perform a correcting flip on that qubit. This does not perturb the actual superposition since we do not know whether 0 became 1 or 1 became 0, but only that a flip occurred.

With a slightly more complicated operation (and more qubits) one can identify and correct phase flips as well. In 1995 Peter Shor showed how such a general error correction scheme can be constructed with nine qubits (Shor 1995), but the number was reduced later to seven (Steane 1996; Calderbank and Shor 1996). Finally the sufficient number of qubits needed was found to be five (Bennett et al. 1996; Laflamme et al. 1996), leading to fault-tolerant error correction (Knill 2001). Systematic studies in the field have produced more general methods such as stabilizer codes (Gottesman 1996). For a recent detailed description of quantum error correction and fault-tolerant computation see Gaitan (2008); see also the related chapter and references therein in Nielsen and Chuang (2000).

For noisy computation and specific cases, one can also utilize optimization of algorithms. For instance, the quantum Fourier transform relies on conditional phase shifts. It can be shown that since the noise per operation severely perturbs the smallest phase shifts, it becomes advantageous to reduce the overall noise by not performing these phase shifts at all (Barenco et al. 1996). The improvement is due to the decreased total number of operations, and additional benefits can come for nearest-neighbor realizations since the size of the implemented phase shift is inversely proportional to the difference in qubit labels (typically equivalent to the physical distance between qubits as well).

3.4 Decoherence-Free Subspaces

Error correction can target both the technical problems as well as the environmental effects. Even for technically perfect qubit operations, one is still left with the coupling of a quantum system with its environment. The thermal heat bath is a typical example of a source for dissipation and decoherence. The first leads to qubit state flips, and the latter erodes the phase relations between amplitudes either within the qubits or (and more rapidly) between qubits. A prime example is the transversal and longitudinal relaxation times in NMR; it also illustrates the general feature that decoherence time scales can be much faster than dissipation time scales, and, thus, in most cases the decoherence time scale compared with the gate operation time is the key figure of merit.

The standard approach to decoherence is to suppress the interaction between the system and its environment. Cooling the computing system to low temperatures is a basic solution; in some cases, such as flux qubits (Sect. 8.2), the cooling is essential for the existence of the qubit system in the first place. Occasionally, the decoherence can actually be of advantage, as

we shall see in connection with the liquid NMR quantum computation in [Sect. 6.1](#), but such situations are exceptions with their own special difficulties.

For some systems, one can find, for some combinations of qubits, that they are more robust to decoherence than other combinations. Thus, one can select a decoherence-free subspace in this extended Hilbert space, and limit the computation there (Palma et al. 1996; Duan and Guo 1997; Zanardi and Rasetti 1997; Lidar et al. 1998).

3.5 Adiabatic and Geometric Quantum Computing

One of the very robust states, when it comes to decoherence and dissipation, is the ground state of a quantum system. In adiabatic evolution, the system remains always in the instantaneous eigenstate of the system. If the computing starts with the ground state of the computer, and one can perform the computation adiabatically (Farhi et al. 2000; Childs et al. 2001), it means that the system remains on the ground state all the time, and it is only the nature of the ground state that evolves into the final result of the computation.

One can describe the adiabatic process as a passage related to a changing parameter. If one performs a closed loop in the parameter space, one can affiliate a global geometric phase to the evolution. This global phase is the famous Berry phase, and it depends on the path that one takes in the parameter space. Later, the approach was generalized to non-closing loops as well. This provides the basis for geometrical quantum computation. An interacting combination of two quantum bits can be made to evolve along its changing ground state so that the global phase change is conditional to the values of the two qubits. It means that in practice one can realize, for example, the phase-flip operation. Another name for computations using geometric phases and closed loops in the parameter space is holonomic quantum computation. There is a nice recent review on the geometric quantum computing and its basic concepts (Sjöqvist 2008).

Recently, some interest was created by the quantum processor developed by the D-Wave company, based on superconducting flux qubits. The actual computation is taking place, assumably, through adiabatic evolution. This is still subject to much debate. One of the problems with adiabatic evolution is that although the lowest eigenstate of the system remains well-defined, it may get very close to some higher state, and nonadiabatic transitions between them can be suppressed only by an extremely slow evolution, which in turn may lead to other problems.

3.6 Cluster State Quantum Computing and Anyonic Quantum Computing

Normally, we assume that the quantum computation follows a very clear cycle. First the input is prepared, then processed, and after that the result is read out by measurement. This is “Hamiltonian computing.” In quantum mechanics, one can challenge this view. One approach is to make the whole computation proceed by measurements instead of unitary evolution. If a two- or three-dimensional lattice of qubits interacts suitably (e.g., Ising-type interaction), it can form a large cluster of entangled states (Briegel and Raussendorf 2001). Computation can then proceed in this cluster by targeted measurements on individual particles (Raussendorf and Briegel 2001). The reliance on irreversible measurements, instead of unitary operations only, has given the cluster approach the name *one-way quantum computing*.

Another idea based on a lattice of qubits with nearest neighbor interactions was proposed by Kitaev (1997). The approach is expected to be fault-tolerant by construction. Computation is performed by using an anyonic excitation of the system, where an exchange of two particles changes the global phase of their wavefunction. The approach relies especially on having non-Abelian anyons, which might be found in two-dimensional degenerate quantum gases (fractional quantum Hall effect). This topic has been recently reviewed thoroughly, see Nayak et al. (2008).

4 Cavity QED

In a confined space, the electromagnetic field modes become quantized into standing waves and form a discrete set of energy states for the field. In the quantum electrodynamical (QED) description, the energy of each mode with frequency ν is quantized into photon states, integer multiples of $\hbar\nu$ including the vacuum state of each mode (\hbar is the Planck constant). Such a cavity with well-defined modes in one direction can be built by opposing mirrors. The photon states of the modes can occur in quantum superpositions of photon numbers n . However, the temperature of the mirrors defines the temperature of the cavity, so to avoid thermal occupancy of the photon states, one must go to very low temperatures. Also, the quality of the mirrors must be high, otherwise the loss of photons broadens the energy spectrum of each mode.

The key issue is the controlled interaction between the cavity modes and the atoms that fly through the cavity or are placed there by means of some electromagnetic trapping. In an ideal case, one can describe the atoms as two-level systems, and limit the interaction to a single mode. One can then explore some key phenomena such as Rabi oscillations between atomic states (Jaynes–Cummings model) but, more importantly, with the atoms, one can manipulate the photonic state of the cavity. States with definite photon number n are actually highly nonclassical states of light, and they can be created with the help of the atoms. Alternatively, the photonic state of the cavity can be explored in a nondestructive way (QND, quantum non-demolishing) with atoms that are highly off-resonant but still coupled with the relevant cavity mode.

Such cavities have been prepared both in the optical and microwave region. It is easier to obtain high quality (high-Q) cavities in the microwave range than in the optical range. For the microwave range, the atoms are usually pumped into a high-energy Rydberg state (main quantum number ≈ 60), so that the transition energy falls into the microwave regime, and in the QND photonic state measurements one applies atom interferometry. The best demonstrations of controlled and coherent dynamics of two-state quantum systems and interactions with a quantized field have been achieved with such systems. These are essential factors in single-qubit operations, and the cavity QED terminology has been adapted for many other physical implementations when demonstrating that the state of a qubit is indeed a quantum superposition and coherence can be established between the qubit states. For example, by implementing Rabi oscillations in a qubit, one can estimate single-qubit decoherence through the damping of these oscillations (and by observing them to prove that there is some coherence to begin with; see, for example, the flux qubit example of Chiorescu et al. (2003)). The concepts of cavity QED were reviewed in detail in the textbook by Haroche and Raimond (2006). In fact, one of the first demonstrations of conditional phase shifts was performed in 1995 with an optical cavity, where photon qubits (0 or 1 photon per polarization

mode) experienced an interaction via off-resonant Cs atoms flying through the cavity (Turchette et al. 1995). See also the 2004 review in van Enk et al. (2004).

5 Physical Realization with Electromagnetically Trapped Particles

5.1 Trapped Ions

The trapping of ions with electromagnetic fields has provided much insight into quantum mechanics since the early 1980s (Horvath et al. 1997). Just the actual observation of the single ions, by detecting the light scattered by them, showed that single atoms are a meaningful concept in physics, not just an abstraction. The observed change of the quantum state of a single ion through a quantum jump in the mid-1980s demonstrated clearly the limits of ensemble quantum mechanics, and also the existence of the until-then hypothetic jumps. In 20 years, these jumps have evolved from a mystery into a practical tool in detecting the state of single ions (see item 5 on the DiVincenzo checklist); already in the first experiments, they were used to show that the scattered light actually originated from a specific integer number of ions.

Laser cooling of these ions was developed alongside the trapping methods, and gave control over the motional states of ions. Especially, it made it possible to reach situations where the motion itself becomes quantized, and in the case of more than one ion, one gets phonons. Thus, it is not perhaps surprising that very soon after the presentation of the Shor algorithm in 1994, Ignacio Cirac and Peter Zoller published their proposal for quantum computing with chains of ultracold ions (Cirac and Zoller 1995). Even now, this scheme provides the basic approach to quantum computing with trapped ions.

Trapping of charged particles by static electromagnetic fields is in principle not possible (Earnshaw theorem), but the problem can be averted by adding either a fast-oscillating electric field (Paul trap) or a static magnetic field (Penning trap). For quantum computing purposes, to obtain individual addressing and controllable interactions, the so far most suitable situation for many ions is provided by the linear ion trap, in which the trapping potential is an elongated cigar, into which the atoms at low temperatures form a chain as they are pushed toward each other by the trapping potential, but this is balanced by the Coulomb repulsion between charged ions. The quantized motion of ions is now a collective effect, and for sufficiently low temperatures any motion perpendicular to the cigar is suppressed, and along the cigar the lowest phonon modes correspond to the oscillation of the whole chain in unison, and to the breathing-like mode where each atom oscillated with a π phase shift compared to its two neighbors.

The cooling of such a chain is based on ions absorbing photons at lower energies, and then radiating them at higher energies, with the energy difference arising from a change of the motional state from a higher one to a lower one (sideband cooling). As the collective motion affects all ions of the chain, Cirac and Zoller proposed that if one codes one qubit per ion, using its internal electronic states, conditional logic can be mediated with the phonons in the chain. The scenario is simple: a change of the state of each qubit can be realized by addressing the particular qubit by laser light. Assuming that the chain is in its lowest motional state, one can address qubit 1 so that if it is in state 1, then a phonon appears in the chain (excitation to the next state of collective motion). When addressing qubit 2, one can make it conditional so that one has four possible outcomes, depending on the original state of the qubit 2, and the phonon state of the chain (and thus the state of qubit 1).

The lifetimes for excited atomic and ionic states are usually short, on the order of nanoseconds, so they are not a priori optimal states for qubit coding. The problem can be circumvented by using either the ground state and a long-living excited state that exists in most ions of alkaline-earth atoms (group II), or two Zeeman substates of the ground state, coupled then with a Raman setup instead of a single laser field. The shortlived electronic states have their purpose, however, as one can detect the state of qubits using the quantum jump approach.

Several groups have adopted the Cirac–Zoller scheme as a basis for demonstrating quantum computing. Basic problems on the road have been the elimination of fluctuations in the electromagnetic trapping fields and actual cooling of the chain of ions (Nägerl et al. 1998; Schmidt et al. 2003a). But successful demonstrations have been achieved. They can be coarsely divided into two groups: demonstration of gate operations, and preparation of specific entangled states of many qubits. Although the latter are often obtained as a result of logical operations and quantum computing, here we concentrate on the gate operations. The success of logical operations is verified by quantum tomography of the final state, and quantified by fidelity with the ideal result. In Innsbruck, in the group of Rainer Blatt, the control-NOT gate between two qubits was demonstrated in 2003 (Schmidt-Kaler et al. 2003b). Already in 1995, such a gate was demonstrated in Boulder (Monroe et al. 1995), but this was a nonscalable demonstration where the two qubits were coded in the same ion, and the Cirac–Zoller scheme was not used. Later in Innsbruck, the three-qubit Toffoli gate was demonstrated in 2009 (Monz et al. 2009). The three-qubit operation was achieved by taking advantage of the three lowest motional states, as an extension of the Cirac–Zoller scheme.

As the ionic chains become longer, their cooling is an increasing challenge. This has led to the proposals involving hot ions. Typically, the phonons are then replaced by photons for the conditional logic. Perhaps the most successful of the proposals is the Mølmer–Sørensen bichromatic gate (Sørensen and Mølmer 1999a, b). It relies on the idea of the conditional phase gate as the basic element. When two ions are simultaneously interacting with two light fields, the phase of the final state will depend on the internal states (acquires -1 if both qubits are in state 1). This scheme works seemingly independent of the motional states of the ions, but their presence is required nevertheless as they are absent only due to the use of the perturbation approach (which means that the gates are not necessarily very fast). Such a gate action was demonstrated recently in Innsbruck (Benhelm et al. 2008).

Despite the success with either cold or hot ions one can ask how scalable the ion trap computers are. Clearly, the lengthening of the chain will pose many problems, from cooling to individual addressing. The possibility to go for 2D structures (Wigner crystals) would allow better scaling, but also more complicated motional states and cooling of the structure. Another possibility is to distribute the computing between several small scale ion trap computers, using, for example, photons as mediators between the computers (and quantum teleportation as a tool for exchanging the quantum information between the ions and mediating photons). Perhaps a simpler alternative is to shuffle ions around, dividing the trap into computing and memory sections (Kielpinski et al. 2002; Schaetz et al. 2004). For qubit states involving the two Zeeman states of the ionic ground state one can utilize microwave fields. This opens the possibility to use high-quality cavities in the control of qubits. Cavity QED is a well-established field and can provide useful tools, although it applies best to transitions between atomic Rydberg states.

For further details, the reader is referred to an excellent review on ion trap quantum computing by Häffner et al. (2008).

5.2 Trapped Neutral Atoms

Trapping of neutral atoms into structures where they can be addressed one-by-one is much harder than for charged particles. There are basically two avenues for this: single atom traps or optical lattices. Light from a highly detuned laser field is unlikely to be absorbed by atoms, but they still interact in a manner that allows one to control the external degrees of freedom of the atoms. Basically, the atoms feel the spatial changes in the field as an external potential. If the light is tuned above the atomic transition (blue detuning) the field intensity maxima repel atoms, and for tuning below the transition (red detuning) the maxima are points of attraction. This is either called dipole or gradient force. In the simplest formulation, atoms are attracted by, for example, the focal point of a laser beam. If one adds some dissipative mechanism, one can confine atoms into such dipole traps (also called FORTs, i.e., far-off-resonant traps). This is one of the key points for laser cooling and trapping of atoms (Metcalf and Van der Straten 1999). If one adds the spin-induced degeneracy of the atomic states and the polarization of the light, one can additionally control the system with external magnetic fields, or use the laser field superpositions to create specific structures such as lattices with potential minima and maxima at regular intervals.

The trapping of atoms into microtraps is a challenging task. For quantum computing purposes, one would prefer a set of such traps with one atom (and only one) in each, which is not easy. The advantage of microtraps compared to optical lattices is that it is easier to address atoms individually, when the separation is in the micrometer range. Several possible setups are currently studied, including atom chip systems, where the atoms are localized very close to a solid surface, and wires on the surface provide the electromagnetic fields that confine the atoms. As the confining forces are weak, low temperatures and elimination of collisions with background atoms are required, not to mention an accurate trap-loading process. Conditional logic is also a challenge. If sufficiently close, the atoms can interact by a state-selective dipole–dipole interaction, which provides one possibility, but there are no proof-of-principle experimental demonstrations for quantum computation in such systems yet.

For optical lattices, the lattice site occupancy is low and not very well controlled with basic laser-cooling and trapping methods. However, the Bose–Einstein condensation of atoms provides a useful tool for preparing lattice occupation. The atoms are trapped and cooled so that they begin to occupy the lowest quantum state of the trap potential. If they are bosons, they undergo eventually a quantum phase transition to a condensed state, where the atoms are correlated completely and the whole sample is described by a single collective quantum wave function (and with a nonlinear Schrödinger equation where the atomic interaction appears as a mean-field potential) (Pethick and Smith 2008). In an optical lattice one can, for sufficiently high barriers between lattice sites, observe a transition to a Mott insulator, where (a) each lattice site is occupied and (b) with exactly the same number of atoms. This has already been achieved experimentally (Bloch 2005), and it opens possibilities for quantum computing, especially if single-atom addressability can be reached. But this is a future method and no demonstrations of quantum logic have been performed so far.

5.3 Trapped Single Electrons

Experimentally, the trapping of single electrons with electromagnetic fields preceded the trapping of ions. In general, many solid-state structures such as quantum dots or

superconducting islands rely on confining potentials for electrons or equivalent quasiparticles. In principle, even atoms and molecules are just traps for electrons. A common theme for many systems is that the dynamics of a complicated system depends only on a few electrons while the nuclei and the majority of the electrons only form a background. Not surprisingly, the popular alkaline atoms for neutral atom trapping and ionized alkaline-earth atoms for ion trapping are effectively hydrogen-like single-electron atoms. In fact, even the nuclei for atoms such as C, N, and F are made of several protons and neutrons, but as composite particles they only appear to one as spin-1/2 systems.

If one nevertheless persists with the concept of an electron trapped by externally generated electromagnetic fields, possibilities for quantum computing have been considered in such systems as well. An example is the proposal where the electrons are confined on the surface of liquid helium and controlled by electromagnetic fields (Platzman and Dykman 1999; Dykman and Platzman 2000). The surface potential for electrons stops them from penetrating the liquid but it is weakly attractive at short distances, so that a few bound states appear, confining the electrons as a two-dimensional gas at the surface. The bound states can be addressed with microwave fields. Due to the Coulomb interaction, the electrons repulse each other (in analogy to trapped ions) and, as their temperature is lowered, one observes a phase transition to a Wigner crystal (Grimes and Adams 1979) (see also Rousseau et al. (2009) and references therein; the Wigner crystal has been observed for trapped ions as well (see, e.g., Birk et al. 1992)).

However, one does not rely on the crystallization in qubit preparation but instead, by embedding electrodes into the liquid, one can provide the localization in the direction of the surface as well, and in addition one can induce individual (local) Stark shifts to resonance energies so that one can select the particular qubit, which is subsequently manipulated with the microwave field pulse (other qubits are off-resonant with the field).

The dipole–dipole interaction between adjacent electrons depends on the bound state because it shapes the electron wavefunction and thus the charge distribution (in very much the same manner as in the first quantum dot proposals that also used the dipole–dipole interaction for conditional logic (Barenco et al. 1995)). The read-out would be obtained by external field-assisted tunneling of the higher state electrons out of the surface potential; they would be individually detected by channel plates. It has been suggested also that the electron spin could be used to represent a qubit (Lyon 2006); the degeneracy of spin states is lifted with an external magnetic field, and the spin-dependence of the dipole–dipole interaction can be used. There is some experimental activity (see, e.g., Sabouret and Lyon 2006) in this field and also some theoretical studies have been done (see, e.g., Dykman and Platzman 2003; Dahm et al. 2003). The main decay channel would be the coupling of electrons to the liquid surface excitations (riplons), but one of the advantages of the system is that this coupling is weak, and compared to other suggestions such as nuclear spins in solid-state structures (see, e.g., [Sect. 6.2](#)) the advantage is that here the electrons are trapped practically in vacuum.

6 Nuclear Spin Computers

An obvious candidate for a quantum bit is the spin-1/2 particle, which is the standard example of many quantum mechanics textbooks and often provides for physics students the first encounter with group theory through SU(2) representations. As discussed in the previous section, although the electron is a typical example, many of the atomic nuclei are spin-1/2

systems as well. As they form the key ingredients in many molecules or solid structures, and can be manipulated and studied with radio-frequency fields, it is natural to consider them as possible basic elements for quantum computing. The nuclear systems, however, behave very differently depending on their surroundings, and this leads to the fact that the quantum computing aspects are best discussed separately for liquid and solid systems.

6.1 Molecular Spins in Liquids

Nuclear magnetic resonance (NMR) is a well-established research topic, and an important method in many fields such as chemistry and medical sciences. The broad success also brings problems: the sophisticated concepts and terminology in modern NMR are not anymore so accessible to physicists, especially to those working on quantum systems such as trapped ions and atoms or solid-state electron systems.

At first, the idea of doing quantum computing with large numbers of molecules in liquid at room temperature seems counterintuitive, especially since it seems to be completely complementary to other approaches, such as trapped ions that rely on single isolated systems at extremely low temperatures. This kind of approach is generally called ensemble quantum computing.

The basic element is a molecule consisting of several atoms, many of which have spin-1/2 nuclei. In a magnetic field, the spin-up and spin-down states obtain different energies, and the spins oscillate with their characteristic frequencies given by the energy difference. This Larmor frequency is typically on the order of 100 MHz. The local environment in the molecule for each nucleus is usually unique and gives a chemical shift to this energy/oscillation frequency, on the order of ~ 1 MHz or less. It means that in principle each spin can be detected and manipulated with a field of specific frequency. The magnetic moment of a single spin is too weak for practical use, and thus one needs macroscopic numbers.

In liquid state, these molecules can move and rotate freely. This has the additional advantage that interactions between the molecules simply average to zero. Furthermore, the same applies to the anisotropic dipolar magnetic spin–spin interaction within each molecule. Eventually, one is left only with the isotropic interaction between nuclear spins mediated by the atomic valence electrons. For two nearby spins, this gets down to a simple Hamiltonian

$$H = \hbar\omega_1 I_1^z + \hbar\omega_2 I_2^z + 2J_{12}I_1^z I_2^z \quad (18)$$

where $\omega_{1,2}$ are the characteristic frequencies, $I_{1,2}^z$ are the spin projection operators in the direction of the external field, and J_{12} quantifies the spin–spin interaction. Thus, one can address spins with different RF-pulses, and the third term in the Hamiltonian allows for conditional logic in the form of controlled phase slips.

Although the large numbers of molecules and the averaging characteristics of the thermal liquid seem to be useful for some aspects, one would also assume that they wash out any quantum coherences as well. In fact, since the thermal energy exceeds well the energy differences of spin states, the total state of the system is almost equal to a statistical mixture of equally populated spin states. The key issue is the small deviation from the mixture, which has the characteristics of a pure quantum state, with 2^N states for N spin-1/2 nuclei of a single molecule. This pseudo-pure state is often written in the form

$$\rho \simeq \frac{1}{2^N} \mathbf{1} + \alpha |\Psi\rangle\langle\Psi| \quad (19)$$

Clearly, since the spin operators are traceless, the thermal part does not contribute to any expectation values. Also, any unitary operation on this density matrix will only reproduce the thermal part, and operate on the pure part in the normal quantum fashion.

Of course, the above picture is highly simplified and in practice one performs complicated sequences of pulses, and uses spin-echo methods to control phase evolution. The advantage is that the decoherence times for nuclear spins are extremely long compared to the pulse durations. However, ensemble quantum computing is beset with several problems.

The factor α is small, typically 10^{-4} to 10^{-5} for a few spins at room temperature, and it scales in the high temperature limit as

$$\alpha \simeq \frac{N\hbar\Delta\omega}{k_B T 2^N} \quad (20)$$

where $\Delta\omega$ is the energy difference between spin-up and spin-down states, T is the temperature, and k_B is the Boltzmann factor. To maintain the signal size means exponentially increasing the number of molecules or repeated measurements, as pointed out by Jones (2000).

The increase in the number of qubits means that one needs larger molecules with identifiable spins, and this is difficult to achieve. For instance, one of the largest systems is an *l*-histidine molecule with 14 spin-1/2 nuclei, of which two hydrogen nuclei are indistinguishable (forming a spin-0 and spin-1 system), providing at maximum a 12-qubit + 1 qutrit system (Negrevergne et al. 2006). It may be extremely difficult to go beyond this.

The pulse sequences needed for computation become exceedingly numerous and also complicated to design, tending to scale exponentially with the number of qubits. Thus, the duration of computation is reaching the decoherence limits. In addition, the small errors in each pulse seem to accumulate quickly as the number of pulses increases. This brings forward the apparently still unresolved question about the “quantumness” of ensemble quantum computing: Is it basically a simulation of actual quantum computing rather than the real thing?

To initiate a computation, one needs to set the pure component to a specific state, typically a state with all qubits in zero state. This is not easy to do and several methods have been applied in practice. Similarly, one can detect the state of the system at the end of a computation, but it is not possible to make projective measurements on selected spins during the computation process (which is a basic ingredient in many error correcting schemes). All this leads to the problem that ensemble quantum computing is not a very suitable candidate, for example, for grid-like quantum computing.

Despite the problems listed above, one should not neglect the many successes of liquid NMR quantum computing in demonstrating or benchmarking the basic operations and algorithms:

- Four-item search (Grover) with chloroform (decoherence time $\simeq 0.3$ s, computation time $\simeq 0.035$ s), in 1998 (Chuang et al. 1998). The two qubits were H and C nuclei. Another demonstration at the same time used partially deuterated cytosine, with two H nuclei as qubits (Jones et al. 1998). In 2000, a three-qubit version with ^{13}C -labeled CHFBr_2 with H, C, and F nuclei as qubits was demonstrated (Vandersypen et al. 2000), and in a more flexible way in 2002 with ^{13}C -labeled alanine, with three C nuclei as spins (Kim et al. 2002).
- Factoring of the number 15 in 2001 with 7-qubit perfluorobutadienyl iron complex (2 C and 5 F nuclei) (Vandersypen et al. 2001).
- The largest known qubit number: 12, with *l*-histidine (Negrevergne et al. 2006).

In addition, the NMR approach has also given special impetus to developing the concept of adiabatic quantum computing. Many reviews on liquid NMR quantum computing have been published during the last 10 years. Among the most comprehensive is the 2004 review article by Vandersypen and Chuang (2004), who have been involved closely with the theoretical and experimental development of the field. Another source is the recent multiauthored book (Oliveira et al. 2007).

6.2 Nuclear and Electron Spins in Silicon

One of the characteristics of solid-state systems is that the electron spin can also play a role. While the nuclear spin can be addressed with the NMR methods, the electron spin-resonance (ESR) method becomes an equivalent tool for control (and not forgetting ENDOR, electron-nuclear spin double resonance). Computing with quantum dots is based only on electrons and will be discussed in [Sect. 7](#).

A much noted suggestion for using spins in solids was made by Kane (1998). His vision was a purified silicon lattice made of zero-spin ^{28}P atoms, with embedded spin $1/2$ ^{31}P atoms (donors) at regular nanoscale intervals (10–20 nm), and at an equivalent distance from the silicon surface. The hyperfine interaction couples the donor nuclear spin strongly with the donor electron. The advantage is that these spins have very long dephasing times that could, at very low temperatures, reach 10^6 s. Low temperatures are also needed to inhibit the actual ionization of the donors. This must be compared with the expected gate operation time, for which Kane estimated 10^{-5} s. Both figures are probably optimistic but clearly the numbers look a priori promising.

The surface of the silicon lattice would be covered with some barrier material such as SiO_2 (to isolate the donor electrons from the leads), and on top of that one would place the metallic leads (A), positioned above each qubit. With the localized constant magnetic field ($B \sim 2$ T) one can split the degeneracy of the hyperfine states, and then further control the hyperfine resonance frequencies of each qubit independently with a current in the A lead (Stark shift induced by the current-generated electric field). Thus, an applied oscillating field ($B_{\text{AC}} \sim 10^{-3}$ T) would create transitions only at the selected qubit since others would be off-resonant.

The large distance between the donors ensures that they can be controlled independently, and electron-mediated interaction between the donor nuclear spins does not play a role. Conditional logic is, however, achieved through this interaction. Additional leads (J) will be placed between the A leads. When charged positively, they pull donor electrons from adjacent qubits and the electron-enhanced interaction, dependent on the spin states (exchange interaction between the electron spins), is sufficiently strong for gate operations, and its duration is controlled by the J lead. This effects conditional logic in very much the same manner as with liquid NMR systems.

The method is in principle scalable, but it is also quite challenging technologically. Further problems are produced by the readout, since now one deals with single spins, so the NMR signal is not sufficiently large for detection (in liquid NMR, this was circumvented by large numbers). The original scheme by Kane was to combine tunneling with single-electron transistor detectors, but this is considered to be problematic both technologically as well as in the light of decoherence times.

An alternative approach that uses the donor electrons was proposed in 2000 (Vrijen et al. 2000). The approach relies on a layered Si–Ge structure. The important gain is the increase in

the distance between qubits by an order of magnitude, to make lithographic implementation easier. The role of the ^{31}P nuclei is now only to provide the location for the donor electrons. Si and Ge have different electronic g factors, so by moving electrons vertically between two Si–Ge layers with different Si/Ge proportion, one can modify the electron spin resonance for control with ESR (instead of NMR). Operation with electron spins alone speeds up the gate rate (75 MHz vs. 1 GHz). Conditional logic is still achieved with the increase of the electron wavefunction overlap but now this is achieved with the A gates (the possibility to achieve conditional logic with exchange interaction in general was suggested in Loss and DiVincenzo (1998) in the context of quantum dot computers; see also DiVincenzo et al. (2000)).

The measuring scheme (conversion of spin into charge) is also simplified. But many of the technological challenges remain, such as the ^{31}P implantation in regular lattices. Although some progress has been made in the fabrication of the Si:P materials (see, e.g., O'Brien et al. 2001; Jamieson et al. 2005), experimental advances in silicon-based quantum computing are still in the future. Laboratory research on the Si:P computers is nevertheless carried out vigorously; see, for example, Andresen et al. (2007), but clearly there are still many problems remaining even for a demonstration of a two-qubit action. Recently, coherent qubit state transfer between nuclear and electron spins was reported (Morton et al. 2008), though under the name of quantum memory rather than quantum computer studies.

6.3 Nitrogen Vacancies in Diamond

Although single-crystal diamond is nowadays considered an old-fashioned form of carbon, compared to fullerenes and graphene, for quantum computing purposes it has recently emerged as a lucrative system (Prawer and Greentree 2008). Impurities in diamond appear as color centers, giving the specific colors for emeralds, rubies, and sapphires. Proper diamonds are not usually completely transparent but have a yellowish shade, for which there is a rigorous classification scheme. This shade is due to nitrogen impurities in diamond.

The basic ingredient for diamond qubits is a nitrogen vacancy (NV). It means that the nitrogen atom (N) replaces one of the carbon atoms, and in addition next to it there is a vacancy (V) in the carbon lattice. Typically, vacancies move in the lattice, until they find a nitrogen impurity, with which they then form a highly stable local structure. For a production of these NV centers, one can use chemical vapor-deposition or ion-implantation. The electron structure of the NV center is a spin triplet ground state that can be optically coupled with the excited triplet state (giving the yellow color, of course). Thus, one can induce and observe quantum coherences on the level of a single system in the same manner as one would observe single ions in electromagnetic traps (Jelezko et al. 2004). Even quantum jumps have been observed with such a transition.

The triplet ground state is split by the dipolar interaction so that the $m_s = 0$ state is the lowest state, and $m_s = \pm 1$ states are degenerate. This degeneracy of the $m_s = \pm 1$ states can be lifted with an external magnetic field. The splitting of the Zeeman substates is about 2.9 GHz, and can be accessed with microwave fields. The interesting point is that optical excitation is only allowed from the $m_s = 0$ state so that it can be used to detect the occupancy on that state.

A nearby ^{13}C nuclear spin can couple with one of the NV electrons through hyperfine interaction, giving an extra 130 MHz splitting to the electronic states. Thus, one has four states involving the nondegenerate combinations of the electron and nuclear spin. Then one can use optical pumping for the preparation of the initial state, and manipulation with microwave and radio-frequency fields, and make the read-out with a laser again. With this, one can effect

conditional logic between the qubits in nuclear and electron spins (Jelezko et al. 2004). This is reminiscent of the first trapped ion two-qubit gate, where the qubits were stored in a single ion. The coherence properties of the electron spin have been determined experimentally (Childress et al. 2006), and recently a second ^{13}C atom was added to demonstrate the quantumness of the system by creation and detection of tripartite quantum entanglement (Neumann et al. 2008). In this room-temperature experiment, the weak point is the electron spin decoherence, which is a few microseconds. If one limits to entangling only the two nuclear spins, the time scale is pushed into the millisecond region.

The main issue with NV centers is obviously scalability, which cannot be achieved with the present setup. One solution could be an optical bus between, for example, different small crystals that host small numbers of qubits. But for the time being the NV centers are mainly suitable for demonstration purposes.

7 Solid-State Qubits with Quantum Dots

Quantum dots emerged as suitable candidates for quantum computing very early (Barenco et al. 1995; Loss and DiVincenzo 1998; Sherwin et al. 1999). The discrete energy states of the dots provide a tempting qubit representation, and the semiconductor technology is highly developed at nanoscale due to the continuous effort to follow Moore's law. The possibility for optical manipulation and the use of electron spin (spintronics) extend the possibilities further (see, e.g., the book Awschalom et al. 2002).

A quantum dot is usually obtained by preparing layered heterostructures, with electrons on the conduction band trapped into one of the layers (quantum wells), and the 2D confinement is augmented with trapping of electrons with fields provided by the electrodes that are etched on the surface of the structure (lateral quantum dot structure). One can obtain single-electron trapping by using the Coulomb blockade: a charge can only tunnel into a dot from the adjacent layer if no other charges are there, as they would block the tunneling by electric repulsion. This aspect can be further controlled by external electric fields that adjust the energy states of the dot in relation to the levels at adjacent layers. This is the basis for single-electron transistors and other such devices. The above is an example of laterally positioned and coupled dots, but vertical, multilayered structures are also possible (Burkard et al. 1999).

For scalable qubits, one could use the quantized motional states of the electrons, and for conditional logic the dipole–dipole interaction, which can be made to depend on the electron state by external fields that affect the electronic charge distributions (as proposed by Barenco et al. 1995). Alternatively, one can use the electron spin as a qubit, control the spin states by ESR fields, and use the spin-dependent exchange interaction for conditional logic, as proposed by Loss and DiVincenzo (1998).

The spin-based systems have been popular and extensive studies have been performed, not only on quantum computation but also on the dynamics and observation of spins in few electron quantum dots (see the recent review by Hanson et al. (2007)). Having more than one electron in a dot creates a singlet-triplet electron-spin structure that can be addressed with magnetic fields. An alternative system with a singlet-triplet structure is a double quantum dot, with a controllable interdot tunneling, see, for example, (Koppens et al. 2007). The tunneling is also controlled by a spin blockade, that is, to conserve the total spin a change between singlet and triplet states (and thus a specific tunneling event between dots) can only occur if a

spin-flip takes place as well. The observed coherence times (Koppens et al. 2006) for a single qubit are promising on the order of microseconds (at temperatures below 1 K), but further work is required to estimate the possibilities of spin-based quantum dot qubits.

Finally, electrons can be transferred from the valence band to the conduction band by optical transition; thus one obtains an exciton, that is, an electron–hole pair, that can act as a qubit (Troiani et al. 2000). One dot can possess more than one exciton. In fact, one of the first demonstrations of a quantum dot quantum gate (Li et al. 2003) took place (in analogy to the trapped ion demonstration (Monroe et al. 1995)) in a single quantum dot with such a biexciton as the two-qubit system. The gate operation relied on different light polarizations since for the qubits the value “1” corresponded to different electron spins in the two electron–hole pairs. The gate that was performed was actually the CROT gate. Earlier, entanglement between the excitons was demonstrated in such a system (Chen et al. 2000). To make the excitonic computers scalable is a challenge, though.

In general, the semiconductors provide a challenging environment for quantum phenomena. In addition to coupling with phonons, the quantum dot electrons feel the other electrons through spin–orbit coupling, and the nuclear spins of the background material via hyperfine interaction. Apparently the latter mechanism is found to be the more relevant mechanism in experiments (Koppens et al. 2005, 2006). For excitons, the recombination of electrons and holes by spontaneous emission (coupling to the electromagnetic degrees of freedom) is a serious obstacle. It is possible to reach gate times on the order of 1 ps, and recombination times on the order of hundreds of picoseconds have been obtained (see discussion in Li et al. 2003), so more than a few gate operations could be achieved. For scalability, qubits in different dots could be coupled by setting them into an optical microcavity, so that excitons at different dots couple to the same cavity mode, and even interactions mediated by intermediate excitons have been proposed; see Chapter 7 in Chen et al. (2007) and references therein.

8 Superconducting Qubits

Superconductivity is a quantum phenomenon that takes place for certain materials at low temperatures (Tinkham 2004). The electric current runs then without resistance in the material, and such currents are quite unaffected by external conditions, except for sufficiently strong magnetic fields. In the standard case of superconductivity, the electrons in the material form Cooper pairs that are the unstoppable current carriers, and the resistance can occur only by a mechanism that provides sufficient energy that breaks the pair. However, the Cooper pairs and the single-electron states are separated by a distinct pairing gap in energy, which at low temperatures exceeds the thermal energy in the system.

Thus superconductivity is a prime example of a macroscopic quantum coherent system, and one can attach to the degenerate electron gas a wavefunction-like order parameter that has a specific phase. The spatial gradient of that phase gives the superconducting current. Superfluidity of liquids and Bose–Einstein condensation of atomic or molecular gases are similar low-temperature examples with overall phase coherence. The connection between the phase and current means that since the phase change in a closed loop can only be a multiple of 2π , the flux related to it must be quantized. If the loop can shrink without limit in the material, it leads to quantized vortices, which are observed especially with rotated or stirred liquids and gases (Tilley and Tilley 1990). For superconductors, one can build circuits so that the loops

cannot shrink to zero size, and thus one can observe the quantization of the persistent superconducting current.

A superconducting current can pass through small regions of insulating material by quantum tunneling. One mechanism is by breaking the Cooper pair during the process (quasiparticle tunneling), or the Josephson tunneling of Cooper pairs. Typically, the latter effect is the more interesting one, and these junctions are often called Josephson or tunneling junctions. It is a quantum phenomena, since it is controlled by the phase difference ϕ over the junction, which means that the tunneling is a coherent process. The time derivative of ϕ gives the voltage over the junction, and the current through it is $I = I_c \sin(\phi)$, where I_c is the critical current that sets the upper limit. Interestingly, a current can exist across the junction even if the phase difference is constant and there is no voltage difference (DC Josephson effect). An externally created electric potential V across the junction gives a phase difference $\phi = 2eVt/\hbar$ and thus an alternating current (AC Josephson effect). The maximum of the current depends on the energy gap of the superconducting material and the temperature. SQUIDs (superconducting quantum interference devices) are based on the sensitivity of this current to external magnetic fields through the field-dependence of the energy gap. For actual insulating material, the barrier thickness should not exceed a few hundred nanometers, but for normal state metal one can go to the micrometer scale.

8.1 Charge Qubits

If one prepares a simple setup with one Josephson junction and two superconducting regions, coupled into a loop by a voltage source, one gets the so called Cooper pair box. Typically, one of the superconducting regions is small and is called an island. The fascinating aspect is that although we have countless Cooper pairs on the island, its electrostatic properties can be governed by a single Cooper pair. We can tune the system so that, initially, there is no net charge on the island (zero excess Cooper pairs). The junction has capacitance C , and there is another one (C_e) between the island and the external circuit. The Hamiltonian for the system is then (Nakahara and Ohmi 2008)

$$H = \frac{1}{2} E_c (n - n_e)^2 - E_J \cos \phi, \quad E_c = \frac{(2e)^2}{C + C_e}, \quad n_e = \frac{C_e V_e}{2e}, \quad E_J = \frac{I_c \hbar}{2e} \quad (21)$$

where V_e is the applied external voltage across the system and n is the Cooper pair number of the island. Note that n_e is a continuous quantity whereas n is limited to integer values.

It shows that one can have a field-induced degeneracy between n and $n + 1$ Cooper pairs on an island. At this degeneracy, tunneling is possible, and the two charge states are in fact coupled by the tunneling coupling E_J (this is not a true coupling; it appears as a frame of reference term since we write the system in terms of Cooper pairs localized on either side of the junction, which is not true because of tunneling). The charge number n and the junction phase ϕ form a conjugate pair of quantities (in analogy to x and p), and thus $\cos \phi$ can be interpreted as a term that induces $n \rightarrow n \pm 1$ transitions between the charge parabola.

One can also have a superposition of charge states, and this forms the charge qubit (Shnirman et al. 1997; Makhlin et al. 1999, 2000). By driving the system into the resonance point (changing n_e) one can observe coherent oscillations (Rabi oscillations) driven by E_J , or by traversing the degeneracy with fixed speed one obtains Landau-Zener transitions. Coherent oscillations and their decay were observed already in 1999 (Nakamura et al. 1999). Such charge

qubits can be coupled, for example, capacitatively, as demonstrated in 2003 (Pashkin et al. 2003), followed closely by a demonstration of conditional logic (Yamamoto et al. 2003). The superconducting systems are very robust to perturbations, but the electronics needed for manipulating and detecting the qubit states seem to be a sufficient source for non-negligible noise.

8.2 Flux Qubits

Parallel to the development of superconducting charge qubits, another idea emerged, namely the use of the current itself (Mooij et al. 1999; Friedman et al. 2000). The idea is to prefabricate the junction as a part of a small ring of superconducting material, so that the tunneling coupling E_J dominates over the charge state separation E_C . This is the standard RF-SQUID setup, and such a ring can then have a persistent current in it, controlled by the external magnetic field flux Φ_e . One can show that, due to the existence of quantum phase coherence in the ring, the total magnetic flux Φ through the ring must obey a quantization rule, so that $\Phi = (\phi + 2\pi n)\Phi_0/(2\pi)$, where the flux quantum is $\Phi_0 = \hbar/(2e)$ and ϕ is the phase change over the junction. It means that to satisfy this relation, a current I arises in the ring, so that $\Phi = \Phi_e + LI$, where L is the self-inductance of the ring. As described, for example, in Friedman et al. (2000), the flux can have the appearance of a position-like quantity, with kinetic energy $C\dot{\Phi}^2/2$ and potential

$$U = U_0 \left[\frac{1}{2} \left(\frac{2\pi(\Phi - \Phi_e)}{\Phi_0} \right)^2 - \beta \cos(2\pi\Phi/\Phi_0) \right] \quad (22)$$

where C is the junction capacitance, $U_0 = \Phi_0^2/(4\pi^2 L)$ and $\beta = 2\pi L I_c / \Phi_0$, with I_c again as the critical current of the junction.

The above potential is for practical purposes a double-well structure with two minima. The β term arises from the phase change over the junction. For $\Phi_e = \Phi_0/2$ one obtains a symmetric structure, and let one consider it for simplicity. The two wells correspond to flux quanta $n = 0$ and $n = 1$, and to equal currents flowing in opposite directions. These fluxes and accompanying supercurrents form the flux qubit. At the bottom of the wells and for large E_J they are quite stable and robust to perturbations. For smaller E_J , the true lowest quantum states spread over the whole double-well and are roughly the symmetric and antisymmetric superpositions of the localized states, and their degeneracy is split by the tunnel coupling. Variations in Φ_e make the potential antisymmetric, separating the left- and right-localized states in energy, and thus reducing the effect of the tunnel coupling. Such coherent superpositions were observed experimentally in 2000 (Friedman et al. 2000; van der Wal et al. 2000).

The system can be further refined by adding more junctions, such as three or four (Mooij et al. 1999); this makes the size of the setup smaller and reduces possible disturbances. This is because with one junction a large inductance L and, thus, a large size is needed to allow for the generation of the sufficiently large compensation flux; in the multijunction setup the compensation flux can be as small as $10^{-3}\Psi_0$. The double-well description is nevertheless still fine (van der Wal et al. 2000). The energy separation of the flux states is in the microwave region and one can thus perform single-qubit operations; here the disturbances in the external flux cause decoherence, and its time scale has been estimated to be around 20 ns by considering Ramsay interferometry with two $\pi/2$ pulses (Chirosescu et al. 2003).

The state of a flux qubit can be read out with various schemes, see van der Wal et al. (2000) and Chiorescu et al. (2003). The conditional logic can be obtained, for example, by circling the qubits with another superconducting ring so that a four-state system emerges, with two-qubit states that can be manipulated with microwave fields (Mooij et al. 1999; Hime et al. 2006; van der Ploeg et al. 2007). Inductive coupling can be achieved by making the superconducting boxes for two qubits share one edge (Majer et al. 2005). A promising scheme is to use the shared edge method but having an auxiliary qubit between the actual qubits with a large tunneling energy, so that it can be adiabatically eliminated. With this method, conditional logic has been demonstrated with four-junction qubits experimentally (Niskanen et al. 2007).

8.3 Phase Qubits and Circuit QED

One can also place a strong bias current $I_e \simeq I_c$ on the Josephson junction when $E_J \gg E_c$. The potential for the phase of the current becomes a washboard potential that has shallow local minima with a few bound states. A similar situation can be obtained with the RF-SQUID by setting a large Φ_e that tilts the potential strongly. The two lowest states form a phase qubit. Transitions between them can be driven with microwave fields, and detection can happen by state-dependent tunneling that can be observed as a direct current. The qubit circuits are coupled with a capacitor, and one can excite entangled states with microwave radiation; this has been demonstrated experimentally (Berkley et al. 2003; Steffen et al. 2006), but an actual demonstration of conditional logic is still missing. Decoherence times are on the order of 100 ns, compared to the duration of single qubit operations that is an order of magnitude less (Steffen et al. 2006).

In above examples, the superconducting system states were usually controlled with microwave radiation. This has also led to the idea of using microwave cavities for transferring quantum information (Blais et al. 2004) and, hence, to the concept of circuit QED. A coupling of a single charge qubit to a cavity mode has been demonstrated (Shuster et al. 2004). Recently, such a coupling of two phase qubits was demonstrated (Majer et al. 2007). The qubits consisted of two parallel junctions so that the tunneling coupling and, thus, the energy separation for the qubit states were controllable with external magnetic flux through the qubit system. Both qubits were coupled to a finite-length coplanar waveguide and separated by a distance of about 4 mm. They did not exchange actual photons with the cavity because they were off-resonant with it, but their presence induced a dispersive qubit–qubit coupling that shifted resonances for joint states, which was spectroscopically detected. Also, resonant coupling of a phase qubit to the cavity mode has been demonstrated, including the transfer of quantum information from the phase qubit to the photon (Houck et al. 2007). The actual exchange of a photon between two qubits through a resonant cavity coupling has been demonstrated as well (Sillanpää et al. 2007). Although still at a very preliminary phase, this approach is especially interesting as it opens up the possibility of well-separated qubits.

9 All-Optical Quantum Computing

Photons are the ultimate tools for quantum communication, so it is natural to ask if they can be applied in large-scale quantum computing as well. The problem with photons is that they are generated or destroyed quite easily, and are thus very different from more physical forms of

qubit systems. Also, they do not interact but require some medium for the process, so conditional logic with linear optical elements is not possible (starting with Bell state detection that would be simple if a linear optics CNOT gate was available (Lütkenhaus et al. 1999)). However, it is possible to circumvent this problem by using photodetectors and feedback loops as was shown in 2001 (Knill et al. 2001).

Demonstrations for simple operations in all-optical quantum computing have been performed (see, e.g., O'Brien et al. 2003), but it remains to be seen whether the actual large scale realization is a feasible goal. For a nice recent review, O'Brien (2007) is suggested.

10 Closing Remarks

The large-scale implementation of quantum computing is still clearly far beyond current experimental capabilities, as the examples given in this review demonstrate. Many ideas are only at a preliminary stage although vigorous work is going on. At least the large number of Science and Nature articles show that many of the proposed systems have broad interest, hopefully because they either hold great promise or are based on ingenious theoretical ideas and innovative experimental work. Both are definitely needed. It has been 15 years since the appearance of the Shor algorithm, but we are not yet even close to a working quantum computer. The future will show how well the attractive character of quantum information persists. Since all these attempts improve our understanding of quantum mechanics and its manifestations, and are usually linked with nanophysics research, it is quite possible that the most valuable outcome will eventually be something other than a working, large-scale quantum computer.

Acknowledgments

The author acknowledges the financial support by the Academy of Finland, the Väisälä Foundation, and the Magnus Ehrnrooth Foundation.

References

- Andresen SES, Brenner R, Wellard CJ et al. (2007) Charge state control and relaxation in an atomically doped silicon device. *Nano Lett* 7:2000–2003
- Awschalom DD, Loss D, Samarth N (eds) (2002) Semiconductor spintronics and quantum computation. Springer, Berlin
- Barenco A, Deutsch D, Ekert A (1995) Conditional quantum dynamics and logic gates. *Phys Rev Lett* 74:4083–4086
- Barenco A, Ekert A, Suominen K-A, Törmä P (1996) Approximate quantum Fourier transform and decoherence. *Phys Rev A* 54:139–146
- Benhelm J, Kirchmair G, Roos CF, Blatt R (2008) Towards fault-tolerant quantum computing with trapped ions. *Nat Phys* 4:463–466
- Bennett CH, DiVincenzo DP, Smolin JA, Wootters WK (1996) Mixed state entanglement and quantum error correction. *Phys Rev A* 54:3824–3851
- Berkley AJ, Xu H, Ramos RC et al. (2003) Entangled macroscopic quantum states in two superconducting qubits. *Science* 300:1548–1550
- Birkl G, Kassner S, Walther H (1992) Multiple-shell structures of laser-cooled $^{24}\text{Mg}^+$ ions in a quadrupole storage ring. *Nature* 357:310–313
- Briegel HJ, Raussendorf R (2001) Persistent entanglement in arrays of interacting particles. *Phys Rev Lett* 86:910–913
- Blais A, Huang R-S, Wallraff A, Girvin SM, Schoelkopf RJ (2004) Cavity quantum electrodynamics for superconducting electrical circuits: an architecture for quantum computation. *Phys Rev A* 69:062320
- Bloch I (2005) Ultracold quantum gases in optical lattices. *Nat Phys* 1:23–30
- Bremner MJ, Dawson CM, Dodd JL et al. (2002) Practical scheme for quantum computation with any two-qubit entangling gate. *Phys Rev Lett* 89:247902

- Burkard G, Loss D, DiVincenzo DP (1999) Coupled quantum dots as quantum gates. *Phys Rev B* 59:2070–2078
- Calderbank AR, Shor PW (1996) Good quantum error-correcting codes exist. *Phys Rev A* 54:1098–1105
- Chen G, Bonadeo NH, Steel DG et al. (2000) Optically induced entanglement of excitons in a single quantum dot. *Science* 289:1906–1909
- Chen G, Church DA, Englert B-G et al. (2007) Quantum computing devices. Chapman & Hall/CRC, Boca Raton, FL
- Childress L, Gurudev Dutt MV, Taylor JM et al. (2006) Coherent dynamics of coupled electron and nuclear spin qubits in diamond. *Science* 314:281–285
- Childs AM, Farhi E, Preskill J (2001) Robustness of adiabatic quantum computation. *Phys Rev A* 65:012322
- Chiorescu I, Nakamura Y, Harmans CJPM, Mooij JE (2003) Coherent quantum dynamics of a superconducting flux qubit. *Science* 299:1869–1871
- Chuang IL, Gershenfeld N, Kubinec M (1998) Experimental implementation of fast quantum searching. *Phys Rev Lett* 80:3408–3411
- Chuang IL, Vandersypen LMK, Zhou X et al. (1998) Experimental realization of a quantum algorithm. *Nature* 393:143–146
- Cirac JI, Zoller P (1995) Quantum computations with cold trapped ions. *Phys Rev Lett* 74:4091–4094
- Cleve R, Ekert A, Macchiavello C, Mosca M (1998) Quantum algorithms revisited. *Proc R Soc Lond A* 454:339–354
- Coppersmith D (1994) An approximate Fourier transform useful in quantum factoring. IBM Research Report RC19642; see also arXiv quant-ph/0201067 (2002)
- Dahm AJ, Heilman JA, Karakurt I, Peshek TJ (2003) Quantum computing with electrons on helium. *Physica E* 18:169–172
- DiVincenzo DP (1995) Quantum computation. *Science* 270:255–261
- DiVincenzo DP (2000) The physical implementation of quantum computation. *Fortschritte der Physik* 48:771–783
- DiVincenzo DP, Bacon D, Kempe J et al. (2000) Universal quantum computation with the exchange interaction. *Nature* 408:339–342
- Duan L-M, Guo G-C (1997) Preserving coherence in quantum computation by pairing quantum bits. *Phys Rev Lett* 79:1953–1956
- Dykman MI, Platzman PM (2000) Quantum computing using electrons floating on liquid helium. *Fortschritte der Physik* 48:1095–1108
- Dykman MI, Platzman PM, Sedighrad P (2003) Qubits with electrons on liquid helium. *Phys Rev B* 67:155402
- van Enk SJ, Kimble HJ, Mabuchi H (2004) Quantum information processing in cavity-QED. *Quantum Inf Process* 3:75–90
- Farhi E, Goldstone J, Gutmann S, Sipser M (2000) Quantum computation by adiabatic evolution. arXiv quant-ph/0001106
- Feynman RP (1982) Simulating physics with computers. *Int J Theor Phys* 21:467–488
- Friedman JR, Patel V, Chen W et al. (2000) Quantum superposition of distinct macroscopic states. *Nature* 406:43–46
- Gaitan F (2008) Quantum error correction and fault tolerant quantum computing. CRC Press, Boca Raton, FL
- Gershenfeld NA, Chuang IL (1997) Bulk spin-resonance quantum computation. *Science* 275:350–356
- Gottesman D (1996) Class of quantum error-correcting codes saturating the quantum Hamming bound. *Phys Rev A* 54:1862–1868
- Grimes CC, Adams G (1979) Evidence for a liquid-to-crystal phase transition in a classical, two-dimensional sheet of electrons. *Phys Rev Lett* 42:795–798
- Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. *Phys Rev Lett* 79:325–328
- Häffner H, Roos CF, Blatt R (2008) Quantum computing with trapped ions. *Phys Rep* 469:155–203
- Hanson R, Kouwenhoven LP, Petta JR et al. (2007) Spins in few-electron quantum dots. *Rev Mod Phys* 79:1217
- Haroche S, Raimond J-M (1996) Quantum computing: dream or nightmare? *Phys Today* 49(8):51–52
- Haroche S, Raimond J-M (2006) Exploring the quantum: atoms, cavities and photons. Oxford University Press, Oxford
- Hime T, Reichardt PA, Plourde BLT et al. (2006) Solid-state qubits with current-controlled coupling. *Science* 314:1427–1429
- Hirvensalo M (2004) Quantum computing, 2nd edn. Springer, Berlin
- Horvath GZsK, Thompson RC, Knight PL (1997) Fundamental physics with trapped ions. *Contemp Phys* 38:25–48
- Houck AA, Schuster DI, Gambetta JM et al. (2007) Generating single microwave photons in a circuit. *Nature* 449:328–331
- Jamieson DN, Yang C, Hopf T et al. (2005) Controlled shallow single-ion implantation in silicon using an active substrate for sub-20-keV ions. *Appl Phys Lett* 86:202101
- Jelezko F, Gaebel T, Popa I et al. (2004) Observation of coherent oscillations in a single electron spin. *Phys Rev Lett* 92:076401
- Jelezko F, Gaebel T, Popa I et al. (2004) Observation of coherent oscillation of a single nuclear spin and realization of a two-qubit conditional quantum gate. *Phys Rev Lett* 93:130501

- Jones JA (2000) NMR quantum computation: a critical evaluation. *Fortschritte der Physik* 48:909–924
- Jones JA, Mosca M, Hansen RH (1998) Implementation of a quantum search algorithm on a quantum computer. *Nature* 393:344
- Kane BE (1998) A silicon-based nuclear spin quantum computer. *Science* 393:133–137
- Kielpinski D, Monroe C, Wineland DJ (2002) Architecture for a large-scale ion-trap quantum computer. *Nature* 417:709
- Kim J, Lee J-S, Lee S (2002) Experimental realization of a target-accepting quantum search by NMR. *Phys Rev A* 65:054301
- Kitaev A (1997) Fault-tolerant quantum computation by anyons. *Ann Phys* 303:2–30. see also arXiv quant-ph/9707021
- Knill E (2005) Quantum computing with realistically noisy devices. *Nature* 434:39–44
- Knill E, Laflamme R, Milburn GJ (2001) A scheme for efficient quantum computation with linear optics. *Nature* 409:46–52
- Koppens FHL, Buzert C, Tielrooij KJ et al. (2006) Driven coherent oscillations of a single electron spin in a quantum dot. *Nature* 442:766–771
- Koppens FHL, Folk JA, Elzerman JM et al. (2005) Control and detection of singlet-triplet mixing in a random nuclear field. *Science* 309:1346–1350
- Laflamme R, Miquel C, Paz JP, Zurek WH (1996) Perfect quantum error correcting code. *Phys Rev Lett* 77:198–201
- Li X, Wu Y, Steel D et al. (2003) An all-optical quantum gate in a semiconductor quantum dot. *Science* 301:809–811
- Lidar DA, Chuang IL, Whaley KB (1998) Decoherence-free subspaces for quantum computation. *Phys Rev Lett* 81:2594–2597
- Loss D, DiVincenzo DP (1998) Quantum computation with quantum dots. *Phys Rev A* 57:120–126
- Lütkenhaus N, Calsamiglia J, Suominen K-A (1999) Bell measurements for teleportation. *Phys Rev A* 59:3295–3300
- Lyon SA (2006) Spin-based quantum computing using electrons on liquid helium. *Phys Rev A* 74:052338
- Majer J, Chow JM, Gambetta JM et al. (2007) Coupling superconducting qubits via a cavity bus. *Nature* 449:443–447
- Majer JB, Paauw FG, ter Haar ACJ et al. (2005) Spectroscopy on two coupled superconducting flux qubits. *Phys Rev Lett* 94:090501
- Makhlin Y, Schön G, Shnirman A (1999) Josephson-junction qubits with controlled couplings. *Nature* 398:305–307
- Makhlin Y, Schön G, Shnirman A (2000) Quantum-state engineering with Josephson-junction devices. *Rev Mod Phys* 73:357–400
- Mermin ND (2007) Quantum computer science: an introduction. Cambridge University Press, Cambridge
- Metcalf HJ, Van der Straten P (1999) Laser cooling and trapping. Springer, Berlin
- Mollin RA (2002) RSA and public-key cryptography. CRC Press, Boca Raton, FL
- Monroe C, Meekhof DM, King BE et al. (1995) Demonstration of a fundamental quantum logic gate. *Phys Rev Lett* 75:4714–4717
- Monz T, Kim K, Hänsel W et al. (2009) Realization of the quantum Toffoli gate with trapped ions. *Phys Rev Lett* 102:040501
- Mooij JE, Orlando TP, Levitov L et al. (1999) Josephson persistent-current qubit. *Science* 285:1036–1039
- Morton JJL, Tyryshkin AM, Brown RM et al. (2008) Solid-state quantum memory using the ^{31}P nuclear spin. *Nature* 455:1085–1088
- Nägerl HC, Bechter W, Eschner J et al. (1998) Ion strings for quantum gates. *Appl Phys B* 66:603–608
- Nakahara M, Ohmi T (2008) Quantum computing: from linear algebra to physical realizations. CRC Press, Boca Raton, FL
- Nakamura Y, Pashkin YA, Tsai JS (1999) Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature* 398:786–788
- Nayak C, Simon SH, Stern A et al. (2008) Non-Abelian anyons and topological quantum computation. *Rev Mod Phys* 80:1083
- Negrevergne C, Mahesh TS, Ryan CA et al. (2006) Benchmarking quantum control methods on a 12-qubit system. *Phys Rev Lett* 96:170501
- Neumann P, Mizuchi N, Rempp F et al. (2008) Multiparticle entanglement among single spins in diamond. *Science* 320:1326–1329
- Nielsen MA, Chuang IL (2000) Quantum computation and quantum information. Cambridge University Press, Cambridge
- Niskanen AO, Harrabi K, Yoshihara F et al. (2007) Quantum coherent tunable coupling of superconducting qubits. *Science* 316:723–726
- O'Brien JL (2007) Optical quantum computing. *Science* 318:1567–1570
- O'Brien JL, Pryde GJ, White AG et al. (2003) Demonstration of an all-optical quantum controlled-NOT gate. *Nature* 426:264–267
- O'Brien JL, Schofield SR, Simmons MY et al. (2001) Towards the fabrication of phosphorus qubits for a silicon quantum computer. *Phys Rev B* 64:161401
- Oliveira I, Sarthour R Jr, Bonagamba T et al. (2007) NMR quantum information processing. Elsevier, Amsterdam
- Palma GM, Suominen K-A, Ekert AK (1996) Quantum computers and dissipation. *Proc R Soc Lond A* 452:567–584

- Pashkin YA, Yamamoto T, Astafiev O et al. (2003) Quantum oscillations in two coupled charge qubits. *Nature* 421:823–826
- Pethick CJ, Smith H (2008) Bose-Einstein condensation in dilute gases, 2nd edn. Cambridge University Press, Cambridge
- Platzman PM, Dykman MI (1999) Quantum computing with electrons floating on liquid helium. *Science* 284:1967–1969
- van der Ploeg SHW, Izmalkov A, van den Brink AM et al. (2007) Controllable coupling of superconducting flux qubits. *Phys Rev Lett* 98:057004
- Prawer S, Greentree AD (2008) Diamond for quantum computing. *Science* 320:1601–1602
- Raussendorf R, Briegel HJ (2001) A one-way quantum computer. *Phys Rev Lett* 86:5188–5191
- Rousseau E, Ponarin D, Hristakos L et al. (2009) Addition spectra of Wigner islands of electrons on superfluid helium. *Phys Rev B* 79:045406
- Sabouret G, Lyon SA (2006) Measurement of the charge transfer efficiency of electrons clocked on superfluid helium. *App Phys Lett* 88:254105
- Sachdev S (1999) Quantum phase transitions. Cambridge University Press, Cambridge
- Schaetz T, Leibfried D, Chiaverini J et al. (2004) Towards a scalable quantum computer/simulator based on trapped ions. *Appl Phys B* 79:979–986
- Schmidt-Kaler F, Häffner H, Gulde S et al. (2003a) How to realize a universal quantum gate with trapped ions. *Appl Phys B* 77:789–796
- Schmidt-Kaler F, Häffner H, Riebe M et al. (2003b) Realization of the Cirac-Zoller controlled-NOT quantum gate. *Nature* 422:408–411
- Schuster DI, Blais A, Frunzio L et al. (2004) Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *Nature* 431:162–167
- Sherwin MS, Imamoglu A, Montroy T (1999) Quantum computation with quantum dots and terahertz cavity quantum electrodynamics. *Phys Rev A* 60:3508–3514
- Shnirman A, Schön G, Hermon Z (1997) Quantum manipulations of small Josephson junctions. *Phys Rev Lett* 79:2371–2374
- Shor PW (1994) Algorithms for quantum computation: discrete log and factoring. Proceedings of the 35th annual IEEE symposium on foundations of computer science, FOCS, Paris, pp 20–22
- Shor PW (1995) Scheme for reducing decoherence in quantum computer memory. *Phys Rev A* 52: R2493–R2496
- Sillanpää MA, Park JI, Simmonds RW (2007) Coherent quantum state storage and transfer between two phase qubits via a resonant cavity. *Nature* 449: 438–442
- Sjöqvist E (2008) A new phase in quantum computation. *Physics* 1:35
- Sørensen A, Mølmer K (1999a) Quantum computation with ions in thermal motion. *Phys Rev Lett* 82: 1971–1974
- Sørensen A, Mølmer K (1999b) Entanglement and quantum computation with ions in thermal motion. *Phys Rev A* 62:022311
- Steane AM (1996) Error correcting codes in quantum theory. *Phys Rev Lett* 77:793–797
- Steane AM (1998) Quantum computing. *Rep Prog Phys* 61:117–173
- Steffen M, Ansmann M, Bialczak RC et al. (2006) Measurement of the entanglement of two superconducting qubits via state tomography. *Science* 313: 1423–1425
- Stenholm S, Suominen K-A (2005) Quantum approach to informatics. Wiley, Hoboken, NJ
- Stolze J, Suter D (2008) Quantum computing, revised and enlarged: a short course from theory to experiment, 2nd edn., Wiley-VCH, Weinheim
- Tilley DR, Tilley J (1990) Superfluidity and superconductivity. Taylor & Francis, London
- Tinkham M (2004) Introduction to superconductivity, 2nd edn. Dover, New York
- Troiani F, Hohenester U, Molinari E (2000) Exploiting exciton-exciton interactions in semiconductor quantum dots for quantum-information processing. *Phys Rev B* 62:R2263–R2266
- Turchette QA, Hood CJ, Lange W, Mabuchi H, Kimble HJ (1995) Measurement of conditional phase shifts for quantum logic. *Phys Rev Lett* 75:4710–4713
- Unruh WG (1995) Maintaining coherence in quantum computers. *Phys Rev A* 51:992–997
- Vandersypen LMK, Chuang IL (2004) NMR techniques for quantum control and computation. *Rev Mod Phys* 76:1037–1069
- Vandersypen LMK, Steffen M, Sherwood MH et al. (2000) Implementation of a three-quantum-bit search algorithm. *Appl Phys Lett* 76:646–649
- Vandersypen LMK, Steffen M, Breyta G et al. (2001) Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature* 414:883–887
- Vedral V (2006) Introduction to quantum information science. Oxford University Press, Oxford
- Vrijen R, Yablonovitch E, Wang K et al. (2000) Electron-spin-resonance transistors for quantum computing in silicon-germanium heterostructures. *Phys Rev A* 62:012306
- van der Wal CH, ter Haar ACJ, Wilhelm FK et al. (2000) Quantum superposition of macroscopic persistent-current states. *Science* 290:773–777
- Yamamoto T, Pashkin YA, Astafiev O et al. (2003) Demonstration of conditional gate operation using superconducting charge qubits. *Nature* 425:941–944
- Zanardi P, Rasetti M (1997) Noiseless quantum codes. *Phys Rev Lett* 79:3306–3309

45 Quantum Cryptography

Takeshi Koshiya

Graduate School of Science and Engineering, Saitama University, Japan
koshiya@mail.saitama-u.ac.jp

1	<i>Introduction</i>	1522
2	<i>Classical Cryptography: Overview</i>	1523
3	<i>Quantum Computational Cryptographic Primitive</i>	1528
4	<i>Quantum Key Distribution</i>	1530
5	<i>Quantum Public-Key Encryption</i>	1532
6	<i>Quantum Digital Signature</i>	1536
7	<i>Quantum Commitment and Oblivious Transfer</i>	1537
8	<i>Quantum Zero-Knowledge Proof</i>	1539

Abstract

Several results in quantum cryptography will be surveyed in this chapter. After a brief introduction to classical cryptography, we provide some cryptographic primitives from the viewpoint of quantum computational complexity theory, which are helpful to get an idea of quantum cryptographic protocols. We then examine cryptographic protocols of quantum key distribution, quantum bit commitment, quantum oblivious transfer, quantum zero-knowledge, quantum public-key encryption, quantum digital signature, and their security issues.

1 Introduction

Due to the rapid growth of electronic communication means, information security has become a crucial issue in the real world. Modern cryptography provides fundamental techniques for securing communication and information. While modern cryptography varies from encryption and digital signatures to cryptographic protocols, we can partition modern cryptography into public-key cryptography and secret-key cryptography. From the theoretical point of view, public-key cryptography has a computational complexity-theoretic flavor and secret-key cryptography has an information-theoretic flavor. Though the two disciplines of cryptography have different flavors, they are not separate from each other but rather complement each other.

The principal and classical task of cryptography is to provide confidentiality. Besides confidentiality, modern cryptography provides authentication, data integrity, and so on. Since Diffie and Hellman (1976) devised the notion of public-key cryptosystems, computational complexity theoretic approaches to cryptology have succeeded in theory and practice. The fundamental study of one-way functions and pseudorandom generators has developed computational complexity theory. Cryptographic protocols such as digital signatures, commitment schemes, oblivious transfer schemes, and zero-knowledge proof systems have contributed to building various information security systems. For each objective mentioned above, we should make models of adversaries so as to enable discussion of whether some cryptographic protocols or methods fulfill the objective. For public-key cryptography, it is typically thought that the adversary should be a probabilistic polynomial-time Turing machine or a polynomial-size circuit family. For secret-key cryptography, the adversary might be the Almighty or those who have some specific power.

It is when the adversary is physically realized that it can become a real threat. It is known that the real world behaves quantum mechanically. Thus, we may suppose that the adversary should run quantum mechanically. Shor's algorithm on quantum computers for the integer factorization problem (Shor 1997) illustrates that quantum adversaries would spoil the RSA cryptosystem (Rivest et al. 1978), which is widely used for secure communications. Shor also proposed an efficient quantum algorithm for the discrete logarithm problem. Moreover, since the security of many cryptographic protocols relies on the computational hardness of these two problems, cryptographic protocols might have to be reconstructed to maintain information security technologies in the future if the quantum adversary has a physical implementation.

The quantum mechanism also has an impact on the other discipline of cryptography, i.e., secret-key cryptography. In 1984, Bennett and Brassard (1984) proposed a quantum key

distribution scheme, which is a key agreement protocol using quantum communication. For the last two decades, so-called quantum cryptography has dramatically developed. For example, its unconditional security proofs were provided, some alternatives were proposed, and so on. We should especially mention that Mayers (1997) and Lo and Chau (1997) independently demonstrated that quantum mechanics cannot necessarily make all cryptographic schemes unconditionally secure.

As mentioned, secret-key cryptography enjoys the benefits of the quantum mechanism. On the other hand, Shor's algorithms can be regarded as a negative effect of the quantum mechanism on public-key cryptography. From the computational point of view, his algorithms illustrate that the quantum mechanism could make the computation more powerful. It is natural to consider that the power of the quantum mechanism could open up an era of neo-modern cryptography. Over the last decade, new possibilities (such as the unconditional security of the BB84 protocol) of quantum cryptography have been explored. Moreover, since the negative impact on public-key cryptography due to Shor, quantum cryptography has been also discussed and developed from the complexity-theoretic point of view. These investigations have promoted the development of quantum information theory and quantum computational complexity theory. In this chapter, we will survey what has been studied in quantum cryptography.

2 Classical Cryptography: Overview

2.1 One-Time Pad

The invention of Shannon's information theory initiated the mathematical study of communication. He also showed that the *one-time pad* has perfect secrecy and it is the only way to achieve perfect secrecy.

Here, a basic definition of classical cryptosystems will be given. A *symmetric-key cryptosystem* is a quintuple $(\mathcal{K}, \mathcal{P}, \mathcal{C}, \mathcal{E}, \mathcal{D})$, where

- \mathcal{K} is a finite set of possible *keys*, which must be shared with legitimate users.
- \mathcal{P} is a finite set of possible *plaintexts*.
- \mathcal{C} is a finite set of possible *ciphertexts*.
- For each k , there is an *encryption rule* $e_k \in \mathcal{E}$ and the corresponding *decryption rule* $d_k \in \mathcal{D}$, where $e_k: \mathcal{P} \rightarrow \mathcal{C}$ and $d_k: \mathcal{C} \rightarrow \mathcal{P}$ must satisfy $d_k(e_k(m)) = m$ for each $m \in \mathcal{P}$. Typically, \mathcal{E} and \mathcal{D} can be considered as the *encryption algorithm* and *decryption algorithm*, respectively. Namely, \mathcal{E} takes a key k and a plaintext m as input and computes $e_k(m)$. Similarly, \mathcal{D} takes a key k and a ciphertext c as input and computes $d_k(c)$.

In the basic scenario, there are two parties who wish to communicate with each other over an insecure channel. These parties are usually referred to as Alice and Bob. The insecurity of the channel means that an eavesdropper, called Eve, may wiretap the messages that are sent over this channel. Alice and Bob can make use of a cryptosystem to keep their information secret in spite of the existence of the eavesdropper, by agreeing with each other on a secret key k via another secure communication method.

In order for Alice to send her secret message m , she uses the key k to compute the ciphertext $c = e_k(m)$, and she sends Bob the ciphertext c over the insecure channel. Since Bob also knows the key k , he can decrypt the ciphertext by using the decryption algorithm and

obtain $d_k(c) = d_k(e_k(m)) = m$. On the other hand, since Eve wiretaps the ciphertext c but she does not know the key k , there is no easy way to obtain the original message m .

The *one-time pad* is the symmetric-key cryptosystem that achieves perfect secrecy. Suppose that $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1\}^n$, $e_k(m) = m \oplus k$ and $d_k(c) = c \oplus k$, where \oplus is a bitwise XOR operation.

Let \mathbf{P} be a random variable over \mathcal{P} and \mathbf{K} a random variable over \mathcal{K} . A cryptosystem is said to have *perfect secrecy* if and only if for every $m \in \mathcal{P}$ and for every $c \in \mathcal{C}$, $\Pr[\mathbf{P} = m | \mathbf{C} = c] = \Pr[\mathbf{P} = m]$. That is, knowing the ciphertext c is not advantageous to the prediction of the original plaintext m . Shannon showed that for every perfectly secure cryptosystem the entropy of the key is at least as large as the entropy of the plaintext. This says that Alice cannot securely send her secret message of n -bits by using a key of length less than n bits.

Moreover, we have a characterization of perfect secrecy that is referred to as Shannon's Theorem. It states that the cryptosystem with $\mathcal{P} = \mathcal{C} = \mathcal{K}$ and $\text{supp}(\mathbf{P}) = \mathcal{P}$ has perfect secrecy if and only if \mathbf{K} is the uniform distribution over \mathcal{K} and there exists a unique key k such that $e_k(m) = c$ for every $m \in \mathcal{P}$ and for every $c \in \mathcal{C}$. By using Shannon's Theorem, it is not hard to see that the one-time pad has perfect secrecy.

2.2 Message Authentication

Confidentiality may be the prime function of cryptographic systems. Besides confidentiality, *authenticity* is another function of cryptographic systems. As a tool to provide message authentication, the universal hash function family (due to Carter and Wegman (1979) and Wegman and Carter (1981)) is well known. \mathcal{H} is a *universal hash function family* if the following holds.

- Each $h \in \mathcal{H}$ is a function from X to Y . Namely, the domain X and the range Y depend only on \mathcal{H} .
- For distinct $x, x' \in X$,

$$\Pr[h(x) = h(x')] = \frac{1}{|Y|}$$

where the probability is taken over the uniform distribution on \mathcal{H} .

If for distinct $x_1, x_2 \in X$ and for $y_1, y_2 \in Y$

$$\Pr[h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{|Y|^2}$$

then \mathcal{H} is a pairwise independent (aka strongly universal) hash function family. It is easy to see that pairwise independent hash functions imply universal hash functions. For some domain X and range Y , there exists an efficient implementation of \mathcal{H} . Namely, there is an efficient algorithm that computes $h(x)$ on input x and the description of h . For example, we have an efficient implementation of the pairwise independent hash function family for $X = \{0, 1\}^n$ and $Y = \{0, 1\}^n$. Let us consider the following:

$$\mathcal{H} = \{h_{a,b}(x) = ax + b \mid a, b \in \mathbb{F}_{2^n}\}$$

where we identify the finite field \mathbb{F}_{2^n} (of order 2^n) with $\{0, 1\}^n$ by the standard encoding and the arithmetic computation is carried on \mathbb{F}_{2^n} . Since the computation over \mathbb{F}_{2^n} can be done

efficiently, \mathcal{H} has an efficient implementation. Also note that there exist efficient implementations for the pairwise independent hash function families whose domain and range are natural.

A typical application of the pairwise independent hash function is message authentication. Suppose that Alice and Bob share the hash key k and agree with the corresponding hash function h_k . The case where Alice sends a message m with $h_k(m)$ can be considered. If Bob receives (m, z) and checks if $z = h_k(m)$ and whenever the check fails, he can judge the message to be forged. On the other hand, since Eve does not know the hash key, the authentication part (i.e., z) seems to be perfectly random.

2.3 Block Ciphers and Cryptographic Hash Functions

Though the one-time pad has perfect secrecy, it is not practical since the key length must be at least as large as the message length. As a practical symmetric cryptosystem, blockwise encryption (called the *block cipher*) has been considered. Some block ciphers (e.g., data encryption standard (DES) and advanced encryption standard (AES)) can withstand some specific attacks. While several design paradigms have been developed so as to withstand known attacks, block ciphers cannot have perfect secrecy.

Universal hashing also has a desirable property. Because of its nature, the number of uses per key is limited. Instead, heuristic cryptographic hash functions such as SHA-1 are practically used.

2.4 Cryptographic Protocols

As mentioned, the era of modern cryptography started in the 1970s. The invention of the notion of one-way functions introduced a computational complexity viewpoint into cryptology. One-way functions are functions $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that, for each $x \in \{0, 1\}^*$, $f(x)$ is efficiently computable but $f^{-1}(y)$ is computationally tractable only for a negligible fraction of all y 's. The notion of one-way functions is one of the most fundamental in cryptology. Though the existence of one-way functions (which is related to the **P** versus **NP** problem) is unproven, modern cryptography has been established based on it. Actually, computational assumptions such as the existence of one-way functions have brought about many cryptographic protocols, such as public-key encryption schemes, digital signature schemes, commitment schemes, oblivious transfer schemes, zero-knowledge proof/argument systems, and so on.

2.4.1 Public-Key Encryption Scheme

A *public-key encryption scheme* is different to a symmetric-key encryption scheme. While the encryption key and the decryption key are the same in the case of symmetric-key encryption, the encryption key is different from the decryption key in the case of public-key encryption. This means that in cases of public-key encryption the encryption key can be publicly issued. The decryption key must be still secret.

The public-key encryption scheme is usually defined by a triple $(\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- \mathcal{G} is a key generation algorithm that generates a pair of the secret key k_s and the public key k_p on input 1^n (the unary representation of the security parameter n). Here, the plaintext space \mathcal{P} and the ciphertext space \mathcal{C} are implicitly defined. \mathcal{P} and \mathcal{C} basically depend on the security parameter n . (They sometimes depend on the key pair (k_s, k_p) .)
- \mathcal{E} is an encryption algorithm that, given the public key k_p and the message m , computes the ciphertext $c = \mathcal{E}(k_p, m)$.
- \mathcal{D} is a decryption algorithm that, given the secret key k_s and the ciphertext c , computes the plaintext $m = \mathcal{D}(k_s, c)$.
- For every possible key pair (k_s, k_s) and for every message $m \in \mathcal{P}$, $\mathcal{D}(k_s, \mathcal{E}(k_p, m)) = m$.

As in the case of the one-time pad, there are two parties, Alice and Bob. In the case of public-key encryption, the roles of Alice and Bob are asymmetric and all parties including Eve are modeled by probabilistic polynomial-time algorithms. First, Bob or the trusted third party runs the key generation algorithm to obtain the key pair (k_s, k_p) . The public key k_p is published. If the trusted third party runs the algorithm, the secret key k_s must be transferred to Bob via a secure communication method. In order for Alice to send her secret message m to Bob, she obtains Bob's public key k_p , computes the ciphertext $c = \mathcal{E}(k_p, m)$ and sends Bob the ciphertext c . After Bob obtains the ciphertext c , he can compute $m = \mathcal{D}(k_s, c)$. Eve wiretaps the ciphertext c and she can also obtain the public key k_p . This means that she can choose an arbitrary message m' on her own and obtain the corresponding ciphertext $c' = \mathcal{E}(k_p, m')$. This ability is called the *chosen plaintext attack*. Thus, the security of public-key encryption must be guaranteed against the chosen plaintext attack at least. In recent theory, a more powerful attack model called the *chosen ciphertext attack* has been considered. Here, the details are not mentioned.

Besides the attack model, it is important to define the security goal. *One-wayness, semantic security* (Goldwasser and Micali 1984) and *non-malleability* (Dolev et al. 2000) are typical goals. One-wayness means that it is infeasible for Eve to recover the whole message from the target ciphertext. Semantic security means that it is infeasible for Eve to extract any partial information of the plaintext from the ciphertext. Non-malleability means that it is infeasible for Eve to obtain another ciphertext whose plaintext is related to the original plaintext of the target ciphertext. The bare RSA public-key encryption scheme (Rivest et al. 1978) cannot have semantic security. Thus, the RSA scheme should be used in a more sophisticated way, such as optimal asymmetric encryption padding (OAEP) (Bellare and Rogaway 1995). Furthermore, hybrid encryptions (e.g., Cramer and Shoup (2003)), among them public-key encryption, symmetric encryption and message authentication techniques, they have been extensively studied, because they easily provide a more practical and more secure public-key encryption scheme.

2.4.2 Digital Signature Scheme

A *digital signature scheme* is one of the most important protocols in cryptology. It is usually defined by a triple $(\mathcal{G}, \mathcal{S}, \mathcal{V})$:

- \mathcal{G} is a key generation algorithm that generates a pair of the secret key k_s and the public key k_p on input 1^n .
- \mathcal{S} is a signature algorithm that, given the secret key k_s and the message m , computes a signature $\sigma = \mathcal{S}(k_s, m)$.

- \mathcal{V} is a verification algorithm that, given the public key k_p and the signed message (m, σ) , checks the validity of σ for the message m .

In the basic scenario, there is a signer Alice who wishes to prove the authenticity of her message and a verifier Bob who verifies the validity of the signature. First, Alice or the trusted third party runs the key generation algorithm to obtain the key pair (k_s, k_p) . The public key k_p is published. If the trusted third party runs the algorithm, the secret key k_s must be transferred to Alice via a secure communication method. In order for Alice to sign her message m , she obtains her secret key k_s , computes the signature $\sigma = \mathcal{S}(k_s, m)$, and issues the signed message (m, σ) . If Bob wishes to verify the validity of the signature σ for the message m , he runs the verification algorithm \mathcal{V} with the public key k_p and checks the validity. Bob rejects the signed message if the check fails and accepts it otherwise.

The most secure notion of digital signature schemes is *existential unforgeability* against the *adaptive chosen message attack*. It means that for any message m even if Eve can adaptively obtain signed messages (m', σ') for her own choice $m' \neq m$ then she cannot forge the signature σ valid for the message m . It is known that digital signature schemes satisfying existential unforgeability against the adaptive chosen message attack are constructible from any one-way function (Naor and Yung 1989).

2.4.3 Bit Commitment Scheme

A *bit commitment* is a fundamental cryptographic protocol between two parties. The protocol consists of two phases: commit phase and reveal phase. In the commit phase, the sender Alice has a bit b in her private space and she wants to commit b to the receiver Bob. They exchange messages and at the end of the commit phase Bob gets some information that represents b . In the reveal phase, Alice confides b to Bob by exchanging messages. At the end of the reveal phase, Bob judges whether the information from the reveal phase really represents b or not. Basically, there are three requirements for secure bit commitment: correctness, the hiding property, and the binding property. Correctness guarantees that if both parties are honest then for any bit $b \in \{0, 1\}$ Alice has, Bob accepts it with certainty. The hiding property guarantees that (cheating) Bob cannot reveal the committed bit during the commit phase. The binding property guarantees that (cheating) Alice cannot commit her bit b such that Alice maliciously reveals $b \oplus 1$ as her committed bit but Bob accepts it.

In the classical case, a simple argument shows the impossibility of bit commitment with the hiding and the binding properties both statistical. Thus, either hiding or binding must be computational. A construction of the statistically binding scheme (of the round complexity $O(1)$) from any pseudorandom generator was given by Naor (1991). Since the existence of one-way functions is equivalent to that of pseudorandom generators (Håstad 1999), a statistically binding scheme can be based on any one-way function. A construction of a statistically hiding scheme from one-way permutation was given by Naor, Ostrovsky, Venkatesan and Yung (Naor et al. 1998). After that, the assumption of the existence of one-way permutation was relaxed to that of a approximable-preimage-size one-way function (Haitner et al. 2005). Finally, Haitner and Reingold (2007) showed that a statistically hiding scheme can be based on any one-way function by using the excellent techniques in Nguyen et al. (2006).

Since a statistically binding (resp., statistically hiding) bit commitment scheme is a building block for zero-knowledge proof (resp., zero-knowledge argument) systems (Goldreich et al. 1991;

Brassard et al. 1988), it is desirable to be efficient from several viewpoints (e.g., the total size of messages exchanged during the protocol, or the round number of communications in the protocol).

2.4.4 Oblivious Transfer

Oblivious transfer (OT) is also an important two-party cryptographic protocol. The first known OT system was introduced by Rabin (1981) where a message is received with probability $1/2$ and the sender cannot know whether his message reaches the receiver. Prior to this, Wiesner (1983) introduced a primitive called multiplexing, which is equivalent to the 1-out-of-2 OT (Even et al. 1985) known today, but it was then not seen as a tool in cryptography. Even et al. defined the 1-out-of-2 OT (Even et al. 1985), where the sender Alice has two secrets σ_0 and σ_1 and the receiver Bob can choose one of them in an oblivious manner. That is, Alice cannot know Bob's choice $i \in \{0, 1\}$ and Bob cannot know any information on σ_{1-i} . The former property is called *receiver's privacy* and the latter *sender's privacy*. Later, Crépeau (1988) showed that Rabin's OT and the 1-out-of-2 OT are equivalent. Furthermore, the more general 1-out-of- N OT (where the sender has N secrets) and the more specific 1-out-of-2 *bit* OT (where the secrets are one bit long) are similarly defined and the reductions among the variants of OT have been discussed in the literature, for example Brassard et al. (1996, 2003) and Crépeau and Savvides (2006).

OT protocols are fundamental building blocks of modern cryptography. Most notably, it is known that any secure function evaluation can be based on OT (Kilian 1988; Goldreich et al. 1987).

2.4.5 Zero-Knowledge Proof/Argument Systems

The notion of *zero-knowledge* was introduced by Goldwasser, Micali and Rackoff (Goldwasser et al. 1989). Roughly speaking, an interactive proof system has the zero-knowledge property if any (possibly cheating) verifier that communicates with the honest prover learns nothing through the interaction except the validity of the claimed statement. In the classical case, there are several variants of zero-knowledge corresponding to how to formally define the notion that the verifier "learns nothing." Anyway, the verifier "learns nothing" if there exists a polynomial-time simulator whose output is indistinguishable in some sense from the information the verifier would have after the interaction with the honest prover.

Goldreich, Micali and Wigderson (Goldreich et al. 1991) showed that any NP language has zero-knowledge proof by using a (nonuniform) computationally concealing statistically binding bit commitment scheme, that is, any (nonuniform) one-way function. This result can be extended for PSPACE languages. Moreover, a statistical zero-knowledge argument for any NP language is constructible from any one-way function (Nguyen et al. 2006).

3 Quantum Computational Cryptographic Primitive

Before showing several quantum cryptographic protocols, some computational primitive for quantum cryptography will be mentioned.

3.1 Quantum One-Way Permutations

In classical cryptography, some cryptographic protocols assume the existence of one-way permutation. Thus, some quantum cryptographic primitive might assume the existence of quantum one-way permutations. As will be seen, Dumais, Mayers and Salvail (Dumais et al. 2000) proposed a quantum bit commitment scheme based on the existence of quantum one-way permutations.

In the classical case, the RSA function and the discrete logarithm-based function could be candidates for one-way permutations. Unfortunately, such candidates are no longer one-way in the quantum setting because Shor's quantum algorithm (Shor 1997) solves the factorization problem and the discrete logarithm problem efficiently. Thus, it is important to consider the existence of quantum one-way permutations.

Kashefi, Nishimura and Vedral (Kashefi et al. 2002) gave a necessary and sufficient condition for the existence of *worst-case* quantum one-way permutations as follows.

Let $f: \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a permutation. Then f is worst-case quantum one-way if and only if there exists a unitary operator in $\mathcal{Q} = \{Q_j(f) \mid j = 0, 1, \dots, n/2 - 1\}$ that is not efficiently computable. The reflection operators $Q_j(f)$ are defined as

$$Q_j(f) = \sum_{x \in \{0,1\}^n} |x\rangle\langle x| \otimes (2|\psi_{j,x}\rangle\langle\psi_{j,x}| - I)$$

where

$$|\psi_{j,x}\rangle = \frac{1}{\sqrt{2^{n-2j}}} \sum_{y: \text{pref}(f(y), 2j) = \text{pref}(x, 2j)} |y\rangle$$

and $\text{pref}(s, i)$ denotes the i bits-long prefix of a string s .

They also considered quantum weakly one-way permutations (i.e., in the cryptographic sense) and gave a sufficient condition on the existence. After that, Kawachi, Kobayashi, Koshiya and Putra (Kawachi et al. 2005b) gave a necessary condition and completed an algorithmic characterization of quantum weakly one-way permutations. The characterization is almost similar to the case of worst-case quantum one-way permutation except that the unitary operators in \mathcal{Q} permit exponentially small errors. While Kawachi et al. (2005b) mentioned a characterization of quantum weakly one-way permutations only, a similar characterization holds for quantum *strongly* one-way permutations. These characterizations might be helpful either to search for candidates of quantum one-way permutations or to disprove their existence.

3.2 Quantum Hard-Core Predicates

Hard-core predicates for one-way functions are also important in computational cryptography. Goldreich and Levin (1989) showed that a hard-core predicate is constructible from any one-way function f . Let $f'(x, r) = (f(x), r)$ be a function where $x, r \in \{0, 1\}^n$. Here, it is easy to see that f' is also one-way. The predicate

$$GL(x, r) = \langle x, r \rangle = \sum_{i=1}^n x_i r_i \bmod 2$$

is a hard-core predicate for the one-way function f' . Moreover, they gave a way to construct a hard-core function of output length $O(\log n)$ for any one-way function. Adcock and

Cleve (2002) considered quantum hard-core predicates (i.e., hard-core predicates against quantum adversaries) for quantum one-way functions. They showed that for any quantum one-way function f , $GL(x, r)$ is also a quantum hard-core predicate for the quantum one-way function $f'(x, r) = (f(x), r)$. Furthermore, they proved that the reduction between quantum hard-core predicates and quantum one-way functions is simpler and tighter than the classical reduction. Actually, they showed that a lower bound on the number of oracle calls in the classical reduction is properly larger than an upper bound in the quantum reduction.

It is widely known that list decodable codes have many computational complexity theoretical applications including hard-core functions (Sudan 2000). Intuitively speaking, if we are given a corrupted codeword of a list decodable code, we may output a short list containing messages close to the correct one rather than uniquely recover it. For example, the Goldreich–Levin Theorem (Goldreich and Levin 1989) mentioned above can be regarded as an efficient list-decoding algorithm for the binary Hadamard code. Actually, for a message x , $GL(x, 0), \dots, GL(x, 2^n - 1)$ corresponds to a codeword of the binary Hadamard code. The prediction algorithm for the hard-core predicate GL also corresponds to an access to a corrupted codeword. Suppose that we could obtain a polynomially long list that contains x with high probability by accessing the corrupted codeword. It implies that inverting the one-way function f is efficiently computable. Thus, we can say that there is no efficient algorithm to predict the hard-core value of GL .

From the quantum computational point of view, a general construction of quantum hard-core functions from quantum one-way functions was obtained due by Kawachi and Yamakami (2006), utilizing a classical code that has a quantum list-decoding algorithm. Roughly speaking, they showed that any code that has an almost orthogonal structure in a sense has a quantum list-decoding algorithm. Consequently, it enables one to discover new (quantum) hard-core functions. Especially, the quantum list-decoding technique affirmatively but quantumly settled an open problem whether Damgård's generator (Damgård 1988) is cryptographically secure or not.

4 Quantum Key Distribution

Quantum key distribution (QKD) is a quantum technique to share a random string between two parties Alice and Bob. It is the most successful protocol in quantum cryptography, which strengthens symmetric-key techniques. Usually, QKD protocols assume the authentication channel. It means that Alice and Bob beforehand have to agree on the hash key for universal hash functions. Thus, QKD can be regarded as a technique for stretching shared randomness between Alice and Bob.

Known QKD protocols can be partitioned into two types. One is based on Heisenberg's uncertainty principle and the other is based on Bell's inequality.

4.1 Based on Heisenberg's Uncertainty Principle

Quantum key distribution in the early stages, such as the BB84 protocol (Bennett and Brassard 1984) and the B92 protocol (Bennett 1992), were based on Heisenberg's uncertainty principle. The principle says that two complementary quantum states cannot be simultaneously measured. If the eavesdropper Eve wiretaps the quantum channel to obtain the secret

information, then she has to measure the quantum states. However, Heisenberg's uncertainty principle says that the measurement of quantum states affects the system. Thus, we can check the presence of an eavesdropper by observing the system change.

4.1.1 The BB84 Protocol

In this protocol, we use two orthogonal bases. One is the computational basis $\{|0\rangle, |1\rangle\}$ and the other is the diagonal basis $\{|0\rangle_{\times}, |1\rangle_{\times}\}$, where

$$\begin{aligned}|0\rangle_{\times} &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\|1\rangle_{\times} &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\end{aligned}$$

We also use $\{|0\rangle_+, |1\rangle_+\}$ to denote the computational basis $\{|0\rangle, |1\rangle\}$. The protocol is as follows:

1. Alice randomly chooses $4n$ qubits, each in one of the four states $|0\rangle_+, |0\rangle_{\times}, |1\rangle_+$ or $|1\rangle_{\times}$, and sends them to Bob.
2. For each qubit that Bob receives, he chooses either the computational basis (+) or the diagonal basis (\times) and measures the qubit with respect to the basis. If the channel is not wiretapped and Alice's basis coincides with Bob's, then his is the same as the bit of Alice. If the bases differ, his measurement result is randomly distributed.
3. Alice tells Bob the basis she used for each qubit via the classical authentication channel. Bob keeps the bits where Bob's basis coincides with Alice's. This happens in about half the cases, so Bob will have about $2n$ bits left.
4. Bob randomly partitions about $2n$ bits into two groups of about n bits. They use one group to estimate the error rate. Bob tells Alice all the bits in one group. If Alice's bits are too different from Bob's bits, they abort the protocol.
5. Alice and Bob obtain a common secret key from the remaining about n bits by performing error correction and privacy amplification.

Firstly, Mayers (1996) gave a (somewhat complex) proof of the unconditional security of the BB84 protocol. After that Shor and Preskill (2000) gave a simple proof by showing that the entanglement-based version of the BB84 protocol is equivalent for Eve to the original BB84 protocol. Note that the security proof in Mayers' fashion has been simplified (see, e.g., Koashi and Preskill (2003)).

4.1.2 The B92 Protocol

This protocol is similar to the BB84 protocol. While the BB84 protocol uses four types of quantum states, the B92 protocol instead uses two non-orthogonal quantum states.

Let $|\psi_0\rangle$ and $|\psi_1\rangle$ be two non-orthogonal quantum states. Then, two projective measurements P_0 and P_1 can be defined as follows:

$$\begin{aligned}P_0 &= I - |\psi_1\rangle\langle\psi_1| \\P_1 &= I - |\psi_0\rangle\langle\psi_0|\end{aligned}$$

The point is that $P_0|\psi_1\rangle = 0$ and $P_1|\psi_0\rangle = 0$.

The sketch of the B92 protocol is as follows: (1) Alice chooses qubits, each in one of the two quantum states $|\psi_0\rangle$ and $|\psi_1\rangle$ and sends them to Bob. (2) For each qubit that Bob receives, he chooses at random one of the two measurements P_0 and P_1 and measures the qubit. He keeps it secret whether he measures something or not. (3) Bob tells Alice the measurement he uses. (4) Alice can extract bits where $|\psi_i\rangle$ matches to P_i .

4.2 Based on Bell's Inequality

Experiments in the 1970s indicated that the correlation function of the entangled quantum states cannot be expressed by classical probability and these entangled quantum states are nonlocal. Ekert (1991) exploited these properties in the QKD protocol (E91 protocol). Namely, he used the property that the entangled photons have strong correlation to yield the secret key. On the other hand, he also used the property that the entangled photons do not satisfy Bell's inequality to detect the presence of the eavesdropper. Since the E91 protocol has a bit complex structure that relates to Bell's inequality, we give the description of a similar but simpler protocol. That is the protocol of Bennett, Brassard and Mermin (Bennett et al. 1992) based on the entanglement.

4.2.1 The BBM92 Protocol

The protocol is as follows:

1. A source S emits a pair of qubits such that

$$\begin{aligned} |\psi_{AB}\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_+|0\rangle_+ + |1\rangle_+|1\rangle_+) \\ &= \frac{1}{\sqrt{2}}(|0\rangle_\times|0\rangle_\times + |1\rangle_\times|1\rangle_\times) \end{aligned}$$

The first qubit is sent in opposite directions. The first qubit is received by Alice and the second by Bob.

2. Both Alice and Bob randomly and independently choose either the computational basis (+) or the diagonal basis (\times) and measure the qubit with respect to the basis.
3. Alice and Bob announce their bases via the classical channel and keep the bits where the bases coincide.

The principle is based on the fact that Alice's bits and Bob's are exactly the same only when the pure state $|\psi_{AB}\rangle$ is prepared. If Eve wants to yield the situation where Alice's bits and Bob's are exactly the same, the only thing she can do is to prepare a pure state on the system AB and send it to Alice and Bob. Thus, Eve cannot leave any correlation between her system and the measurement results by Alice and Bob.

5 Quantum Public-Key Encryption

As mentioned, public-key cryptosystems are indispensable, even in the future if quantum computers are physically realized. Unfortunately, almost all practical public-key cryptosystems are vulnerable to Shor's algorithm. On the other hand, there are public-key

cryptosystems that have not been shown to be vulnerable to quantum adversaries. Lattice-based cryptography (Micciancio and Regev 2009) is one of them. In this section, we focus on public-key cryptosystems that use quantum computation.

5.1 The OTU00 Scheme

Okamoto, Tanaka and Uchiyama (Okamoto et al. 2000) first proposed a knapsack-based cryptosystem as a quantum public-key cryptosystem. Since knapsack-based cryptosystems in the early stages used linearity in the public key, attack algorithms were developed. To avoid attack algorithms, Chor and Rivest (1988) introduced nonlinearity into the public-key generation by using an easy discrete logarithm problem and proposed yet another knapsack-based cryptosystem. The OTU00 scheme can be regarded as an extension of the Chor–Rivest cryptosystem. This is because they use general discrete logarithm problems to generate public keys with the help of quantum computation.

The original OTU00 scheme uses the algebraic number field. Here, we give a simpler description of the OTU00 scheme based on the finite field.

[Key Generation Algorithm \mathcal{G}]

1. For the security parameter n , determine the parameters k, ℓ bounded by a polynomial in n .
2. Randomly generate a prime p and a generator g of \mathbb{F}_p^* .
3. Randomly choose k relatively prime integers $p_1, \dots, p_k \in \mathbb{F}_p^*$ such that the product of any ℓ elements out of the k elements is bounded by p .
4. Run Shor's algorithm to compute a_i such that $p_i = g^{a_i} \pmod{p}$ for each i ($1 \leq i \leq k$).
5. Randomly choose $d \in \mathbb{F}_{p-1}$.
6. Compute $b_i = (a_i + d) \pmod{(p-1)}$ for each i .
7. Return the public key (ℓ, b_1, \dots, b_k) and the secret key $(g, d, p, p_1, \dots, p_k)$.

[Encryption Algorithm \mathcal{E}]

1. Encode the message $m \in \{0, 1\}^\nu$ ($\nu = \lfloor \log_2 (\binom{k}{\ell}) \rfloor$) into a k -bit string (s_1, \dots, s_k) of the Hamming weight ℓ as follows:
 - (a) Initialize the variable j to ℓ .
 - (b) Repeat the following from $i = 1$ to $i = k$. For convenience, we let $\begin{pmatrix} a \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ q \end{pmatrix} = 1$ for $a \geq 1$.
 - If $m \geq \binom{k-i}{j}$ then set $s_i \leftarrow 1$ and $m \leftarrow m - \binom{k-i}{j}$ and decrease j by 1.
 - If $m < \binom{k-i}{j}$ then set $s_i = 0$.
2. Compute the ciphertext $c = \sum_{i=1}^k b_i s_i$ and return it.

[Decryption Algorithm \mathcal{D}]

1. Compute $r = c - \ell d \pmod{p-1}$.
2. Compute $u = g^r \pmod{p}$.
3. Compute (s_1, \dots, s_k) as follows:
 - If $p_i | u$ then set $s_i = 1$.
 - If $p_i \nmid u$ then set $s_i = 0$.

4. Recover the plaintext m from (s_1, \dots, s_k) as follows:
 - (a) Initialize m to 0 and the variable j to ℓ .
 - (b) Repeat the following from $i = 1$ to $i = k$.

If $s_i = 1$ then set $m \leftarrow m + \binom{k-i}{j}$ and decrease j by 1.
5. Return m .

The correctness of the decryption comes from the following fact:

$$u \equiv g^r \equiv g^{c-\ell d} \equiv g^{\sum_i s_i b_i - \ell d} \equiv g^{\sum_i s_i a_i} \equiv \prod_i (g^{a_i})^{s_i} \equiv \prod_i p_i^{s_i} \pmod{p}$$

Nguyen and Stern (2005) show that breaking the OTU00 scheme is reducible to the shortest vector problem or the closest vector problem. However, since these problems are NP-hard, the reductions do not provide an effective attack.

5.2 The KKNY05 Scheme

Kawachi, Koshiya, Nishimura and Yamakami (Kawachi et al. 2005a) proposed a semantically secure (in a weak sense) quantum public-key cryptosystem based on the worst-case hardness of the graph automorphism problem. Actually, they introduced a computational problem of distinguishing between two specific quantum states as a new cryptographic problem to design the quantum public-key cryptosystem. The computational indistinguishability between quantum states is a generalization of the classical indistinguishability between two probability distributions, which plays an important role in computational cryptography. Their problem QSCD_{ff} asks whether we can distinguish between two sequences of identical samples of $\rho_{\pi}^+(n)$ and of $\rho_{\pi}^-(n)$ for each fixed hidden permutation π for each length parameter n of a certain form. Let S_n be the *symmetric group* of degree n and let

$$\mathcal{K}_n = \{\pi \in S_n : \pi^2 = id \text{ and } \forall i \in \{1, \dots, n\} [\pi(i) \neq i]\}$$

for $n \in N$, where id stands for the identity permutation and $N = \{2(2n' + 1) : n' \in \mathbb{N}\}$. For each $\pi \in \mathcal{K}_n$, let $\rho_{\pi}^+(n)$ and $\rho_{\pi}^-(n)$ be two quantum states defined by

$$\begin{aligned} \rho_{\pi}^+(n) &= \frac{1}{2n!} \sum_{\sigma \in S_n} (|\sigma\rangle + |\sigma\pi\rangle)(\langle\sigma| + \langle\sigma\pi|) \text{ and} \\ \rho_{\pi}^-(n) &= \frac{1}{2n!} \sum_{\sigma \in S_n} (|\sigma\rangle - |\sigma\pi\rangle)(\langle\sigma| - \langle\sigma\pi|) \end{aligned}$$

The cryptographic properties of QSCD_{ff} follow mainly from the definition of the set \mathcal{K}_n of the hidden permutations. Although the definition seems somewhat artificial, the following properties of \mathcal{K}_n lead to cryptographic and complexity-theoretic properties of QSCD_{ff}:

- (i) $\pi \in \mathcal{K}_n$ is of order 2, which provides the trapdoor property of QSCD_{ff}
- (ii) For any $\pi \in \mathcal{K}_n$, the conjugacy class of π is equal to \mathcal{K}_n , which enables one to prove the equivalence between the worst-case/average-case hardness of QSCD_{ff}
- (iii) The graph automorphism problem is (polynomial-time Turing) equivalent to its subproblem with the promise that a given graph has a unique non-trivial automorphism in \mathcal{K}_n or none at all. This equivalence is exploited to give a complexity-theoretic lower bound of QSCD_{ff} that is, the worst-case hardness of the graph automorphism problem.

Their problem QSCD_{ff} is closely related to a much harder problem: the hidden subgroup problem on the *symmetric groups* (SHSP). Note that no known subexponential-time quantum

algorithm exists for SHSP. Hallgren et al. (2003) introduced a distinction problem between certain two quantum states, similar to QSCD_{ff}, to discuss the computational intractability of SHSP by a “natural” extension of Shor’s algorithm (1997) with the quantum Fourier transformation. An efficient solution to this distinction problem gives an answer to a pending question on a certain special case of SHSP. To solve this distinction problem, as they showed, the so-called *weak Fourier sampling* on a single sample should require an exponential number of samples. This result was improved by Grigni et al. (2004) and Kempe et al. (2006). On the contrary, Hallgren et al. (2006) and Hayashi et al. (2008) proved that no time-unbounded quantum algorithm solves the distinction problem even from $o(n \log n)$ samples. Kawachi et al. (2005a) showed that the above distinction problem is polynomial-time reducible to QSCD_{ff}. This immediately implies that we have no time-unbounded quantum algorithm for QSCD_{ff} from $o(n \log n)$ samples. Even with sufficiently many samples for QSCD_{ff} there is no known subexponential-time quantum algorithm for QSCD_{ff} and thus finding such an algorithm seems a daunting task.

One can move to the description of the quantum public-key cryptosystem in Kawachi et al. (2005a). Usually public-key cryptosystems consist of three algorithms for key generation, encryption, and decryption.

[Public-Key Generation Algorithm]

Input: $\pi \in \mathcal{K}_n$

Procedure:

Step 1. Choose a permutation σ from S_n uniformly at random and store it in the second register. Then, the entire system is in the state $|0\rangle|\sigma\rangle$.

Step 2. Apply the Hadamard transformation to the first register.

Step 3. Apply the Controlled- π to both registers.

Step 4. Apply the Hadamard transformation to the first register again.

Step 5. Measure the first register in the computational basis. If 0 is observed, then the quantum state in the second register is ρ_π^+ . Otherwise, the state of the second register is ρ_π^- . Now, apply the conversion algorithm to ρ_π^- .

The encryption algorithm consists of two parts: one is the conversion algorithm below and the other is so simple that we describe it in the description of the total system.

[Conversion Algorithm]

The following transformation inverts, given ρ_π^+ , its phase according to the sign of the permutation with certainty.

$$|\sigma\rangle + |\sigma\pi\rangle \mapsto (-1)^{\text{sgn}(\sigma)}|\sigma\rangle + (-1)^{\text{sgn}(\sigma\pi)}|\sigma\pi\rangle$$

Since π is odd, the above algorithm converts ρ_π^+ into ρ_π^- .

Finally, we give a description of the decryption algorithm.

[Decryption Algorithm]

Input: unknown state χ which is either ρ_π^+ or ρ_π^- .

Procedure:

Step 1. Prepare two quantum registers: The first register holds a control bit and the second one holds χ . Apply the Hadamard transformation H to the first register. The state of the system now becomes

$$H|0\rangle\langle 0|H \otimes \chi$$

Step 2. Apply the Controlled- π operator C_π to the two registers, where $C_\pi |0\rangle|\sigma\rangle = |0\rangle|\sigma\rangle$ and $C_\pi |1\rangle|\sigma\rangle = |1\rangle|\sigma\pi\rangle$ for any $\sigma \in S_n$.

Step 3. Apply the Hadamard transformation to the first register.

Step 4. Measure the first register in the computational basis. Output the observed result.

The following is the description of the total cryptosystem consisting of two phases: key transmission phase and message transmission phase.

[Key Transmission Phase]

1. Bob chooses a decryption key π uniformly at random from \mathcal{K}_n .
2. Bob generates sufficiently many copies of the encryption key ρ_π^+ by using the public-key generation algorithm.
3. Alice obtains encryption keys from Bob.

[Message Transmission Phase]

1. Alice encrypts 0 or 1 into ρ_π^+ or ρ_π^- , respectively, by using the conversion algorithm, and sends it to Bob.
2. Bob decrypts Alice's message using the decryption algorithm.

6 Quantum Digital Signature

Gottesman and Chuang (2001) proposed a quantum digital signature scheme. In the classical case, the Lamport signature (Lamport 1979) is based on one-way functions. The GC01 scheme is a quantum counterpart of the Lamport signature scheme. In the classical case, the existence of one-way functions is unproven. They allowed one-way functions to have the image of quantum states and showed the existence of (extended) one-way functions in the information-theoretic sense. Note that if the one-way function in the GC01 scheme is replaced with any one-way function in the computational sense then the security (the existential unforgeability against the chosen message attack) of the GC01 scheme holds in the computational sense. The original GC01 scheme has the non-repudiation property based on our choice of parameters. (The non-repudiation property means that once the signer creates a signed message he cannot repudiate the validity of the signed message.)

6.1 The GC01 scheme

For simplicity, we give the description of a version of the GC01 scheme without the non-repudiation property.

[Key Generation Algorithm \mathcal{G}]

1. Let f be an extended quantum one-way function.
2. Randomly choose $x_0^{(i)}$ and $x_i^{(i)}$ from $\{0, 1\}^n$ for $i = 1, \dots, s$ and return them as the secret key.
(How to determine the value of s will be mentioned later.)
3. For $i = 1, \dots, s$, make t copies of $(|\psi_0^{(i)}\rangle, |\psi_1^{(i)}\rangle) = (|f(x_0^{(i)})\rangle, |f(x_i^{(i)})\rangle)$ and distribute them to at most t verifiers.

[Signature Algorithm \mathcal{S}]

For a target bit $b \in \{0, 1\}$ to be signed, produce its signature σ as follows:

$$\sigma = (\sigma_1, \dots, \sigma_s) = (x_b^{(1)}, \dots, x_b^{(s)})$$

[Verification Algorithm \mathcal{V}]

1. Check the validity of (b, σ) by using the public key $(|\psi_0^{(i)}\rangle, |\psi_1^{(i)}\rangle)$.
2. Compute $(f(\sigma_1), \dots, f(\sigma_s))$ from the signature $(\sigma_1, \dots, \sigma_s)$.
3. Check the equivalence between $f(\sigma_i)$ and $\psi_b^{(i)}$ for $i = 1, \dots, s$ by using the swap-test (Buhrman 2001). If all the swap-tests pass, then accept (b, σ) as a valid signed message. Otherwise, reject (b, σ) .

6.2 Swap Test

As seen, the swap-test is used to check the equivalence between $|f(\sigma_i)\rangle$ and $|\psi_b^{(i)}\rangle$ in the verification algorithm \mathcal{V} . In general, the swap-test checks the equivalence between $|\phi\rangle$ and $|\psi\rangle$. Exactly speaking, the swap-test performs the following:

$$(H \otimes I)(c\text{-SWAP})(H \otimes I)|0\rangle|\phi\rangle|\psi\rangle \quad (1)$$

where c-SWAP exchanges the last two registers if the first qubit is 1. Then [Eq. 1](#) is written as

$$\frac{1}{2}|0\rangle(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) + \frac{1}{2}|1\rangle(|\phi\rangle|\psi\rangle - |\psi\rangle|\phi\rangle)$$

If $|\phi\rangle = |\psi\rangle$ then the measurements on the first register always give the value 0. Let $\delta = \langle\phi|\psi\rangle$. If $|\phi\rangle \neq |\psi\rangle$ then the probability that the measurements on the first register give the value 0 is $\frac{1}{2}(1 + \delta^2)$. Since the swap-test is a one-sided error algorithm, the error probability is at most $\frac{1}{2}(1 + \delta^2)$.

Now we assume that $|\langle f(\sigma_i)|\psi_b^{(i)}\rangle| < \delta$ for every $i = 1, \dots, s$. Then, the probability that \mathcal{V} accepts an invalid signed message is the same as the probability of all the swap-tests in the case that $\sigma_i \neq x_b^{(i)}$ for all $i = 1, \dots, s$. Since it is less than $\left(\frac{1+\delta^2}{2}\right)^s$, we have to set s so that $\left(\frac{1+\delta^2}{2}\right)^s$ becomes negligible.

7 Quantum Commitment and Oblivious Transfer

Since the BB84 protocol (Bennett and Brassard 1984) was shown to be secure unconditionally, some other cryptographic protocols such as bit commitment and oblivious transfer schemes were expected to achieve security without any computational assumption.

The security of bit commitment schemes consists of a concealing property and a binding property. A concealing property satisfies even if a cheating receiver cannot predict the committed value before the reveal phase. A binding property satisfies even if a cheating sender

cannot reveal two different values in the reveal phase. In the classical setting, either the concealing or the binding property must be computational. By introducing quantum states into the bit commitment scheme, we expected that unconditionally secure bit commitment schemes would be realized.

First, Crépeau and Kilian (1988) showed a construction from unconditionally secure quantum commitment schemes to 1-out-of 2 quantum oblivious transfer schemes. Its information-theoretic security was proved in Crépeau (1994), Mayers and Salvail (1994), and Yao (1995). (Note that such a reduction in the classical setting is unknown.) Unfortunately, it was later shown that commitment schemes with unconditional concealing and binding properties are impossible even if quantum states are applicable (Lo and Chau 1997; Mayers 1997). Thus, alternative approaches such as quantum string commitment (Kent 2003) or cheat-sensitive quantum bit commitment (Aharonov et al. 2000; Hardy and Kent 2004; Buhrman et al. 2008) have been studied. Since those approaches do not reach a satisfactory level, yet another approach will be considered. If either the concealing or the binding condition of the bit commitment scheme is allowed to be computational, interesting commitment schemes utilizing quantum information are still possible. Dumais, Mayers and Salvail (Dumais et al. 2000) proposed a perfectly concealing and computationally binding noninteractive quantum bit commitment scheme under the assumption of the existence of quantum one-way permutations. Koshiba and Odaira (2009) generalized the Dumais–Mayers–Salvail (DMS) scheme to obtain a statistically concealing and computationally binding noninteractive quantum bit commitment scheme under the assumption of the existence of a quantum one-way function of the special form. The use of quantum information enables us to transform from a computationally concealing and computationally binding bit commitment scheme to a statistically concealing and computationally binding (interactive) scheme (Crépeau et al. 2001). Since the former is constructible from any quantum one-way function, so is the latter.

Here, let us see some issues in defining quantum bit commitment. Though we can define the concealing condition of quantum bit commitments as in the classical case, some care must be taken to define the binding condition in the quantum case. In the classical case, the binding property means that the probability that the bit value committed in the commit phase can be revealed as both 0 and 1 is negligibly small. However, this definition is too strong in the quantum case. Suppose that a sender in some quantum bit commitment scheme could generate the following state:

$$\frac{1}{\sqrt{2}}|0\rangle|\phi_0\rangle + \frac{1}{\sqrt{2}}|1\rangle|\phi_1\rangle$$

where $|\phi_0\rangle$ (resp., $|\phi_1\rangle$) is a quantum state to be sent when 0 (resp., 1) is honestly committed in the commit phase. Then the sender Alice sends the quantum state only in the second register to the receiver Bob and keeps the quantum state in the first register at her side. Alice can change the committed value with probability 1/2 by measuring the quantum state left at her side just before the reveal phase. Since the above situation is essentially inevitable, the straightforward extension of the binding condition in the classical case is not satisfied in the quantum case. Thus, we have to weaken the definition of the binding condition in the quantum case. Actually, Dumais, Mayers and Salvail (Dumais et al. 2000) considered that the binding condition is satisfied if $p_0 + p_1 - 1$ is negligibly small, where p_0 (resp., p_1) is the probability that the committed value is revealed as 0 (resp., 1).

The following is the description of the quantum bit commitment scheme in Dumais et al. (2000). Let $f: \{0, 1\}^n \rightarrow \{0, 1\}^n$ be any quantum one-way permutation.

[Commit Phase]

1. Alice decides $w \in \{0, 1\}$ as her committing bit. She also chooses $x \in \{0, 1\}^n$ uniformly at random, and then computes $f(x)$.
2. Alice sends $|f(x)\rangle$ to Bob if $w = 0$ and $H^{\otimes n}|f(x)\rangle$ otherwise.

[Reveal Phase]

1. Alice announces x and w to Bob.
2. If $w = 1$, Bob applies $H^{\otimes n}$ to the state sent from Alice. Otherwise, he does nothing to the state.
3. Bob measures the resulting state in the computational basis. Let y be the outcome. He accepts if and only if $y = f(x)$.

Finally in this section, we mention a conversion from quantum bit commitment scheme with computational assumption to quantum oblivious transfer with computational assumption. In Crépeau and Kilian's conversion from quantum bit commitment to 1-out-of 2 quantum oblivious transfer, the commitment scheme must be statistically secure. Thus, since the security proof (Yao 1995) is information-theoretic, his proof is not applicable to the computational security. Crépeau et al. have shown a way to convert a statistically concealing and computationally binding string commitment scheme of some non-standard type to 1-out-of-2 quantum oblivious transfer (Crépeau et al. 2004).

8 Quantum Zero-Knowledge Proof

At any round of interaction to simulate, a simulator typically generates a pair of a question from the verifier and a response from the honest prover. If this produces a pair that is inconsistent in itself (or with the other parts of the transcript of the interaction simulated so far), the simulator “rewinds” the process to simulate this round again. We now mention why the “rewinding” technique is not generally applicable to quantum verifiers. First, quantum information cannot be copied and the fact is also known as the no-cloning theorem. Second, measurements are generally irreversible processes. This difficulty was explicitly pointed out by van de Graaf (1997).

Since then, the study of quantum zero-knowledge has developed from the computational point of view. For example, Watrous (2002) defined the quantum counterpart of honest verifier statistical zero-knowledge and studied its properties. We denote by **HVQSZK** the class of languages that have honest verifier quantum statistical zero-knowledge proof systems. In Watrous (2002), the following statements are shown:

1. There exists a complete promise problem, which is a natural generalization of a complete promise problem in statistical zero-knowledge (**SZK**) due to Sahai and Vadhan (2003).
2. **HVQSZK** is contained by **PSPACE**.
3. **HVQSZK** is closed under the complement.
4. Any **HVQSZK** protocol can be parallelized to a one-round **HVQSZK** protocol.

Kobayashi (2003) defined noninteractive quantum perfect zero-knowledge (**NIQPZK**) and noninteractive quantum statistical zero-knowledge (**NIQSZK**) and studied their properties. Specifically:

1. If the prover and the verifier do not beforehand have any shared randomness and any shared entanglement, languages that have noninteractive quantum zero-knowledge proof systems are in **BQP**.
2. Assuming that the verifier and the prover share polynomially many Einstein–Podolsky–Rosen (EPR) pairs a priori, **NIQPZK** with the one-sided error has a natural complete promise problem, which is a generalization of a complete promise problem in noninteractive statistical zero-knowledge (**NISZK**) due to Goldreich, Sahai and Vadhan (Goldreich et al. 1999).
3. The graph non-automorphism (GNA) problem has a **NIQPZK** proof system with prior EPR pairs. (Since it is unknown whether GNA is in **BQP** or not, **NIQPZK** with prior EPR pairs includes a nontrivial language.)

In order to circumvent problematic issues caused by the rewinding technique, Damgård, Fehr and Salvail (Damgård et al. 2004) showed a way to construct a computational QZK proof and perfect QZK argument against quantum adversaries in the *common reference string* model, wherein it is assumed that an honest third party samples a string from some specified distribution and provides both the prover and verifier with this string at the start of the interaction. They have also given a construction of unconditionally concealing and computationally binding string commitment protocols against quantum adversaries.

Watrous (2006) resolved in many cases the problematic issues caused by the rewinding technique in quantum zero-knowledge by introducing a success probability amplifying technique, which is successfully utilized to amplify the success probability in the Quantum Merlin–Arthur protocol without increasing witness sizes (Marriott and Watrous 2004). Specifically he showed:

1. The graph isomorphism problem is in perfect zero-knowledge (**PZK**) even against the adversarial verifier who uses quantum computation to cheat.
2. If quantum one-way permutations exist, every problem in **NP** has ZK proof systems even against the adversarial verifier who uses quantum computation to cheat.
3. **HVQSZK=QSZK**, where **QSZK** denotes the class of problems having quantum statistical zero-knowledge proof systems.

Together with his proof construction, this implies that all the properties proved for **HVQSZK** in Watrous (2002) are inherited to **QSZK** (except for those related to the round complexity).

Kobayashi (2008) proves a number of general properties on quantum zero-knowledge proofs, not restricted to the statistical zero-knowledge case. Specifically, for quantum computational zero-knowledge proofs, letting **QZK** and **HVQZK** denote the classes of problems having quantum computational zero-knowledge proofs and honest-verifier quantum computational zero-knowledge proof systems, respectively, he showed:

1. **HVQZK = QZK**.
2. Any problem in **QZK** has a public-coin quantum computational zero-knowledge proof system.
3. Any problem in **QZK** has a quantum computational zero-knowledge proof systems of perfect completeness.

References

- Adcock M, Cleve R (2002) A quantum Goldreich-Levin theorem with cryptographic applications. In: Proceedings of the 19th annual symposium on theoretical aspects of computer science, Antibes-Juan les Pins, France, March 2002. Lecture notes in computer science, vol 2285. Springer, Berlin, pp 323–334
- Aharonov D, Ta-Shma A, Vazirani UV, Yao AC-C (2000) Quantum bit escrow. In: Proceedings of the 32nd ACM symposium on theory of computing, ACM, Portland, OR, May 2000, pp 705–714
- Bellare M, Rogaway P (1995) Optimal asymmetric encryption. In: EUROCRYPT '94: Advances in cryptology, Perugia, Italy, May 1994. Lecture notes in computer science, vol 950. Springer, Berlin, pp 92–111
- Bennett CH (1992) Quantum cryptography using any two nonorthogonal states. *Phys Rev Lett* 68:3121–3124
- Bennett CH, Brassard G (1984) Quantum cryptography: public key distribution and coin tossing. In: Proceeding of the IEEE international conference on computers, systems, and signal processing, Bangalore, India, December 1984. IEEE, New York, pp 175–179
- Bennett CH, Brassard G, Mermin ND (1992) Quantum cryptography without Bell's theorem. *Phys Rev Lett* 68:557–559
- Brassard G, Chaum D, Crépeau C (1988) Minimum disclosure proofs of knowledge. *J Comput Syst Sci* 37(2):156–189
- Brassard G, Crépeau C, Santha M (1996) Oblivious transfers and intersecting codes. *IEEE Trans Info Theory* 42(6):1769–1780
- Brassard G, Crépeau C, Wolf S (2003) Oblivious transfers and privacy amplification. *J Cryptol* 16(4):219–237
- Buhrman H, Cleve R, Watrous J, de Wolf R (2001) Quantum fingerprinting. *Phys Rev Lett* 87:167902
- Buhrman H, Christandl M, Hayden P, Lo H-K, Wehner S (2008) Possibility, impossibility and cheat-sensitivity of quantum bit string commitment. *Phys Rev A* 78(32):022316
- Carter JL, Wegman MN (1979) Universal classes of hash functions. *J Comput Syst Sci* 18(2):143–154
- Chor B, Rivest RL (1988) A knapsack-type public key cryptosystems based on arithmetic in finite fields. *IEEE Trans Info Theory* 34:901–909
- Cramer R, Shoup V (2003) Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J Comput* 33(1):167–226
- Crépeau C (1988) Equivalence between two flavours of oblivious transfer. In: CRYPTO'87: Advances in cryptology, University of California, Santa Barbara, CA, August 1987. Lecture notes in computer science, vol 293. Springer, New York, pp 350–354
- Crépeau C (1994) Quantum oblivious transfer. *J Mod Opt* 41(12):2445–2454
- Crépeau C, Kilian J (1988) Achieving oblivious transfer using weakened security assumptions. In: Proceedings of the 29th annual IEEE symposium on foundations of computer science, IEEE, White Plains, NY, October 1988, pp 42–52
- Crépeau C, Savvides G (2006) Optimal reductions between oblivious transfers using interactive hashing. In: EUROCRYPT 2006: Advances in cryptology, St. Petersburg, Russia, May–June 2006. Lecture notes in computer science vol 4004. Springer, Heidelberg, pp 201–221
- Crépeau C, Legare F, Salvail L (2001) How to convert the flavor of a quantum bit commitment. In: EUROCRYPT 2001: Advances in cryptology, Innsbruck, Austria, May 2001. Lecture notes in computer science vol 2045. Springer, Berlin, pp 60–77
- Crépeau C, Dumais P, Mayers D, Salvail L (2004) Computational collapse of quantum state with application to oblivious transfer. In: Proceedings of the 1st theory of cryptography conference, Cambridge, MA, February 2004. Lecture notes in computer science, vol 2951. Springer, Berlin, pp 374–393
- Damgård I (1988) On the randomness of Legendre and Jacobi sequences. In: CRYPTO'88: Advances in cryptology, Santa Barbara, CA, August 1988. Lecture notes in computer science vol 403. Springer, Berlin, pp 163–172
- Damgård I, Fehr S, Salvail L (2004) Zero-knowledge proofs and string commitments notwithstanding quantum attacks. In: CRYPTO 2004: Advances in cryptology, Santa Barbara, CA, August 2004. Lecture notes in computer science, vol 3152. Springer, Berlin, pp 254–272
- Diffie W, Hellman ME (1976) New directions in cryptography. *IEEE Trans Info Theory* 22(5):644–654
- Dolev D, Dwork C, Naor M (2000) Non-malleable cryptography. *SIAM J Comput* 30(2):391–437
- Dumais P, Mayers D, Salvail L (2000) Perfectly concealing quantum bit commitment from any quantum one-way permutation. In: EUROCRYPT 2000: Advances in cryptology, Bruges, Belgium, May 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 300–315
- Ekert AK (1991) Quantum cryptography based on Bell's theorem. *Phys Rev Lett* 67:661–663
- Even S, Goldreich O, Lempel A (1985) A randomized protocol for signing contracts. *Commun ACM* 28(6):637–647
- Goldreich O, Levin LA (1989) A hard-core predicate for all one-way functions. In: Proceedings of the 21st

- ACM symposium on theory of computing, ACM, Seattle, WA, May 1989, pp 25–32
- Goldreich O, Micali S, Wigderson A (1987) How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the 19th ACM symposium on theory of computing, ACM, New York, May 1987, pp 218–229
- Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J Assoc Comput Mach* 38(3):691–729
- Goldwasser S, Micali S (1984) Probabilistic encryption. *J Comput Syst Sci* 28(2):270–299
- Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. *SIAM J Comput* 18(1):186–208
- Goldreich O, Sahai A, Vadhan S (1999) Can statistical zero knowledge be made non-interactive? Or on the relationship of SZK and NISZK. In: CRYPTO 1999: Advances in cryptology, Santa Barbara, CA, August 1999. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 467–484
- Gottesman D, Chuang I (2001) Quantum digital signatures. Available via ArXiv:quant-ph/0103032v2
- Grigni M, Schulman LJ, Vazirani M, Vazirani UV (2004) Quantum mechanical algorithms for the nonabelian hidden subgroup problem. *Combinatorica* 24(1):137–154
- Haitner I, Reingold O (2007) Statistically-hiding commitment from any one-way function. In: Proceedings of the 39th ACM symposium on theory of computing, San Diego, CA, June 2007, pp 1–10
- Haitner I, Horvitz O, Katz J, Koo C-Y, Morselli R, Shaltiel R (2005) Reducing complexity assumptions for statistically-hiding commitment. In: EUROCRYPT 2005: Advances in cryptology, Aarhus, Denmark, May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 58–77
- Hallgren S, Russell A, Ta-Shma A (2003) The hidden subgroup problem and quantum computation using group representations. *SIAM J Comput* 32(4):916–934
- Hallgren S, Moore C, Rötteler M, Russell A, Sen P (2006) Limitations of quantum coset states for graph isomorphism. In: Proceedings of the 38th ACM symposium on theory of computing, ACM, Seattle, WA, May 2006, pp 604–617
- Hardy L, Kent A (2004) Cheat sensitive quantum bit commitment. *Phys Rev Lett* 92(15):157901
- Håstad J, Impagliazzo R, Levin LA, Luby M (1999) A pseudorandom generator from any one-way function. *SIAM J Comput* 28(4):1364–1396
- Hayashi M, Kawachi A, Kobayashi H (2008) Quantum measurements for hidden subgroup problems with optimal sample complexity. *Quantum Info Comput* 8:345–358
- Kashefi E, Nishimura H, Vedral V (2002) On quantum one-way permutations. *Quantum Info Comput* 2(5):379–398
- Kawachi A, Yamakami T (2006) Quantum hardcore functions by complexity-theoretical quantum list decoding. In: Proceedings of the 33rd international colloquium on automata, languages and programming, Venice, Italy, July 2006. Lecture notes in computer science, vol 4052. Springer, Berlin, pp 216–227
- Kawachi A, Koshiya T, Nishimura H, Yamakami T (2005a) Computational indistinguishability between quantum states and its cryptographic application. In: EUROCRYPT 2005: Advances in cryptology, Aarhus, Denmark, May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 268–284
- Kawachi A, Kobayashi H, Koshiya T, Putra RRH (2005b) Universal test for quantum one-way permutations. *Theor Comput Sci* 345(2–3):370–385
- Kempe J, Pyber L, Shalev A (2007) Permutation groups, minimal degrees and quantum computing. *Groups Geometry Dyn* 1(4):553–584
- Kent A (2003) Quantum bit string commitment. *Phys Rev Lett* 90(23):237901
- Kilian J (1988) Founding cryptography on oblivious transfer. In: Proceedings of the 20th ACM symposium on theory of computing, ACM, Chicago, IL, May 1988, pp 20–31
- Koashi M, Preskill J (2003) Secure quantum key distribution with an uncharacterized source. *Phys Rev Lett* 90:057902
- Kobayashi H (2003) Non-interactive quantum perfect and statistical zero-knowledge. In: Proceedings of the 14th international symposium on algorithms and computation, Kyoto, Japan, December 2003. Lecture notes in computer science, vol 2906. Springer, Berlin, pp 178–188
- Kobayashi H (2008) General properties of quantum zero-knowledge proofs. In: Proceedings of the 5th theory of cryptography conference, New York, March 2008. Lecture notes in computer science, vol 4948. Springer, New York, pp 107–124
- Kobayashi T, Odaira T (2009) Statistically-hiding quantum bit commitment from approximable-preimage-size quantum one-way function. In: Proceedings of the 4th workshop on theory of quantum computation, communication and cryptography, Waterloo, ON, Canada, May 2009. Lecture notes in computer science, vol 5906. Springer, Berlin, pp 33–46
- Lamport L (1979) Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International
- Lo H-K, Chau HF (1997) Is quantum bit commitment really possible? *Phys Rev Lett* 78(17):3410–3413
- Marriott C, Watrous J (2004) Quantum Arthur-Merlin games. In: Proceedings of the 19th IEEE conference

- on computational complexity, IEEE, Amherst, MA, June 2004, pp 275–285
- Mayers D (1996) Quantum key distribution and string oblivious transfer in noisy channels. In: CRYPTO'96: Advances in cryptology, Santa Barbara, CA, August 1996. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 343–357
- Mayers D (1997) Unconditionally secure quantum bit commitment is impossible. *Phys Rev Lett* 78(17):3414–3417
- Mayers D, Salvail L (1994) Quantum oblivious transfer is secure against all individual measurements. In: Proceedings of workshop on physics and computation, IEEE, Dallas, TX, November 1994, pp 69–77
- Micciancio D, Regev O (2009) Lattice-based cryptography. In Bernstein DJ, Buchmann J, Dahmen E (eds) Post-quantum cryptography. Springer, Berlin, pp 147–191
- Naor M (1991) Bit commitment using pseudorandomness. *J Cryptol* 4(2):151–158
- Naor M, Yung M (1989) Universal one-way hash functions and their cryptographic applications. In: Proceedings of the 21st ACM symposium on theory of computing, ACM, Seattle, WA, May 1989, pp 33–43
- Naor M, Ostrovsky R, Venkatesan R, Yung M (1998) Perfect zero-knowledge arguments for NP using any one-way permutation. *J Cryptol* 11(2):87–108
- Nguyen M-H, Ong S-J, Vadhan SP (2006) Statistical zero-knowledge arguments for NP from any one-way function. In: Proceedings of the 47th IEEE symposium on foundations of computer science, IEEE, Berkeley, CA, October 2006, pp 3–14
- Nguyen PQ, Stern J (2005) Adapting density attacks to low-weight knapsacks. In: ASIACRYPT 2005: Advances in cryptology, Chennai, India, December 2005. Lecture notes in computer science, vol 3788. Springer, Berlin, pp 41–58
- Okamoto T, Tanaka K, Uchiyama S (2000) Quantum public-key cryptosystems. In: CRYPTO 2000: Advances in cryptology, Santa Barbara, CA, August 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 147–165
- Rabin M (1981) How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University
- Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signature and public key cryptosystems. *Commun ACM* 21(2):120–126
- Sahai A, Vadhan S (2003) A complete problem for statistical zero knowledge. *J ACM* 50(2):196–249
- Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput* 26(5):1484–1509
- Shor PW, Preskill J (2000) Simple proof of security of the BB84 quantum key distribution protocol. *Phys Rev Lett* 85:441–444
- Sudan M (2000) List decoding: algorithms and applications. *SIGACT News* 31(1):16–27
- van de Graaf J (1997) Towards a formal definition of security for quantum protocols. PhD thesis, Université de Montréal
- Watrous J (2002) Limits on the power of quantum statistical zero-knowledge. In: Proceedings of the 43rd IEEE symposium on foundations of computer science, IEEE, Vancouver, BC, Canada, November 2002, pp 459–470
- Watrous J (2006) Zero-knowledge against quantum attacks. In: Proceedings of the 38th annual ACM symposium on theory of computing, ACM, Seattle, WA, May 2006, pp 296–305
- Wegman MN, Carter JL (1981) New hash functions and their use in authentication and set equality. *J Comput Syst Sci* 22(3):265–279
- Wiesner S (1983) Conjugate coding. *SIGACT News* 15(1):78–88
- Yao AC-C (1995) Security of quantum protocols against coherent measurements. In: Proceedings of the 27th annual ACM symposium on theory of computing, ACM, Las Vegas, NV, May–June 1995, pp 67–75

46 BQP-Complete Problems

Shengyu Zhang

Department of Computer Science and Engineering, The Chinese
University of Hong Kong, Hong Kong S.A.R., China
syzhang@cse.cuhk.edu.hk

1	<i>Introduction</i>	1546
2	<i>Preliminaries</i>	1548
3	<i>Approximate Eigenvalue Sampling</i>	1549
4	<i>Additive Approximation of the Jones Polynomials and Tutte Polynomials</i>	1557
5	<i>Concluding Remarks</i>	1568

Abstract

The concept of completeness is one of the most important notions in theoretical computer science. **PromiseBQP**-complete problems are those in **PromiseBQP** to which all other **PromiseBQP** problems can be reduced in classically probabilistic polynomial time. Studies of **PromiseBQP**-complete problems can deepen our understanding of both the power and limitation of efficient quantum computation. In this chapter we give a review of known **PromiseBQP**-complete problems, including various problems related to the eigenvalues of sparse Hamiltonians and problems about additive approximation of Jones polynomials and Tutte polynomials.

1 Introduction

A celebrated discovery in theoretical computer science is the existence of **NP**-complete problems. A decision problem is **NP**-complete if it is in **NP**, and all other problems in **NP** reduce to it in deterministic polynomial time. After Cook (1971) (and Levin (1973) independently) showed that the Satisfiability problem is **NP**-complete, Karp (1972) found that 21 natural combinatorial problems, mostly problems on graphs, are **NP**-complete as well. Since then, thousands of problems arising from various disciplines such as mathematics, physics, chemistry, biology, information science, etc. have been found to be **NP**-complete (Garey and Johnson 1979).

Why is the notion of **NP**-completeness so important? First, showing a problem to be **NP**-complete means that the problem is very unlikely to be solvable in polynomial time. This makes **NP**-completeness an “important intellectual export of computer science to other disciplines” (Papadimitriou 1997). Second, the number of **NP**-complete problems that arise naturally is huge; thus **NP**-complete problems give a nice classification of **NP** problems. Actually, all but a few **NP** problems are known to be either in **P** or **NP**-complete. Nowadays, whenever people find a new problem in **NP** and cannot quickly find a polynomial-time algorithm, they start to consider whether it is **NP**-complete. Third, the notion of completeness provides a useful method to study a whole class by concentrating on one specific problem. For example, if one wants to separate **NP** from **P**, it is enough to show that any **NP**-complete problem does not have a deterministic polynomial-time algorithm. Of course, if one tries to explore the possibility of **NP** = **P** by designing efficient algorithms for **NP**, it is also sufficient to find an efficient algorithm on any one of the **NP**-complete problems.

Complete problems do not only exist in **NP**, they also exist in other computational classes, although the requirement for the reduction algorithm may vary from case to case. To study a property of a class, it is enough to show the property for one complete problem in the class, as long as the reduction used in the completeness definition does not destroy the property. For example, the fact that Graph 3-Coloring has a (computational) Zero-Knowledge protocol immediately implies that all **NP** problems have a Zero-Knowledge protocol, because for an arbitrary **NP** problem, the prover and verifier can first map the input to a Graph 3-Coloring instance and then run the Zero-Knowledge protocol for the latter (Goldreich et al. 1991).

Quantum computing is a new paradigm rapidly developed since the mid-1990s. Since Shor’s fast quantum algorithm for Factoring and Discrete Log caused great excitement in both the physics and computer science communities, many efforts have been put into designing quantum algorithms with exponential speedup over their classical counterparts. However, the progress of this line of research has been disappointingly slower than what people had

expected. There are a few quantum algorithms with exponential speedup, such as Hallgren's polynomial-time quantum algorithm for solving Pell's equation (Hallgren 2007) and Kuperberg's $2^{O(\sqrt{n})}$ time quantum algorithm for solving the Hidden Subgroup Problem (Lomont 2004) for a dihedral group (Kuperberg 2005); see a recent survey (Childs and van Dam 2010) for a more comprehensive review. But all the problems are number-theoretical or algebraic. (One exception is the oracle separation for the Glued Tree problem by Childs et al. based on a continuous quantum walk (Childs et al. 2003), but the problem is somewhat artificial.) Why does it seem that fast quantum algorithms are much harder to design and what should one do next? In a survey (Shor 2004), Shor gave a couple of possible reasons, including that one does not have enough intuition and experience in dealing with quantum information, and that there may not be many natural problems in quantum computers that can have speedup. He also suggested to try to first study problems that are solvable on a classical computer, aiming at developing algorithmic designing tools with extensive usefulness. It is hoped that studies of BQP-complete problems can shed light on these questions and deepen our understanding of the power and limitation of quantum computation.

BQP is the computational class containing decision problems that are solvable probabilistically on a polynomial-time quantum computer. Since a central problem in quantum computing is the comparison between the computational power of quantum and classical computation, it is natural to see what extra power a quantum computer gives one. In this regard, let **BQP**-completeness be defined with the reduction being **BPP** algorithms, that is, those (classical) probabilistic polynomial-time algorithms. In other words, a problem is **BQP**-complete if it is in **BQP**, and all other **BQP** problems can reduce it by probabilistic polynomial-time classical algorithms. Analogous to the fact that **NP**-complete problems are the “hardest” problems in **NP** and thus capture the computational power of efficient nondeterministic computation, **BQP**-complete problems are the hardest problems in **BQP** and thus capture the computational power of efficient quantum computation.

Like many other “semantic” complexity classes, **BQP** is not known to contain complete problems. What people usually study for completeness, in such a scenario, is the class containing the *promise* problems, that is, those decision problems for which the union of Yes and No input instances is not necessarily the whole set of $\{0,1\}$ strings. In our quantum case, it is the class **PromiseBQP**, that is, the collection of promise problems solvable in polynomial time on a quantum computer. There are mainly two tracks of **PromiseBQP**-complete problems. The first track contains problems on the eigenvalues of a local or sparse Hamiltonian or unitary matrix (Wocjan and Zhang 2006; Janzing and Wocjan 2007; Janzing et al. 2008). In the first work (Wocjan and Zhang 2006) along this line, Wocjan and Zhang considered the local Hamiltonian eigenvalue sampling (LHES) and local unitary phase sampling (LUPS) problems: Given a classical string $x \in \{0, 1\}^n$ and a $2^n \times 2^n$ dimensional local Hamiltonian $H = \sum_j H_j$ or $U = \prod_j U_j$, where each H_j or U_j are operating on a constant number of qubits, one is approximately sampling the eigenvalues of H or U under the distribution $\langle x | \eta_j \rangle$, where $|\eta_j\rangle$ are the corresponding eigenvectors. These two problems have close connections to other well-known problems in quantum algorithm and complexity theory: LHES is the natural sampling variant of estimating the minimum eigenvalue of a local Hamiltonian, a **QMA**-complete problem (Kitaev et al. 2002; Kempe et al. 2006), and LUPS is the natural sampling variant of estimating the eigenvalue of a unitary on a given eigenvector, a powerful algorithmic tool called phase estimation (Kitaev 1995). Though not defined as a promise problem, it will be shown in this survey that one can easily transform them into promise problems and thus be **PromiseBQP**-complete.

Later, the work was extended in two ways. In Janzing et al. (2008), Janzing, Wocjan, and Zhang showed that even if one restricts the local Hamiltonians in LHES to translationally invariant ones operating on a one-dimensional qudit chain, the problem is still **PromiseBQP**-complete. The other extension, done by Janzing and Wocjan (2007), views the Hamiltonian as a ± 1 weighted graph and lifts the graph to power k , where k is part of the input. Then estimating a diagonal entry is **PromiseBQP**-complete.

The second line of research on **PromiseBQP**-complete problems is about approximating the Jones polynomial and Tutte polynomial (Freedman et al. 2002b, c; Aharonov and Arad 2006; Aharonov et al. 2006, 2007a; Wocjan and Yard 2006). The Jones polynomial is an important knot invariant with rich connections to topological quantum field theory, statistical physics (Wu 1992) and DNA recombination (Podtelezhnikov et al. 1999). The main result for the Jones polynomial in the studies of **PromiseBQP**-completeness is that approximating the Jones polynomial of the plat closure of the braid group at $e^{2\pi i/k}$ to within some precision is **PromiseBQP**-complete. Both the efficient quantum algorithm and the universality for constant k were implicitly given by Freedman et al. (2002b, c), but recent results by Aharonov et al. (2006) and Aharonov and Arad (2006) gave an explicit and simpler algorithm and a hardness proof, which also extend the results to any k bounded by a polynomial of the size of the input braid.

The multivariate Tutte polynomial is more general than the Jones polynomial; the Tutte polynomial has many connections to algebraic graph theory and the Potts model in statistical physics. In Aharonov et al. (2007a), Aharonov, Arad, Eban, and Landau gave an efficient algorithm for additively approximating this general polynomial and also showed that the approximation for some ranges is **PromiseBQP**-complete.

The rest of the chapter is organized as follows. ◻ [Section 2](#) gives precise definitions of **BQP** and **PromiseBQP**. In ◻ [Sects. 3](#) and ◻ [4](#), the two lines of research on **PromiseBQP**-complete problems are studied. ◻ [Section 5](#) concludes with some questions raised.

2 Preliminaries

A promise problem is a pair $(L_{\text{Yes}}, L_{\text{No}})$ of nonintersecting subsets of $\{0, 1\}^*$. A language is the set of Yes instances of a promise $(L_{\text{Yes}}, L_{\text{No}})$ satisfying $L_{\text{Yes}} \cup L_{\text{No}} = \{0, 1\}^*$. For more discussions on promise problems versus languages (and why the former is important and sometimes necessary for complexity theory), the readers are referred to Goldreich's survey (Goldreich 2005).

Definition 1 (**PromiseBQP**) **PromiseBQP** is the class of promise problems $(L_{\text{Yes}}, L_{\text{No}})$ such that there is a uniform family of quantum circuits $\{U_n\}$ operating on $p(n) = \text{poly}(n)$ qubits, with the promise that for any n , applying U_n on $|x, 0^{p(n)-n}\rangle$ and measuring the first qubit gives the outcome 1 with probability at least $2/3$ for all $x \in L_{\text{Yes}}$, and at most $1/3$ for all $x \in L_{\text{No}}$.

Definition 2 (**BQP**) **BQP** is the class of languages L_{Yes} with $(L_{\text{Yes}}, \{0, 1\}^* - L_{\text{Yes}})$ in **PromiseBQP**.

Like many other “semantic” classes, **BQP** is not known to have complete problems. However, if one extends languages to promise problems, then **PromiseBQP** has canonical complete

problems for any model, such as the quantum Turing machine (Bernstein and Vazirani 1997), quantum circuit (Yao 1993), adiabatic quantum computer (Aharonov et al. 2004), and quantum one-way computer (Browne and Briegel 2006). For the quantum circuit model, for example, the canonical complete problem is the following.

Definition 3 The *canonical complete problem* for **PromiseBQP** in the circuit model has input $(\langle U \rangle, x)$, where $\langle U \rangle$ is the description of a uniform family of quantum circuits $\{U_n\}$ working on $p(n) = \text{poly}(n)$ qubits, with the promise that applying U_n on $|x, 0^{p(n)-n}\rangle$ and measuring the first qubit gives outcome 1 with probability at least $2/3$ for all $x \in L_{\text{Yes}}$ and at most $1/3$ for all $x \in L_{\text{No}}$. The problem is to distinguish the two cases.

The fact that this is a complete problem for **PromiseBQP** is almost by definition, which also makes the problem not so interesting. We hope to have more “natural” problems that can help us understand the class **PromiseBQP**.

3 Approximate Eigenvalue Sampling

This section provides an overview of the first track of studies on **PromiseBQP**-complete problems, that is on problems about eigenvalues of the local or sparse Hamiltonians.

➊ **Section 3.1** starts with the local Hamiltonian eigenvalue sampling (LHES) problem, defined in Wocjan and Zhang (2006), and we show that it is **PromiseBQP**-complete. The problem together with the proof contains the core ideas and techniques for this line of research.

Two extensions are then shown. First, the result in Janzing et al. (2008) is mentioned: even if the Hamiltonians are restricted to be translationally invariant on a one-dimensional qudit chain (with $d = O(1)$), the problem is still **PromiseBQP**-complete. Then ➋ **Sect. 3.2** shows the result in Janzing and Wocjan (2007) that estimating an entry in a sparse real symmetric matrix to some power k , where k is part of the input, is also **PromiseBQP**-complete.

The proofs in the three sections have similar ingredients, and an attempt is made to unify the notation and give a consistent treatment.

3.1 Phase Sampling and Local Hamiltonian Eigenvalue Sampling

The problems that one is going to see in this section have close relation to two well-known ones, local Hamiltonian minimum eigenvalue and phase estimation, which will be defined below. The local Hamiltonian minimum eigenvalue (LHME) problem is usually abbreviated as the local Hamiltonian (LH) problem, but here LHME is used to distinguish it from the other related problems, which are defined later.

Definition 4 (Local Hamiltonian minimum eigenvalue (LHME)) A tuple (H, a, b) is given where

1. $H = \sum_j H_j$ is a Hamiltonian operating on n qubits, with j ranging over a set of size polynomial in n , and each H_j operating on a constant number of qubits; it is promised that either $\lambda(H) < a$ or $\lambda(H) > b$, where $\lambda(H)$ is the minimum eigenvalue of H .
2. a and b are two real numbers such that $a < b$ and the gap $b - a = \Omega(1/\text{poly}(n))$.

The task is to distinguish between the case $\lambda(H) < a$ and the case $\lambda(H) > b$.

Definition 5 (Phase estimation (PE)) A unitary matrix U is given by black-boxes of controlled- U , controlled- U^{2^2}, \dots , controlled- $U^{2^{t-1}}$ operations, and an eigenvector $|u\rangle$ of U with eigenvalue $e^{2\pi i\varphi}$ with the value of $\varphi \in [0, 1)$ unknown. The task is to output an n -bit estimation of φ .

These two problems are both well studied in quantum computing. The Local Hamiltonian problem was shown by Kitaev et al. (2002) to be **PromiseQMA**-complete when each H_j operates on five qubits; actually it remains **PromiseQMA**-complete even when each H_j operates on only two qubits (Kempe et al. 2006). Kitaev's efficient quantum algorithm for phase estimation (Kitaev 1995) was a powerful tool for quantum algorithm design, such as for factoring (Shor 1997; Nielsen and Chuang 2000) and some recent quantum-walk based algorithms such as the general quantum walk search (Magniez 2007), formula evaluation (Ambainis et al. 2007), and its extension to span-program evaluation (Reichardt and Spalek 2008).

Let us now consider the sampling variant of the above two problems.

Definition 6 A probability distribution q on \mathbb{R} is said to *approximate* another probability distribution p on a discrete set $S \subseteq \mathbb{R}$ with *error* δ and *precision* ε if

$$\Pr_{x \sim q}[s - \varepsilon \leq x \leq s + \varepsilon] \geq (1 - \delta)p(s) \quad (1)$$

for any $s \in S$.

Intuitively, to approximate the probability distribution p , one draws a sample from another distribution q , and the outcome x is ε -close to s at least $(1 - \delta)$ times the correct probability $p(s)$, for each $s \in S$. Now one can define the sampling version of the two problems.

Definition 7 (Local Hamiltonian eigenvalue sampling (LHES)) We are given $(H, \varepsilon, \delta, b)$ where

1. $H = \sum H_j$ is a Hamiltonian operating on n qubits, with j ranging over a set of size polynomial in n , and each H_j operating on a constant number of qubits.
2. $\varepsilon = \Omega(1/\text{poly}(n))$ is the required estimation precision.
3. $\delta = \Omega(1/\text{poly}(n))$ is the required sampling error probability.
4. $b \in \{0, 1\}^n$ is a classical n -bit string.

Suppose the eigenvalues and the corresponding eigenvectors of H are $\{(\lambda_k, |\eta_k\rangle) : k \in [2^n]\}$ satisfying $|\lambda_k| < \text{poly}(n)$ for each k . Define the probability distribution $D(H, b)$ over the spectrum of H by

$$D(H, b) = \{(\lambda, \Pr(\lambda)) : \Pr(\lambda) = \sum_{\lambda_k=\lambda} |\langle b|\eta_k\rangle|^2\}. \quad (2)$$

The task is to draw a sample from some probability distribution p approximating $D(H, b)$ with error δ and precision ε .

Definition 8 (Local unitary phase sampling (LUPS)) We are given $(H, \varepsilon, \delta, b)$ where

1. U is (the description of) an n -qubit quantum circuit.
2. $\varepsilon = \Omega(1/\text{poly}(n))$ is the required estimation precision.

3. $\delta = \Omega(1/\text{poly}(n))$ is the required sampling error probability.
4. $b \in \{0, 1\}^n$ is a classical n -bit string.

Suppose the eigenvalues of U are $\{\lambda_j = e^{2\pi i \varphi_j}\}_{j=1, \dots, 2^n}$ (where $\varphi_j \in [0, 1)$ for each j), with the corresponding eigenvectors $\{|\eta_j\rangle\}_{j=1, \dots, 2^n}$. The task is to estimate φ_j with error δ and precision ε .

One can immediately see that these two problems are not even promise problems, since the output is a (sampling from a) distribution rather than a Yes/No answer. Nevertheless, they capture the power of efficient quantum computers in the following sense: First, a polynomial-time uniform family of quantum circuits can achieve the sampling requirement. Second, given an oracle for either problem, any **BQP** problem can be solved by a classical polynomial-time algorithm. These facts will be proved in the following two sections. After that, a mention will be made about how to change LHES to a promise problem so that it really becomes **PromiseBQP**-complete.

3.1.1 BQP Algorithm for LHES and LUES

In this section we prove that LHES and LUES are solvable by an efficient quantum circuit. The standard algorithm for phase estimation is first reviewed, then it is observed that the same algorithm actually gives the desired LUES solution. This is then used to show an algorithm for LHES.

Phase estimation can be solved by a quantum algorithm as follows. The working space has two registers. The first register consists of $t = n + \lceil \log(2 + 1/2\delta) \rceil$ qubits and is prepared in $|0\dots0\rangle$. The second register contains the eigenvector $|u\rangle$. Measuring $\tilde{\varphi}$ in the first register after carrying out the transformations described below gives the desired n -bit estimation of φ with probability of at least $1 - \delta$.

$$|0\rangle^{\otimes t}|u\rangle \quad (3)$$

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle \quad (\text{apply the Fourier transform}) \quad (4)$$

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j|u\rangle \quad (\text{apply the controlled powers of } U) \quad (5)$$

$$= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j \varphi} |j\rangle|u\rangle \quad (6)$$

$$\rightarrow |\tilde{\varphi}\rangle|u\rangle \quad (\text{apply the inverse Fourier transform}) \quad (7)$$

The following observation says that the same algorithm actually works for LUES.

Fact If one feeds $|0\rangle|b\rangle$ instead of $|0\rangle|u\rangle$ as input to the above algorithm for the phase estimation problem and $t = \lceil \log \frac{1}{\varepsilon} \rceil + \lceil \log(2 + \frac{1}{2\delta}) \rceil$, then the measurement of the first register gives the desired sampling output. This implies that LUES can be solved by a polynomial-size quantum circuit.

This actually holds not only for $|b\rangle$ but also for a general state $|\eta\rangle$. To see why this is true, write $|\eta\rangle$ as $\sum_{j=1}^n \alpha_j |\eta_j\rangle$, then by the linearity of the operations, the final state is $\alpha_j |\tilde{\varphi}_j\rangle |\eta_j\rangle$. For more details, the readers are referred elsewhere (Nielsen and Chuang 2000; Chap. 5). Note that to implement the controlled- U^{2^j} operations for $j = 0, \dots, 2^t - 1$ in the above algorithm, one needs to run U for 2^t times, which can be done efficiently since $t = \lceil \log \frac{1}{\varepsilon} \rceil + \lceil \log(2 + \frac{1}{2\delta}) \rceil$ and $\varepsilon = \Omega(1/\text{poly}(n))$, $\delta = \Omega(1/\text{poly}(n))$.

Now we can give the efficient algorithm for LHES.

Theorem 1 *LHES can be implemented by a uniform family of quantum circuits of polynomial size.*

Proof By a simple scaling ($H' = H/\Lambda = \sum_j H_j/\Lambda$ where $\Lambda = \max_k |\lambda_k| = \text{poly}(n)$), one can assume that all the eigenvalues λ_k of H satisfy $|\lambda_k| < 1/4$. The basic idea to design the quantum algorithm is to use LUPS on $e^{2\pi i H}$. Note that $e^{2\pi i H}$ is unitary, and if the eigenvalues and eigenvectors of H are $\{\lambda_k |\eta_k\rangle\}$, then those of $e^{2\pi i H}$ are just $\{e^{2\pi i \lambda_k} |\eta_k\rangle\}$. Therefore, it seems that it is enough to run the LUES algorithm on $(e^{2\pi i H}, \varepsilon, \delta, b)$, and if one gets some $\lambda > 1/2$, then output $\lambda - 1$. However, note that H is of exponential dimension, so $e^{2\pi i H}$ is not ready to compute in the straightforward way. Fortunately, this issue is well studied in the quantum simulation algorithms, and the standard approach is the following asymptotic approximation using the Trotter formula (Trotter 1959; Chernoff 1968; Nielsen and Chuang 2000) or its variants. Here using the simulation technique, one obtains

$$\left(e^{2\pi i \sum_j H_j/m}\right)^m = \left(\prod_j e^{2\pi i H_j/m}\right)^m + O(1/m) \quad (8)$$

Now one runs LUES on $(b, \varepsilon, \delta/2, e^{2\pi i H})$. Whenever one needs to call $e^{2\pi i H}$, one uses $\prod_j e^{2\pi i H_j/m}$ for m times instead. Note that such substitution yields $O(1/m)$ deviation, so $t = \log \frac{2}{\varepsilon\delta} + O(1)$ calls yield $O\left(\frac{1}{\varepsilon\delta m}\right) \leq \frac{c}{m\varepsilon\delta}$ deviation for some constant c . Let $m = \frac{2c}{\varepsilon\delta^2}$, thus the final error probability is less than $\frac{\delta}{2} + \frac{c}{m\varepsilon\delta} = \delta$, achieving the desired estimation and sampling precisions.

From the proof one can see that as long as $e^{2\pi i H}$ can be simulated efficiently from the description of H , one can sample the eigenvalues of H as desired. Since sparse Hamiltonians, which contain local Hamiltonians as special cases, can be simulated efficiently (Aharonov and Ta-Shma 2003; Berry et al. 2007), we know that if we modify the definition of LHES by allowing H to be sparse then it is also BQP-complete.

3.1.2 BQP Hardness of LHES

Theorem 2 $\text{PromiseBQP} \subseteq \mathbf{P}^{\text{LHES}}$

Proof For any $L \in \text{PromiseBQP}$, there is a uniform family of polynomial size quantum circuits with ε -bounded error (for a small constant ε) that decides if $x \in L$ or $x \notin L$. Denote by

U the corresponding quantum circuit (of size n) and suppose the size of U is M , which is bounded by a polynomial in n . Further suppose that the computation is described by $U|x, \mathbf{0}\rangle = \alpha_{x,0}|0\rangle|\psi_{x,0}\rangle + \alpha_{x,1}|1\rangle|\psi_{x,1}\rangle$, where $\mathbf{0}$ is the initial state of the ancillary qubits, and $|\psi_{x,0}\rangle$ and $|\psi_{x,1}\rangle$ are pure states. After the U transform, the first qubit is measured and the algorithm outputs the result based on the outcome of the measurement. The correctness of the algorithm requires that $|\alpha_{x,0}|^2 < \varepsilon$ if $x \in L$, and $|\alpha_{x,1}|^2 < \varepsilon$ if $x \notin L$.

Now, a local Hamiltonian H is constructed encoding the circuit U and a binary string b encoding the inputs x , such that eigenvalue sampling applied to H and b yields significantly different probability distributions for the two cases of $x \in L$ and $x \notin L$.

To this end, the circuit V is constructed as follows: one first applies the original circuit U , then flips the sign for the component with the first qubit being 1, and finally reverses the computation of U . See  Fig. 1.

Suppose the original circuit U is decomposed as the product of elementary gates, that is, $U = U_{m-1} \dots U_0$, where each U_j is an elementary gate. Let V_j be the j -th gate in V , where $j = 0, 1, \dots, 2m$. Let $M = 2m + 1$, the number of gates in V . Attach a clock register to the system and define the operator

$$F = \sum_{j=0}^{M-1} V_j \otimes |j+1\rangle\langle j| \quad (9)$$

where the summation in the index is module M . Note that F is an $O(\log N)$ -local operator; we will remark how to slightly modify it to be a 4-local operator at the end of the proof. Define

$$|\varphi_{x,j}\rangle = F^j(|x, \mathbf{0}\rangle|0\rangle) \quad (10)$$

for $j \geq 0$, where F^j means that F is applied j times. It is easy to see that $|\varphi_{x,j}\rangle = |\varphi_{x, j \bmod 2M}\rangle$ for $j \geq 2M$.

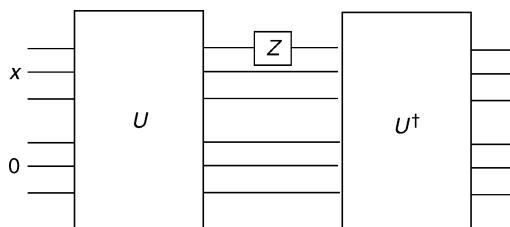
Note that due to the clock register for different j and j' in $\{0, \dots, 2M-1\}$, $|\varphi_{x,j}\rangle$ and $|\varphi_{x,j'}\rangle$ are orthogonal if $|j-j'| \neq M$. Also note that $U_{m-1} \dots U_0|x, \mathbf{0}\rangle = U|x, \mathbf{0}\rangle = \alpha_{x,0}|0\rangle|\psi_{x,0}\rangle + \alpha_{x,1}|1\rangle|\psi_{x,1}\rangle$ by definition. Therefore, we have

$$\langle \varphi_{x,j} | \varphi_{x,M+j} \rangle = \langle x, \mathbf{0} | U_0^\dagger \dots U_{m-1}^\dagger (P_0 - P_1) U_{m-1} \dots U_0 | x, \mathbf{0} \rangle = |\alpha_{x,0}|^2 - |\alpha_{x,1}|^2 \quad (11)$$

for $j = 0, \dots, M-1$, where P_0 and P_1 are the projections onto the subspaces corresponding to the first qubit being 0 and 1, respectively. To summarize, we have

$$\langle \varphi_{x,j} | \varphi_{x,j'} \rangle = \begin{cases} 0 & |j-j'| \neq M \\ |\alpha_{x,0}|^2 - |\alpha_{x,1}|^2 & |j-j'| = M \end{cases} \quad (12)$$

 Fig. 1
Circuit V .



Define the subspace $S_x = \text{span}\{|\varphi_{x,j}\rangle : j = 0, \dots, 2M - 1\}$. The key property here is that though F has an exponentially large dimension, $F|_{S_x}$ is of only polynomial dimension. Depending on the probability, we consider three cases.

Case $|\alpha_{x,0}| = 1$: The dimension of S_x is M , and $F|_{S_x}$ is a shift operator on the basis $\{|\varphi_{x,j}\rangle\}_{j=0, \dots, M-1}$, that is $F|\varphi_{x,j}\rangle = |\varphi_{x,j+1 \bmod M}\rangle$. It is not hard to see that this operator has eigenvalues and the corresponding eigenvectors

$$\lambda_k = \omega_M^k, \quad |\xi_k\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \omega_M^{-kj} |\varphi_{x,j}\rangle, \quad k = 0, 1, \dots, M-1 \quad (13)$$

where $\omega_M = e^{2\pi i/M}$.

Case $\alpha_{x,0} = 0$: The dimension of S_x is M , and $F|_{S_x}$ is almost a shift operator on the basis $\{|\varphi_{x,j}\rangle\}_{j=0, \dots, M-1}$: $F|\varphi_{x,j}\rangle = |\varphi_{x,j+1}\rangle$ for all $j = 0, \dots, M-2$, and $F|\varphi_{x,M-1}\rangle = -|\varphi_{x,0}\rangle$. It is not hard to see that this operator has eigenvalues and the corresponding eigenvectors

$$\mu_k = \omega_M^{k+1/2}, \quad |\eta_k\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \omega_M^{-(k+1/2)j} |\varphi_{x,j}\rangle, \quad k = 0, 1, \dots, M-1. \quad (14)$$

Case $0 < |\alpha_{x,0}| < 1$: In this general case, S_x has dimension $2M$. To find the eigenvalues and eigenvectors of $F|_{S_x}$, define

$$|\phi_{x,j}\rangle = \frac{|\varphi_{x,j}\rangle + |\varphi_{x,M+j}\rangle}{\||\varphi_{x,j}\rangle + |\varphi_{x,M+j}\rangle\|}, \quad |\gamma_{x,j}\rangle = \frac{|\varphi_{x,j}\rangle - |\varphi_{x,M+j}\rangle}{\||\varphi_{x,j}\rangle - |\varphi_{x,M+j}\rangle\|} \quad (15)$$

for $j = 0, \dots, M-1$. Then first, because $\langle \varphi_{x,j} | \varphi_{x,N+j} \rangle = |\alpha_{x,0}|^2 - |\alpha_{x,1}|^2$ is a real number, we have $\langle \phi_{x,j} | \gamma_{x,j} \rangle = 0$. Together with [Eq. 12](#), we know that

$$\{|\phi_{x,0}\rangle, \dots, |\phi_{x,M-1}\rangle, |\gamma_{x,0}\rangle, \dots, |\gamma_{x,M-1}\rangle\} \quad (16)$$

forms an orthonormal basis of S_x . One can further observe that $F|_{S_x} = F_0 \oplus F_1$, where F_0 and F_1 act on $S_{x,+} = \text{span}\{|\phi_{x,0}\rangle, \dots, |\phi_{x,M-1}\rangle\}$ and $S_{x,-} = \text{span}\{|\gamma_{x,0}\rangle, \dots, |\gamma_{x,M-1}\rangle\}$, respectively, with the matrix representations (in the basis $\{|\phi_{x,j}\rangle\}$ and $\{|\gamma_{x,j}\rangle\}$, respectively) as follows:

$$F_0 = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}, \quad F_1 = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \quad (17)$$

So F_0 and F_1 have the same matrix representations as the two operators in the previous two cases, thus having the same eigenvalues and eigenvectors (with respect to different basis vectors though). Precisely, F_0 has eigenvalues λ_k with eigenvectors $|\xi_k\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \omega_M^{-kj} |\phi_{x,j}\rangle$; F_1 has eigenvalues μ_k with eigenvectors $|\eta_k\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \omega_M^{-(k+1/2)j} |\gamma_{x,j}\rangle$. (Here one uses the same notation $|\xi_k\rangle$ and $|\eta_k\rangle$ because they are consistent with the previous cases when $|\alpha_{x,0}|^2 = 0$ or 1 respectively.)

By this, it is not hard to find the eigenvalue sampling probabilities:

$$|\langle \varphi_{x,0} | \xi_k \rangle|^2 = |\langle \varphi_{x,0} | \frac{1}{M} \sum_j w_M^{-kj} \phi_{x,j} \rangle|^2 \quad (18)$$

$$= \frac{1}{M} |\langle \varphi_{x,0} | \phi_{x,0} \rangle|^2 \quad (\text{by Eq. 12}) \quad (19)$$

$$= \frac{1}{M} \left| \left\langle \varphi_{x,0} \left| \frac{\varphi_{x,0} + \varphi_{x,M}}{\|\varphi_{x,0}\rangle + |\varphi_{x,M}\rangle\|} \right. \right\rangle \right|^2 \quad (\text{by def of } |\phi_{x,0}\rangle) \quad (20)$$

$$= \frac{|\alpha_{x,0}|^2}{M} \quad (\text{by Eq. 12}) \quad (21)$$

and similarly we have

$$|\langle \varphi_{x,0} | \eta_k \rangle|^2 = \frac{|\alpha_{x,1}|^2}{M} \quad (22)$$

Now the local Hamiltonian is constructed as

$$H = (F + F^\dagger)/2. \quad (23)$$

It is easy to verify that H is a local Hamiltonian. And further, suppose the eigenvalues of F are $\{\kappa_j\}$ with the eigenvectors $\{|\psi_j\rangle\}$, then the eigenvalues of H are just $\{(\kappa_j + \kappa_j^*)/2\}$ with the same corresponding eigenvectors. Thus H has eigenvalues $\cos(2k\pi/M)$ and $\cos((2k+1)\pi/M)$ for $k=0, 1, \dots, (M-1)/2$, and the distribution $D(H, |\varphi_{x,0}\rangle)$ is $\{\cos(2k\pi/M)\}$ with probability $|\alpha_{x,0}|^2/M$, and $\cos((2k+1)\pi/M)$ with probability $|\alpha_{x,1}|^2/M$.

Now if $x \in L_{\text{Yes}}$, then $|\alpha_{x,0}|^2$ is very small, thus with high probability the sample gives a random value uniformly chosen from $\{\cos(2k\pi/M): k = 0, \dots, (M-1)/2\}$. If $x \in L_{\text{No}}$, then $|\alpha_{x,1}|^2$ is very small, thus with high probability the sample gives a random value uniformly chosen from $\{\cos((2k+1)\pi/M): k = 0, \dots, (M-1)/2\}$. Since any two values chosen from these two sets are at least $\Omega(M^{-2})$ away from each other, an approximator with precision $O(M^{-2})$ suffices to distinguish between these two cases.

Finally, to obtain a 4-local LHES, $|i\rangle$ is replaced by $|e_i\rangle = |0\dots010\dots0\rangle$ for $i = 0, \dots, N-1$, where the only 1 appears at coordinate i . Modify the operator F to be

$$F = \sum_{j=0}^{M-1} V_j \otimes |e_{j+1}\rangle\langle e_j| \quad (24)$$

Note that $|e_j\rangle\langle e_{j-1}|$ and $|e_0\rangle\langle e_{N-1}|$ are 2-local. The remaining proof passes through. This completes the proof for $\textcircled{2}$ [Theorem 2](#).

Changing It to a Promise Problem

Now that we have completed both the algorithm and the completeness, it is not hard to see that the sampling problem can be modified to a promise problem as follows. We can derive its **PromiseBQP**-completeness from the above proofs. Since this was not mentioned in the previous work (Wocjan and Zhang 2006), a bit more detail is given below. Let

$$p_0(M) = \text{uniform distribution over } \{\cos(2k\pi/M): k = 0, 1, \dots, (M-1)/2\};$$

$$p_1(M) = \text{uniform distribution over } \{\cos((2k+1)\pi/M): k = 0, 1, \dots, (M-1)/2\}.$$

Definition 9 (Local Hamiltonian eigenvalue distribution (LHED)). We are given $(H, \varepsilon, \delta, b, M)$ where

1. $H = \sum H_j$ is a Hamiltonian operating on n qubits, with j ranging over a set of size polynomial in n , and each H_j operating on a constant number of qubits.
2. $\varepsilon = \Omega(1/\text{poly}(n))$, $\delta = o(M^{-2})$ is the required estimation precision.

3. $\delta = \Omega(1/\text{poly}(n))$, $\delta < 1/10$ is the required sampling error probability.
4. $b \in \{0, 1\}^n$ is a classical n -bit string.
5. $M = \text{poly}(n)$.

Suppose the eigenvalues and the corresponding eigenvectors of H are $\{(\lambda_k | \eta_k)\}$: $k \in [2^n]$ satisfying $|\lambda_k| \leq \text{poly}(n)$ for each k . The input has the promise that the probability distribution $D(H, b)$, given by [Eq. 2](#), approximates exactly one of $p_0(M)$ and $p_1(M)$ with error δ and precision ε . The task is to determine which one, $p_0(M)$ or $p_1(M)$, $D(H, b)$ approximates.

Restricted to One-Dimension Chain

It turns out that even if one restricts the LHER problem to the Hamiltonian on a one-dimensional qudit chain for $d = O(1)$, the problem is still **PromiseBQP**-complete (Janzing et al. 2003). The Hamiltonian is constructed along the same lines as above, with some additional treatment, following the ideas of (Aharonov et al. 2007b), to make it translationally invariant on a one-dimensional qudit chain.

3.2 Lifting the Matrix to a Power

One may find the sampling nature of the previous problems not natural, and would like to consider the average eigenvalue of a local Hamiltonian. (The authors of Wocjan and Zhang (2006) gave the credit for the question about the average eigenvalue to Yaoyun Shi.) If the distribution is, as before, induced by a vector $|b\rangle$, such that the average is $\sum_k |\langle b | \xi_k \rangle|^2 \lambda_k$, then it is not an interesting problem to study for **PromiseBQP**: Note that the value is equal to $\langle b | H | b \rangle$, and further

$$\langle b | H | b \rangle = \langle b | \sum_j H_j | b \rangle = \sum_j \langle b | H_j | b \rangle \quad (25)$$

Since each $\langle b | H_j | b \rangle$ can be easily computed even deterministically, one can obtain the exact average eigenvalue of a local Hamiltonian deterministically in polynomial time.

However, if one raises the matrix to some power, then the problem becomes **PromiseBQP**-complete again. This is the subject of this section, based on the result in Janzing and Wocjan (2007).

Definition 10 The sparse matrix powered entry (SMPE) problem is defined as follows. The input is a tuple $(A, b, m, j, \varepsilon, g)$ where

1. $A \in \mathbb{R}^{N \times N}$ is a symmetric matrix with the operator norm $\|A\| \leq b$, A has no more than $s = \text{polylog}(N)$ nonzero entries in each row, and there is an efficiently computable function f specifying for each given row the nonzero entries and their positions.
2. $m = \text{polylog}(N)$ is a positive integer, $j \in [N]$, $\varepsilon = 1/\text{polylog}(N)$ and $g \in [-b^m, b^m]$.

The input has the promise that either $(A^m)_{jj} \geq g + \varepsilon b^m$ or $(A^m)_{jj} \leq g - \varepsilon b^m$. The task is to distinguish between these two cases.

The main theorem is now given.

Theorem 3 *The problem SMPE is **PromiseBQP**-complete.*

The algorithm is very similar to the one in the last section, except that now the efficient quantum algorithm is used to simulate the process e^{-iAt} for general sparse Hamiltonians A (Berry et al. 2007). Next, an account of the proof of the **PromiseBQP**-hardness is given.

From the definition of H in [Eq. 23](#), one can see that

$$H^t = \sum_k \cos^t(2k\pi/M) |\xi_k\rangle\langle\xi_k| + \sum_k \cos^t((2k+1)\pi/M) |\eta_k\rangle\langle\eta_k| \quad (26)$$

Thus the distribution $D(H^t|\varphi_{x,0})$ is $\{\cos^t(2k\pi/M)\}$ with probability $|\alpha_{x,0}|^2/M$, and $\cos^t((2k+1)\pi/M)$ with probability $|\alpha_{x,1}|^2/M\}$, whose average is

$$\begin{aligned} \langle \varphi_{x,0} | H^t | \varphi_{x,0} \rangle &= \frac{1}{M} |\alpha_{x,0}|^2 \left(1 + \sum_{k=1}^{(M-1)/2} \cos^t(2k\pi/M) \right) \\ &\quad + \frac{1}{M} |\alpha_{x,1}|^2 \left(-1 + \sum_{k=0}^{(M-3)/2} \cos^t((2k+1)\pi/M) \right) \end{aligned} \quad (27)$$

When t is large enough, say $t = M^3$, all the cosine terms combined become negligible compared to the 1 or -1 in the summation. Thus, depending on whether $|\alpha_{x,1}|^2$ is close to 1 or 0, the average eigenvalue $\langle \varphi_{x,0} | H^t | \varphi_{x,0} \rangle$ will be close to either $-1/M$ or $1/M$, respectively. Therefore, estimating the average eigenvalue can determine whether $|\alpha_{x,1}|^2$ is close to 1 or 0, solving the starting **PromiseBQP** problem.

4 Additive Approximation of the Jones Polynomials and Tutte Polynomials

A completely different vein of research on **PromiseBQP**-completeness is the study of approximation of Jones polynomials and Tutte polynomials (Freedman 2002b, c; Aharonov et al. 2006, 2007a; Aharonov and Arad 2006; Wocjan and Yard 2008). There are at least two interesting aspects of this line compared to the problems discussed in the previous section. First, the problems look less quantum, at least by the definition. Second, the algorithms do not use quantum Fourier transform as many other quantum algorithms with exponential speedup over their classical counterparts do. What the algorithms use is the homomorphism property of a representation.

Next, this line of research is introduced in [Sect. 4.1](#) to give some background about the Jones polynomial and Tutte polynomial, mainly about their connections to physics and combinatorics. [Section 4.2](#) presents the efficient quantum algorithms for approximating Jones polynomials of trace closure of the braid group at some roots of unity. Finally a brief account is given of two subsequent (unpublished) results: the **PromiseBQP**-hardness of approximating Jones polynomials (Aharonov and Arad 2006) in [Sect. 4.3](#) and the algorithms and complexity of approximating the Tutte polynomials (Aharonov et al. 2007a) in [Sect. 4.4](#).

4.1 About Jones Polynomials and Tutte Polynomials

A short discussion about the connections of the Jones polynomial and the Tutte polynomial to other fields is given. The definitions are a bit involved and thus deferred to later sections.

In knot theory, the Jones polynomial is a knot invariant discovered by Jones (1985). Specifically, it is, for each oriented knot or link, a Laurent polynomial in the variable \sqrt{t} with integer coefficients. The Jones polynomial is an important knot invariant in low dimensional topology; it is also related to statistical physics (Wu 1992) and DNA recombination (Podtelezhnikov et al. 1999).

The connection between the Jones polynomial and quantum computing is already known. Freedman et al. (2002b, c) showed that a model of quantum computing based on topological quantum field theory and Chern–Simons theory (Freedman 1998; Freedman et al. 2002a) is equivalent to the standard quantum computation model up to a polynomial relation. Note that these results actually already imply an efficient quantum algorithm for approximating the Jones polynomial at $e^{2\pi i/5}$, though the algorithm was not explicitly given. In Aharonov et al. (2006b), Aharonov, Jones and Landau gave a simple and explicit quantum algorithm to approximate the Jones polynomial at all points of the form $e^{2\pi i/k}$, even if k grows polynomially with n . At the universality side, the result in Freedman et al. (2002) implies that approximating the Jones polynomial of the plat closure of a braid at $e^{2\pi i/k}$ is **PromiseBQP**-hard for any constant k . In Aharonov and Arad (2006), Aharonov and Arad generalized this by showing that the problem is **PromiseBQP**-hard also for asymptotically growing k , bounded by a polynomial of size of the braid. These together give a new class of **PromiseBQP**-complete problems.

The usual Tutte polynomial is a two-variable polynomial defined for graphs (or more generally for matroids). It is, essentially, the ordinary generating function for the number of edge sets of a given size and connected components. Containing information about how the graph is connected, it plays an important role in algebraic graph theory. It contains as special cases several other famous polynomials, such as the chromatic polynomial, the flow polynomial and the reliability polynomial, and it is equivalent to Whitney’s rank polynomial, Tutte’s own dichromatic polynomial, and the Fortuin–Kasteleyn’s random-cluster model under simple transformations. It is the most general graph invariant defined by a deletion–contraction recurrence. See textbooks (Bollobás 1998; Biggs 1993; Godsil and Royle 2001) for more detailed treatment.

The result in Aharonov et al. (2007a) focuses on the multivariate Tutte polynomial, which generalizes the two-variable case by assigning to each edge a different variable; see survey (Sokal 2005). This generalized version has rich connections to the Potts model in statistical physics (Wu 1982).

On the complexity side, the exact evaluation of the two-variable Tutte polynomial for planar graphs is **#P**-hard. Both positive and negative results are known for the multiplicative approximation FPRAS (fully polynomial randomized approximation scheme); see Aharonov et al. (2007a) for more details.

4.2 Additive Approximation of the Jones Polynomials

4.2.1 Definitions and the Main Theorems

This section presents an efficient quantum algorithm to additively approximate the Jones polynomial of trace and plat closures of braids at roots of unity. We begin by defining the Jones polynomial. A set of circles embedded in \mathbb{R}^3 is called a *link* L . If each circle has a direction, then we say that the link is oriented. A link invariant is a function on links that is

invariant under isotopy of links. That is, the function remains unchanged when the links are distorted (without being broken). An important link invariant is the Jones polynomial $V_L(t)$, which is a Laurent polynomial in \sqrt{t} over \mathbb{Z} .

Definition 11 (Writhe) The *writhe* $w(L)$ of an oriented link L is the number of positive crossings (I_+ in Fig. 2) minus the number of negative crossings (I_- in Fig. 2) of L .

Definition 12 (Bracket Kauffman polynomial) The *bracket Kauffman polynomial* $\langle L \rangle$ of an oriented link L is

$$\langle L \rangle = \sum_{\sigma} \sigma(L) \quad (28)$$

where the summation is over all the crossing-breaking states. A crossing-breaking state σ is an n -bit string indicating how to break all crossings by changing each crossing in one of the two ways as shown in Fig. 3. Define $\sigma(L) = A^{\sigma_+ - \sigma_-} d^{|\sigma|-1}$, where σ_+ and σ_- are the numbers of choices of the first and the second case respectively, when breaking all crossings in L , $|\sigma|$ is the number of closed loops in the resulting link, and $d = -A^2 - A^{-2}$.

Definition 13 (Jones polynomial) The *Jones polynomial* of an oriented link L is

$$V_L(t) = V_L(A^{-4}) = (-A)^{3w(L)} \langle L \rangle \quad (29)$$

where $w(L)$ is the writhe of L , and $\langle L \rangle$ is the bracket Kauffman polynomial of L .

There are different ways of forming links. In particular, one can obtain a link from a *braid*. First, an intuitive geometrical definition of a braid is given here. Consider two horizontal bars each with n pegs, one on top of the other; see Fig. 4. The pegs on the upper bar are called the *upper pegs*, and we index them from left to right using $i = 1, \dots, n$; similarly for the *lower pegs*. Each strand goes from an upper peg, always downwards, to a lower peg, so that finally each peg has exactly one strand attached to it. The n -strand braid group B_n is the set of n -strand braids

Fig. 2

Positive and negative crossings.

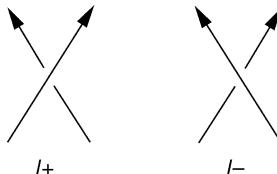


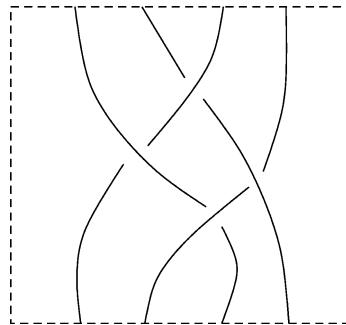
Fig. 3

Two ways to break a crossing.



Fig. 4

A braid with 4 strands.



with the multiplication defined by putting the first braid on top of the second. To be more precise, for braids b_1 and b_2 , the new braid $b_1 \cdot b_2$ is obtained by putting b_1 on top of b_2 , and removing the pegs of the lower bar of b_1 and the upper bar of b_2 with strands on the same peg connected.

The braid group B_n has an algebraic presentation by generators and relations.

Definition 14 Let B_n be the group with generators $\{1, \sigma_1, \dots, \sigma_{n-1}\}$ with relations

1. $\sigma_i \sigma_j = \sigma_j \sigma_i$ for all $|i - j| \geq 2$
2. $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$

The correspondence between the geometrical pictures and the algebraic presentation is as follows: 1 corresponds to the braid where all strands j go from upper peg j to lower peg j without crossing any other strand; σ_i corresponds to the same braid except that strand i and $i + 1$ have one crossing, with strand i in front. The above two relations are easily verified by this correspondence.

Braids are not links since the strands are not closed circles, but there are different ways of forming links from braids. Two simple ones are the *trace closure* and the *plat closure*. The trace closure B^{tr} of a braid B connects upper peg i and lower peg i without crossing any strand in B . The plat closure B^{pl} of a $2n$ -strand braid B connects its upper peg $2i - 1$ and upper peg $2i$ for all $i \in [n]$; similarly for the lower pegs. See [Fig. 5](#).

The main theorem of this section says that there are quantum algorithms additively approximating the Jones polynomials of B^{tr} and B^{pl} at $A^{-4} = e^{2\pi i/k}$ within some precision.

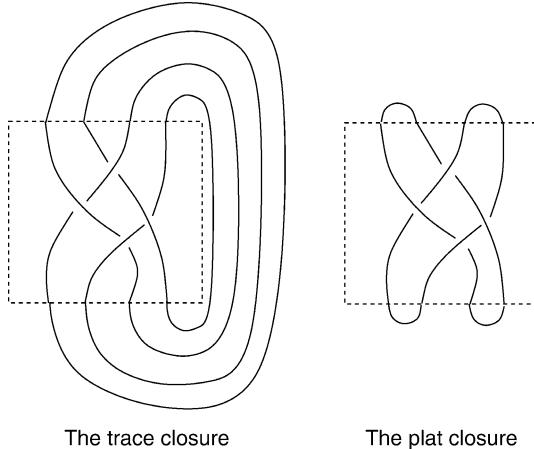
Theorem 4 There is a quantum algorithm, which on input n -strand m -crossing braid B outputs a complex number c satisfying $|c - V_{B^{\text{tr}}}(e^{2\pi i/k})| < \varepsilon(-A^2 - A^{-2})^{n-1}$ with probability $1 - 1/\exp(n, m, k)$ for some $\varepsilon = 1/\text{poly}(n, m, k)$.

Theorem 5 There is a quantum algorithm, which on input n -strand m -crossing braid B outputs a complex number c satisfying $|c - V_{B^{\text{pl}}}(e^{2\pi i/k})| < \varepsilon(-A^2 - A^{-2})^{3n/2}/N$ with probability $1 - 1/\exp(n, m, k)$ for some $\varepsilon = 1/\text{poly}(n, m, k)$. Here

$$N = \sum_{l=1}^{k-1} \sin(\pi l/k) |P_{n,k,l}| \quad (30)$$

Fig. 5

The trace and plat closures of a 4-braid.



where

$$P_{n,k,l} = \left\{ x \in \{0,1\}^n : 1 \leq 2 \sum_{i=1}^j x_i - j \leq k-1, \forall j \in [n]; 2 \sum_{i=1}^n x_i - n = l \right\} \quad (31)$$

4.2.2 The Approximation Algorithms

Main Idea and Overview

The main idea of designing the efficient quantum is to design a representation of the braid group and use the homomorphism property of the representation to decompose the computation into ones for each crossing in the braid, which can be done efficiently. This idea was also used to design classical algorithms. Let us recall an example to illustrate the idea. Consider the word problem of the free group generated by $\{a, b\}$. That is, given a sequence of elements from $\{a, b\}$, decide whether the multiplication (in that order) is the identity element of the group. There is a simple linear time algorithm using the stack (or pushdown machine, in other words), but the algorithm also uses linear space. Interestingly, Lipton and Zalcstein showed in 1977 that it can be solved in log space in a very cute manner. Basically, they used the following theorem proved by Ivan Sanov in 1947.

Theorem 6 *There are two integer matrices A, B so that the mapping $a \rightarrow A$ and $b \rightarrow B$ is a faithful representation of the free group on $\{a, b\}$.*

That means that we can replace the word problem by this question: Does a sequence of matrices over $\{A, B, A^{-1}, B^{-1}\}$ equal the identity matrix I ? Since multiplication of 2×2 matrices does not need the stack anymore, this basically gives a log space algorithm. (There is actually one more difficulty: during the computation there may be entries that are too large and cannot be stored in log-space. The solution is to do the multiplication mod p for all

p using $O(\log(n))$ bits. Then the correctness is guaranteed by the Chinese Remainder Theorem and the following well-known fact: $\prod_{p \leq t} p \geq c^t$, where the product is over primes and $c > 1$.)

The quantum algorithm to approximate the Jones polynomial also uses representation of the braid group. A high level picture is:

$$B_n \xrightarrow{\rho_A} \text{TL}_n(d) \xrightarrow{\Phi} \mathbb{C}^{r \times r} \quad (32)$$

Here, $\text{TL}_n(d)$ is the *Temperley–Lieb algebra*, which will be defined later. The mapping ρ_A is a homomorphism from B_n to $\text{TL}_n(d)$. The mapping Φ is the representation of $\text{TL}_n(d)$ by $r \times r$ unitary matrices for some r .

The analysis can be divided into three steps:

1. For the link B^{tr} , the Jones polynomial is

$$V_{B^{\text{tr}}}(A^{-4}) = (-A)^{3w(B^{\text{tr}})} d^{n-1} \text{tr}(\rho_A(B)) \quad (33)$$

The function $\text{tr}(\cdot)$ is some mapping from $\text{TL}_n(d)$ to \mathbb{C} to be defined later. By this equality, it is enough to approximate $\text{tr}(\rho_A(B))$.

2. One can calculate $\text{tr}(\rho_A(B))$ by

$$\text{tr}(\rho_A(B)) = \text{Tr}_n(\Phi \circ \rho_A(B)) \quad (34)$$

where \circ is the standard function composition, and Tr_n is defined as follows. For a matrix $W \in \mathbb{C}^{r \times r}$,

$$\text{Tr}_n(\Phi \circ \rho_A(B)) = \frac{1}{N} \sum_{l=1}^{k-1} \sin(\pi l/k) \text{Tr}(\Phi \circ \rho_A(B)|_l) \quad (35)$$

where Tr is the standard matrix trace, N is as defined in [Theorem 5](#), $W|_l$ is W restricted on some subspace. If an orthonormal basis of $\Phi \circ \rho_A(B)|_l$ is $\{|p\rangle\}$, then one can further the calculation by

$$\text{Tr}(\Phi \circ \rho_A(B)|_l) = \sum_p \langle p | \Phi \circ \rho_A(B) | p \rangle \quad (36)$$

It turns out that for each l , one can efficiently sample p almost uniformly at random (even on a classical computer). One also needs the following well-known fact in quantum computing.

Fact Given a quantum state $|\psi\rangle$ and a quantum circuit Q , one can generate a random variable $b \in \mathbb{C}$ with $|b| \leq 1$ and $E[b] = \langle \psi | Q | \psi \rangle$.

By the above analysis, one can achieve the approximation as long as one can implement $\Phi \circ \rho_A(B)$ efficiently on a quantum computer.

3. This is where the homomorphism comes into the picture. Each braid B can be decomposed into the product of a sequence of basic braids σ_i . The A and Φ are carefully designed such that $\Phi \circ \rho_A$ is a unitary representation. Thus $\Phi \circ \rho_A(B) = \Phi \circ \rho_A(\prod_i \sigma_i) = \prod_i \Phi \circ \rho_A(\sigma_i)$. Therefore, it is enough to implement $\Phi \circ \rho_A(\sigma_i)$ for each basis σ_i , and it turns out that this is not hard to do. This finishes the overview of the whole idea of designing the quantum algorithm.

More Details

Next, more details to carry out the above strategy are given. To start with, the Temperley–Lieb algebra is defined.

Definition 15 (Temperley–Lieb algebra) The *Temperley–Lieb algebra* $\text{TL}_n(d)$ is the algebra generated by $\{1, E_1, \dots, E_{n-1}\}$ with relations

1. $E_i E_j = E_j E_i$ $|i - j| \geq 2$
2. $E_i E_{i+1} E_i = E_i E_{i-1} E_i = E_i$
3. $E_i^2 = d E_i$

As the braid group B_n , the Temperley–Lieb algebra also has a nice pictorial description called Kauffman diagrams. A Kauffman n -diagram has an upper bar with n top pegs and a lower bar with n lower pegs as in a braid, but it does not have crossings or loops. A typical Kauffman 4-diagram is as shown in [Fig. 6](#).

The multiplication of two Kauffman n -diagrams $K_1 \cdot K_2$ is obtained similarly for braids: that is, putting K_1 on top of K_2 . Note that $K_1 \cdot K_2$ now may contain loops. If there are m loops, then the loops are removed and we add a factor of d^m in the resulting diagram. See [Fig. 7](#). The Kauffman diagrams over \mathbb{C} form an algebra. The correspondence between the pictures and the original definition of Temperley–Lieb algebra is shown in [Fig. 8](#).

The first mapping ρ_A is now defined in [Eq. 32](#).

Fig. 6

A typical Kauffman 4-diagram.

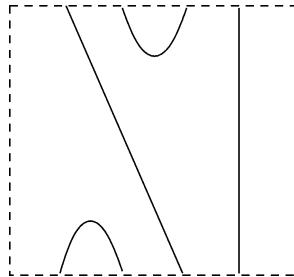


Fig. 7

Product of two Kauffman 4-diagrams.

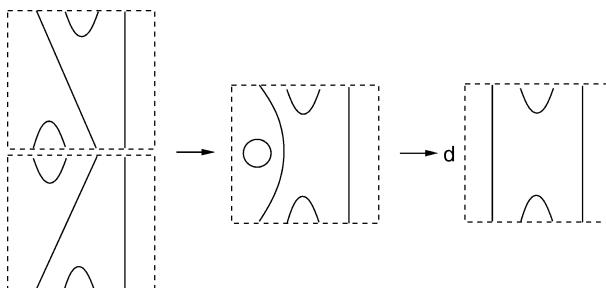
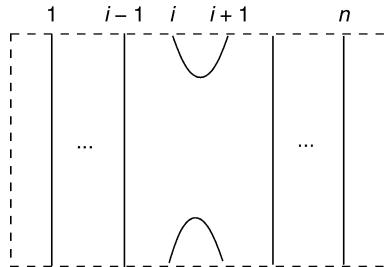


Fig. 8

Correspondence between the pictures and the Temperley–Lieb algebra.

**Fig. 9**

The graph L_{k-1} .



Definition 16 $\rho_A: B_n \mapsto \text{TL}_n(d)$ is defined by $\rho_A(\sigma_i) = AE_i + A^{-1}1$.

The following facts are easily verified.

Fact 1 The mapping ρ_A respects the two relations in the definition of the braid group.

Fact 2 If $|A|=1$ and $\Phi(E_i)$ is Hermitian for each i , then the map $\Phi \circ \rho_A$ is a unitary representation of B_n .

Now the trace function in Step 1 and 2 is defined. It is easier to define the Kauffman diagrams.

Definition 17 The *Markov trace* $tr: \text{TL}_n(d) \rightarrow \mathbb{C}$ on a Kauffman n -diagram K connects the upper n pegs to the lower n pegs of K with nonintersecting curves, as in the trace closure case. Then $tr(K) = d^{a-n}$, where a is the number of loops of the resulting diagram. Extend tr to all of $\text{TL}_n(d)$ by linearity.

For this definition, it is not hard to check that the equality in Step 1 holds.

Lemma 1 For any braid B , one has

$$V_{B^n}(A^{-4}) = (-A)^{3w(B^{\text{tr}})} d^{n-1} tr(\rho_A(B)) \quad (37)$$

Now the second mapping Φ is defined, for which the path model representation of $\text{TL}_n(d)$ will be needed. Consider a graph L_{k-1} of a line with $k-1$ points (connected by $k-2$ edges); see **Fig. 9**. Let $Q_{n,k}$ be the set of all paths of length n on L_{k-1} . That is, $Q_{n,k}$ can be identified with $\{q \in [k-1]^{n+1}: |q_i - q_{i+1}| = 1, \forall i \in [n-1]\}$. Let $r = |Q_{n,k}|$. That is, the size of $Q_{n,k}$ is the dimension of matrix $\Phi(K)$; thus one can index the row/column of the matrix by a path in $Q_{n,k}$.

To specify Φ easily, let one use the bit string representation of a path. Each path is specified by a string $p \in \{0, 1\}^n$ such that $p_i = 1$ if the i -th step goes right and $p_i = 0$ if the i -th step goes left. To guarantee that the path is always within the graph, one requires $1 \leq z_j \leq k - 1$ for any $j \in [n]$, where $z_j = 2 \sum_{i=1}^j p_i - j$ is the location of the path after first j steps. Let $H_{n,k}$ be the span of all these paths, each treated as a basis state. Denote $p_{<i} = p_1 \dots p_{i-1}$ and $p_{\geq i} = p_i \dots p_n$.

To define Φ , it is enough to define its action on each E_i , done as follows, where $\lambda_j = \sin(j\pi/k)$.

$$\Phi(E_i)|p_{<i}00p_{\geq i+2}\rangle = 0 \quad (38)$$

$$\Phi(E_i)|p_{<i}01p_{\geq i+2}\rangle = \frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p_{<i}01p_{\geq i+2}\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p_{<i}10p_{\geq i+2}\rangle \quad (39)$$

$$\Phi(E_i)|p_{<i}10p_{\geq i+2}\rangle = \frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p_{<i}10p_{\geq i+2}\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p_{<i}01p_{\geq i+2}\rangle \quad (40)$$

$$\Phi(E_i)|p_{<i}11p_{\geq i+2}\rangle = 0 \quad (41)$$

Theorem 7 If $d = 2 \cos(\pi/k)$, then $\Phi \circ \rho_A$ is a unitary representation of B_n in r -dimensional vector space.

Now define the subspace $H_{n,k,l}$ of $H_{n,k}$ by $H_{n,k,l} = \text{span}\{|p\rangle : z_n(p) = l\}$. Define Tr_n as in [Eq. 35](#). We are ready to show the algorithm after some final comments: The Fact in Step 2 is covered by the standard Hadamard Test on both the real and the imaginary parts; see Aharonov et al. (2006b) for details. For each basis σ_i , it is not hard to check that $\Phi \circ \rho_A(\sigma_i)$ can be implemented in polynomial time on a quantum computer. The algorithm for approximating the Jones polynomial on B^{tr} is as follows. The averaging over polynomial number of samples at the last step can be shown to give enough approximation by the standard Chernoff bound.

Algorithm 1 Approximate Jones trace closure

1. Repeat for $j = 1$ to $t = \text{poly}(n, m, k)$:
 - a. Classically, pick a random path $p \in P_{n,k}$ with probability $\Pr(p) \propto \sin(\pi l/k)$, where l is the index of the site at which p ends.
 - b. Use Hadamard Test to output a random variable x_j with $E[x_j] = \text{Re}\langle p | Q(B) | p \rangle$.
2. Use Hadamard Test to output a random variable y_j with $E[y_j] = \text{Im}\langle p | Q(B) | p \rangle$.
3. Let $r = \frac{1}{t} \sum_j (x_j + iy_j)$. Output $(-A)^{3w(\delta^{\text{tr}})} d^{n-1} r$.

The case of plat closure can be reduced to the trace closure case by the following observation. The plat closure of a braid B is isotopic to the trace closure of C , obtained by putting B on top of $n/2$ capcups, where a capcup is a cup on top of a cap. The algorithm is largely the same as the one for trace closure, except that one needs an observation that the state $|\alpha\rangle = |1, 0, 1, 0, \dots, 1, 0\rangle$ can be used to connect the Jones polynomial of the plat closure and the trace function. To be more precise, one has

$$\text{Fact 3 } \langle \alpha | \Phi \circ \rho_A | \alpha \rangle = \frac{N}{\sin(\pi/k)d^{n/2}} Tr_n(\Phi \circ \rho_A(C)).$$

By this, it is enough to estimate $\langle \alpha | \Phi \circ \rho_A | \alpha \rangle$, which can be done by Hadamard test again like in the trace closure case. The algorithm is as follows.

Algorithm 2 Approximate Jones plat closure

1. Repeat for $j = 1$ to $t = \text{poly}(n, m, k)$:
 - a. Generate the state $|\alpha\rangle = |1, 0, 1, 0, \dots, 1, 0\rangle$.
 - b. Use Hadamard Test to output a random variable x_j with $E[x_j] = \text{Re}(\langle \alpha | Q(B) | \alpha \rangle)$.
2. Use Hadamard Test to output a random variable y_j with $E[y_j] = \text{Im}(\langle \alpha | Q(B) | \alpha \rangle)$.
3. Let $r = \frac{1}{t} \sum_j (x_j + iy_j)$. Output $(-A)^{3w(B^t)} d^{3n/2-1} r \sin(\pi/k)/N$.

4.3 The PromiseBQP-Hardness of Approximating Jones Polynomials

A brief discussion of the idea in Aharonov and Arad (2006) of simulating a quantum circuit by an oracle U approximating the Jones polynomials will be presented here. First, based on a simple procedure similar to the one in [Sect. 3.1.2](#), it is enough to approximate $\langle 0 | U | 0 \rangle$ for any polynomial size quantum circuit U . So we want to efficiently construct a braid B such that B has polynomially many crossings and $\langle \alpha | \Phi \circ \rho_A(B) | \alpha \rangle \approx \langle 0 | U | 0 \rangle$. (Here the \approx sign means approximation to any desired accuracy.)

Note that in this approach, the working space has to be encoded by paths on graph L_{k-1} . Thus we need to encode the space for the original circuit U by the path space. A simple encoding uses the following four-step paths:

$$|0\rangle \rightarrow |1010\rangle, \quad |1\rangle \rightarrow |1100\rangle \tag{42}$$

Suppose U is decomposed as $U = U_m \dots U_1$, where each U_i is an elementary gate acting on at most two qubits. It can be assumed without loss of generality, that each U_i operates on adjacent qubits. Using the path encoding, U_i operates on eight qubits. Note that the path is not arbitrary in $Q_{4n, k}$ since it always returns to the original point every four steps. Denote by S the subspace spanned by all these paths; then it is sufficient if one can efficiently find $B_i \in B_{4n}$ such that $\Phi \circ \rho_A(B_i) \approx U_i$ on the subspace S to the polynomially small accuracy. It turns out that it is doable, as the following density theorem shows for B_8 .

Theorem 8 Suppose \tilde{U} is an encoded two-qubit quantum gate, then for any $\delta > 0$ and $k \geq 11$, one can find a braid $B \in B_8$ with $\text{poly}(k, 1/\delta)$ generators of B_8 such that $\|(\Phi \circ \rho_A(B))|p\rangle - (\tilde{U})|p\rangle\| \leq \delta$ for all $|p\rangle \in H_{n,k,1}$.

The proof of this theorem is the core technical part needed to show the hardness, but it is a bit far away from the **PromiseBQP**-completeness notion that is being discussed. The readers are referred to Aharonov and Arad (2006) for the details.

4.4 Additive Approximation of the Tutte Polynomials

The results for the Jones polynomials in Aharonov et al. (2006b) and Aharonov and Arad (2006) are generalized to the Tutte polynomials in Aharonov et al. (2007a). The multivariate Tutte polynomial is defined as follows.

Definition 18 (Tutte polynomial) Given an undirected graph $G = (V, E)$ with a weight function on edges $\mathbf{v} = \{\nu_e : e \in E\}$, the Tutte polynomial is

$$Z_G(q, \mathbf{v}) = \sum_{A \subseteq E} q^{k(A)} \prod_{e \in A} \nu_e \quad (43)$$

where $k(A)$ is the number of connected components in the subgraph (V, A) .

When all ν_e 's are equal to the same v , then the polynomial becomes

$$Z_G(q, v) = \sum_{A \subseteq E} q^{k(A)} v^{|A|},$$

which is essentially the same as the standard Tutte polynomial

$$T_G(x, y) = \sum_{A \subseteq E} (x - 1)^{k(A) - k(E)} (y - 1)^{|A| + k(A) - |V|} \quad (44)$$

under the change of variables

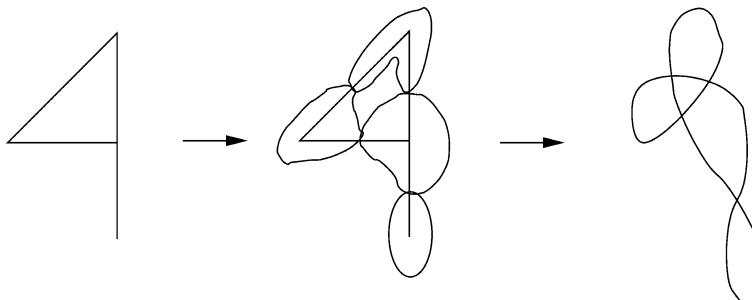
$$x = 1 + q/v, \quad y = 1 + v \quad (45)$$

$$q = (x - 1)(y - 1), \quad v = y - 1 \quad (46)$$

The main results in Aharonov et al. (2007a) are as follows. First, there is an efficient algorithm to additively approximate the multivariate Tutte polynomial of a given weighted planar graph. Second, there exists a range of complex weights and complex values of q such that the additive approximation of the multivariate Tutte polynomial at those points to within some scale is **PromiseBQP**-hard. The approximation windows in the above two results do not match that nicely in the non-unitary case, in which they modified the definition to get the **PromiseBQP**-complete problems.

The rest of this section will mainly illustrate some ideas of the algorithm. The algorithm actually takes an approach similar to that the Jones polynomial, with generalizations of various objects. First, we start at a planar graph rather than a braid. Like using closures of braids, we also change the planar graph to a knot-like object. Here the *medial graph* is used. For a planar graph G , the medial graph L_G is obtained in the following way. First encircle the facets of G with lines, and then cross the lines that surround each edge by putting a vertex in the middle of the edge. See  Fig. 10 for an illustration. Note that the resulting graph L_G is 4-regular.

 **Fig. 10**
Medial graph of a planar graph.



The regions of the medial graph can be {black,white}-colored so that no two adjacent regions have the same color. This coloring is unique up to an overall flip; let the outer region be fixed to be white.

For each crossing, there are two ways to break it, depending on whether one connects the two black or the two white regions. Let σ be an m -bit string indicating the crossing-breaking choices, where $m = |E| = |\sigma|$ is the number of the crossings. Denote by $\text{Black}(\sigma)$ the set of edges, the crossings corresponding to which are broken by σ with two black regions connected. The Kauffman bracket of the medial graph L_G can now be defined by

$$\langle L_G \rangle = \sum_{\sigma} d^{|\sigma|} \prod_{e \in \text{Black}(\sigma)} u_e \quad (47)$$

where the new variables u_e and the old ones v_e are related by $v_e = dv_e$. The following fact connects the Kauffman bracket of L_G and the Tutte polynomial of G .

Fact 4 $\langle L_G \rangle(d, \mathbf{u}) = d^{-n} Z_G(d^2, d\mathbf{u})$.

Thus it is enough to calculate the Kauffman bracket of the medial graph. To this end, one needs a generalized Temperley–Lieb algebra $\text{GTL}(d)$, where one allows an arbitrary number of strands and allows cases with different numbers of upper and lower pegs in one diagram. A diagram does not “change” by adding some trivial strands; that is, those going from an upper peg directly to a lower peg without crossing any other strand. With these relaxations, the product can be defined similarly to the Jones polynomial case (by putting one on top of the other) in a consistent manner.

Using this bridge, generalized versions of the mapping Φ and the path model representation of $\text{GTL}(d)$ can be defined. But now the representative does not need to be unitary. Again the homomorphism property of representation reduces the task to approximating each basic tangle diagram.

There is one catch, however: The final scale of the approximation window depends on the decomposition of L_G into some simple structures called *basis tangles*. The optimal decomposition is not known to be easy to obtain. This is also a drawback of the algorithm: the approximation window contains a quantity which is complicated and not directly about the graph itself (but about the layout of it on the two-dimensional plane).

5 Concluding Remarks

5.1 Some Other PromiseBQP-Completeness Related Problems

In Wocjan and Zhang (2006), it is shown that the problem of local unitary phase sampling is also **PromiseBQP**-complete, as is a problem called local unitary average eigenvalue. Even earlier, Knill and Laflamme found that the quadratically signed weight enumerators problem is also **PromiseBQP**-complete (Knill and Laflamme 2001).

In the track of the Jones polynomial and Tutte polynomial, Wocjan and Yard gave new quantum algorithms for approximating HOMFLYPT two-variable polynomials of trace closures of braids (Wocjan and Yard 2008). They also gave algorithms for approximating the Jones polynomial of a general class of closures of braids.

There are some equivalent models of efficient quantum computation, such as adiabatic quantum computation (Aharonov et al. 2004) and one-way quantum computation (Browne and Briegel 2006). One can also view the canonical problems in those models as complete problems for the standard quantum Turing machine or quantum circuit model.

One can interpret the problem of SMPE in Sect. 3.2 as approximating the total weight of cycles of length t passing a vertex j on a weighted graph. The work (Janzing and Wocjan 2007) can actually show the **PromiseBQP**-completeness even for Hamiltonians with $\{1, -1, 0\}$ entries. However, allowing the weight to be -1 makes weights on cycles cancel, giving the problem a quantum flavor. Childs recently studied the problem with the weight to be chosen only from $\{0, 1\}$ (Childs 2009).

5.2 Future Directions

There are two issues in the state of the art of **PromiseBQP**-completeness. One is that all the known **PromiseBQP**-complete problems are not “natural” enough. In some sense, they are all describing the same class using different languages, though the difficulty of translation may be at different levels. One can say that all completeness results have this feature, but the key reason why **NP**-completeness is so important is that there are so many natural and seemingly unrelated combinatorial problems in theoretical computer science, discrete mathematics, and various other branches of mathematics and science. But not many natural **PromiseBQP**-complete problems are known so far.

Another direction is to try to use the **PromiseBQP**-complete problem to study the classes **PromiseBQP** and **BQP**. For example, one of the main open questions in quantum complexity theory is whether **BQP** is in **PH**, the polynomial hierarchy. The current best known upper bound of **BQP** is **AWPP**, a not-so-natural subclass of **PP**. (See the textbooks (Arora and Barak 2009; Goldreich 2008; and Papadimitriou 1994) and the “complexity zoo” (currently at http://qwiki.stanford.edu/wiki/Complexity_Zoo) for definitions of these complexity classes.) Could the known **PromiseBQP**-complete problems shed any light on the open problem?

References

- Aharonov D, Arad I (2006) The **BQP**-hardness of approximating the Jones polynomial. quant-ph/0605181
- Aharonov D, Ta-Shma A (2003) Adiabatic quantum state generation and statistical zero knowledge. In: Proceedings of 35th annual ACM symposium on theory of computing (STOC). ACM, New York, pp 20–29
- Aharonov D, van Dam W, Kempe J, Landau Z, Lloyd S, Regev O (2004) Adiabatic quantum computation is equivalent to standard quantum computation. In: Proceedings of the 45th annual IEEE symposium on foundations of computer science (FOCS). IEEE Computer Society, Washington, DC, pp 42–51
- Aharonov D, Jones V, Landau Z (2006) A polynomial quantum algorithm for approximating the Jones polynomial. In: Proceedings of the 38th annual ACM symposium on theory of computing (STOC). ACM, New York, pp 427–436
- Aharonov D, Arad I, Eban E, Landau Z (2007a) Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane. arXiv:quant-ph/0702008
- Aharonov D, Gottesman D, Irani S, Kempe J (2007b) The power of quantum systems on a line. In: Proceedings of the 48th annual IEEE symposium on foundations of computer science (FOCS). IEEE Computer Society, Washington, DC, pp 373–383
- Ambainis A, Childs AM, Reichardt BW, Spalek R, Zhang S (2007) Any and-or formula of size n can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer. In: Proceedings of the 48th annual IEEE symposium on foundations of computer science

- (FOCS). IEEE Computer Society, Washington, DC, pp 363–372
- Arora S, Barak B (2009) Computational complexity: a modern approach. Cambridge University Press, Cambridge, UK
- Bernstein E, Vazirani U (1997) Quantum complexity theory. SIAM J Comput 26(5):1411–1473
- Berry D, Ahokas G, Cleve R, Sanders B (2007) Efficient quantum algorithms for simulating sparse Hamiltonians. Commun Math Phys 270(2):359–371
- Biggs N (1993) Algebraic graph theory, 2nd edn. Cambridge University Press, New York
- Bollobás B (1998) Modern graph theory. Springer, New York
- Browne D, Briegel H (2006) One-way quantum computation – a tutorial introduction. arXiv:quant-ph/0603226
- Chernoff P (1968) Note on product formulas for operator semigroups. J Funct Anal 2:238–242
- Childs A (2009) Universal computation by quantum walk. Phys Rev Lett 102:180501
- Childs A, van Dam W (2010) Quantum algorithms for algebraic problems. Rev Mod Phys arXiv:0812.0380, 82:1–52
- Childs A, Cleve R, Deotto E, Farhi E, Gutmann S, Spielman D (2003) Exponential algorithmic speedup by a quantum walk. Proceedings of the 35th annual ACM symposium on theory of computing (STOC). ACM Press, New York, pp 59–68
- Cook S (1971) The complexity of theorem-proving procedures. In: Proceedings of 3rd annual ACM symposium on theory of computing (STOC). ACM Press, New York, pp 151–158
- Freedman M (1998) P/NP, and the quantum field computer. Proc Natl Acad Sci 95(1):98–101
- Freedman M, Kitaev A, Larsen M, Wang Z (2002a) Topological quantum computation. Bull Amer Math Soc 40(1):31–38
- Freedman M, Kitaev A, Wang Z (2002b) Simulation of topological field theories by quantum computers. Commun Math Phys 227(3):587–603
- Freedman M, Larsen M, Wang Z (2002c) A modular functor which is universal for quantum computation. Commun Math Phys 227:605–622
- Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman, New York
- Godsil C, Royle G (2001) Algebraic graph theory. Springer, New York
- Goldreich O (2005) On promise problems (a survey in memory of Shimon Even [1935–2004]). Electronic colloquium on computational complexity (ECCC). TR05–018
- Goldreich O (2008) Computational complexity: a conceptual perspective. Cambridge University Press, Cambridge, UK
- Goldreich O, Micali S, Wigderson A (1991) Proofs that yield nothing but their validity. J ACM 38(3):690–728
- Hallgren S (2007) Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. J ACM 54(1):1–19
- Janzing D, Wocjan P (2007) A simple PromiseBQP-complete matrix problem. Theor Comput 3(1):61–79
- Janzing D, Wocjan P, Zhang S (2008) Measuring energy of basis states in translationally invariant nearest-neighbor interactions in qudit chains is universal for quantum computing. New J Phys 10:093004
- Jones V (1985) A polynomial invariant for knots via von Neumann algebras. Bull Amer Math Soc 12(1): 103–111
- Karp R (1972) Reducibility among combinatorial problems. In: Thatcher JW, Miller RE (eds) Complexity of computer computations. Plenum Press, New York
- Kempe J, Kitaev A, Regev O (2006) The complexity of the local Hamiltonian problem. SIAM J Comput 35(5): 1070–1097
- Kitaev A (1995) Quantum measurements and the Abelian stabilizer problem. arXiv:quant-ph/9511026
- Kitaev A, Shen A, Vyalyi M (2002) Classical and quantum computation. American Mathematical Society, Providence, RI
- Knill E, Laflamme R (2001) Quantum computing and quadratically signed weight enumerators. Infor Process Lett 79(4):173–179
- Kuperberg G (2005) A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM J Comput 35(1):170–188
- Levin L (1973) Universal search problems (in Russian). Problemy Peredachi Informatsii 9(3):265–266
- Lipton R, Zalcstein Y (1977) Word problems solvable in logspace. J ACM 24(3):522–526
- Lomont C (2004) The hidden subgroup problem – review and open problems. arXiv:quant-ph/0411037
- Magniez F, Nayak A, Roland J, Santha M (2007) Search via quantum walk. In: Proceedings of the 39th annual ACM symposium on theory of computing (STOC). ACM Press, New York, pp 575–584
- Nielsen M, Chuang I (2000) Quantum computation and quantum information. Cambridge University Press, Cambridge, UK
- Papadimitriou C (1994) Computational complexity. Addison-Wesley, Reading
- Papadimitriou C (1997) NP-completeness: a retrospective. In: Proceedings of the 24th international colloquium on automata, languages and programming (ICALP), Lecture notes in computer science, vol. 1256. Springer, Berlin, pp 2–6
- Podtelezhnikov A, Cozzarelli N, Vologodskii A (1999) Equilibrium distributions of topological states in

- circular DNA: interplay of supercoiling and knotting. *Proc Natl Acad Sci USA* 96(23):12974–12979
- Reichardt B, Spalek R (2008) Span-program-based quantum algorithm for evaluating formulas. In: Proceedings of 40th annual ACM symposium on theory of computing (STOC). ACM Press, New York, pp 103–112
- Shor P (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput* 26:1484–1509
- Shor P (2004) Progress in quantum algorithms. *Quant Inform Process* 3:5–13
- Sokal A (2005) The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: Webb BS (ed) Surveys in combinatorics. Cambridge University Press, Cambridge, UK, pp 173–226
- Trotter H (1959) On the product of semigroups of operators. *Proc Amer Math Soc* 10:545–551
- Wocjan P, Yard J (2008) The Jones polynomial: quantum algorithms and applications in quantum complexity theory. *Quant Inform Comput* 8(1&2): 147–180
- Wocjan P, Zhang S (2006) Several natural BQP-complete problems. arXiv:quant-ph/0606179
- Wu FY (1982) The Potts model. *Rev Mod Phys* 54: 235–268
- Wu FY (1992) Knot theory and statistical mechanics. *Rev Mod Phys* 64:1099–1131
- Yao A (1993) Quantum circuit complexity. In: Proceedings of the 34th annual symposium on foundations of computer science (FOCS). IEEE Computer Society, Washington, DC, pp 352–361

Section VI

Broader Perspective – Nature-Inspired Algorithms

David W. Corne

47 An Introduction to Artificial Immune Systems

Mark Read¹ · Paul S. Andrews² · Jon Timmis³

¹Department of Computer Science, University of York, UK
markread@cs.york.ac.uk

²Department of Computer Science, University of York, UK
psa@cs.york.ac.uk

³Department of Computer Science and Department of Electronics,
University of York, UK
jtimmis@cs.york.ac.uk

1	<i>Introduction</i>	1576
2	<i>The Immune System</i>	1577
3	<i>Engineering Artificial Immune Systems</i>	1578
4	<i>Reflections and Projections</i>	1586
5	<i>Summary</i>	1592

Abstract

The field of artificial immune systems (AIS) comprises two threads of research: the employment of mathematical and computational techniques in the modeling of immunology, and the incorporation of immune system metaphors in the development of engineering solutions. The former permits the integration of immunological data and sub-models into a coherent whole, which can be of value to immunologists in the facilitation of immunological understanding, hypothesis testing, and the direction of future research. The latter attempts to harness the perceived properties of the immune system in the solving of engineering problems. This chapter concentrates on the latter: the development and application of immune inspiration to engineering solutions.

1 Introduction

Artificial Immune Systems (AIS) is a branch of biologically inspired computation focusing on many aspects of immune systems. AIS development can be seen as having two target domains: the provision of solutions to engineering problems through the adoption of immune system inspired concepts; and the provision of models and simulations with which to study immune system theories.

The motivation for building immune-inspired solutions to engineering problems arises from the identification of properties within the immune system that are attractive from an engineering perspective. These include (de Castro and Timmis 2002a): the self-organization of huge numbers of immune cells; the distributed operation of the immune system throughout the body; pattern recognition and anomaly detection to enable the immune system to recognize pathogens; and optimization and memory to improve and remember immune responses. AIS take inspiration from these properties and associated immune processes, and have been defined as:

- ▶ “adaptive systems, inspired by theoretical immunology and observed immune function, principles and models, which are applied to problem solving.” (de Castro and Timmis 2002a)

The field of AIS also encompasses modeling and simulation techniques to understand the immune system in general (see Timmis et al. (2008a) for a review), however, this chapter focuses on immune-inspired systems for engineering problems.

This chapter is not intended as an extensive review chapter, but its purpose is to present a general introduction to the area and provide discussion on the major research issues relating to the field of AIS. Therefore, in this chapter we briefly explore the underlying immunology that has served as an inspiration for the development of immune-inspired algorithms. We have chosen not to focus on the modeling aspect of AIS, but rather on the main algorithms that have been developed over recent years. This is undertaken in [Sect. 3](#) where we discuss four main immune-inspired algorithms that dominate the literature, namely, clonal selection, immune networks, negative selection, and dendritic cell algorithms, and highlight their usage in terms of applications. [Section 4](#) follows with a discussion on AIS and how researchers have begun to evaluate current AIS and describes new frameworks and methodologies that aim to help develop AIS in a more principled manner. We also briefly discuss the application of AIS to a variety of different domains and the types of applications that AIS might be better suited to, and finally we provide a very brief outline of the modeling

approaches that can be found in the literature that are employed to help further our understanding of immunology.  [Section 5](#) provides a chapter summary.

2 The Immune System

Immunology concerns the study of the immune system and the effects of its operation on the body. The immune system is normally defined in relation to its perceived function: a defense system that has evolved to protect its host from pathogens (harmful microorganisms such as bacteria, viruses, and parasites) (Goldsby et al. 2003). It comprises a variety of specialized cells that circulate and monitor the body, various extracellular molecules, and immune organs that provide an environment within which immune cells interact, mature, and respond. The collective action of immune cells and molecules forms a complex network leading to the detection and recognition of pathogens within the body. This is followed by a specific effector response aimed at eliminating the pathogen. This recognition and response process is very complicated with many details not yet properly understood.

In mammals, the immune system can be classified into two components based on functionality: a less specific component termed *innate* immunity and a more specific component termed *adaptive* (or acquired) immunity. The mechanisms of innate immunity are generic defense mechanisms that are nonspecific to particular examples of pathogen, but act against general classes of pathogen. They are encoded within the genes of the species, and do not adapt during the lifetime of the individual. Examples include the inflammatory response, phagocytic immune cells (those that can ingest and kill pathogens), anatomic barriers such as skin, and physiologic barriers such as temperature.

By contrast, the mechanisms of adaptive immunity enable the immune system to adapt to previously unseen pathogens based upon exposure to them (Goldsby et al. 2003). This is achieved through a learning mechanism that operates during the lifetime of the individual. Additionally, once exposed to a pathogen, memory mechanisms exist to allow the immune system to remember the *shape* of the pathogen. This enables a faster and more effective secondary response that can be elicited against the pathogen if it is encountered again. The adaptive and innate arms of the immune system interact to provide the body with a comprehensive defense mechanism against pathogens.

All immune cells, and the majority of other cells of the body, possess protein molecules on their surface that act as receptors to other extracellular molecules. When a sufficiently strong chemical bond occurs between a receptor and another molecule (a ligand), a cascade of intracellular signals is initiated, the outcome of which depends on the initiating receptors. This process provides the immune system with a mechanism for recognition at the molecular level. Two types of immune cell receptors exist: innate receptors that have evolved to recognize specific molecules; and the unique receptors of lymphocytes that are generated during the life time of the individual to recognize previously unseen molecules. The latter of these molecules are generically known as *antigens*, a term given to any molecular structure that can chemically bind to the unique receptors of adaptive immune cells, known as T and B-cells. The antigen receptors of the B-cell are called *antibodies*, and those of the T-cell are called T cell receptors (TCR). They are both generated via a stochastic process, and are vital to the body's adaptive immune response. Communication between immune cells involves a number of immune molecules. They include cytokines, immune cell receptors, antibodies, enzymes, plasma proteins, and adhesion molecules. The cytokines, for example, are signaling molecules

secreted by both immune and other bodily cells, which are then detected via specific cellular receptors. Many different types of cytokine exist and their effects include the activation, differentiation, growth, movement, and death of many types of cells (Cohen 2000).

2.1 Motivation for Immune Inspired Engineering Solutions

Why is it that engineers are attracted to the immune system for inspiration? The immune system exhibits several properties that engineers recognize as being desirable in their systems. Timmis and Andrews (2007), Timmis et al. (2008a), de Castro and Timmis (2002a) have identified these as the following.

Distribution and self-organization. The behavior of the immune system is deployed through the actions of billions of agents (cells and molecules) distributed throughout the body. Their collective effects can be highly complex with no central controller. An organized response emerges as a system-wide property derived from the low-level agent behaviors. These immune agents act concurrently making immune processes naturally parallelized.

Learning, adaption, and memory. The immune system is capable of recognizing previously unseen pathogens, thus exhibiting the ability to learn. Learning implies the presence of memory, and the immune system is able to “remember” previously encountered pathogens, as demonstrated by the phenomenon of primary and secondary immune responses. The first time a pathogen is encountered, an immune response (the primary response) is elicited; the next time that pathogen is encountered, a faster and often more aggressive response is mounted (the secondary response).

Pattern recognition. Through its various receptors and molecules, the immune system is capable of recognizing a diverse range of patterns. This is accomplished through receptors that perceive antigenic materials in differing contexts (processed molecules, whole molecules, additional signals, etc.). Receptors of the innate immune system vary little, whilst receptors of the adaptive immune system, such as antibodies and T-cell receptors, are subject to huge diversity.

Classification. The immune system is very effective at distinguishing harmful substances (typically viewed as *nonself*) from the body’s own tissues (typically viewed as *self*), and directing its actions accordingly. From a computational perspective, it does this with access to only a single class of data, self-molecules (Stibor et al. 2005). The creation of a system that effectively classifies data into two classes, having been trained on examples from only one, is a challenging task.

3 Engineering Artificial Immune Systems

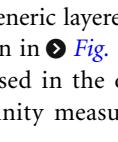
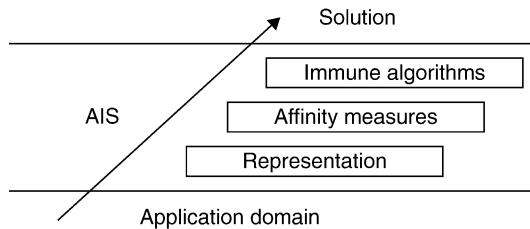
de Castro and Timmis (2002a) have proposed a flexible and generic layered approach to the development of immune-inspired engineering solutions, shown in  Fig. 1. This framework identifies the main design decisions that need to be addressed in the deployment of an immune-inspired engineering solution: representations, affinity measures, and immune algorithms.

Fig. 1

The layered framework approach to constructing AIS solutions. (Taken from de Castro and Timmis (2002a).)



Given a particular application domain, an appropriate representation of the data must be chosen. In AIS, this typically follows the notion of *shape-space* (Perelson and Oster 1979). Here, molecules m (such as receptors and antigen) exist as points in a shape space S , and can be represented as a string of attributes $m = \langle m_1, m_2, \dots, m_L \rangle$ in an L dimensional space, $m \in S^L$. The attributes m_i will represent aspects of a problem domain: patterns to be recognized, functional values to be optimized, combinations of input and proposed actions, etc. de Castro and Timmis (2002a) suggest four data types of which these attributes may belong: real valued; integer valued; hamming valued, finite length strings composed of digits with a finite alphabet; and categorically valued, where values include items such as “name,” or “color.” The affinity measures are functions or criteria through which interactions of the AIS elements are quantified. They are highly dependent on the representation chosen, for example, continuous variables typically employ the Euclidean distance measure, whereas bit string representations may use the hamming distance. Work in McEwan et al. (2008) provides a convincing critique of the shape-space paradigm and discusses the limitations of such an approach. Their paper is discussed in more detail in **Sect. 4.1**.

The highest layer of the framework details the selection of an immune-inspired algorithm to operate over the immune elements of the system. Various types of immune-inspired algorithms exist, which can operate independently of the choice of representation and affinity measure, adding dynamics to the algorithm populations based on measurements that the affinity functions provide. Despite this, immune algorithms should be chosen with care based on the problem’s data (Freitas and Timmis 2007).

In what is considered to be one of the first papers in AIS, Farmer et al. (1986) examined the immune system in the context of classifier systems, essentially highlighting the parallels of the immune network theory (Jerne 1974) and artificial intelligence. AIS has since been applied to a large range of domains that can broadly be classified as learning, anomaly detection, and optimization problems (Hart and Timmis 2008). Four main classes of AIS algorithm have been applied to these problems and each is outlined below.

3.1 Clonal Selection Theory-Inspired AIS

Clonal selection-based algorithms attempt to capture mechanisms of the antigen-driven proliferation of B-cells that results in their improved binding abilities. Using a process known as affinity maturation, the receptors of B-cell are mutated and subsequent B-cell selection results in a population of B-cells with better overall affinity for the antigen.

Clonal selection algorithms capture the properties of learning, memory, adaption, and pattern recognition (Timmis et al. 2008a).

A generic clonal selection inspired algorithm, based on CLONALG (de Castro and Von Zuben 2002, 2000), is presented in **Algorithm 1**. A set of patterns (antigens) is input to the algorithm, and output is a set of memory B-cells capable of recognizing unseen patterns. A randomly initialized set of B-cells are preferentially selected based on their affinity for the antigen. The higher affinity cells are cloned proportionally to their affinity, and mutated at a rate inversely proportional to affinity. The higher affinity clones will replace the lower affinity cells of the previous generation. Very high affinity clones compete for a place in the set of memory cells. This algorithm can be tailored toward optimization problems by removing the antigen set S , and directly representing the function or domain to be optimized as the affinity function. As clonal selection algorithms employ mutation and selection of a population of candidate solutions, they tend to be similar to other evolutionary algorithms (Newborough and Stepney 2005).

In **Algorithm 1**, a generic clonal selection algorithm is outlined, however, there are many variants in the literature that have been augmented and altered to fit specific application areas. For example, work in Watkins et al. (2004) developed a reinforcement learning approach known as AIRS (artificial immune recognition system), based on the ideas of clonal selection for the classification of unseen data items. In effect AIRS is an instance creation algorithm which acts as a preprocessor to the k -nearest neighbor approach that has been found to perform well on certain types of classification problems (Secker and Freitas 2007). In the context of dynamic learning, work by Kim and Bentley (2002a, b, c) developed a network intrusion detection system based on a dynamic variant of the clonal selection paradigm that was capable of identifying potential attacks to computer networks in an online manner and then be able to, in a limited manner, adapt to new types of attacks. As a final example, work by Kelsey and Timmis (2003), and Cutello et al. (2004a, b, 2005) have developed particularly effective optimization algorithms based on variants of clonal selection

Algorithm 1 A generic clonal selection algorithm, based on CLONALG (de Castro and Von Zuben 2000, 2002)

```

input:  $S$  = a set of antigens, representing data elements to be recognized.
output:  $M$  = set of memory B-cells capable of classifying unseen data elements.
begin
    Generate set of random specificity B-cells  $B$ .
    for all antigens  $ag \in S$  do
        Calculate affinity of all B-cells  $b \in B$  with  $ag$ .
        Select highest affinity B-cells, perform affinity proportional cloning, place clones in  $C$ .
        for all B-cell clones  $c \in C$  do
            Mutate  $c$  at rate inversely proportional to affinity.
            Determine affinity of  $c$  with  $ag$ .
        end for
        Copy all  $c \in C$  into  $B$ .
        Copy the highest affinity clones  $c \in C$  into memory set  $M$ .
        Replace lowest affinity B-cells  $b \in B$  with randomly generated alternatives.
    end for
```

by making use of novel selection and mutation mechanisms tailored specifically for certain types of optimization problems.

3.2 Immune Network Theory AIS

The immune network theory as proposed by Jerne (1974) views the immune system as a regulated network of molecules and cells that recognize each other which acts in a self-organizing manner to produce memory, even in the absence of antigen. B-cells interact via receptors to stimulate and suppress each other. This forms a regulatory network that represents an *internal image* of the antigenic patterns that the immune system observes (Farmer et al. 1986).

As with clonal selection, the immune network theory has provided inspiration for many algorithms ranging from optimization to machine learning (de Castro and Timmis 2002b; Honorio et al. 2007; Timmis and Neal 2000; de Castro and Von Zuben 2001; Bezerra et al. 2004). From a machine learning perspective, many of the systems are unsupervised and produce an instant reduction of the data space. They present clusters of this reduced data as networks of connected B-cells, where a B-cell may be considered a point m in the shape space S^L discussed above. The motivation for such algorithms is that the resulting networks highlight structures inherent in the data set and reduce the dimensionality and complexity of the data (Neal 2003). A generic immune network algorithm, based on aiNet (de Castro and Von Zuben 2001), is presented in [Algorithm 2](#). It is a modified version of CLONALG that incorporates a mechanism of suppression amongst B-cells.

In aiNet, data items are represented as antigen which B-cells (detectors) recognize. Like clonal selection algorithms ([Algorithm 1](#)), affinity maturation produces B-cells with differing specificities, and competition removes the worst of these cells from the population. A suppressive mechanism then prunes cells of similar specificities from the population. The resulting network of B-cells is then representative of clusters within the data.

Despite possessing suppressive mechanisms, early immune network algorithms suffered from an excess of B-cells, which hindered run time efficiency and rendered the resulting networks overly complex (Timmis and Neal 2000). To address this, work by Timmis and Neal (2000) incorporated the notion of an artificial recognition ball (ARB), a bounded area surrounding a point in antigenic space. All B-cells exhibiting specificities within an ARB's area are represented by that ARB, thus removing the requirement to explicitly represent each of them. To further regulate the network's population size, ARBs lie in competition with one another for a share of finite system-wide resource; ARBs that are unable to claim sufficient resource are removed from the network. Resource is allocated on the basis of ARB stimulation, derived from antigen affinity, and from low affinity to the other ARBs with which they are linked. Hence, the pressures of the algorithm are to derive clusters of linked but well spread out ARBs that represent structure in the data.

A similar, but modified, immune network algorithm was published by Neal (2003). Both cloning and hypermutation are absent in this algorithm; new ARBs are created from antigen that fall outside the range of existing ARBs in the network. The algorithm does not incorporate any stopping criteria, and can be used to create cluster-based representations of dynamically changing data. This algorithm removed the requirement for central control over the allocation of resources; ARBs are responsible for determining their own stimulation and acting accordingly. The nature of the stimulation calculation prevents ARB population explosion and

Algorithm 2 A generic immune network algorithm, based on aiNet (de Castro and Von Zuben 2001) (Taken from Timmis et al. (2008a).)

input: S = a set of antigens, representing data elements to be clustered, nt network affinity threshold, ct clonal pool threshold, h number of highest affinity clones, a number of new antibodies to introduce.

output: N = set of memory detectors capable of classifying unseen patterns.

begin

 Generate set of random specificity B-cells N .

repeat

for all antigens $ag \in S$ **do**

 Calculate affinity of all B-cells $b \in N$ with ag .

 Select highest affinity B-cells, perform affinity proportional cloning, place clones in C .

for all B-cell clones $c \in C$ **do**

 Mutate c at rate inversely proportional to affinity.

 Determine affinity of c with ag .

end for

 Select h highest affinity clones $c \in C$ and place in D .

 Remove all elements of D whose affinity with ag is less than ct .

 Remove elements of D whose affinity with other elements in D is less than ct .

 Insert remaining elements of D into N .

end for

 Determine affinity between each pair of B-cells in N .

 Systemically remove all B cells whose affinity to another B cell is less than nt .

 Introduce a new, randomly generated, B-cells into N .

until a stopping condition has been satisfied

renders the algorithm robust regarding exact parameter values. The algorithm captures well the properties of self-organization and population regulation as exhibited by the immune system. Galeano et al. (2005) provides a good review of many other immune networks that appear in the literature.

3.3 Negative Selection AIS

Inspired by the observation that the immune system protects the host body from invading pathogens, early AIS mapped these qualities to the invasion of computers and computer networks by viruses, worms, and intruders. The concept of self–nonself discrimination provided the basis for the development of various security-based AIS algorithms. Specifically, a process called negative selection was used, as inspiration, to derive a set of detectors capable of recognizing only nonself. An example of an algorithm based on Forrest et al. (1994) is shown in  *Algorithm 3*.

This algorithm was applied to protecting a computer from unauthorized changes, such as infection with a virus. There are two main stages to the algorithm: the generation of detectors; and the online monitoring of data and programs for changes. In the detector-generation stage, the collection of self strings S represents data and programs stored on the computer. The randomly generated detectors D are matched against elements in S , and those $d \in D$ that match (based on an affinity function) are removed. In Forrest et al. (1994) (this work was the

Algorithm 3 Generic negative selection algorithm (Based on Forrest et al. (1994).)

```
input:  $S$  = set of self strings characterizing benign, normal data.  
output:  $A$  = Stream of nonself strings detected.  
begin  
    Create empty set of detector strings  $D$                                 ▷ Generation of detector strings  
    Generate random strings  $C$ .  
    for all random strings  $c \in C$  do  
        for all self strings  $s \in S$  do  
            if  $c$  matches  $s$  then  
                Discard  $c$   
            else  
                Place  $c$  in  $D$   
            end if  
        end for  
    end for  
    while There exist protected strings  $p$  to check do                                ▷ Detection stage  
        Retrieve protected string  $p$   
        for all detector strings  $d \in D$  do  
            if  $p$  matches  $d$  then  
                Place  $p$  in  $A$  and output.                                         ▷ Nonself string detected  
            end if  
        end for  
    end while
```

first instance of negative selection being employed in the context of computer security) an affinity function that checked for the similarity of r consecutive characters at any point in the detector and self strings, called the r -contiguous matching rule, was used. The randomly generated detectors that are not removed from the detector collection are used to check for alterations to the system.

Negative selection algorithms have not been constrained to detection of viruses; they have also found application as intrusion detection systems. In this context the self strings S could be a concatenation of source IP, destination IP, and port addresses (Forrest and Beauchemin 2007). The detector-generation stage would be executed during a time when the network was known to be secure. Consequently, a match during the monitoring phase could indicate an anomalous connection, an intrusion. A large amount of work has been dedicated to the development of negative selection algorithms in a variety of application areas and from a theoretical perspective (Balthrop et al. 2002; Gonzalez and Dasgupta 2003; Esponda et al. 2004).

Despite a considerable amount of examples in the literature, it has been argued that negative selection suffers several drawbacks. Defining self can prove problematic; in the case of a network the total variety of safe packets can be enormous, the logistics of capturing this self set can prove difficult. In deriving the set D a huge quantity of randomly generated detectors that match self will have been deleted, thus it can become very inefficient (Freitas and Timmis 2007). Furthermore, algorithms of this variety have been seen to suffer certain scaling problems: as the universe in which self and nonself elements are defined grows (reflecting the complexity of the detection problem), the number of detectors required to effectively cover the nonself space becomes difficult to generate (Stibor et al. 2005; Timmis et al. 2008b).

3.4 Danger Theory AIS

It has been suggested that the integration of mechanisms derived from *danger theory* (Matzinger 1994) could provide for more effective intrusion detection algorithms than traditional negative selection approaches (Aickelin and Cayzer 2002). Rather than monitor for the explicit presence of the intruder, danger-theory-inspired systems could be alerted by the anomalous intruder behavior. The shift in emphasis is subtle, but significant. Such an intrusion detection system would monitor for signs of “danger,” such as abnormalities in memory usage or disk activity, unexpected or unwarranted frequencies of file changes (Aickelin and Cayzer 2002; Aickelin et al. 2003).

An interesting consequence of danger-inspired AIS lies in the interpretation of the *danger zone*. *In vivo*, this is the spatial neighborhood from where the danger signals originate. In the artificial domain, this concept need not be spatial, Secker et al. (2003) place the danger zone in the temporal domain. The concept of danger signals provides danger-theory-inspired engineering solutions with several advantages over self- and nonself-inspired approaches. Danger signals restrict the domain of nonself to a manageable size, remove the requirement to observe all self, and instill adaptability regarding scenarios where self and nonself boundaries are dynamic (Aickelin and Cayzer 2002).

The main danger-theory-inspired algorithm that has been developed is the dendritic cell algorithm (DCA) (Greensmith et al. 2005, 2006a). The DCA is a signal-processing algorithm, inspired by the behavior of dendritic cells. These reside in the body tissues and collect antigen and other (danger) signals that provide a picture of the current state of the tissues. This picture determines whether the antigen has been collected in a safe or dangerous context, and causes dendritic cells to change into a *semi-mature* or *mature* state. The task of the DCA is to classify data items (antigens) as being either benign or malignant in nature. Antigen are associated with concentrations of pathogen associated molecular pattern (PAMP) signal, danger signal, safe signal, and pro-inflammatory signals. These signals are derived from real biological signals and are mapped onto attributes associated with the data items as follows (Greensmith et al. 2006a):

- *PAMP*. A known signature of abnormal behavior. This attribute of the data item is highly indicative of an anomaly.
- *Danger signal*. A moderate degree of confidence that this attribute of the data item is associated with abnormal behavior.
- *Safe signal*. Indicative of normal system operation.
- *Pro-inflammatory signal*. A general sign of system distress.

The main challenge in implementing the DCA is defining how these signals map onto the data items derived from the problem domain (Greensmith et al. 2006a).

The DCA, shown in  [Algorithm 4](#), operates by maintaining a pool of dendritic cells (DCs). From this pool, dendritic cells are randomly selected to sample data items (and related signals) that are presented to the algorithm in a sequential manner. Based on the signals received, dendritic cells produce *semi-mature* and *mature* cytokines (immune signaling molecules). At the end of antigen processing, DCs are assigned semi-mature or mature status according to the levels of the cytokines produced. Every data item is then classified as being benign or malignant on the basis of a majority vote amongst the DCs that sampled it, each voting in accordance to its level of maturity.

Through its focus on behavioral consequences (derived from the signals described above) as opposed to physical presence (in the case of negative selection algorithms), the

Algorithm 4 The Dendritic Cell Algorithm (DCA) (Greensmith et al. 2005)

```

input:  $S$  = a set of antigens, representing data elements classified as safe or dangerous.
output:  $K$  = set of antigens classified as safe.
           $L$  = set of antigens classified as dangerous.

begin
  Create DC pool of 100 dendritic cells.
  for all antigen  $ag \in S$  do                                ▷ Perform signal processing on  $ag$ 
    for 10 randomly selected dendritic cells  $dc \in DC$  do
      Sample  $ag$ .
      Update  $dc.danger$ ,  $dc.PAMP$ , and  $dc.safe$  signals based on  $ag$ .
      Calculate and update concentration of  $dc.semimatureCytokine$  output cytokine.
      Calculate and update concentration of  $dc.matureCytokine$  output cytokine.
      Calculate and update concentration of  $dc.coStimulatory$  output molecules.
      if concentration of  $dc.coStimulatory$  > threshold then
        Remove  $dc$  from  $DC$  and place in  $M$ .
        Insert new  $dc$  into  $DC$ .
      end if
    end for
  end for

  for all dendritic cells  $dc \in M$  do                      ▷ Differentiation of dendritic cells.
    if concentration of  $dc.semimatureCytokine$  >  $dc.matureCytokine$  then
       $dc.class$  = semi/mature.
    else
       $dc.class$  = mature.
    end if
  end for

  for all antigen  $ag \in S$  do                                ▷ Classification of antigens
    for all dendritic cells  $dc \in M$  that sampled  $ag$  do
      Calculate if  $ag$  presented in mature or semimature context by  $dc$ .
    end for
    if  $ag$  presented as semimature majority of time then
      Place  $ag$  in  $K$ .                                         ▷  $ag$  is benign
    else
      Place  $ag$  in  $L$ .                                         ▷  $ag$  is malignant
    end if
  end for

```

DCA is able to operate in the presence of dynamically changing environments. However, in its current state (Greensmith et al. 2005, 2006a, b), the DCA is not able to operate in a true online fashion; data must be collected *a priori* and classification is performed as a final batch operation. Hence, anomalies cannot be detected as they occur. A second potential problem for the DCA is that misclassification can occur around the boundaries where data items switch between *safe* and *dangerous* contexts. This is due to multiple sampling of antigen by each DC. The consequence is that the DCA will exhibit significant misclassification when applied to problems where context switches in the data items are frequent (Greensmith et al. 2005). In order to overcome the limitation of operating in an off-line manner, Lay and Bate (2007) have

developed a real-time, online DCA that is capable of altering schedule overruns in real-time operating systems. The DCA has also been used for behavior classification on a robotics platform (de Castro et al. 2007a).

4 Reflections and Projections

Artificial immune systems has matured into a well recognized field that tackles a broad range of problem domains. This is best illustrated from the proceedings of the International Conference of Artificial Immune Systems ICARIS (Timmis et al. 2003; Nicosia et al. 2004; Jacob et al. 2005; Bersini and Carneiro 2006; de Castro et al. 2007b; Bentley et al. 2008). The field is now at a stage where a number of researchers are reflecting upon its contributions to the wider academic and engineering communities. A number of these reflections and proposed future directions for AIS are assessed here.

4.1 Evaluation of Current AIS

Hart and Timmis (2008) analyze a large collection of AIS engineering applications and categorize these into three classes of problem: anomaly detection, optimization, and clustering and classification. Considering key works from each class in turn, they attempt to assess and evaluate whether the application of AIS brings any benefits that could not be derived from applying alternative, existing techniques to the problem. Their criteria asserts that it is not sufficient to simply outperform other algorithms on benchmark tests; to be truly successful, the AIS must contain features that are not present in alternative paradigms.

Anomaly detection AIS are assessed by Hart and Timmis (2008) as having had limited success, but the authors make note of recent advances that danger-theory-inspired algorithms have provided, and state that significant breakthroughs are still possible. For optimization problems, it is concluded that although optimization-based AIS can and will provide comparable performance to existing methods, they will not offer any distinguishing features that cannot be found elsewhere. For classification and clustering applications, the authors conclude that the naturally distributed nature of some AIS algorithms allows for natural parallelization and distribution across several processors, offering something potentially distinctive. Regarding operation over dynamic data sets, the authors state that by definition, AIS algorithms incorporate some notion of memory, and could therefore outperform alternative learning systems which are purely reactive in nature.

Though their assessment of AIS accomplishments concludes that many are not truly successful, the authors note that this is partly due to several shortcomings that have characterized AIS design and application to date (Hart and Timmis 2008). These include: the methodology through which AIS algorithms capture their inspiring immunology; the attention paid to the effects that certain design decisions impose when engineering AIS systems; the theoretical understanding of AIS algorithms; and the nature of the problems to which AIS have been applied.

In a similar vein to Hart and Timmis (2008), Garrett (2005) studies various AIS to attempt to answer the question of whether AIS research has delivered anything *useful* to date. A useful algorithm in this context is defined by being *distinct* and *effective*. An algorithm's distinctiveness is assessed through criteria covering the algorithm's internal

representation of the problem and potential solutions, and its computational components. Effectiveness is assessed on the algorithm's performance, including the path through which solutions are obtained, the quality of results obtained through its application to benchmark problems, and the speed at which results can be obtained. In combining the two sets of criteria, an algorithm is said to be useful if it is both effective and distinctive.

The fact that work reflecting on the state of AIS is being conducted is encouraging, and is healthy for the discipline. However, it should be noted that the method and criteria employed by Garrett (2005) in arriving at its conclusions has been criticized for being more of an exercise in classification than in detailed evaluation, and for being highly subjective in nature (Timmis et al. 2008a). Additionally, the criteria focuses on performance in relation to benchmark problems. It has been suggested that a downfall of AIS research to date has been its repeated application to benchmark problems, and to areas for which many quality solutions already exist (Hart and Timmis 2008; Timmis et al. 2008a). The effectiveness criteria do not reflect the need for AIS to carve its own niche (Hart and Timmis 2008; Timmis et al. 2008a), and provide quality solutions in a problem domain that no other technique can match.

McEwan et al. (2008) question the appropriateness of the shape space representation for AIS with respect to machine-learning problems. Typical machine-learning problems entail data sets of very high dimensionality. In such a scenario, the adoption of the shape space representation can lead to the “curse of dimensionality”: as the dimension of the space increases linearly, its volume increases exponentially, and the quality of locality that affinity measures attempt to discern becomes meaningless as all points approach in equidistance to one another. It has been noted by Stibor et al. (2005) that the task of generating, maintaining, and exploiting an effective set of detectors within such a high-dimensional space is computationally intractable.

As an alternative, McEwan et al. (2008) propose marrying the machine-learning technique of *boosting* with immune inspiration. Boosting proposes a strong learning strategy that is derived as a compound decision between multiple (slightly better than random) weak learners. The authors draw analogy to the cooperative nature in which many varieties of immune cells with differing specificities and recognition targets are able to cooperatively mount an effective immune response that hones on a specific target (Cohen's *correspondence* (Cohen 2000)).

4.2 Inspiration, Frameworks, and Methodologies

In recent years, there has been a gradual shift in some AIS toward paying more attention to the underlying biological system that serves as inspiration. For example, the development of the DCA (see  Sect. 3.4) involved the input from real biological experimentation as inspiration. However, there was no reported sophisticated biological modeling to understand the underlying biology as is suggested by Stepney et al. (2005) and Timmis et al. (2006). Other examples of this shift back to the underlying biology include Wilson and Garrett (2004) and Jacob et al. (2004), who have used modeling techniques to build AIS in order to understand underlying immune properties.

The majority of AIS such as those detailed in  Sects. 3.1–3.3 have taken their inspiration from well-established immunological perspectives. In contrast, Andrews and Timmis (2005, 2007) advocate exploiting conflicting immune theories as a rich source of potential ideas for the engineer. This is an approach that has been successfully carried out by Aickelin and

Cayzer (2002), Secker et al. (2003), and Greensmith et al. (2005) in exploiting danger theory and the development of the DCA. AIS can draw significantly more inspiration from the immune system, and the immunological debate surrounding its higher functions, than the relatively simplistic subset of concepts that have served thus far.

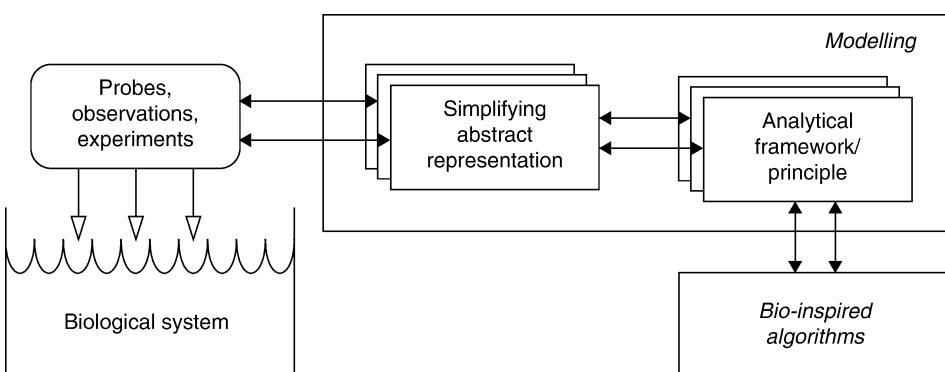
Taking this approach, the AIS engineer enjoys the freedom of adopting various immunological models and concepts that best suit the application domain. It is, however, essential to ensure that the concepts employed are correctly abstracted and reasoned about to accurately capture the emergent phenomena from which they are inspired. It has been argued that various immune-inspired algorithms have been hampered with a lack of biological accuracy (Timmis 2007). Typically, immune-inspired algorithms have fallen prey to “reasoning by metaphor,” wherein their operation and structures bear only a weak resemblance to the biological phenomenon that inspired them (Stepney et al. 2005), and, thus, consequentially fail to unlock their full potential (Hart and Timmis 2008).

To combat the problems associated with this apparent weakness in biological metaphors, a conceptual framework approach to the development of bio-inspired algorithms has been proposed (Stepney et al. 2005), shown in Fig. 2. This provides a structure and methodology for biological investigation, abstraction, modeling, and ultimately the construction of algorithms. The process should be interdisciplinary, involving at the very least biologists, mathematicians, and computer scientists. The framework aims to facilitate a better understanding of the targeted underlying biological concepts and to ultimately build more powerful bio-inspired algorithms whilst simultaneously gaining a better understanding of which application domains these algorithms are best suited to.

The first step in the conceptual framework approach is to probe the biological system through observation and experimentation. These probes are biased toward extracting information concerning the particular biological phenomena of interest. From the information gained, careful abstraction and mathematical modeling will highlight the central processes responsible for the observed biological phenomena. Analytical computational models may be constructed, which allow for the execution and animation of any underlying model, and can

Fig. 2

The conceptual framework approach to deriving biologically inspired algorithms (Stepney et al. 2005).



provide a deeper insight into its workings. The observations and mechanisms perceived at this stage will be free from any particular application bias. Finally, these insights can serve as design principles for bio-inspired algorithms, which may be applied to non-biological problems (Stepney et al. 2005; Hart and Timmis 2008).

A number of AIS works have been inspired by the conceptual framework principles. These include: a computational model of degenerate T-cell receptors (Andrews and Timmis 2006) and adaptable degenerate immune cell receptors (Andrews and Timmis 2008); and an instantiation of an artificial cytokine network (Hone and van den Berg 2007), which examined the behavior of the network to elicit any useful properties that could be applied to solving engineering problems (Read et al. 2008). Newborough and Stepney (2005) also apply many of the conceptual framework ideas to produce a generic framework for population-based bio-inspired algorithms including genetic algorithms, negative selection, clonal selection, particle swarm optimization, and ant colony optimization.

The conceptual framework of Stepney et al. (2005) also influenced Twycross and Aickelin (2005) who present a general meta-framework for models incorporating innate immunity. A table of six general properties of the innate immune system is presented and it is claimed that AIS will need to incorporate properties such as these to realize functions of the immune system. Similarly, Guzella et al. (2007) highlight a class of T cell, T regulatory cells, as inspiration for AIS. They suggest that incorporating these cells might lead to more biologically plausible models and algorithms that achieve better results in real-life problems.

While the conceptual framework offers a structured methodology for the development of immune- (and other biologically) inspired algorithms, the deployment of these AIS in a particular engineering context also requires careful consideration. Through their examination of AIS application to classification problems Freitas and Timmis (2007) note several considerations, frequently overlooked, which can significantly affect an algorithm's suitability and performance. They state that the implementor of an AIS algorithm should note the nature of the problem's data, and chose a representation that intuitively maps the data's characteristics. Altering the data to suit a particular representation, in particular, discarding data that is of a different type (e.g., disposing of categorical data to fit a continuous valued representation), is bad practice. Rather, the immune-inspired algorithm's representation should be tailored to suit the problem's data.

Freitas and Timmis (2007) also advise careful consideration of the choice of affinity measure for the chosen representation. An affinity measure can be associated with an inductive bias: some basis through which one hypothesis will be favored over another. An inductive bias is not an undesirable trait, it forms the basis of learning. Yet, care must be taken to ensure that the inductive bias incurred is appropriate for the problem at hand. For example, certain affinity measures have a positional bias, whereby the order of data within the representation can affect the outcome of the affinity measure. If the order of the data is irrelevant to the problem being tackled, then an affinity measure yielding a positional bias might be an inappropriate choice. This work is supported by empirical investigations into the effects of different affinity measures by Hart and Ross (2004) and Hart (2005).

4.3 Application Domains

It has been suggested by Hart and Timmis (2008) that there will be little benefit from applying AIS algorithms to problems of a static nature, over existing and established paradigms. The

authors conjecture that the distinctive “killer application” niche for AIS will require algorithms to exhibit the following properties (quoted verbatim):

- They will be *embodied*.
- They will exhibit *homeostasis*.
- They will benefit from interactions between *innate* and *adaptive* immune models.
- They will consist of *multiple, heterogeneous interacting, communicating components*.
- Components can be easily and naturally *distributed*.
- They will be required to perform *life-long learning*.

Recent applications of AIS in novel problem domains have started to show indications of satisfying these properties, which are reviewed here.

A central function of the immune system is its cooperation with the endocrine and neural systems in the provision of homeostasis to the host (Hart and Timmis 2008). Homeostasis is “the tendency of a system, esp. the physiological system of higher animals, to maintain internal stability, owing to the coordinated response of its parts to any situation or stimulus tending to disturb its normal condition or function” [American Psychological Association \(APA\)](#). Hence, since the domain in which *in vivo* immune systems operate is inherently dynamic; it is not unreasonable to surmise that immune-inspired algorithms might be particularly well suited to operation in dynamic environments. The immune system’s potential as inspiration for homeostasis in robotics is investigated by Owens et al. (2007). Here, homeostasis requires: the system to perceive the environment from multiple perspectives to overcome the inherent problems of sensory malfunction; a repertoire of innate responses that can affect change in the environment or the system directly; the cognition that facilitates the selection of an appropriate effector action in response to perceived input state; and the ability to adaptively correlate sensory information and effector mechanisms, such that its actions can dynamically evolve with a changing environment. Similarly, Neal et al. (2006) outline an endocrine-immune-inspired homeostatic control system. The artificial immune system allows for low-level faults (e.g., an overheating motor) to be corrected locally (e.g., by turning on a local fan), while integration with an artificial endocrine system allows for chronic faults to propagate inflammation throughout the robot’s systems. System-wide inflammation influences the higher level function of the robot in a global attempt to rectify the fault, for example, the decision by the robot to stop moving, thus allowing the motor to cool down.

The potential for AIS application in the domain of real-time systems was demonstrated by Lay and Bate (2007), who employed the dendritic cell algorithm in the detection of process deadline over runs in an embedded system. The analysis of process executions, and the insurance that all deadlines are met is typically performed statically during the development process. By incorporating adaptive AIS techniques, it is hoped that the system is rendered robust, while simultaneously reducing development time and costs.

Embodiment in bio-inspired engineering has been investigated by Stepney (2007), who examines the intimate coupled nature of a system and its environment. This includes their perceptions and consequent reactions in perturbing one another through complex high bandwidth feedback networks. The environment is open, with a quantitatively large and rich variety of information flowing through it, while the system exhibits highly nonlinear dynamics; small input perturbations need not equate to small behavioral modifications. A consequence of embodiment is the coevolution of the environment with the system. In the context of the danger theory of the immune system (Matzinger 2002), immune cells (system) have learnt to perceive danger signals just as the body (environment) has learnt to

provide them. Pathogens experience evolutionary pressure to evade detection, thus contributing to the environment's dynamics (Stepney 2007). Thus, the two are intimately bound. This concurs with the argument for the complex systems view of immunology presented by Cohen (2000). Thus, for engineers to truly capture the complexity of the biology from which they derive their inspiration, they must embody the artificial system within its artificial environment, rather than deliberately engineer the interfaces, sensors, and actuators through which the system interacts with its environment.

Though no AIS currently satisfies the conceptual features of embodiment as outlined in Stepney (2007), Bentley et al. (2005) go some way in addressing similar issues by suggesting that a layer is missing from AIS design. They outline the concept of an artificial tissue layer acting as an interface between a problem space and an AIS. The tissue layer performs some data preprocessing before presenting it to the AIS, allowing for the incorporation of domain-specific knowledge and the integration of several data sources, and is the medium through which the AIS responds. An analogy is drawn by Bentley et al. (2005) between the artificial tissue providing an innate response and the AIS providing the adaptive response. As a preliminary investigation, two tissue algorithms are presented by Bentley et al. (2005). In a similar work, Twycross and Aickelin (2006) present a framework that facilitates the complete encapsulation of an AIS, providing: artificial anatomical compartments within which the AISs immune elements may operate; and generic receptors for other immune cells, antigens, and cytokines contained within the compartment. The framework was partially motivated through the possibility to evaluate the performance of several alternative AIS algorithms on the same problem, but the manner in which it interfaces the AIS with the environment, and performs preprocessing is interesting from the perspective of embodiment.

4.4 Modeling and Simulating the Immune System

In recent years, building models and simulations of the immune system have become an important aspect of AIS, both as stand-alone pieces of work, and as steps toward producing engineering applications using methodologies such as the conceptual framework approach of Stepney et al. (2005). Forrest and Beauchemin (2007) note that there is a vast range of modeling approaches applicable to modeling the immune system, each with their own advantages and disadvantages operating at different levels of abstraction.

An overview of many mathematical techniques used for modeling the immune system is provided by Perelson and Weisbuch (1997). A large number of these approaches involve the use of differential equations, although other techniques can be applied, such as Boolean networks (Weisbuch and Atlan 1988) and the work of Kelsey et al. (2008) which present and analyze a Markov chain model of a cytokine network. Recently, process calculi have been applied to models of the immune system such as Owens et al. (2008). Process calculi are formal languages from computer science that are used to specify concurrent systems. As biological systems are inherently concurrent, these types of languages seem well suited to biological modeling.

Forrest and Beauchemin (2007) provide a review of many of the modeling approaches in immunology, with a focus on agent based modeling (ABM). In ABM, components such as cells (and sometimes molecules) are represented individually as *agents*, rather than as homogenous populations such as in differential equation techniques. Different agent types typically represent different immune cell types. These agent types are encoded with simple

rules extracted from the real biology that govern how they behave and interact. ABM techniques typically employ an explicit notion of space, such as that used in cellular automata-like models (Kleinsteiner and Seiden 2000). The advantage of ABM is that it allows the observation of agent population dynamics as they emerge from the interactions of individual agents. An example of ABM is Beauchemin et al. (2006) who investigate the dynamics of in vitro infection with a strain of influenza. ABM has also been used to study more computational aspects of applied AIS such as a series of work by Hart and Ross (2004), Hart (2005, 2006), and Hart et al. (2006). These works use a simulation of an idiotypic network to investigate how different models of shape-space and affinity affect the dynamics of the network, such as memory capacity and the structures formed, emphasizing the need for careful choice of parameters in the engineered systems.

Diagrammatic tools have also been used to model the immune system, the most widely used being the unified modeling language (UML) (Fowler 2000), which consists of a set of 13 different types of diagram that can model different aspects of structure and behavior. The advantage of the UML is its non-domain-specific nature and subsequent ability to capture abstractions. The UML (and related diagrams such as statecharts) have started to become a powerful tool in modeling aspects of biological systems. By far the most advanced use of the UML and statecharts in immunology is that of Efroni et al. (2003), who have built a sophisticated and predictive model of T cell maturation in the thymus using a tool called reactive animation, which combines the use of statecharts and other UML diagrams. In addition to the UML, there are other techniques used in software engineering, which Bersini (2006) suggests can facilitate the development and *communication* of immune modeling. These include object-oriented technologies such as object-oriented programming and design patterns (Gamma et al. 1995). The perceived benefit is the clarification of immune objects and their relationships. To support this, Bersini (2006) provides an example of how clonal selection can be modeled with a simple state diagram.

5 Summary

This chapter is intended as an overview of the area of artificial immune systems (AIS), predominantly from an engineering solutions perspective. It is not meant to be an exhaustive bibliography, but serves to illustrate that AIS is an area of great diversity, actively reflecting upon itself, and expanding into new areas and meeting new challenges. The spectrum of AIS research ranges from the modeling of immune systems in aid of immunological study, to the development of algorithms for specific engineering applications. While the predominant focus of this chapter has been on the algorithmic aspect of AIS, the other aspects of AIS research are no less significant. The principled development of immune-inspired algorithms that capture, in a more than superficial manner, the properties and characteristics of the immune system are equally valuable to the discipline's continuing success. This is highlighted to different degrees in Stepney et al. (2005) and Timmis et al. (2006) who advocate the careful consideration of the underlying biological system, the use of modeling to help understand that system, and the principled abstraction of algorithms and general frameworks from those models.

It is worth noting that there are many different aspects of the immune system that have served as inspiration for AIS. Only the main strands have been reviewed in this chapter, namely: clonal selection, immune networks, negative selection, and danger theory. However, there remain many untapped possibilities that are worthy of consideration and study within

principled frameworks, such as the conceptual framework of Stepney et al. (2005). In addition, the area of AIS should seek out challenging application areas that exploit the immune metaphor further than it has to date.

As pointed out by Timmis et al. (2008a), a recent paper by Cohen (2007) identifies three types of AIS researcher. The first type he calls the literal school, they build artificial systems that attempt to perform analogous tasks to the actual immune system (e.g., build computer security systems that discriminate between self and nonself); the second type are those of the metaphorical school who take inspiration from the immune system and build artificial systems based on analogies (so the application may be far from analogous to what the immune system does); and a third type of researcher aims to understand immunity through the development of computational and mathematical models. This goes to illustrate the diversity of research that lies within the discipline of AIS, and renders it a promising avenue for truly interdisciplinary research.

Acknowledgments

Mark Read is sponsored by the Department of Computer Science, University of York, and Paul Andrews is supported by EPSRC grant number EP/E053505/1.

References

- Aickelin U, Cayzer S (2002) The danger theory and its application to artificial immune systems. In: Timmis J, Bentley PJ (eds) ICARIS 2002: Proceedings of the 1st international conference on artificial immune systems, University of Kent Printing unit, Canterbury, UK, September 2002, pp 141–148
- Aickelin U, Bentley PJ, Cayzer S, Kim J, McLeod J (2003) Danger theory: The link between AIS and IDS? In: Bentley PJ, Hart E (eds) ICARIS 2003: 2nd international conference on artificial immune systems, Edinburgh, Scotland, September 2003. Lecture notes in computer science, vol 2787. Springer, New York, pp 147–155
- American Psychological Association (APA): Homeostasis. (n.d.). Dictionary.com Unabridged (v 1.1). Retrieved June 25, 2008, from Dictionary.com Web site: <http://dictionary.reference.com/browse/homeostasis>
- Andrews PS, Timmis J (2005) Inspiration for the next generation of artificial immune systems. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 126–138
- Andrews PS, Timmis J (2006) A computational model of degeneracy in a lymph node. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 164–177
- Andrews PS, Timmis J (2007) Alternative inspiration for artificial immune systems: exploiting Cohen's cognitive immune model. In: Flower D, Timmis J (eds) In silico-immunology. Springer, New York, Chap 7 (2007)
- Andrews PS, Timmis J (2008) Adaptable lymphocytes for artificial immune systems. In: Bentley PJ, Lee D, Jung S (eds) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 376–386
- Balthrop J, Esponda F, Forrest S, Glickman M (2002) Coverage and generalisation in an artificial immune system. In: Genetic and evolutionary computation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, July 2002, pp 3–10
- Beauchemin C, Forrest S, Koster FT (2006) Modeling influenza viral dynamics in tissue. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 23–36
- Bentley PJ, Greensmith J, Ujjin S (2005) Two ways to grow tissue for artificial immune systems. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune

- systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 139–152
- Bentley PJ, Lee D, Jung S (eds) (2008) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 5132. Springer, New York <http://www.artificial-immune-systems.org/icaris.shtml>
- Bersini H (2006) Immune system modeling: the OO way. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin <http://www.artificial-immune-systems.org/icaris.shtml>
- Bezerra GB, de Castro LN, Zuben FJV (2004) A hierarchical immune network applied to gene expression data. In: ICARIS 2004: Proceedings of the 3rd international conference on artificial immune systems. Catania, Springer, Berlin/Heidelberg, September 2004, pp 14–27
- Cohen IR (2000) Tending Adam's garden: evolving the cognitive immune self. Elsevier Academic Press, London, UK
- Cohen IR (2007) Real and artificial immune systems: computing the state of the body. *Nat Rev Immunol* 7:569–574
- Cutello V, Nicosia G, Pavone M (2004a) Exploring the capability of immune algorithms: a characterization of hypermutation operators. In: Nicosia G, Cutello V, Bentley PJ, Timmis J (eds) ICARIS 2004: 3rd international conference on artificial immune systems? Catania, Italy, September 2004. Lecture notes in computer science, vol 3239. Springer, Berlin, pp 263–276
- Cutello V, Nicosia G, Pavone M (2004b) An immune algorithm with hyper-macromutations for the Dill's 2D hydrophobic-hydrophilic model. IEEE congress on evolutionary computation, CEC 2004, Portland, Oregon, USA, June 19–23, 2004. IEEE Press, 1:1074–1080
- Cutello V, Narzisi G, Nicosia G, Pavone M (2005) Clonal selection algorithms: A comparative case study using effective mutation potentials. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 263–276
- de Castro LN, Timmis J (2002a) Artificial immune systems: a new computational approach. Springer-Verlag, London
- de Castro L, Timmis J (2002b) An artificial immune network for multi modal optimisation. In: WCCI: Proceedings of the world congress on computational intelligence, Honolulu, HI, May 2002. IEEE, New York, NY, USA, pp 699–704
- de Castro LN, Von Zuben FJ (2000) The clonal selection algorithm with engineering applications. In: Proceedings of GECCO'00, workshop on artificial immune systems and their applications. Las Vegas, NV
- de Castro LN, Von Zuben FJ (2001) aiNet: an artificial immune network for data analysis. Idea Group Publishing, Hershey, PA, pp 231–259
- de Castro LN, Von Zuben FJ (2002) Learning and optimization using the clonal selection principle. *IEEE Trans Evol Comput* 6(2):239–251
- de Castro LN, Von Zuben FJ, Knidell H (eds) (2007a) The application of a dendritic cell algorithm to a robotic classifier. In: ICARIS 2007: Proceedings of 6th international conference on artificial immune systems, Santos, Brazil, August 2007. Lecture notes in computer science, vol 4628. Springer, Berlin
- de Castro LN, Von Zuben FJ, Knidell H (eds) (2007b) ICARIS 2007: Proceedings of 6th international conference on artificial immune systems, Santos, Brazil, August 2007. Lecture notes in computer science, vol 4628. Springer, Berlin <http://www.artificial-immune-systems.org/icaris.shtml>
- Efroni S, Harel D, Cohen IR (2003) Towards rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic t-cell maturation. *Gen Res* 13:2485–2497
- Esponda F, Forrest S, Helman P (2004) A formal framework for positive and negative detection schemes. *IEEE Trans Syst Man Cybern B Cybern* 34(1):357–373
- Farmer JD, Packard NH, Perelson AS (1986) The immune system, adaptation, and machine learning. *Phys D* 2(1–3):187–204
- Flower D, Timmis J (eds) (2007) *In silico immunology*. Springer, New York
- Forrest S, Beauchemin C (2007) Computer immunology. *Immunol Rev* 216(1):176–197
- Forrest S, Perelson AS, Allen L, Cherukuri R (1994) Self-nonself discrimination in a computer. In: SP '94: Proceedings of the 1994 IEEE symposium on security and privacy, Oakland, CA, May 1994. IEEE Computer Society, Washington DC, pp 202–212
- Fowler M (2000) UML distilled: a brief guide to the standard object modeling language, 2nd edn. Addison-Wesley, Reading, MA
- Freitas A, Timmis J (2007) Revisiting the foundations of artificial immune systems for data mining. *IEEE Trans Evol Comput* 11(4):521–540
- Galeano JC, Veloza-Suan A, González FA (2005) A comparative analysis of artificial immune network

- models. In: GECCO 2005: Proceedings of the genetic and evolutionary computation conference, Washington, DC, June 2005. Springer, Berlin
- Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA
- Garrett S (2005) How do we evaluate artificial immune systems? *Evol Comput* 13(2):145–177
- Goldsby RA, Kindt TJ, Osborne BA, Kuby J (2003) Immunology, 5th edn. W. H. Freeman and Company, New York
- Gonzalez FA, Dasgupta D (2003) Anomaly detection using real-valued negative selection. *Genet Programming Evolvable Mach* 4(4):383–403
- Greensmith J, Aickelin U, Cayzer S (2005) Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 153–167
- Greensmith J, Aickelin U, Twycross J (2006a) Articulation and clarification of the dendritic cell algorithm. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 404–417
- Greensmith J, Twycross J, Aickelin U (2006b) Dendritic cells for anomaly detection. In: CEC 2006: IEEE congress on evolutionary computation, Vancouver, Canada, July 2006, pp 664–671
- Guzella TS, Mota-Santos TA, Caminhos WM (2007) Regulatory t cells: inspiration for artificial immune systems. In: de Castro LN, Von Zuben FJ, Knidel H (eds) ICARIS 2007: of 6th international conference on artificial immune systems, Santos, Brazil, August 2007. Lecture notes in computer science, vol 4628. Springer, Berlin, pp 312–323
- Hart E (2005) Not all balls are round: an investigation of alternative recognition-region shapes. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 29–42
- Hart E (2006) Analysis of a growth model for idiotypic networks. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 66–80
- Hart E, Ross P (2004) Studies on the implications of shape-space models for idiotypic networks. In: Nicosia G, Cutello V, Bentley PJ, Timmis J (eds) ICARIS 2004: 3rd international conference on artificial immune systems, Catania, Italy, September 2004. Lecture notes in computer science, vol 3239. Springer, Berlin, pp 413–426
- Hart E, Timmis J (2008) Application areas of AIS: the past, the present and the future. *J Appl Soft Comput* 8(1):191–201
- Hart E, Bersini H, Santos F (2006) Tolerance vs intolerance: How affinity defines topology in an idiotypic network. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 109–121
- Hone A, van den Berg H (2007) Modelling a cytokine network (special session: Foundations of artificial immune systems). In: Foundations of computational intelligence, Honolulu, HI, April 2007. IEEE, New York, pp 389–393
- Honorio L, Leite da Silva A, Barbosa D (2007) A gradient-based artificial immune system applied to optimal power flow problems. In: de Castro LN, Von Zuben FJ, Knidel H (eds) ICARIS 2007: 6th international conference on artificial immune systems, Santos, Brazil, August 2007. Lecture notes in computer science, vol 4628. Springer, Berlin, pp 1–12
- Jacob C, Litorco J, Lee L (2004) Immunity through swarms: Agent-based simulations of the human immune system. In: Nicosia G, Cutello V, Bentley PJ, Timmis J (eds) ICARIS 2004: 3rd international conference on artificial immune systems, Calania, Italy, September 2004. Lecture notes in computer science, vol 3239. Springer, Berlin, pp 400–412
- Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) (2005) ICARIS 2005: 4th international conference on Artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg <http://www.artificial-immune-systems.org/icaris.shtml>
- Jerne NK (1974) Towards a network theory of the immune system. *Ann Immunol (Inst Pasteur)* 125C:373–389
- Kelsey J, Timmis J (2003) Immune inspired somatic contiguous hypermutation for function optimisation. In: GECCO 2003: Genetic and evolutionary computation conference, Chicago, IL, July 2003. Springer, New York, pp 207–218
- Kelsey J, Henderson B, Seymour R, Hone A (2008) A stochastic model of the interleukin (IL)-1 β network. In: Bentley PJ, Lee D, Jung S (eds) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 5132. Springer, New York, pp 1–11

- Kim J, Bentley PJ (2002a) A model of gene library evolution in the dynamic clonal selection algorithm. In: Timmis J, Bentley PJ (eds) ICARIS 2002: Proceedings of the 1st international conference on artificial immune systems. University of Kent Printing Unit, Canterbury, UK, September 2002, pp 182–189
- Kim J, Bentley P (2002b) Immune memory in the dynamic clonal selection algorithm. In: Timmis J, Bentley PJ (eds) ICARIS 2002: Proceedings of the 1st international conference on artificial immune systems. University of Kent Printing Unit, Canterbury, UK, September 2002, pp 59–67
- Kim J, Bentley PJ (2002c) Towards an artificial immune system for network intrusion detection: an investigation of dynamic clonal selection. In: CEC2002: Proceedings of the 2002 congress on evolutionary computation. Honolulu, HI, May 2002
- Kleinsteiner SH, Seiden PE (2000) Simulating the immune system. *Comput Sci Eng* 2(4):69–77
- Lay N, Bate I (2007) Applying artificial immune systems to real-time embedded systems. In: IEEE congress on evolutionary computation 2007, Singapore, September 2007, pp 3743–3750
- Matzinger P (1994) Tolerance, danger, and the extended family. *Annu Rev Immunol* 12:991–1045
- Matzinger P (2002) The danger model: a renewed sense of self. *Science* 296(5566):301–305
- McEwan C, Hart E, Paechter B (2008) Boosting the immune system. In: Bentley PJ, Lee D, Jung S (eds) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 5132. Springer, New York, pp 316–327
- Neal M (2003) Meta-stable memory in an artificial immune network. In: Timmis J, Bentley PJ, Hart E (eds) ICARIS 2003: 2nd international conference on artificial immune systems, Edinburgh, Scotland, September 2003. Lecture notes in computer science, vol 2787. Springer, New York, pp 168–180
- Neal M, Feyereisl J, Rascunà R, Wang X (2006) Don't touch me, I'm fine: robot autonomy using an artificial innate immune system. In: Bersini H, Carneiro J (eds) ICARIS 2006: 5th international conference on artificial immune systems, Oeiras, Portugal, September 2006. Lecture notes in computer science, vol 4163. Springer, Berlin, pp 349–361
- Newborough J, Stepney S (2005) A generic framework for population-based algorithms, implemented on multiple FPGAs. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 43–55
- Nicosia G, Cutello V, Bentley PJ, Timmis J (eds) (2004) ICARIS 2004: 3rd international conference on artificial immune systems, Catania, Italy, September 2004. Lecture notes in computer science, vol 3239. Springer, Berlin <http://www.artificial-immune-systems.org/icaris.shtml>
- Owens ND, Timmis J, Greensted AJ, Tyrell AM (2007) On immune inspired homeostasis for electronic systems. In: de Castro CN, Von Zuben FJ, Kniel H (eds) ICARIS 2007: 6th international conference on artificial immune systems, Santos, Brazil, August 2007. Lecture notes in computer science, vol 4628. Springer, Berlin, pp 216–227
- Owens NDL, Timmis J, Greensted A, Tyrrell A (2008) Modelling the tunability of early t cell signalling events. In: Bentley PJ, Lee D, Jung S (eds) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 5132. Springer, New York, pp 12–23
- Perelson AS, Oster GF (1979) Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self-non-self discrimination. *J Theor Biol* 81(4):645–670
- Perelson AS, Weisbuch G (1997) Immunology for physicists. *Rev Mod Phys* 69(4):1219–1267
- Read M, Timmis J, Andrews PS (2008) Empirical investigation of an artificial cytokine network. In: Bentley PJ, Lee D, Jung S (eds) ICARIS 2008: 7th international conference on artificial immune systems, Phuket, Thailand, August 2008. Lecture notes in computer science, vol 5132. Springer, New York, pp 340–351
- Secker A, Freitas A (2007) WAIRS: Improving classification accuracy by weighting attributes in the AIRS classifier. In: Proceedings of the congress on evolutionary computation, Singapore, September 2007. IEEE Press, Singapore, pp 3759–3765
- Secker A, Freitas A, Timmis J (2003) A danger theory inspired approach to web mining. In: Timmis J, Bentley PJ, Hart E (eds) ICARIS 2003: 2nd international conference on artificial immune systems, Edinburgh, Scotland, September 2003. Lecture notes in computer science, vol 2787. Springer, New York, pp 156–167
- Stepney S (2007) Embodiment. In: Flower D, Timmis J (eds) *In silico immunology*. Springer, New York, Chap 12
- Stepney S, Smith RE, Timmis J, Tyrrell AM, Neal MJ, Hone ANW (2005) Conceptual frameworks for artificial immune systems. *Int J Unconventional Comput* 1(3):315–338
- Stibor T, Mohr P, Timmis J, Eckert C (2005) Is negative selection appropriate for anomaly detection? In: GECCO '05: Proceedings of the 2005 conference on genetic and evolutionary computation, Washington, DC, June 2005. ACM, New York, pp 321–328. doi: <http://doi.acm.org/10.1145/1068009.1068061>

- Timmis J (2007) Artificial immune systems – today and tomorrow. *Nat Comput* 6(1):1–18
- Timmis J, Andrews PS (2007) A beginners guide to artificial immune systems. In: Flower D, Timmis J (eds) *In silico immunology*. Springer, New York, Chap 3 (2007)
- Timmis J, Bentley PJ (eds) (2002) ICARIS 2002: Proceedings of the 1st international conference on artificial immune systems. University of Kent Printing Unit, Canterbury, UK, September 2002
- Timmis J, Neal MJ (2000) A resource limited artificial immune system for data analysis. In: Proceedings of ES2000 – Research and development in intelligent systems XVII, Cambridge, UK, December 2000. URL <http://www.cs.kent.ac.uk/pubs/2000/1121>, pp 19–32
- Timmis J, Bentley P, Hart E (eds) (2003) ICARIS 2003: 2nd international conference on artificial immune systems, Edinburgh, Scotland, September 2003. Lecture notes in computer science, vol 2787. Springer, New York <http://www.artificial-immune-systems.org/icaris.shtml>
- Timmis J, Amos M, Banzhaf W, Tyrrell A (2006) “Going back to our roots”: second generation biocomputing. *Int J Unconventional Comput* 2(4):349–382
- Timmis J, Andrews P, Owens N, Clark E (2008a) An interdisciplinary perspectives on artificial immune systems. *Evol Intell* 1(1):5–26
- Timmis J, Hone A, Stibor T, Clark E (2008b) Theoretical advances in artificial immune systems. *J Theor Comput Sci.* doi: 10.1016/j.tcs.2008.02.011
- Twycross J, Aickelin U (2005) Towards a conceptual framework for innate immunity. In: Jacob C, Pilat ML, Bentley PJ, Timmis J (eds) ICARIS 2005: 4th international conference on artificial immune systems, Banff, Canada, April 2005. Lecture notes in computer science, vol 3627. Springer, Heidelberg, pp 112–125
- Twycross J, Aickelin U (2006) Libtissue – implementing innate immunity. In: IEEE congress on evolutionary computation, Vancouver, Canada, July 2006. pp 499–506
- Watkins A, Timmis J, Boggess L (2004) Artificial immune recognition system (AIRS): an immune-inspired supervised machine learning algorithm. *Genet Programming Evolvable Mach* 5(3):291–317. URL citeseer.ist.psu.edu/watkins04artificial.html
- Weisbuch G, Atlan H (1988) Control of the immune response. *J Phys A Math Gen* 21(3):189–192
- Wilson WO, Garrett SM (2004) Modelling immune memory for prediction and computation. In: Nicosia G, Cutello V, Bentley PJ, Timmis J (eds) ICARIS 2004: 3rd international conference on artificial immune systems, Catania, Italy, September 2004. Lecture notes in computer science, vol 3239. Springer, Berlin, pp 386–399

48 Swarm Intelligence

David W. Corne¹ · Alan Reynolds² · Eric Bonabeau³

¹School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
dwcorne@macs.hw.ac.uk

²School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
a.reynolds@hw.ac.uk

³Icosystem Corporation, Cambridge, MA, USA
eric@icosystem.com

1	<i>Introduction</i>	1600
2	<i>Inspiration from Nature</i>	1601
3	<i>Two Main Concepts for Swarm Intelligence Algorithms</i>	1613
4	<i>Current Trends and Concluding Notes</i>	1618

Abstract

Increasing numbers of books, websites, and articles are devoted to the concept of “swarm intelligence.” Meanwhile, a perhaps confusing variety of computational techniques are seen to be associated with this term, such as “agents,” “emergence,” “boids,” “ant colony optimization,” and so forth. In this chapter, we attempt to clarify the concept of swarm intelligence and its associations, and to provide a perspective on its inspirations, history, and current state. We focus on the most popular and successful algorithms that are associated with swarm intelligence, namely, ant colony optimization, particle swarm optimization, and (more recently) foraging algorithms, and we cover the sources of natural inspiration with these foci in mind. We then round off the chapter with a brief review of current trends.

1 Introduction

Nature provides inspiration to computer scientists in many ways. One source of such inspiration is the way in which natural organisms behave when they are in groups. Consider a swarm of ants, a swarm of bees, a colony of bacteria, or a flock of starlings. In these cases and in many more, biologists are of the opinion (and as we have often seen for ourselves) that the group of individuals itself exhibits a behavior that the individual members do not, or cannot. In other words, if the group itself is considered as an individual – the *swarm* – in some ways, at least, it seems to be more intelligent than any of the individuals within it.

This observation is the seed for a cloud of concepts and algorithms, some of which have become associated with swarm intelligence. Indeed, it turns out that swarm intelligence is only closely associated with a small portion of this cloud. If one searches nature for scenarios in which a collection of agents exhibit behavior that the individuals do not (or cannot), it is easy to find entire and vast subareas of science, especially in the biosciences. For example, any biological organism seems to exemplify this concept, when the individual organism is considered as the “swarm” and its cellular components as the agents.

One might consider brains, and nervous systems, in general, as a supreme exemplar of this concept, when individual neurons are considered as the agents. Or one might zoom in on certain inhomogeneous sets of biomolecules as our “agents,” and herald gene transcription, say, as an example of swarm behavior. Fortunately, for the sake of this chapter’s brevity and depth, it turns out that the swarm intelligence literature has come to refer to a small and rather specific set of observations and associated algorithms. This is not to say that computer scientists are uninspired by the totality of nature’s wonders that exhibit such “more than the sum of the parts” behavior – much of this volume makes it clear that this is not so at all. However, if the focus is on the specific concept of swarm intelligence and the attempt to define it intentionally, the result might be a useful behavior that emerges from the cooperative efforts of a group of individual agents in which

1. The individual agents are largely homogeneous
2. The individual agents act asynchronously in parallel
3. There is little or no centralized control
4. Communication between agents is largely effected by some form of stigmergy
5. The “useful behavior” is relatively simple (finding a good place for food, or building a nest – not writing a symphony, or surviving for many years in a dynamic environment)

So, swarm intelligence is not about how collections of cells yield brains (which falls foul of at least items 1, 4, and 5), and it is not about how individuals form civilizations (violating mainly items 4 and 5), and it is not about such things as the life cycle of the slime mould (item 5). However, it is about individuals cooperating (knowingly or not) to achieve a definite goal, such as, ants finding the shortest path between their nest and a good source of food, or bees finding the best sources of nectar within the range of their hive. These and similar natural processes have led directly to families of algorithms that have proved to be very substantial contributions to the sciences of computational optimization and machine learning.

So, originally inspired, respectively, by certain natural behaviors of swarms of ants, and flocks of birds, the backbone of swarm intelligence research is built mainly upon two families of algorithms: ant colony optimization, and particle swarm optimization. Seminal works on ant colony optimization were by Dorigo et al. (1991) and Colorni et al. (1992a, b), and particle swarm optimization harks back to Kennedy and Eberhart (1995). More recently, alternative inspirations have led to new algorithms that are becoming accepted under the swarm intelligence umbrella; among these are search strategies inspired by bee swarm behavior, bacterial foraging, and the way in which ants manage to cluster and sort items. Notably, this latter behavior is explored algorithmically in a subfield known as swarm robotics. Meanwhile, the way in which insect colonies collectively build complex and functional constructions is a very intriguing study that continues to be carried out in the swarm intelligence arena. Finally, another field that is often considered in the swarm intelligence community is the synchronized movement of swarms, in particular, the problem of defining simple rules for individual behavior that led to realistic and natural behavior in a simulated swarm. “Reynolds’ rules” provided a general solution to this problem in 1987, and this can be considered an early triumph for swarm intelligence, which has been exploited much in the film and entertainment industries.

In the remainder of this chapter we expand on each of these matters. ➤ [Section 2](#) gives an account of the natural behaviors that have inspired the main swarm intelligence algorithms. ➤ [Section 3](#) then discusses the more prominent algorithms that have been inspired by the techniques in ➤ [Sect. 2](#), and ➤ [Sect. 4](#) notes some current trends and developments and offers some concluding remarks.

2 Inspiration from Nature

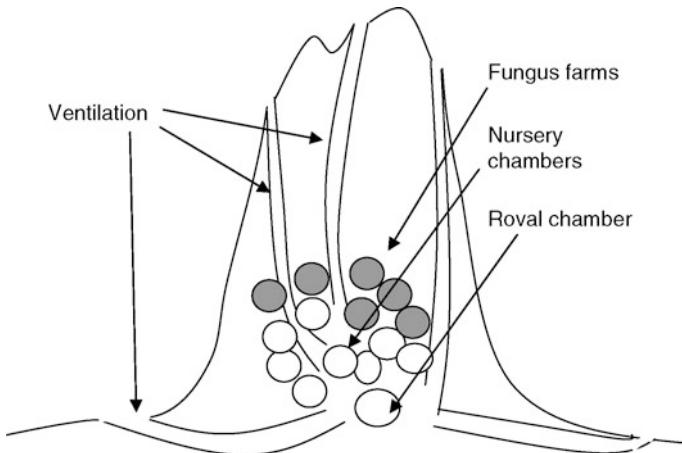
2.1 Social Insects and Stigmergy

Ants, termites, and bees, among many other insect species, are known to have a complex social structure. Swarm behavior is one of several emergent properties of colonies of such so-called *social insects*. A ubiquitous characteristic that is seen again and again in such scenarios is *stigmergy*. Stigmergy is the name for the *indirect* communication that seems to underpin cooperation among social insects (as well as between cells, or between arbitrary entities, so long as the communication is *indirect*).

The term was introduced by Pierre-Paul Grassé in the late 1950s (Grassé 1959). Quite simply, stigmergy means communication via signs or cues placed in the environment by one entity, which affect the behavior of other entities who encounter them. Stigmergy was originally defined by Grassé in his research on the construction of termite nests. ➤ [Figure 1](#) shows a simplified schematic of a termite nest. We will say more about termite nests in

Fig. 1

A highly simplified schematic of a termite nest.



➤ Sect. 2.1.3, but for now it suffices to point out that these can be huge structures, several meters high, constructed largely from mud and from the saliva of termite workers. Naturally, the complexity and functionality of the structure is quite astounding, given what we understand to be the cognitive capabilities of a single termite.

Following several field trips to Africa in the late 1930s and 1940s studying termites and their nests, among other things, Grassé showed that the regulation and the coordination of the nest-building activity did not depend on the termite workers themselves, but was instead achieved by the nest itself. That is, some kind of stimulating configuration of materials triggers a response in a termite worker, where that response transforms the configuration into another configuration that may, in turn, trigger yet another, possibly different, action performed by the same termite or by any other termite worker in the colony. This concept of stigmergy was attractive and stimulating, but at the time, and often today, it was and is often overlooked by students of social insects, because it leaves open the important operational issue of how the specific trigger-response configurations and stimuli must be organized in time and space to allow appropriate coordination. But despite the general vagueness of Grassé's formulation, stigmergy is recognized as a very profound concept, the consequences of which are still to be fully explored. Stigmergy is not only of potential importance for our understanding of the evolution and maintenance of social behavior in animals, from communally breeding species to highly social insects, but it is also turning out to be a crucial concept in other fields, such as artificial intelligence, robotics, or the social, political, and economic sciences. Meanwhile, in the arena of natural computing, stigmergy is the fundamental concept behind one of the main swarm intelligence algorithms, as well as several others.

Apart from termite nests, another exemplary case of stigmergy in nature is that of pheromone deposition. Ants deposit pheromone along their paths as they travel; an ant striking out on its own will detect pheromone trails, and prefer to follow such trails already travelled. In general, the concept of stigmergy captures underlying commonalities in (usually) insect behaviors that are underpinned by indirect communication. This covers more emergent behaviors than trail following, and (the original inspiration for the term) the construction of

structures such as termite mounds and bee hives. Stigmergy also seems key to behaviors such as brood sorting and cemetery clustering – some ant species are known to spatially cluster their young into age-groups within the nest, and they keep their nests tidy by removing dead nest mates and piling them into clusters outside.

The phenomenon of stigmergy has early evolutionary roots; it is now used to explain the morphologies of multicellular organisms, seashells, and so forth. Essentially, individual cells position themselves in a way influenced by deposits left behind by their colleagues or precursors. A useful way to think of it is that stigmergic communication involves a “stigmergy structure,” which is like a notepad, or an actual structure, built from cues left by individuals.

The structure itself may be a spatially distributed accumulation of pheromone, or a partially built hive, or a partially constructed extracellular matrix. The structure itself influences the behaviors of the individuals that “read” it, and these individuals usually also add to the structure. Army ants find their directions of travel influenced by pheromone trails, and they add to the trails themselves. Termites are triggered by particular patterns that they see locally in the partially built mound, and act in simple and specific ways, resulting in additions to the structure itself. An authoritative overview of stigmergy-associated behaviors in nature is by Bonabeau et al. (1997), while Theraulaz (1994) provides a comprehensive survey of self-organization processes in insect colonies. As hinted above, when the stigmergic processes often observed in nature are considered, the most prominent sources of inspiration from the swarm intelligence viewpoint are those of navigation to food sources, sorting/clustering, and the collective building of structures. Each of these processes are briefly considered next.

2.1.1 Natural Navigation

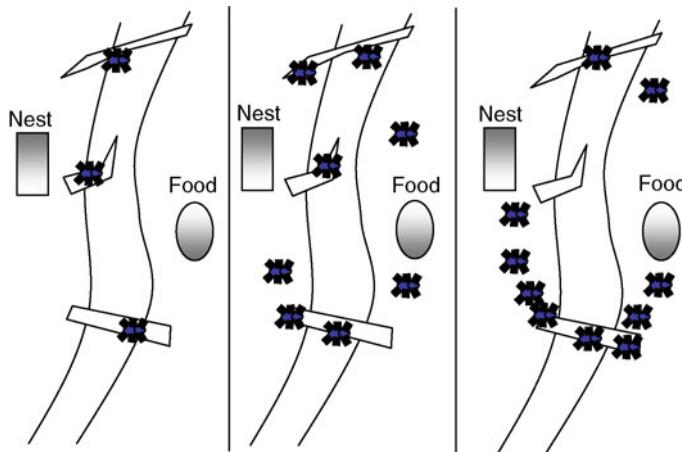
Navigation to food sources seems to depend on the deposition of pheromone by individual ants. In the natural environment, the initial behavior of a colony of ants in seeking a new food source is for individual ants to wander randomly. When an ant happens to find a suitable food source it will return to its colony; throughout, the individual ants have been laying pheromone trails. Subsequent ants setting out to seek food will sense the pheromone laid down by their precursors, and this will influence the path they take up. Over time, of course, pheromone trails evaporate. However, consider what happens in the case of a particularly close food source (or, alternatively, a faster or safer route to a food source). The first ant to find this source will return relatively quickly. Other ants that take this route will also return relatively quickly, so that the best routes will enjoy a greater frequency of pheromone laying over time, becoming strongly fancied by other ants. The overall collective behavior amounts to finding the best path to a nearby food source, and there is enough stochasticity in the process to avoid convergence to poor local optima – trail evaporation ensures that suboptimal paths discovered earlier are not converged upon too quickly, while individual ants maintain stochasticity in their choices, being influenced by but not enslaved by the strongest pheromone trail they sense.

Figure 2 shows a simple illustration, indicating how ants will converge via stigmergy toward a safer and faster way to cross a flow of water between their nest and a food source.

Figure 2 shows three contrived snapshots of a simple scenario over time. On the left, ants need to cross a narrow stream of flowing water toward a tempting food source, and three crossings – fallen twigs – present themselves. Initially, ants are equally likely to try each one. Each of these ants lays its trail of pheromone as it makes its journey toward the right. Over time, however, the path toward the middle twig becomes less laid with pheromone, simply

Fig. 2

Convergence to a safer crossing over time.



because, unlike the situation with the upper and lower twigs, there are no ants laying pheromone *on their way back* from that particular path. Eventually, on the right, we see several ants following the path defined by the lower twig, both attracted by, and further multiplying, the steady buildup of pheromone on this path, which is faster than that on the path defined by the upper twig. Although the upper twig provides a fairly short journey, it is more perilous, since it is quite narrow, and several ants fall off before being able to strengthen the path.

Actually, this is one of the simplest and most straightforward activities in social insects related to pheromones. The term “pheromone” itself was promoted for this context in 1959 (the late 1950s was clearly a fruitful period for swarm intelligence vocabulary) (Karlson and Lüscher 1959), to encompass the broad range of biologically active chemicals used by insects for varieties of communicative purposes. The context in which we describe it above is known in biology as “recruitment,” referring broadly to tasks in which individuals discover an opportunity (usually a food source) and need to recruit others to help exploit it. However, there are many other behaviors that are associated with pheromones, such as indicating alarms or warnings, interactions between queens and workers, and mating. An excellent source of further information is Vander Meer et al. (1998b), and in particular Vander Meer et al. (1998a).

2.1.2 Natural Clustering

Turning now to a different style of swarm behavior, it is well known that ant species (as well as other insect species) exhibit emergent sorting and clustering of objects. Two of the most well-known examples are the clustering of corpses of dead ants into cemeteries (achieved by the species *Lasius niger*, *Pheidole pallidula*, and *Messor sancta* (Depickere et al. 2004)), and the arrangement of larvae into similar age groups (so-called brood-sorting, achieved by *Leptothorax unifasciatus* (Franks and Sendova-Franks 1992)).

For example, in *L. unifasciatus* ant colonies, the ants' young are organized into concentric rings called annuli. Each ring comprises young of a different type. The youngest (eggs, and micro-larvae) are clustered together in the central cluster. As one moves outward from the center, progressively older groups of larvae are encountered. Also, the spaces between these rings increase as one moves outward.

In cemetery formation, certain ant species are known to cluster their dead into piles, with individual piles maintained at least a minimal distance from each other. In this way, corpses are removed from the living surroundings, and cease to be a hindrance to the colony. One aspect of this behavior in particular is that it is arguably not exemplary of swarm behavior; that is, it is perhaps not *collective* intelligence. The explanatory model that seems intuitively correct and confirmed by observation (Theraulaz et al. 2002) is one in which an individual operates according to quite simple rules while otherwise generally wandering around randomly:

1. If not carrying a corpse, and a single corpse (or quite small cluster of corpses) is encountered, pick it up.
2. If carrying a corpse and a relatively large cluster of corpses is encountered, put it down.

A single ant could achieve the clustering observed in nature that seems to operate according to these rules (Theraulaz et al. 2002). However, the emergent clusters are produced faster when a collection of ants are involved. Gaubert et al. (2007) is a useful reference for discussion of mathematical and other models of these behaviors. Meanwhile, a steady line of recent research is investigating computational clustering methods that are directly inspired by these natural phenomena (Handl and Meyer 2007). The first such inspired clustering algorithm was proposed by Lumer and Faieta (1994), closely based on the Deneubourg et al. (1991) model of the natural process. In recent work (Handl et al. 2006), an ant-based clustering method called ATTA is tested, and the case is made convincingly that ant-based clustering algorithms certainly have a niche in data mining, performing particularly well on problems where the number of clusters are not known in advance, and where the clusters themselves are highly variable in size and shape. In this article, our focus stays with optimization and we will say a little more about clustering; the reader is again referred to a review by Handl and Meyer (2007) for further study on this topic.

2.1.3 Natural Construction

We now consider the extraordinary collective behavior that leads to the construction of achievements such as wasp nests, termite mounds, and bee hives. Brood-sorting, considered above, exemplifies a simple structure that arises from collective behavior. However, the more visible and impressive structures such as termite mounds have always impressed observers, and often confounded one when one tries to imagine how such simple minds can lead to such creations. As will be clear from context, stigmergy seems to be the key to understanding these buildings; patterns inherent in partial elements of structures are thought to trigger simple rule-based behavior in the insect, which in turn changes the perceived patterns, and so on, until a complete hive or nest is built. Much computation-based study has been made of this by Bonabeau, Theraulaz, and coworkers.

In nature, the sizes of such social insect structures can reach an astounding 30 m in diameter (Grassé 1984). An impressive example of the complexity of these structures comes from the African termites *Macrotermes*, “the fungus growers.” In a mature nest of this

species, there are typically seven distinct elements of structure (this description is adapted from Bonabeau et al. 1998):

1. The protective and ribbed cone-shaped outer walls (also featuring ventilation ducts).
2. Brood chambers within the central “hive” area. They have a laminar structure and contain the nurseries where the young termites are raised.
3. The hive consists of thin horizontal lamellae supported by pillars.
4. A flat floor structure, sometimes exhibiting cooling vents in a spiral formation.
5. The royal chamber: a thick walled enclosure for the queen, with small holes in the walls to allow workers in and out. This is usually well protected underneath the hive structure, and is where the queen lays her eggs.
6. Garden areas dedicated to cultivating fungi. These are arranged around the hive, and have a comb-like structure, arranged between the central hive and the outer walls.
7. Tunnels and galleries constructed both above and below ground which provide pathways from the termite mound to the colony’s known foraging sites.

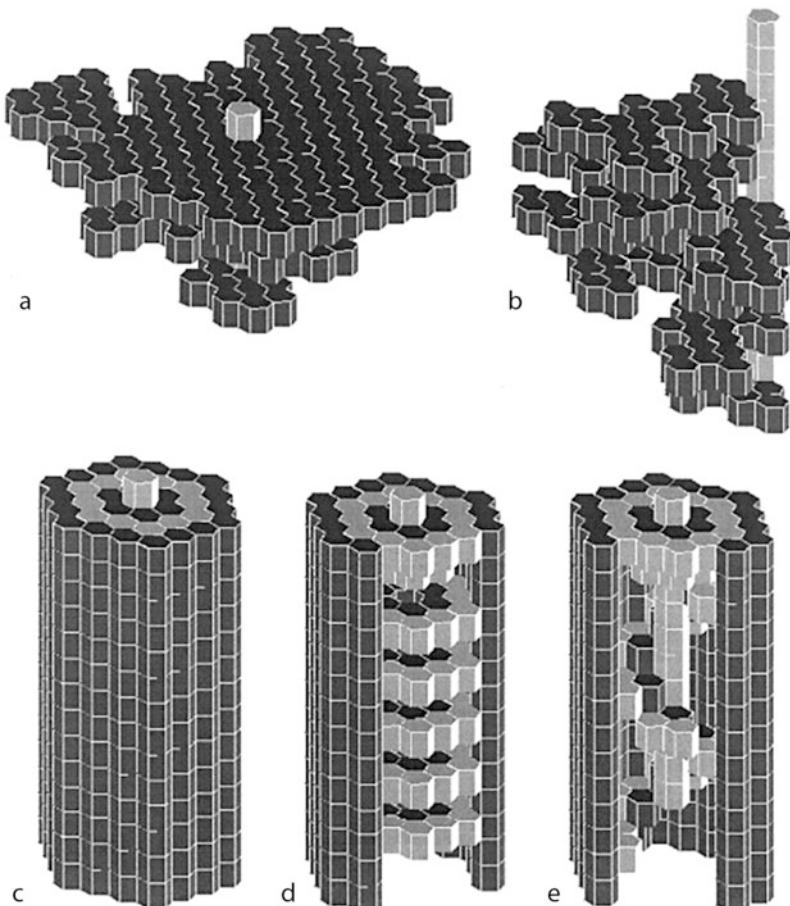
So, how does a collection of termites make such a structure? Perhaps this is the most astonishing example of natural swarm intelligence at work. The observations and descriptions of such structures from the biology literature have tended to focus on *description*, elucidating further and finer details from a variety of species, but have done little to clarify the mechanism. However, computational simulation work, such as Theraulaz and Bonabeau (1995), has indicated how such behavior can emerge from collections of “micro-rules,” where patterns of the growing structure perceived by an individual termite (or ant, wasp, etc.) act as a stigmergic trigger, perhaps in tandem with other environmental influences, leading to a specific response that adds a little new structure. Theraulaz and Bonabeau (1995) and Bonabeau et al. (2000) have shown how specific collections of such micro-rules can lead to, in simulation, a variety of emergent structures, each of which seems convincingly similar to wasp nests from specific wasp species. ↗ *Figure 3* shows examples of three artificial nests, constructed in this fashion, that closely resemble the nest structures of three specific wasp species; several more are presented in Theraulaz and Bonabeau (1995) and in Bonabeau et al. (2000).

↗ *Figure 4* shows some examples of micro-rules of the kind that can lead to the types of structures shown in ↗ *Fig. 3*. A micro-rule simply describes a three-dimensional pattern of “bricks”; in the case of the experiments that led to ↗ *Fig. 3*, a brick has a hexagonal horizontal cross section, and there are two types of bricks (it was observed that at least two different brick types seem necessary for interesting results). The three-dimensional pattern describes the immediate neighborhood of a single central brick, including the seven hexagonal cells above it (the upper patches of hexagons in ↗ *Fig. 4*), the six that surround it, and the seven below it. In the figure, a “white” hexagonal cell is empty – meaning no brick here; otherwise there are two kinds of bricks, distinguished by different shadings in the figure. A micro-rule expresses the following building instruction: “if the substructure defined by this pattern is found, with the central cell empty, then add the indicated type of brick in the central position.”

Intuition suggests how the construction by a collection of agents such as wasps of artifacts such as those in ↗ *Fig. 3*, or even more complex artifacts, may be facilitated by specific collections of micro-rules; however, that does not make it easy to design a set of micro-rules for a specific target construction. Sets of micro-rules achieving the illustrated results were obtained by using a carefully designed evolutionary algorithm. Interested readers should

Fig. 3

Figure 2 from Bonabeau et al. (2000), reproduced with permission. These show results from artificial colonies of ten wasps, operating under the influence of stimulus-response micro-rules based on patterns in a 3D hexagonal brick lattice. (a) A nest-like architecture resembling the nests of *Vespa* wasps, obtained after 20,000 building steps; (b) an architecture resembling the nests of *Parachartergus* wasps, obtained after 20,000 building steps; (c) resembling *Chatergus* nests, obtained after 100,000 building steps; (d, e) showing internal structure of (c).

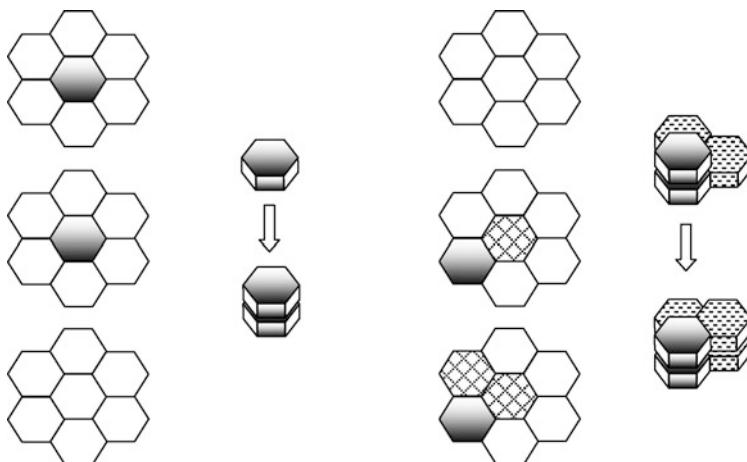


consult Bonabeau et al. (2000) for further details, including analyses of the operation of the emerging rule sets, revealing the requirement for various types of coordination implicitly built into the micro-rule collection.

Meanwhile, it is a long way from wasp nests to termite mounds, especially mounds of the complexity hinted at above. However, by considering and extending partial models for elements of termite mounds, in Bonabeau et al. (1998) some basis is provided for the suspicion that such complexity may be explained by the interaction of stimulus-response-based (“micro-rule”) processes and pheromone-based triggers that modify the stimulus-response behavior, unfolding over time as a controlled morphogenetic process.

Fig. 4

An illustration of two “micro-rules” from the space of such rules that can lead to structures such as those in Fig. 3. A single micro-rule defines a building instruction based on matching a pattern of existing bricks. A single column of three groups of seven hexagonal cells is a micro-rule, by describing a structure around the neighborhood of the central cell (which is empty in the matching pattern). The building instruction is to fill the central cell with the indicated type of “brick.” In this figure, two examples of micro-rules are shown, each further illustrated by “before” and “after” building patterns on the right.



Finally, we note that much of what was discussed in this subsection forms part of the basis for the new field of “swarm robotics.” This area of research (Sahin 2005; Mondada et al. 2005) focuses on what may be achievable by collections of small, simple robots furnished with relatively non-sophisticated ways to communicate. Chief among the motivating elements of such research are the qualities of robustness, flexibility, and scalability that swarm robots could bring to a number of tasks, ranging through agriculture, construction, exploration, and other fields. Imagine that one wishes to build a factory on Mars, for example. One might imagine this could be performed, in some possible future, by a relatively small collection of very sophisticated and intelligent robots. However, in the harsh conditions of Mars, one can expect that loss or destruction of one or more of these is quite likely. Swarms of simple robots, instead, are far more robust to loss and damage, and may present an altogether more manageable, shorter-term, and more adaptable approach than the “super-robot” style.

As yet, swarm robotics has not risen to such heights, although there is published discussion along these lines, drawing from the sources of natural inspiration already discussed (Cicirello and Smith 2001). Meanwhile, interesting and useful behaviors have been demonstrated in swarm robotics projects, after, in almost all cases, considerable work in design, engineering, and construction of the individual “bots.” Supporting these studies, a large part of swarm robotics research is into how to design the individual robots’ behaviors in such a way that the swarm (or team) achieves an overall goal or behavior. Unsurprisingly to many, it turns out that the judicious use of evolutionary computation proves effective for this difficult design problem (Waibel et al. 2009). However, hand-design or alternative principled methods for designing behaviors remains a backbone of this research, especially when the desired overall

behavior is complex, involving many tasks (e.g., Ducatelle et al. 2009), and in general there are several emerging issues in swarm robotics that have sparked active current lines of research, such as the problem of “interference” – swarm robots, whether cooperating on the same task or not, often physically interfere with each others’ operation (Pini et al. 2009) – and the problem of achieving tasks with minimal energy requirements (Roberts et al. 2008).

Relatively impressive behaviors from swarm robotics research has so far included cooperative transport of one or more objects, and cooperation toward moving up a vertical step (as large as the bots involved); readers may visit the European project “Swarmbots” and “Swarmanoids” websites for explanation and many other resources, at <http://www.swarmbots.org/>, and at <http://www.swarmanoid.org/> as well as similar resources such as <http://www.pherobot.com/>. Hardware and related technology issues remain a bottleneck that still inhibit a full exploration of social insect swarm intelligence in robotics; however, this work continues toward that end and will be observed with great interest.

2.2 Foraging

There are broadly two types of natural processes that go by the term “foraging,” and in turn provide sources of inspiration for optimization (or resource allocation) methods. In both cases, the overall emergent behavior is that the swarm finds and exploits good food sources, adaptively moves to good new sources as current ones become depleted, and does all of this with efficient expenditure of energy (as opposed to, e.g., brute force search of their environment). The means by which this behavior is achieved is rather different in these two sources of inspiration.

In one case, that of “bacterial foraging” (Passino 2002), individual bacteria are (essentially) directed toward rich areas via *chemotaxis*; that is, they exist in an environment in which their food source diffuses, so they can detect and respond to its presence. In particular, chemotaxis refers to movement along a chemical gradient. An individual *Escherichia coli* bacterium has helical appendages called *flagellae* which spin either clockwise or anticlockwise (one can think of them as analogous to propellers). When they spin in one direction, the bacterium will “tumble”; this is an operation which ends up moving the bacterium a short distance, and leaving it with an essentially random new orientation. When the flagellae spin in the other direction, the bacterium’s movement will be a “run” – this is a straight-line movement in the direction the bacterium was facing at the beginning, and continues as long as the flagellae continue to spin in the same direction.

In a nutrient-free and toxin-free environment, an individual bacterium will alternate between clockwise and anticlockwise movement of its flagellae. So, it tumbles, runs, tumbles, runs, and so forth. The effect of this behavior is a random search for nutrients. However, when the bacterium encounters an increasing nutrient gradient, that is, a higher concentration of nutrient in its direction of movement, its internal chemistry operates in a way that causes the runs to be of longer duration. It still alternates between tumbles and runs, but maintains longer run lengths so long as the gradient continues to increase. The effect of this is to explore and exploit the food source, moving upward along the nutrient gradient, while maintaining an element of stochastic exploration.

In addition, under certain conditions we know that bacteria secrete chemicals that attract each other. There is speculation that this can happen in response to nutrient-rich environments, so that additional bacteria are recruited to exploit the food source, where

the attractive secretions build further on the attraction provided by the chemical gradient. Also, there is evidence that bacteria release such an attractant under stressful conditions too, which in turn may be a protective response; as they swarm into a sizeable cluster many individuals are protected from the stressful agent. These self-attractant and chemotactic behaviors are known to lead to pattern formations under certain conditions (Budrene and Berg 1991). These and many other details have been elucidated for *E. coli* and similar bacteria via careful experimentation, for example: Berg and Brown (1972), Segall et al. (1986), and DeRosier (1998).

The other broad style of efficient collective foraging behavior is that exhibited by the honeybee (among other insects). When a bee discovers a food source some distance from the hive, it returns to the hive and *communicates the direction, distance, and quality* of the food source to other bees. The details of this communication, achieved by specialized “dances,” are quite remarkable, and have emerged from a series of ingenious experiments and observations, largely by Karl von Frisch (1967). The essential details are these: in context, the dance is performed in alignment with a particular aspect of the hive structure, which provides an absolute reference against which the bee audience can perceive specific angles. The main dance is the “waggle” dance, which consists of a straight-line movement, during which the bee waggles from side to side along the way. This straight-line movement is done upward at a particular angle from the vertical. At the end of the straight-line part, the bee loops round to the starting point and repeats the dance (actually, it alternates the direction of this loop, drawing a figure “8”). The angle of this dance from vertical indicates to the bee audience the direction they need to take with respect to the current position of the sun. Among various extraordinary aspects of this, it is known that the bee automatically corrects for movement of the sun during the day, and communicates the correct direction. Also, at times, the bee will pause its dance and allow watching bees to sample the nectar it is carrying, giving an indication of the quality of the food source.

More interesting from the algorithmic viewpoint, however, is that the abundance of the food source is communicated by the duration of the dance (essentially, the number of times the figure “8” is repeated). An individual enjoying this performance may or may not decide to follow these directions and be “recruited” to this particular source. Such an individual may also be exposed to rival performances. However, the longer the duration, the more bees will see this dance, and the more will be recruited to this dance rather than others. In this way, the bee colony sports the emergent behavior of smart resource allocation, with more bees assigned to better sources, and adaptation over time as returning bees gradually provide shorter and shorter dances as the source becomes depleted.

As we will see later, both bee and bacterial foraging have been taken as the inspiration for general optimization methods, as well as for approaches to the specific problem domain of optimal resource allocation.

2.3 Flocking

Perhaps the most visible phenomenon that brings to mind swarm intelligence is the travelling behavior of groups (flocks, swarms, herds, etc.) of individuals that all are familiar with. The mesmerizing behavior of large flocks of starlings is a common morning sight over river estuaries. Swarms of billions of monarch butterflies, herds of wildebeest, schools of tuna fish, swarms of bees, all share common emergent behaviors, chiefly being:

1. The individuals stay close to each other, but not too close, and there seem to be no collisions.
2. Swarms change direction smoothly, as if the swarm was a single organism.
3. *Unlike* a single organism, yet still smoothly and cleanly, swarms sometimes pass directly through narrow obstacles (in the way that a stream of water passes around a vertical stick placed centrally in the stream's path).

In some ways, such swarm behavior is arguably less mysterious than other emergent behaviors; it seems clear that we might be able to explain this behavior via a built-in predisposition for individuals to stay with their colleagues, and we can readily imagine how evolution will have favored such behavior: there is safety in numbers. However, the devil is in the detail, and it took seminal work by Reynolds (1987) to outline and demonstrate convincing mechanisms that can explain these behaviors. Reynolds' work was within the computer graphics community, and has had a volcanic impact there. Now known as "Reynolds' rules," the recipe that achieves realistic swarm behavior (with some, but not obtrusively much, parameter investigation needed depending on the species simulated) is this triplet of steering behaviors to be followed by each individual in a swarm:

Separation: steer to avoid coming too close to others.

Alignment: steer toward the mean heading of others.

Cohesion: steer toward the mean *position* of others.

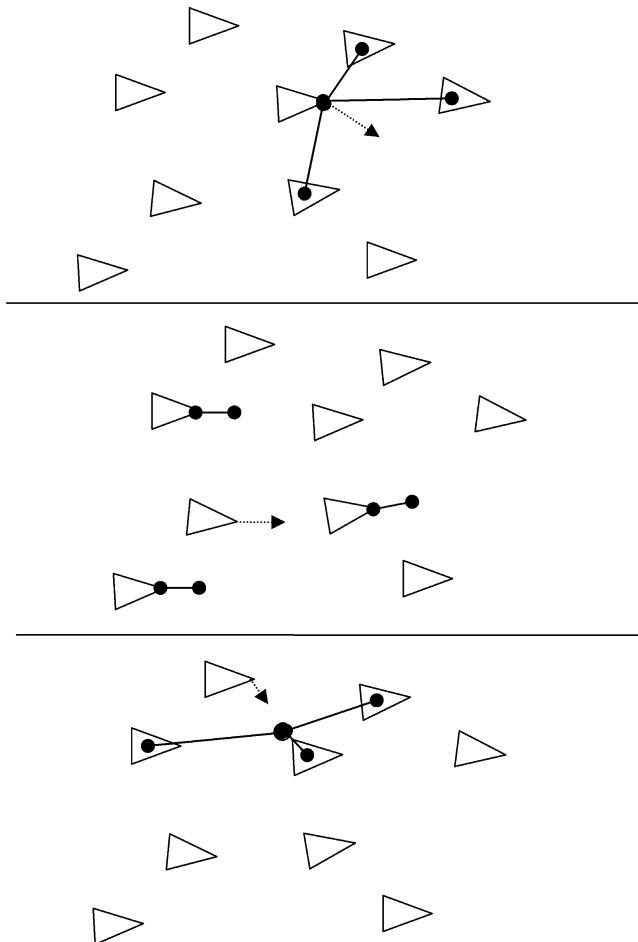
A basic illustration of each rule is given in Fig. 5. In the figure, the common terminology of "boid" is taken to refer to an individual in a flock. The figure shows examples of the adjustments that might be made under the guidance of the rules. To understand how realistic swarm simulation works, it is important to note that each boid has its own perceptual field – that is, it could only "see" a certain distance, and had a specific field of view (e.g., boids cannot see behind them). The adjustments it makes to its velocity at any time are therefore a function of the positions and velocities of the boids in its perceptual field, rather than a function of the flock as a whole.

The rules are key ingredients to a realistic appearance in simulated flocks, but there are several other details, particularly regarding obstacle avoidance and goal-seeking behavior. Interested readers may consult Reynolds (1987) and the many works that cite it. It is important to note that these rules are not strictly nature inspired, in the sense that Reynolds was not attempting to explain natural swarming behavior, he was simply attempting to emulate it. However, the resulting behavior was found to agree well with observations of natural flocking behavior (e.g., Partidge (1982) and Potts (1984)), and Reynolds (1987) reported that "many people who view these animated flocks immediately recognize them as a representation of a natural flock, and find them similarly delightful to watch." These techniques are now common in the film industry; among the earliest uses were in the film Batman Returns (1992, director Tim Burton), in which Reynolds' rules lay behind the simulated bat swarms and flocks of penguins.

Meanwhile, natural flocking behavior also turns out to be one of the sources of inspiration for the highly popular and successful particle swarm optimization algorithm, which appears in the next section as one of the prominent flagships for swarm intelligence. It is not obvious why flocking behavior might lead to an optimization algorithm; however, it soon becomes clear when the dynamics of flocking and the tendency of optimization landscapes to be locally smooth are considered. In the case of bacterial foraging, the dynamics of the natural behavior are such that individuals will tend to congregate around good areas. With the bacterial

Fig. 5

Illustrating Reynolds' rules, which lead to natural-looking behavior in simulated swarms. *Upper:* Separation: each boid makes an adjustment to velocity which prevents it from coming too close to the flockmates in its perceptual field. *Middle:* Alignment: a boid adjusts its heading toward the average of those in its perceptual field. *Lower:* a boid makes an adjustment to velocity that moves it toward the mean position of the flockmates in its perceptual field.



example, nature provides mechanisms for suggesting appropriate directions of movement, while there is a clear goal for the bacterial colony to achieve – find nutrient-rich (and toxin free) areas. For the current context in particular, the secretion of attractant chemicals is a mechanism that promotes bacteria swarming together, while an individual's position in its environment directly provides it with a level of “fitness” that it can sense in terms of nutrient concentration.

When the flocking behavior in birds is considered, however, Reynolds' work provides clues about appropriate ways to move together as a swarm, but there is no clear mirror of a “fitness” in the environment. Often, birds will migrate from A to B, knowing where they are going,

rather than seeking new environments. However, if flocks *did* have a goal to move toward “fitter” positions in the landscape they travel, then it becomes intuitively reasonable to consider the cohesive swarm behavior as a sensible way to achieve local exploration around fit areas, perhaps enabling the sensing of even fitter areas that may then sway the overall movement of the flock. In this way, flocking behavior combines with a little algorithm engineering to achieve a very successful optimization mechanism.

3 Two Main Concepts for Swarm Intelligence Algorithms

When we consider the impact of swarm intelligence so far on computer science, two families of algorithms clearly stand out in terms of the amount of work published, degree of current activity, and the overall impact on industry. One such family is inspired directly by the pheromone-trail following behavior of ant species, and this field is known as ant colony optimization (ACO). The other such family is inspired by flocking and swarming behavior, and the main exemplar algorithm family is known as particle swarm optimization (PSO). Also in this family are algorithms based on bacterial foraging, and some of the algorithms that are based on bee foraging; these share with PSO the broad way in which the natural phenomenon is mapped onto the concept of search within a landscape. In this section, these two main families will be discussed in turn.

3.1 Ant Colony Optimization

Ant colony optimization (ACO) was introduced in 1996 via an algorithm called “Ant System” (AS) (Dorigo et al. 1996). The basic approach used in AS remains highly characteristic of most ACO methods in current use, as we will describe next.

Recall that, in the natural case, an ant finds a path from its nest to a food source by following the influences of pheromone trials laid down by ants that have previously sought food (and usually returned). AS, and ACO algorithms in general, mirror aspects of this behavior quite faithfully. In short, an artificial ant builds a solution to the optimization problem at hand, and lays down simple “artificial pheromone” along the route it took toward that solution. Following artificial ants then build solutions of their own, but are influenced by the pheromone trails left behind by their precursors. This is the essential idea, and starts to indicate the mapping from the natural to the artificial case. However, there are various further issues necessary to consider making this an effective optimization algorithm. This is discussed further in the next section, with a focus on the mapping from the basic ideas of ACO to applications in optimization.

3.1.1 Applying ACO to Optimization Problems

In order to apply AS to an optimization problem, the problem needs to be represented in such a way that any candidate solution to it is a specific path along a network. This network can be conceived as having a single start node, from which (usually) every ant starts, and a single finish node, reaching which indicates that the path taken encodes a complete solution. A clear example might be the network of roads in a city, where each junction of roads is a node in the

network, and each road is an arc or edge between nodes. Consider the problem of finding the shortest path between a specific junction A and another specific junction B. In this case, A and B are clearly the start and finish nodes, and one can imagine an ACO approach which maps very closely indeed to the natural case. However, with a little thought, it is clear that each individual ant's path construction should be constrained so that it does not return to a junction it has already visited (unless this is a valid move for the problem under consideration). Also, though one might choose to simulate the preferential recruitment of new ants to shorter paths by closely following the natural case, it seems more sensible and straightforward to make the pheromone trail strength directly a function of the solution quality. That is, when an ant has completed its path, one evaluates the quality of its solution, and renders things such that better solutions lead to stronger pheromone deposits along its arcs. These pheromone deposits will decay over time; however, just as in the natural case, we can see that this will prevent premature convergence to poor solutions that happen to be popular in the early stages.

Finally, since such information is often available, and would seem useful in cases where ants have large numbers of choices, one might bias the paths available at each junction with the aid of a simple heuristic evaluation of the potential of that arc. For a shortest path problem, for example, this could be based on how much closer to B each arc would leave the ant.

With such considerations in mind, one can envisage artificial ants travelling the road network from A to B via distinct but sensible routes. At each junction, the ant senses the pheromone levels that await it at each of the arcs it can feasibly take. These levels are made from many components; arcs that are highly attractive will probably enjoy the remnants of trails from prior ants that have reported good solutions, and/or may have a good heuristic component. Arcs with low pheromone levels will probably be losers in the heuristic stakes, and have seen little activity that has led to good solutions; however, the ant may still choose such an arc, since our algorithm is stochastic.

Finally, as an individual artificial ant arrives at B, it retrospectively lays pheromone on the path it took, where the strength of that pheromone trail will reflect the quality of its solution. The next artificial ant starts from A and sees a slightly updated pheromone trail (stigmergic) pattern, and so it continues.

To apply this method to other problems, one simply needs (implicitly, at least) a network-based representation of the problem as described. If one is solving the travelling salesperson problem (TSP), for example, the network is the complete graph of the cities, each arc between cities indicates the distance or cost of that arc, and in this case an individual ant can start anywhere. As one follows an individual ant's route, one sensibly constrains its potential next-hops to avoid cycles, and along the way one may bias its choices simply by using the distance of the next arc, and it retrospectively lays pheromone once it has completed a tour of all cities. In general, an optimization problem can always be approached in this way, by suitable choices of semantics for nodes and arcs, and well-designed routines for generating and constraining an ant's available choices at each junction.

This explication can now be finished by clarifying the AS algorithm, which in fact has already been covered verbosely in the above. Once the transformation of the problem has been designed, so that an ant's path through a network provides a candidate solution, the algorithm cycles through the following two steps until a suitable termination condition is reached:

Solution Construction: A number of ants individually construct solutions based on the current pheromone trail strengths (initially, pheromone is randomly distributed). Each ant steps through the network choosing among feasible paths. At each choice point, the ant chooses among available arcs according to a function of the pheromone strength on each

arc, and of the heuristic values of each arc. In the original, and still commonly used version of this function (see Dorigo et al. (1996) and many more), the pheromone and heuristic components for each arc are exponentiated to parameters α and β respectively, allowing for tuning of the level to which the algorithm relies on exploration and heuristic support. Also, the overall attractiveness value of each arc is scaled so that the ant can treat these values as a probability distribution over its available choices.

Pheromone Update: When the ants' paths are complete, for each ant, the corresponding solution is evaluated, and pheromone is laid on each arc travelled in proportion to the overall solution quality. Also, the component of pheromone strength that arises from earlier ants is reduced. Quite simply, for any particular arc, its updated pheromone strength p_{new} is $(1 - \rho)p_{\text{old}} + \rho f$, where ρ controls the speed of pheromone decay, and f accumulates the overall quality of solutions found which involved that arc in the current iteration.

ACO has now been applied to very many problems and (clearly, or we would probably not have devoted so much time to it) has been very successful, especially when hybridized with local search or some other metaheuristic (in such hybridized algorithms, an ant will typically use the additional heuristic for a short time to find an improvement locally to its solution). Initially demonstrated for the TSP (Dorigo et al. 1996), there are an enormous number of applications of ACO now published. We mention vehicle routing (Gambardella et al. 1999), rule discovery (Parpinelli et al. 2002), and protein/ligand docking (Korb et al. 2007) just to give some initial idea of the range of applications. To discover more, recent surveys include Blum (2005a) and Gutjahr (2007), the latter concentrating on theoretical analyses. Meanwhile, Socha and Dorigo (2008) show how to apply ACO to continuous domains (essentially, ants select parameters via a probability density function, rather than a discrete distribution over a fixed set of arcs).

3.1.2 Ant-Based Routing in Telecommunications

The basic ACO idea of exploiting pheromone-trail-based recruitment is also the inspiration for a healthy subarea of research in communications networks; therein, ant-inspired algorithms are designed to assist with network routing and other network tasks, leading to systems that combine high performance with a high level of robustness, able to adapt with current network traffic and robust to network damage. Early and prominent studies in this line were by Schoonderwoerd et al. (1996, 1997), which were soon built upon by di Caro and Dorigo's AntNet (1998). To explain this application area, and the way that ACO ideas are applied therein, it will be helpful to first explore the problem and the associated solution that was studied in Schoonderwoerd et al.'s seminal work.

Schoonderwoerd et al. were concerned with load balancing in telecommunication networks. The task of a telecommunication network is to connect calls that can arise at any node, and which may need to be routed to any other node. The networks themselves, as a function of the capacities of the constituent equipment at each node, cannot guarantee successful call connections in all cases, but they aim to maintain overall acceptable performance under standard conditions. At very busy times, and/or if a particular node is overwhelmingly flooded with calls, then typically many calls will be "dropped." It is worth noting that there are problems with central control of such systems (landline telecommunication networks, mobile networks, traffic networks, and so forth). To achieve centralized control in a way that manages load-balancing or any other such target, several disadvantages are apparent. The controller

usually needs current knowledge about many aspects of the entire system, which in turn necessitates using communication links from every part of the system to the controller itself. Centrally managed control mechanisms therefore scale badly as a result of the rapidly increasing overheads of this communication, interfering with the performance of the system itself. Also, failure in the controller will often lead to complete failure of the complete system.

Schoonderwoerd et al.'s ant-inspired approach, which remains a central part of the majority of more recent ant-based approaches, was to replace the routing tables in such networks with so-called "pheromone tables." Networks of the type of interest invariably have a routing table at each node, specifying which "next-hop" neighboring node to pass an incoming call to, given the ultimate destination of that call. In Schoonderwoerd et al.'s "Ant-Based Control" (ABC) method, the routing table at a network node instead provided n probability distributions over its neighboring nodes, one for each of the n possible destinations in the network. When a routing decision is to be made, it is made stochastically according to these probabilities – that is, it is most likely that the next-hop with the highest probability will be taken, but there is a chance that the next-hop with the lowest probability will be taken instead. The entries in the pheromone table were considered analogous to pheromone trail strengths, and changed adaptively during the operation of the network. Updating of these trails in ABC is very simple – whenever a call is routed from node A to B, the entries for A in node B's routing table are all increased, with corresponding reductions to other entries. However, this simple idea has obvious intuitive benefits; first, by testing various routing decisions over time (rather than deterministic decisions), the process effectively monitors the current health of a wide variety of different routing strategies; when a link is over-used, or down, this naturally leads to diminution in its probability of use, since the associated entries in routing tables will not be updated, and hence will naturally reduce as alternatives are updated. Also, as it turns out, the pheromone levels can adapt quite quickly to changes in call patterns and loads. The ABC strategy turned out to be surprisingly effective, despite its simplicity, when Schoonderwoerd et al. compared it with a contemporary agent-based strategy developed by Appleby and Steward (1994), and found it superior over a wide range of different situations.

Research in ant-based approaches for decentralized management is increasingly very active (e.g., Hussein and Saadawi 2003; Rosati et al. 2008; Di Caro et al. 2008). The essential idea, to replace static built-in routing strategies with stochastic "pheromone tables" or similar, is applicable in almost all modern communication scenarios, ranging through ad hoc computer networks, mobile telephone networks, and various layers of the Internet. Ongoing research continues to explore alternative strategies for making the routing decisions, controlling the updates to pheromone trails, and so forth, while investigating various distinct application domains, and continuing to find competitive or better performance than alternative state-of-the-art methods used in network engineering.

3.2 Particle Swarm Optimization and Foraging

PSO was established in 1995 with Kennedy and Eberhart's paper in the international joint conference on neural networks (IJCNN) (Kennedy and Eberhart 1995). The paper described a rather simple algorithm (and time has seen no need to alter its straightforward fundamentals), citing Craig Reynolds' work as inspiration (Reynolds 1987), along with slightly later work in the modeling of bird flocks (Heppner and Grenander 1990). The basic idea is to unite the

following two notions: (a) the behavior of a flock of birds moving in 3D space toward some goal; (b) a swarm of solutions to an optimization problem, moving through the multidimensional search space toward good solutions.

Thus, a “particle” is equated with a candidate solution to an optimization problem. Such a particle has both a *position* and a *velocity*. Its position is, in fact, precisely the candidate solution it currently represents. Its velocity is a displacement vector in the search space, which (it hopes) will contribute toward a fruitful change in its position at the next iteration.

The heart of the classic PSO algorithm is in the step which calculates a new position for the particle based on three influences. The inspiration from Reynolds (1987) is clear, but the details are quite different, and, of course, exploit the fact that the particle is moving in a search space and can measure the “fitness” of any position. The influences – the components that lead to the updated position – are:

Current velocity: the particle’s current velocity (obviously).

Personal best: the particle remembers the fittest position it has yet encountered, called the personal best. A component of its updated velocity is the direction from its current position to the personal best.

Global best: every particle in the swarm is aware of the best position that any particle has yet discovered (i.e., the best of the personal bests). The final component of velocity update, shared by all particles, is a vector in the direction from its current position to this globally best-known position.

Following a random initialization of positions and velocities, the evaluation of the fitness of the particles’ current positions, and consequent initialization of the personal bests and global best, PSO proceeds in a remarkably straightforward manner. First, each particle updates its velocity by adding a vector in each of the above three component directions. To provide these vectors, in the classic algorithm, the current velocity component is left undisturbed, while the personal and global best components are each scaled by a random scalar drawn uniformly from [0,2]. The resulting vector is used to update the current velocity, and the new velocity vector is used to update the current position. The new position is evaluated, book-keeping is done to update personal and global bests, and then this is repeated.

Kennedy and Eberhart initially reported that PSO appeared to do very well over a wide range of test problems, including its use as an alternative to backpropagation for training an artificial neural network (Kennedy and Eberhart 1995). Perhaps helped by the ease of implementation of this algorithm (remarkably few lines of code are needed for the classic algorithm), an avalanche of papers began to follow, almost invariably adding to the evidence that this algorithm provides a very substantial contribution to optimization practice. Naturally, this field is now rich in variants and extensions to the original design – a number of recent surveys are available (e.g., Reyes-Sierra and Coello 2006; Yang et al. 2007) – while the published applications are as varied as one might expect from such a generally applicable algorithm.

3.2.1 Bacteria and Bees

Newer to the ranks of swarm-intelligence-based optimization, and yet to prove quite as widely successful, are techniques inspired by bacterial and bee foraging. For the most part, these algorithms follow the broad direction of PSO, in that individuals in a swarm represent solutions moving through a landscape, with the fitness of their current solution easily obtained by evaluating their position. Meanwhile, just as with PSO, an individual’s movement through

this landscape is influenced by the movements and discoveries of other individuals. The fine details of a bacterial foraging algorithm (BFA), however, are quite distinct, and in one of the more popular methods draw quite closely from what is known (and briefly touched upon above) about bacterial swarming in nature. Passino (2002) presents a fine and detailed explication of both the natural case and the BFA. It turns out that BFA-style algorithms are enjoying quite some success in recent application to a range of engineering problems (e.g., Niu et al. 2006; Tripathy and Mishra 2007; Guney and Basbug 2008).

Also inspiring, so far, a small following are algorithms that are inspired by bee foraging behavior. The authoritative sources for this are Quijano and Passino's papers, respectively, outlining the design and theory (Quijano and Passino 2007a), and application (Quijano and Passino 2007b) of a bee foraging algorithm. In Quijano and Passino (2007a) the design of a bee-foraging algorithm is presented in intimate connection with an elaboration of the mechanisms of natural bee foraging (such as briefly described earlier). The algorithm is as much a model of the natural process as it is a routine applicable to certain kinds of problem. Considering individual bees as resources, the concept here is to use bee foraging behavior as a way to ideally distribute those resources in the environment, and maintain an ideal distribution over time as it adapts to changing patterns of supply. Just as natural bees maintain an efficient distribution of individuals among the available sources of nectar, the idea is that this can be mapped to control problems which aim to maintain a distribution of resources (such as power or voltage) in such a way that some goal is maintained (such as even temperature or maximal efficiency). In Quijano and Passino (2007b), we see the algorithm tested successfully on a dynamic voltage-allocation problem, in which the task is to maintain a uniform and maximal temperature across an interconnected grid of temperature zones.

Finally, we note that bee-foraging behavior has also directly inspired techniques for internet search, again based on the notion of maintaining a maximally effective use of server resources, adapting appropriately and effectively to the relative richness of new discoveries (Walker 2000; Nakrani and Tovey 2003).

4 Current Trends and Concluding Notes

We have pointed to a number of survey papers and other works from which the reader can attain a full grasp of the current activity in swarm intelligence algorithms, but in this brief section we attempt a few notes that outline major current trends, and then we wrap up.

A notable trend in recent work on PSO, and indeed on metaheuristics in general, is toward the creation of hybrid algorithms. While themes from evolutionary computation continue to be incorporated in PSO (Shi et al. 2005), others have explored the idea of hybridization with less frequently used techniques such as scatter search and path relinking (Yin et al. 2007), immune system methods (Zhang and Wu 2008), and, indeed, ACO (Holden and Freitas 2007). Meanwhile, the range of problems to which PSO may be applied has been greatly increased with the development of multi-objective forms of PSO (Coello Coello et al. 2004).

Other work has involved the use of multiple swarms. This may allow each swarm to optimize a different section of the solution (van den Bergh and Engelbrecht 2004). Alternatively, each swarm may be configured differently to take advantage of the strengths of different PSO variants (e.g., Jordan et al. 2008), in an attempt to create a more reliable algorithm that can be applied to a wide range of problem domains.

The themes of multi-objective optimization and hybridization equally apply to recent research into ACO. While multi-objective ACO is a more mature field than multi-objective PSO (see, e.g., Mariano and Morales 1999), work continues in categorizing and comparing multi-objective approaches to ACO (e.g., García-Martínez et al. 2007), in creating generic frameworks for multi-objective ACO and in creating new multi-objective variants (e.g., Alaya 2007). Recent applications have seen single-objective ACO hybridized with genetic algorithms (Lee et al. 2008), beam search (Blum 2005b), and immune systems (Yuan et al. 2008) and multi-objective ACO used in combination with dynamic programming (Häckel et al. 2008) and integer linear programming (Doerner et al. 2006).

Other recent work has seen ACO adapted for use in continuous domains (Dreo and Siarry 2006; Socha and Dorigo 2008), while research continues into variations of ACO and new algorithm features, for example, different types of pheromone and the use of dominance rules to warn ants from searching among solutions known to be of low quality (Lin et al. 2008).

Recent work on bacterial foraging algorithms has concentrated on exploiting the effectiveness of the local search ability of the algorithm, while adapting it to improve the global search ability on high-dimensional and multimodal problems. With this aim, bacterial foraging has been hybridized with more effective global optimizers such as genetic algorithms (Chen et al. 2007; Kim et al. 2007) and PSO (Tang et al. 2007; Biswas et al. 2007).

In conclusion, we attempted to demystify the concept of swarm intelligence, and, after touring through the chief sources of natural inspiration, we distilled the essence of its impact and presence in computer science down to two major families of algorithms for optimization. No less intriguing and exciting additional topics in the swarm intelligence arena, that have also been discussed, are stigmergic construction, ant-based clustering, and swarm robotics. It is abundantly clear that the natural inspirations from swarming ants, bees, and birds (among others) have provided new ideas for optimization algorithms that have extended the state of the art in performance on many problems, sometimes with and sometimes without additional tailoring and hybridization. Ant-based clustering also seems to provide a valuable contribution, while swarm robotics, stigmergy-based construction, and a variety of other emerging subtopics have considerable promise, and will doubtless develop in directions rather difficult to foresee.

References

- Alaya I (2007) Ant colony optimization for multi-objective optimization problems. In: Proceedings of the 19th IEEE international conference on tools with artificial intelligence. Patras, Greece, pp 450–457
- Appleby S, Steward S (1994) Mobile software agents for control in telecommunications networks. *BT Technol J* 12(2):104–113
- Berg H, Brown D (1972) Chemotaxis in *Escherichia coli* analysed by three-dimensional tracking. *Nature* 239:500–504
- van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. *IEEE Trans Evol Comput* 8(3):225–239
- Biswas A, Dasgupta S, Das S, Abraham A (2007) Synergy of PSO and bacterial foraging optimization – A comparative study on numerical benchmarks. In: Corchado E et al. (eds) Innovations in hybrid intelligent systems. Advances in soft computing, vol 44. Springer, Germany, pp 255–263
- Blum C (2005a) Ant colony optimization: Introduction and recent trends. *Phys Life Rev* 2(4):353–373
- Blum C (2005b) Beam-ACO – hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Comp Oper Res* 32(6):1565–1591
- Bonabeau E, Guérin S, Snyers D, Kuntz P, Theraulaz G (2000) Three-dimensional architectures grown by simple ‘stigmergic’ agents. *Biosystems* 56:13–32
- Bonabeau E, Theraulaz G, Deneubourg J-L, Aron S, Camazine S (1997) Self-organization in social insects. *Trends Ecol Evol* 12(5):188–193

- Bonabeau E, Theraulaz G, Deneubourg J-L, Franks NR, Rafelsberger O, Joly J-L, Blanco S (1998) A model for the emergence of pillars, walls and royal chambers in termite nests. *Phil Trans Royal Soc B Biol Sci* 353(1375):1561–1576
- Budrene E, Berg H (1991) Complex patterns formed by motile cells of *Escherichia coli*. *Nature* 349: 630–633
- Chen T-C, Tsai P-W, Chu S-C, Pan J-S (2007) A novel optimization approach: bacterial-GA foraging. In: Proceedings of the second international conference on innovative computing, information and control (ICICIC). IEEE Computer Press, Washington, DC, p 391
- Cicirello VA, Smith SF (2001) Wasp nests for self-configurable factories. In: Proceedings of fifth international conference on autonomous agents. ACM, New York, pp 473–480
- Coello Coello CA, Toscano Pulido G, Salazar Lechuga M (2004) Handling multiple objectives with particle swarm optimization. *IEEE Trans Evol Comp* 8 (3):256–279
- Colorni A, Dorigo M, Maniezzo V (1992a) Distributed optimization by ant colonies. In: Varela F, Bourgine P (eds) Proceedings of the first European conference on artificial life, Elsevier, Paris, France, pp 134–142
- Colorni A, Dorigo M, Maniezzo V (1992b) An investigation of some properties of an ant algorithm. In: Männer R, Manderick B (eds) Proceedings of the parallel problem solving from nature conference (PPSN 92), Elsevier, Brussels, Belgium, pp 509–520
- Deneubourg J-L, Goss S, Franks N, Sendova-Franks A, Detrain C, Chretien L (1991) The dynamics of collective sorting: Robot-like ants and ant-like robots. In: Arcady-Meyer J, Wilson S (eds) From animals to animats: proceedings of first international conference on simulation of adaptive behavior. MIT Press, Cambridge, pp 356–365
- Depickere S, Fresneau D, Deneubourg J-L (2004) Dynamics of aggregation in *Lasius niger* (Formicidae): Influence of polyethism. *Insectes Sociaux* 51(1): 81–90
- DeRosier D (1998) The turn of the screw: the bacterial flagellar motor. *Cell* 93:17–20
- Di Caro G, Dorigo M (1998) AntNet: distributed stigmergetic control for communications networks. *JAIR* 9:317–365
- Di Caro G, Ducatelle F, Gambardella LM (2008) Theory and practice of ant colony optimization for routing in dynamic telecommunications networks. In: Sala N, Orsucci F (eds) Reflecting interfaces: the complex coevolution of information technology ecosystems. Idea Group, Hershey
- Doerner KF, Gutjahr WJ, Hartl RF, Strauss C, Stummer C (2006) Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection. *Eur J Oper Res* 171:830–841
- Dorigo M, Maniezzo V, Colorni A (1991) The ant system: an autocatalytic optimizing process. Technical Report No. 91-016 Revised. Politecnico di Milano, Italy
- Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of co-operating agents. *IEEE Trans Syst Man Cybernetics – Part B: Cybernetics* 26(1):29–41
- Dreo J, Siarry P (2006) An ant colony algorithm aimed at dynamic continuous optimization. *Appl Math Comput* 181:457–467
- Ducatelle F, Förster A, Di Caro G, Gambardella LM (2009) New task allocation methods for robotic swarms. In: Ninth IEEE/RAS conference on autonomous robot systems and competitions. Castelo Branco, Portugal, May 2009
- Franks NR, Sendova-Franks A (1992) Brood sorting by ants: Distributing the workload over the work-surface. *Behav Ecol Sociobiol* 30(2):109–123
- Gambardella LM, Taillard É, Agazzi G (1999) MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, London, pp 63–76
- García-Martínez C, Cordón O, Herrera F (2007) A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *Eur J Oper Res* 180:116–148
- Gaubert L, Redou P, Harrouet F, Tisseau J (2007) A first mathematical model of brood sorting by ants: Functional self organisation without swarm-intelligence. *Ecol Complexity* 4:234–241
- Grassé P-P (1959) La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux* 6:41–84
- Grassé P-P (1984) Termitología, Tome II – Fondation des sociétés construction. Masson, Paris
- Guney K, Basbug S (2008) Interference suppression of linear antenna arrays by amplitude-only control using a bacterial foraging algorithm. *Prog Electromagnet Res* 79:475–497
- Gutjahr WJ (2007) Mathematical runtime analysis of ACO algorithms: Survey on an emerging issue. *Swarm Intell* 1(1):59–79
- Häckel S, Fischer M, Zechel D, Teich T (2008) A multi-objective ant colony approach for pareto-optimization using dynamic programming. In: Proceedings of the tenth annual conference on genetic and evolutionary computation (GECCO). ACM, New York, pp 33–40
- Handl J, Meyer B (2007) Ant-based and swarm-based clustering. *Swarm Intell* 1(2):95–113

- Handl J, Knowles J, Dorigo M (2006) Ant-based clustering and topographic mapping. *Artif Life* 12(1):35–61
- Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinated bird flocks. In: Krasner S (ed) *The ubiquity of chaos*. AAAS, Washington, DC
- Holden N, Freitas AA (2007) A hybrid PSO/ACO algorithm for classification. In: Proceedings of the 2007 GECCO conference companion on genetic and evolutionary computation. London, UK, pp 2745–2750
- Hussein O, Saadawi T (2003) Ant routing algorithm for mobile ad-hoc networks (ARAMA). In: Proceedings of IEEE conference on performance, computing and communications, Phoenix, Arizona, USA, pp 281–290
- Jordan J, Helwig S, Wanka R (2008) Social interaction in particle swarm optimization, the ranked FIPS, and adaptive multi-swarms. In: Proceedings of the genetic and evolutionary computation conference (GECCO). Atlanta, Georgia, USA, pp 49–56
- Karlson P, Luscher M (1959) Pheromones: A new term for a class of biologically active substances. *Nature* 183:155–176
- Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international joint conference on neural networks. IEEE Press, Piscataway, pp 1942–1948
- Kim DH, Abraham A, Cho JH (2007) A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Inf Sci* 177:3918–3937
- Korb O, Stützle T, Exner TE (2007) An ant colony optimization approach to flexible protein-ligand docking. *Swarm Intell* 1(2):115–134
- Lee Z-J, Su S-F, Chuang C-C, Liu K-H (2008) Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. *Appl Soft Comput* 8:55–78
- Lin BMT, Lu CY, Shyu SJ, Tsai CY (2008) Development of new features of ant colony optimization for flowshop scheduling. *Int J Prod Econ* 112:742–755
- Lumer E, Faieta B (1994) Diversity and adaptation in populations of clustering ants. In: Cliff D et al. (eds) *From animals to animats 3: Proceedings of third international conference on simulation of adaptive behaviour*. MIT Press, Cambridge, pp 501–508
- Mariano CE, Morales E (1999) MOAQ: An ant-Q algorithm for multiple objective optimization problems. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakielka M, Smith RE (eds) *Proceedings of the genetic and evolutionary computation conference (GECCO 99)*. Orlando, Florida, USA, pp 894–901
- Mondada F, Gambardella LM, Floreano D, Nolfi S, Deneubourg J-L, Dorigo M (2005) The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robot Automat Mag* 12(2):21–28
- Nakrani S, Tovey C (2003) On honey bees and dynamic allocation in an internet server ecology. In: *Proceedings of second international workshop on the mathematics and algorithms of social insects*
- Niu B, Zhu Y, He X, Zeng X (2006) Optimum design of PID controllers using only a germ of intelligence. In: *Proceedings of sixth world congress on intelligent control and automation*. IEEE Press, Piscataway, NJ, pp 3584–3588
- Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. *IEEE Trans Evol Comput* 6(4):321–332
- Partridge BL (1982) The structure and function of fish schools. *Scient Am June*:114–123
- Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Cont Syst Mag June*:52–68
- Pini G, Brutschy A, Birattari M, Dorigo M (2009) Interference reduction through task partitioning in a robotic swarm. In: *Sixth international conference on informatics in control, automation and robotics (ICINCO 09)*. Milan, Italy
- Potts WK (1984) The chorus-line hypothesis of manoeuvre coordination in avian flocks. *Lett Nat* 309:344–345
- Quijano N, Passino KM (2007a) Honey bee social foraging algorithms for resource allocation. Part I: Algorithm and theory. In: *Proceedings of 2007 American control conference*. New York, USA, pp 3383–3388
- Quijano N, Passino KM (2007b) Honey bee social foraging algorithms for resource allocation. Part II: Application. In: *Proceedings of 2007 American control conference*, New York City, New York, USA, pp 3389–3394
- Reyes-Sierra M, Coello Coello CA (2006) Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *Int J Comput Intell Res* 2(3):287–308
- Reynolds C (1987) Flocks, herds and schools: A distributed behavioral model. *Comput Grap* 21(4):25–34
- Roberts J, Zufferey J, Floreano D (2008) Energy management for indoor hovering robots. In: *IEEE (eds) IEEE/RSJ international conference on intelligent robots and systems (IROS-2008)*. Nice, France
- Rosati L, Berioli M, Reali G (2008) On ant routing algorithms in ad hoc networks with critical connectivity. *Ad Hoc Netw* 6(6):827–859
- Sahin E (2005) Swarm robotics: From sources of inspiration to domains of application. In: *Swarm robotics. LNCS*, vol 3342. Springer, Berlin, pp 10–20
- Schoonderwoerd R, Holland O, Bruton J, Rothkrantz L (1996) Ant-based load balancing in telecommunications networks. *Adap Behav* 5(2):169–207
- Schoonderwoerd R, Holland O, Bruton J (1997) Ant-like agents for load balancing in telecommunications networks. In: *Proceedings of the first international*

- conference on autonomous agents. ACM, New York, pp 209–216
- Segall J, Block S, Berg H (1986) Temporal comparisons in bacterial chemotaxis. *PNAS* 83:8987–8991
- Shi XH, Liang YC, Lee HP, Lu C, Wang LM (2005) An improved GA and a novel PSO-GA-based hybrid algorithm. *Inf Process Lett* 93:255–261
- Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. *Eur J Oper Res* 185:1155–1173
- Tang WJ, Wu QH, Saunders JR (2007) A bacterial swarming algorithm for global optimization. In: Proceedings of the 2007 IEEE congress on evolutionary computation (CEC 2007). IEEE Service Center, Piscataway, pp 1207–1212
- Theraulaz G (1994) Du super-organisme à l'intelligence en essaim: modèles et représentations du fonctionnement des sociétés d'insectes. In: Bonabeau E, Theraulaz G (eds) Intelligence collective. Hermès, Paris, pp 29–109
- Theraulaz G, Bonabeau E (1995) Modelling the collective building of complex architectures in social insects with lattice swarms. *J Theor Biol* 177(4):381–400
- Theraulaz G, Bonabeau E, Nicolis SC, Sole RV, Fourcassié V, Blanco S, Fournier R, Joly JL, Fernandez P, Grimal A, Dalle P, Deneubourg JL (2002) Spatial patterns in ant colonies. *PNAS* 99 (15):9645–9649
- Tripathy M, Mishra S (2007) Bacteria foraging-based solution to optimize both real power loss and voltage stability limit. *IEEE Trans Power Syst* 22 (1):240–248
- Vander Meer RK, Alonso LE (1998a) Pheromone directed behaviour in ants. In: Vander Meer RK et al. (eds) Pheromone communication in social insects. Westview, Boulder, CO, pp 159–192
- Vander Meer RK, Breed M, Winston M, Espelie KE (eds) (1998b) Pheromone communication in social insects. Westview, Boulder, CO, pp 368
- von Frisch K (1967) The dance language and orientation of bees. Harvard University Press, Cambridge, MA
- Waibel M, Keller L, Floreano D (2009) Genetic team composition and level of selection in the evolution of multi-agent systems. *IEEE Trans Evol Comput* 13 (3):648–660
- Walker RL (2000) Dynamic load balancing model: Preliminary assessment of a biological model for a pseudo-search engine. In: Parallel and distributed processing. LNCS, vol 1800. Springer, Berlin, pp 620–627
- Yang B, Chen Y, Zhao Z (2007) Survey on applications of particle swarm optimization in electric power systems. In: IEEE international conference on control and automation. Guangzhou, China, pp 481–486
- Yin P-Y, Glover F, Laguna M, Zhu J-X (2007) Scatter PSO – A more effective form of particle swarm optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007). IEEE Press, Piscataway, NJ, pp 2289–2296
- Yuan H, Li Y, Li W, Zhao K, Wang D, Yi R (2008) Combining immune with ant colony algorithm for geometric constraint solving. In: Proceedings of the 2008 workshop on knowledge discovery and data mining. IEEE Computer Society, Washington, DC, pp 524–527
- Zhang R, Wu C (2008) An effective immune particle swarm optimization algorithm for scheduling job shops. In: Proceedings of the third IEEE conference on industrial electronics and applications. Singapore, pp 758–763

49 Simulated Annealing

Kathryn A. Dowsland¹ · Jonathan M. Thompson²

¹Gower Optimal Algorithms, Ltd., Swansea, UK

k.a.dowsland@btconnect.com

²School of Mathematics, Cardiff University, UK

thompsonjm1@cardiff.ac.uk

1	<i>Introduction</i>	1624
2	<i>Combinatorial Optimization and Local Search</i>	1625
3	<i>Overview of Simulated Annealing</i>	1628
4	<i>Theory</i>	1630
5	<i>Generic Decisions</i>	1632
6	<i>Problem-Specific Decisions</i>	1634
7	<i>Enhancements and Modifications</i>	1638
8	<i>Applications</i>	1648
9	<i>Conclusions</i>	1649

Abstract

Since its introduction as a generic heuristic for discrete optimization in 1983, simulated annealing (SA) has become a popular tool for tackling both discrete and continuous problems across a broad range of application areas. This chapter provides an overview of the technique with the emphasis being on the use of simulated annealing in the solution of practical problems. A detailed statement of the algorithm is given, together with an explanation of its inspiration from the field of statistical thermodynamics. This is followed by a brief overview of the theory with emphasis on those results that are important to the decisions that need to be made for a practical implementation. It then goes on to look at some of the ways in which the basic algorithm has been modified in order to improve its performance in the solution of a variety of problems. It also includes a brief section on application areas and concludes with general observations and pointers to other sources of information such as survey articles and websites offering downloadable simulated annealing code.

1 Introduction

Combinatorial optimization problems occur whenever there is a requirement to select the best option from a finite set of alternatives. They arise in a wide range of application areas including business, medicine, and engineering, and are renowned for being relatively simple to state but are often very difficult to solve. Until the 1980s much of the research effort into the solution of such problems concentrated on exact solution techniques, that is, techniques that are guaranteed to produce optimal solutions. Although such approaches cut down considerably on the computational requirements of complete enumeration, for many classes of problems the number of steps needed to guarantee optimality still grows exponentially with problem size – a factor that has serious repercussions for the scalability of any solution approach. By the end of the 1960s, these ideas had been formalized with the definition of an efficient algorithm as one in which the number of steps required increases polynomially with problem size (Edmonds 1965), and the conjecture that there are some problems for which no efficient solution algorithm exists. Subsequent work in theoretical computer science during the 1970s resulted in the now well-established theory of computational complexity and NP-completeness due to Cook (1971, 1972) and Karp (1972). A full treatment of these developments can be found in Garey and Johnson (1979). From a practical point-of-view, the key finding of this work is that there are many combinatorial optimization problems for which efficient algorithms that can guarantee optimality may never be found. This fact, combined with wider availability and decreasing cost of computer power during the early 1980s, lead to a change of focus from exact techniques. Instead, many researchers concentrated their efforts on the search for generic heuristic approaches that could be applied to any combinatorial optimization problem and produce high quality, if not optimal, results. Simulated annealing is one of a number of heuristics developed at this time that derive their inspiration from the natural world. In the case of simulated annealing, this inspiration comes from the behavior of fluids when they are subjected to controlled cooling such as in the production of large crystals. Since its introduction in 1983 there has been a wealth of literature on both theoretical and practical aspects. The purpose of this chapter is to provide an overview of simulated annealing, with emphasis on its use in the solution of practical problems. Nevertheless, some theoretical results are necessary in order to get the best out of a simulated annealing implementation and so these are covered

in [Sect. 4](#). More detailed treatments can be found in Aarts and Korst (1989), Aarts et al. (2005), and Salamon et al. (2002). It should also be noted that although simulated annealing was first introduced as a heuristic for combinatorial optimization, it is now also used for continuous optimization.

Simulated annealing (SA) can be regarded as an improved version of the older technique of local search. In a local search heuristic, an initial solution is gradually improved by considering small perturbations or changes; for example, changing the value of a single variable, or swapping the values of two variables. The set of solutions that can be produced in this way is known as the neighborhood of the original solution and its elements are referred to as neighbors. Starting from the initial solution, a trial solution is chosen from the set of neighbors. If this is “better” than the current solution, then it replaces it. This process continues until no improving solution exists. The replacement of the current solution by the trial solution is known as a neighborhood move. The problem with this type of approach is that the search terminates at a local, rather than global, optimum. This problem can be partially overcome by repeating the algorithm from different starting solutions, or by using more complex neighborhoods, thus widening the scope of the search. A third option used in heuristics, such as SA and tabu search (Glover 1989; Gendreau and Potvin 2005), is to allow some non-improving moves to be accepted. However, to ensure that the overall trajectory of the search is in the direction of improvement, this must be done in a controlled manner. In the case of SA, the control mechanism is inspired by the way in which thermodynamic energy is reduced when a substance is subjected to controlled cooling. These ideas are formalized in the following section.

2 Combinatorial Optimization and Local Search

A combinatorial optimization problem (S, f) consists of a finite space, S , of all possible solutions and a cost function $f : S \rightarrow \mathbb{R}$ that assigns a real cost to each $s \in S$. For a minimization problem the objective is to find s^* such that $f(s^*) \leq f(s) \forall s \in S$.

In order to apply a local search heuristic, we need to define a neighborhood structure $N(S)$ on the solution space S . For each $s \in S$, $N(s)$ defines a subset of S whose members are close to s in some way. Given the above definitions, the generic local search algorithm can be stated as follows:

Procedure local search

Begin

 Initialise (s_0)

$s_c = s_0$

 Repeat

 Select $s \in N(s_c)$

 If $f(s) < f(s_c)$ then $s_c = s$

 Until $f(s) \geq f(s_c) \forall s \in N(s_c)$

End

The above definition requires further detail concerning the selection of s . There are two common strategies; random descent and steepest descent. In random descent, the neighborhood is sampled uniformly and the first improving solution is selected, while in steepest descent, the whole neighborhood is evaluated and the lowest cost neighbor is selected.

Two examples of combinatorial optimization problems with appropriate neighborhood definitions are given below.

2.1.1 Example 1: The p-median Problem (Teitz and Bart 1968)

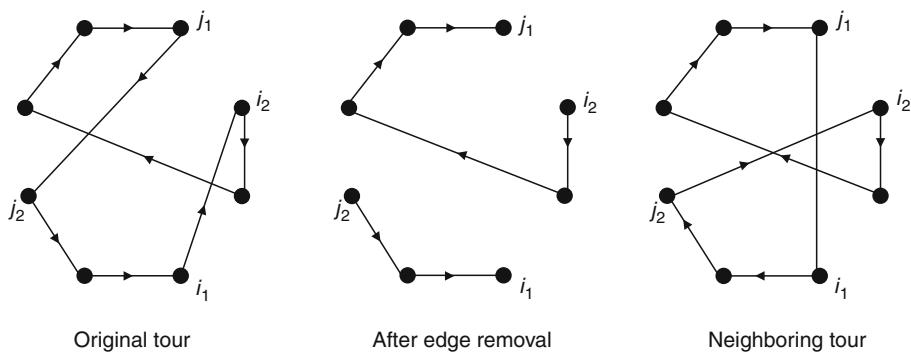
Given a set I of n supply locations, a set J of m demand locations, a cost matrix C , where c_{ij} is the cost of supplying demand location j from supply location i , and an integer constant p , find a subset of p supply locations such that the total cost of supplying J is minimized. Any feasible solution is a partition of I into a set P , containing p elements of I , and the set \bar{P} containing the remaining elements of I . The set of neighboring solutions is defined as those solutions that can be obtained by swapping an element of P for an element in \bar{P} .

2.1.2 Example 2: The Traveling Salesman Problem (TSP)

Given a set of n cities and an $n \times n$ matrix D where d_{ij} defines the distance between city i and city j , find a tour (i.e., a circular route visiting each city exactly once and then returning to the starting point) that minimizes the total distance traveled. The set of feasible solutions is the set of permutations of the cities, and any permutation can be regarded as the set of links between adjacent cities in the permutation (the n th city is regarded as being adjacent to the first city). A suitable neighborhood definition is then the set of solutions that can be obtained by removing two edges and then reconnecting the two resulting paths in such a way as to produce a feasible tour. This is illustrated in Fig. 1. Note that if the removed links are (i_1, i_2) and (j_1, j_2) the new links must be (j_1, i_1) and (j_2, i_2) if the result is to be a feasible tour. Note also that the segment between i_1 and j_2 will be reversed. Thus, if the distances are symmetric, tours in $N(s)$ will differ only by the difference in the four edges involved in the swap. However, in the case of the asymmetric problem, this may not be such a good neighborhood as the cost of the reversed segment will also have changed.

Fig. 1

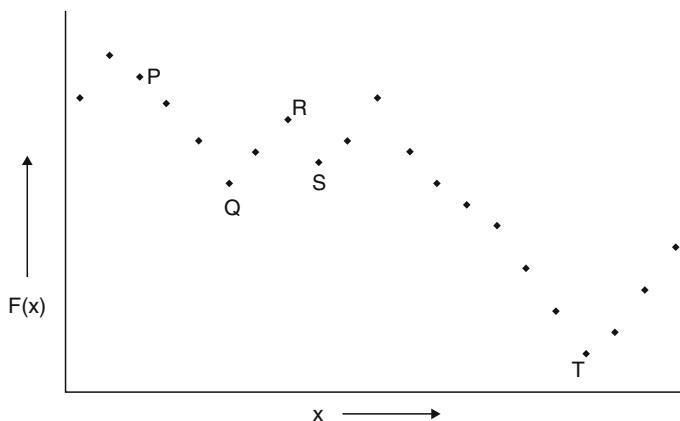
An example of a 2-opt neighborhood move for the TSP.



The advantage of local search is that it is a generic approach that simply requires the definition of a neighborhood structure on the solution space. Thus, it can be applied to almost any combinatorial optimization problem. The disadvantage is that it will converge to a local rather than a global optimum, and thus the quality of the final solution is dependent on the starting point. As a simple illustration, consider the one-dimensional function shown in Fig. 2. Let the set of neighbors of a given point, s , be the two points adjacent to s . Then, if we apply local search from solution P the result will be the local optimum at Q, whereas, if we start from R we may reach Q or S. Both solutions are clearly worse than the global optimum at T. Although this is a very simple example, the same behavior is apparent in multidimensional solution spaces associated with more complex problems.

This difficulty can be ameliorated somewhat by selecting several starting solutions. For realistically sized problems, simply selecting a set of random starting points is rarely sufficient, however, techniques such as greedy randomized adaptive search procedure (GRASP) (Feo et al. 1994) that attempt to use heuristic rules to obtain promising starting solutions and adjust these using feedback from previous iterations have had some success. An alternative approach is to define larger neighborhoods, for example, swapping k locations for the p-median problem, or k edges for the TSP. Such neighborhoods are referred to as k -opt neighborhoods. This can improve results and, in general, as k increases, the number of local optima decreases. Unfortunately, the size of such neighborhoods is exponential in k and so only small values of k are practical and solution quality remains highly dependent on the starting solution. For example, in Fig. 2 if we increase the neighborhood to those solutions within two steps of the current solution we will not be able to reach the global optimum from S, or any point to the left of S. More intelligent ways of expanding the neighborhood such as those used in variable neighborhood search or very large neighborhood search (VLNS) have been successful for some problems. The third option for dealing with local optima is to allow some uphill moves in a controlled manner. This is the method employed by simulated annealing.

Fig. 2
An example landscape for a one-dimensional function.



3 Overview of Simulated Annealing

Simulated annealing for combinatorial optimization was introduced by Kirkpatrick et al. (1983) and independently by Černý (1985). The approach is based on an algorithm to simulate the cooling of material in a heat bath, due to Metropolis et al. (1953), a process known as annealing. When solid material is heated past its melting point and then cooled again, the structural properties of the resulting material depend on the residual energy in the material, which in turn depends on the rate of cooling. For example, large crystals represent a state of low energy and result from very slow cooling, whereas fast cooling or quenching results in crystals with high energy, resulting in imperfections. The laws of thermodynamics state that for a substance in a state of equilibrium at an ambient temperature T , the probability of an increase in energy of magnitude δE is given by

$$P(\delta E) = \exp\left(\frac{-\delta E}{kT}\right) \quad (1)$$

where k is a constant known as Boltzmann's constant. Metropolis proposed a simulation model to determine the energy of a system at its ground (or frozen) state, that is, when the temperature is reduced to its limiting value. In the simulation, the material is considered as a system of atoms. For each possible configuration of atoms, the energy of the system can be calculated. At each iteration, one atom in the current configuration is subjected to a small displacement and the increase in energy, δE , is calculated. If $\delta E \leq 0$ a new configuration is accepted automatically. Otherwise it is accepted with probability $P(\delta E)$. This is facilitated by generating a random number, r , in the interval $[0,1]$ and accepting the new configuration if $r < P(\delta E)$. The algorithm can be used to simulate the annealing process by repeatedly reducing the value of T . Once the system has reached equilibrium at the current temperature, until the system freezes at its ground state. Note that this approach can be regarded as a random descent approach in which some uphill moves are accepted according to Eq. 1. Kirkpatrick formalized these ideas by mapping the elements of the Metropolis algorithm onto the elements of a combinatorial optimization problem as given in Table 1. Note that as Boltzmann's constant is a physical constant, it is essentially suppressed and the term kT is considered as a single constant, t , which is usually referred to as the temperature.

Although this analogy is an attractive one, in order to be of use in an optimization context it must work effectively for problems where the landscape of the energy function over the states of the system displays similar features to those found in combinatorial optimization problems. This is not the case for most fluids as all the atoms are alike and are therefore

Table 1

Mapping of the Metropolis algorithm onto a combinatorial optimization problem

Thermodynamic simulation	Combinatorial optimization
System states	Feasible solutions
Energy	Cost
Change of state	Neighborhood move
Temperature	Control parameter
Frozen state	Heuristic solution

interchangeable in the solution, resulting in a single ground state. However, there are some classes of fluids that do not obey these rules. Kirkpatrick uses the example of spin-glasses. These are materials that exhibit a feature known as frustration, caused by competing sources of magnetic energy within the system. This has a similar effect on the solution landscape to the interaction between conflicting objectives and constraints in a combinatorial optimization problem and results in many different possible ground states. Thus the analogy between searching the solution space for a combinatorial optimization problem and the annealing of spin-glasses was considered worthy of further experimental investigation. Kirkpatrick's first experiments were based on a partitioning problem in computer design in which the cost function had the same form as the energy function for spin glasses. This was followed by work on problems with more general cost functions including the TSP.

The mapping in [Table 1](#) allows one to convert any local search algorithm into an annealing algorithm by accepting non-improving moves according to [Eq. 1](#). The probability of accepting such a move will be dependent on the magnitude of the increase and the temperature. Thus small increases will be accepted more often than large ones and the probability of accepting any such moves will gradually reduce to 0 as the temperature approaches 0.

A formal statement of the simulated annealing algorithm is given below.

Procedure simulated annealing

Begin

 Initialise (s_0)

$s_c = s_0$

 Repeat

 For $its = 1, nrep$ do

 Begin

 Select $s \in N(s_c)$

$\delta = f(s) - f(s_c)$

 If $\delta < 0$ then $s_c = s$

 else

$r = U[0,1]$

 if $\exp(-\delta/t) > r$ then $s_c = s$

 End

$t = \alpha(t)$

 Until stopping condition = 'true'

 End

where $U[0,1]$ is a random number in the range $[0,1]$.

In its original form, the Metropolis algorithm returns the current solution when the stopping point is reached. However, this may not be the minimum cost solution visited during the search and it is now common practice to return the best solution found rather than the final one.

The above procedure is a generic statement of the algorithm and a number of decisions must be made in order to implement it for the solution of a particular problem. These can be partitioned into two categories; those relating to the cooling process (the generic decisions) and those relating to the way in which the problem is modeled within a local search framework (the problem-specific decisions). The generic decisions include the starting temperature, the rate of cooling (governed by the function α and the parameter $nrep$) and the stopping

condition. The problem-specific decisions involve the definition of the solution space, the cost function, and the choice of neighborhood structure. These combine to define a solution landscape over which the search takes place.

Empirical research has shown that both types of decisions need to be made with care as they influence both the speed of the algorithm and the quality of the solutions obtained, while theoretical results have shown that under the right conditions asymptotic convergence to an optimal solution can be guaranteed. Unfortunately, it has also been shown that in order to guarantee a solution that is arbitrarily close to the optimum, the required number of iterations is exponential in problem size, that is, no better than an exact approach based on complete enumeration. Thus, the theory does not lead directly to a set of generic decisions. Nevertheless, it does provide some pointers as to what factors should be taken into account when making both types of decisions. We therefore review some of the results in the next section before the generic and problem-specific decisions are discussed in more detail.

4 Theory

Much of the theoretical background to simulated annealing is derived from concepts in statistical mechanics and from the theory of Markov chains. This section gives a very brief introduction to these two concepts and cites a number of sources that go into more detail.

For a constant temperature parameter, t , the behavior of simulated annealing can be modeled as a Markov chain. Thus, Markov chain theory underpins much of the theoretical work on SA. In the model, the solutions correspond to the possible states of the system and for any pair of solutions i and j the probability that the system will move from state i to state j is given by the product of the probability of generating solution j as a neighbor of i , denoted by $G_{ij}(t)$ and the probability that j will be accepted, denoted by $A_{ij}(t)$. In the basic model, it is usually assumed that each neighbor of i will be selected with equal probability and that the acceptance probability is given by \bullet Eq. 1. Thus we have

$$P_{ij}(t) = \begin{cases} G_{ij}(t)A_{ij}(t) & \text{if } i \neq j \\ 1 - \sum_{l \in S, l \neq i} G_{il}(t)A_{il}(t) & \text{if } i = j \end{cases}$$

where

$$G_{ij}(t) = \begin{cases} \frac{1}{|N(i)|} & \text{if } j \in N(i) \\ 0 & \text{otherwise} \end{cases} \quad \text{for any } t$$

and

$$A_{ij}(t) = \begin{cases} 1 & \text{if } f(j) \leq f(i) \\ \exp\left(\frac{(f(i)-f(j))}{t}\right) & \text{otherwise} \end{cases}$$

As long as the neighborhoods are strongly connected (i.e., it is possible to reach any solution from any other using a sequence of neighborhood moves) Markov chain theory states that there exists a unique stationary distribution. This distribution defines the probability of reaching solution i after an infinite number of moves at temperature t (denoted $Q_t(i)$), for each solution i . It is given by

$$Q_t(i) = \frac{\exp\left(-\frac{f(i)}{t}\right)}{\sum_{j \in S} \exp\left(-\frac{f(j)}{t}\right)}$$

Moreover, if we define the set of optimal solutions to be S^* then

$$\lim_{t \downarrow 0} Q_t(i) = \begin{cases} \frac{1}{|S^*|} & \text{if } i \in S^* \\ 0 & \text{otherwise} \end{cases}$$

This implies asymptotic convergence to an optimal solution as long as the process reaches the stationary distribution at each temperature. See Aarts and Korst (1989) for a more detailed treatment of the derivation of these results.

A corresponding result can be obtained by modeling the problem as an inhomogeneous Markov chain, in which t is reduced after each move, as long as the sequence of temperatures t_0, t_1, \dots, t_k , obey certain assumptions. In particular, Hajek (1988) showed that the rate of cooling should be related to the shape of the solution landscape and that the sequence of temperatures should satisfy $t_k \geq c/\log(2+k)$, where c is the depth of the deepest local (but not global) minimum. Although bounds on c have been derived for some combinatorial optimization problems (Kern 1986), it has also been shown that calculating c is itself an NP-hard problem. Even if c can be approximated, the number of steps required by Hajek's formula for even moderately sized problems is too large to be practically useful. Other results have suggested that the number of moves required to guarantee a stationary distribution that is arbitrarily close to the optimum will typically require a number of iterations that is exponential in problem size. For example Mitra et al. (1986) show that for the inhomogeneous case, the number of moves required is greater than the size of the solution space, while Aarts and Van Laarhoven (1985) showed that for the homogeneous model the number of steps required at each temperature would be quadratic in the size of the solution space.

The results presented above made quite stringent assumptions concerning the generation and acceptance probabilities that correspond to the basic statement of SA. However, it is possible to prove convergence under more general conditions. In particular, the generation probabilities do not need to be uniform over the neighborhood as long as they are symmetric, that is $G_{ij}(t) = G_{ji}(t)$.

In addition to results based on Markov chain theory, it has also been shown that other concepts from the field of statistical thermodynamics are relevant in the context of optimization. These include the specific heat and entropy. The specific heat of a system is defined as the rate of change in energy with respect to temperature and is given by σ_T^2/kT^2 , where σ_T^2 is the variance in energy at the current temperature T . In a physical system, a high level of specific heat indicates a fundamental change in the state of the system, and it has been suggested that the corresponding situation in an optimization context may signal the need for slow cooling. For example, in the solution of the TSP, Kirkpatrick et al. (1983) observed an increase in specific heat once the basic structure of the tour was in place. A related concept is entropy, which is a measure of the disorder of a system. The entropy at equilibrium can be defined as $S_T = -\sum_{i \in S} Q_T(i) \ln(Q_T(i))$. Fleischer and Jacobson (1999) carried out an empirical study

based on different implementations for the maximum clique problem and showed that the expected cost of the final solution was inversely related to the entropy. From an optimization viewpoint, it can be shown that as long as equilibrium is reached at each value of t , both the expected cost and entropy decrease monotonically. Furthermore, the probability of finding an

optimal solution at each temperature increases monotonically as t decreases. For a full derivation of these results, refer to Aarts and Korst (1989).

Although the above results do not provide any definitive rules for defining polynomial time cooling schedules with any performance guarantees, they still provide some guidance when making problem-specific decisions. These will be discussed in the following sections.

5 Generic Decisions

The generic decisions define the cooling schedule and comprise the starting temperature t_0 , the final temperature t_f and the temperature reduction function α . Each of these is discussed in turn.

5.1 Starting Temperature

If the final solution is to be independent of the starting point then the initial temperature must be hot enough to allow free movement through the solution space. In some cases, this can be estimated from the problem data. For example, it may be easy to calculate the mean or maximum change in cost over all uphill neighborhood moves. This can then be used to estimate the temperature at which such a move would be accepted with a reasonable probability. One commonly used formula is $t_0 = (-U)/\ln(R_{\text{acc}})$, where U is the mean increase in cost over all uphill moves and R_{acc} is the required acceptance ratio. Even if it is not possible to estimate U theoretically, it can be estimated by conducting a random walk on the solution space. However, as pointed out by Triki et al. (2005), if the landscape contains some shallow and some very deep local optima, then it may be better to base the starting temperature on an estimate of the maximum, rather than the mean. An alternative is simply to start from a low temperature and to increase it rapidly until the desired acceptance ratio is reached. At this point cooling can begin. In fact, this method is closer to the physical analogy in which a solid is heated past its melting point before being annealed slowly. This approach was taken by Kirkpatrick et al. (1983) in their experiments on a chip-placement problem arising in computer design.

5.2 Reduction Function

The success of any simulated annealing algorithm is highly sensitive to the rate at which the temperature parameter is reduced. This is governed by the function α and the number of steps at each temperature, $nrep$. It is apparent from the theory that t needs to be reduced slowly. Many different schedules have been suggested and two of the most popular are a geometric schedule based on the homogeneous model and a schedule introduced by Lundy and Mees (1986) based on the inhomogeneous model. The geometric schedule takes the form $\alpha(t) = at$, where $a < 1$. Empirical evidence from the literature supports the need for slow cooling and typical values for a are in the range 0.8 – 0.99 with a bias to higher values, that is, to slower cooling. The value of $nrep$ is often related to the size of the neighborhood, and may vary from temperature to temperature. For example, it is important to spend enough time at low temperatures to ensure that the regions around a local optimum have been fully explored.

Thus, it is common to increase $nrep$ as the temperature is reduced. $nrep$ may also be related more directly to the state of the search by using feedback from the search statistics. This results in a dynamic schedule. Simple dynamic schedules include reducing the temperature with reference to the number of moves accepted, the ratio of accepted moves to trial moves (the acceptance rate), or the ratio of accepted moves to cost increasing moves. As any of these statistics may be very small at low temperatures it is usual to impose a maximum number of moves per temperature as well. The Lundy and Mees schedule is based on the inhomogeneous model and reduces the temperature every iteration according to $\alpha(t) = t/(1+\beta t)$ where β is a suitably small value. Lundy and Mees suggest that β should be significantly smaller than $1/D$, where D is an upper bound on the maximum cost difference between any two points in the solution space. It has been shown that if the parameters for these two schedules are such that t is reduced at a similar rate over a similar range then there is little to choose between them in terms of solution speed or quality. Note that as the gradient of the Lundy and Mees schedule gets shallower as temperature decreases, an equivalent geometric schedule would involve increasing $nrep$ with every decrease in t .

In all the above schedules, the function α is constant and predetermined. Suitable parameters are often derived by trial and error and may not be robust, even for different instances of the same problem. As the theory of simulated annealing is based on the concept of equilibrium at each temperature, it has been argued that an adaptive schedule based on this idea may be a better choice. One way of achieving this is to ensure that equilibrium is reached at the initial temperature and then reduce the temperature by a sufficiently small amount so that equilibrium will be reached quickly at the new temperature. This will be true for t_0 if it is high enough to allow free movement between solutions. Thus all that is necessary is to ensure that each temperature value is sufficiently close to its predecessor. A number of reduction functions have been suggested based on estimates of the mean energy at different temperatures. An early example is the schedule of Van Laarhoven and Aarts (1987). The aim of their schedule is for the stationary distributions at consecutive temperatures to be close. This will be true if $Q_{\alpha(t)}(i) < (1+\varepsilon)Q_t(i) \forall i, t$, and for a suitably small value of ε . This leads to the derivation of the function

$$\alpha(t) = t \frac{1}{1 + \frac{\ln(1+\varepsilon)}{3\sigma_t} t}$$

where σ_t is the standard deviation of the cost at temperature t . In practice, suitably small values of ε and suitable starting and ending temperatures mean that this cooling schedule is usually too slow. One of the most popular schedules in practice is due to Huang et al. (1986). The reduction in temperature is determined by the difference in the average cost at consecutive temperatures. They use

$$\alpha(t) = t \cdot \exp(-\lambda t / \sigma_t)$$

where λ is an estimate of Δ/σ_t where Δ is the difference in the expected cost at t and $\alpha(t)$. They suggest using a value of 0.7 for λ .

Although both these schedules are designed to cope with different solution landscapes, we note that neither is parameter free. More recently, Triki et al. (2005) suggested a more general representation of this type of schedule using

$$\alpha(t) = t \cdot \left(1 - t \frac{\Delta(t)}{\sigma_t^2}\right)$$

where $\Delta(t)$ is an estimate of Δ as defined above. They point out that an appropriate choice of $\Delta(t)$ will yield both Van Laarhoven and Huang's schedules, as well as other schedules from the literature. They suggest an implementation in which the value of $\Delta(t)$ is initialized to a quantity proportional to σ (estimated from a random walk) and *nrep* is governed by a limit on the number of iterations and the number of acceptances, whichever occurs first. At this point, if the decrease in cost is sufficiently close to the target $\Delta(t)$, equilibrium is assumed and the temperature is reduced. Otherwise, another set of repetitions is carried out at this temperature. If the test for equilibrium is failed four times at the same temperature, the assumption is that the temperature was reduced too rapidly so the temperature is increased and the target function Δ is decreased.

5.3 Freezing Point

In theory, the temperature should be reduced to zero when no further uphill moves are possible and the system will terminate at a local minimum. In practice, a frozen state where no further moves are possible is often reached well before this. Lundy and Mees suggest that if the objective is to produce a solution within ε of the local optimum with probability θ then the termination point should satisfy $t_f \leq \varepsilon / \ln((|S|-1)/\theta)$. Others have suggested stopping after a predefined number of iterations without acceptance, or when the acceptance ratio falls below a certain value. Alternatively, if the average cost is measured over a small number of moves, the system may be considered frozen when the difference in consecutive calculations falls below a predefined parameter.

6 Problem-Specific Decisions

The problem-specific decisions cover the definition of the solution space, neighborhood structure, and cost function. These factors combine to define the solution landscape. Empirical results suggest that this has a significant impact on the success of a simulated annealing implementation. This coincides with the theoretical results such as Hajek's and fits well with the concept of entropy, as at the same temperature sparsely connected regions of the solution space display lower levels of entropy than denser areas, indicating that there is a smaller choice of acceptable moves. As with the generic decisions it is not possible to derive a set of definitive rules that can be applied to a given problem. Nevertheless, it is possible to outline some desirable properties. These are based on common sense, empirical investigations, and theoretical results. For many problems it may not be possible to meet all these guidelines at the same time, and some of them may even be contradictory.

The first consideration is to maintain the conditions for convergence. As outlined in the previous section, the original convergence results relied on a set of stringent conditions, but it has since been shown that it is sufficient that every solution be reachable from every other by a chain of neighborhood moves. This is usually easy to verify for well-structured problems, but may be more difficult for messy problems that occur in practice.

A second consideration stems from the requirement for slow cooling. This means that it will be necessary to execute a large number of iterations within the available computing time. Thus the calculations involved in each iteration need to be as efficient as possible. Two processes that are related to the solution landscape are required at each iteration. The first is

the generation of a randomly selected trial solution from the set of neighbors of the current solution. For many problems, a combination of the most obvious definitions of solution space and neighborhood structure results in a fast and simple neighborhood generation routine. For example, for the p-median problem all that is required is to select a member of P and a member of $I \setminus P$. This can be achieved by generating two integer random numbers within the appropriate ranges. However, this may not be the case for complex neighborhood definitions or if the solution space is highly constrained. In such cases, it may be beneficial to relax some of the constraints in the definition of the solution space and to include a penalty term in the cost function to guide the search away from infeasible solutions. This will be illustrated in the graph coloring example in the next section.

The second routine that is required at every iteration is the calculation of the difference in cost between the current solution and the trial solution. One way of doing this is to calculate the cost of the trial solution from scratch, but this is often unnecessary as it is more efficient to calculate the change in cost. For example, in the traveling salesman example, as long as the problem is symmetric, the change in cost is just the difference between the cost of the two removed edges and that of the two replacement edges. The cost calculation for the p-median problem is more complex. In an optimal solution, each demand point will be allocated to the nearest element of P . Thus changing one supply point may change the allocations of any of the demand points. Nevertheless, it is still worthwhile calculating the cost in terms of those demand points that have been reallocated rather than completely evaluating the new solution. The efficiency of such calculations can be improved by using appropriate data structures. It is worth noting that even if suitable data structures require a significant amount of updating every time a move is accepted, the number of accepted moves over a whole run is small compared to the number of trial moves evaluated. Thus their use may still save time.

In order to reach good solutions, it is desirable that the cooling schedule be as close as possible to the ideal. Hajek's result suggests that neighborhood/cost function combinations that give rise to spiky landscapes with deep troughs should be avoided. On the other hand, it is also clear that the cost function should guide the search over the landscape, rather than behave like an unbiased random walk. Thus, flat plateau-like regions resulting from groups of neighboring solutions of equal cost should also be avoided. This point will also be illustrated in the graph coloring example below. A number of theoretical and empirical results also suggest that the rate of cooling should be related to the size of the solution space and/or neighborhoods. This suggests that these should be kept as small as possible. In the case of the solution space, this observation conflicts with the idea of increasing the solution space to allow some infeasible solutions as suggested above. In some cases, it may be possible to reduce the set of solutions by applying a reduction technique to the problem beforehand. For many classical combinatorial optimization problems (e.g., set covering, maximum clique, steiner problem in graphs), such techniques have been well documented. It is however worth remembering that, by definition, these techniques remove points from the solution space and therefore care must be taken that this does not destroy the reachability property. Even if this is not the case the reduction may remove neighbors that are close in cost to the current solution, resulting in a landscape with a spiky topography and lower entropy that will be more difficult to search. In the case of the neighborhoods, it is generally agreed that neighborhoods should be small, so that neighborhood moves are truly local. Nevertheless, it is not always beneficial to use the smallest possible neighborhood as larger neighborhoods may lead to a smoother topography. Conversely, Goldstein and Waterman (1988) suggest that if the landscape is too smooth, it becomes too easy to escape from valleys and thus good optima may be missed. However, at low

temperatures, they are likely to need more iterations in order to ensure that they are sampled adequately so that the local minimum in the region is located.

The above discussion suggests that the combination of solution space, neighborhood structure, and cost function suggested for the local search solution for the TSP should also work well for simulated annealing. Neighbors are easy to generate and the change in cost is simple to calculate. The size of the solution space is $(n-1)!/2$ and the size of the neighborhood is $n(n-1)/2$. Thus the neighborhood is small compared to the size of the solution space. It is also easy to see that any solution can be reached from any other by a chain of feasible moves. In the case of the p-median problem, the local search framework is also suitable for SA, although the change in cost function cannot be calculated quite as easily as in the TSP case. The fact that more elements in the cost function are subject to change also suggests that the solution landscape may have more spikes and troughs. The next section looks at one further example – that of graph coloring, for which the problem-specific decisions present a greater challenge.

6.1 Graph Coloring

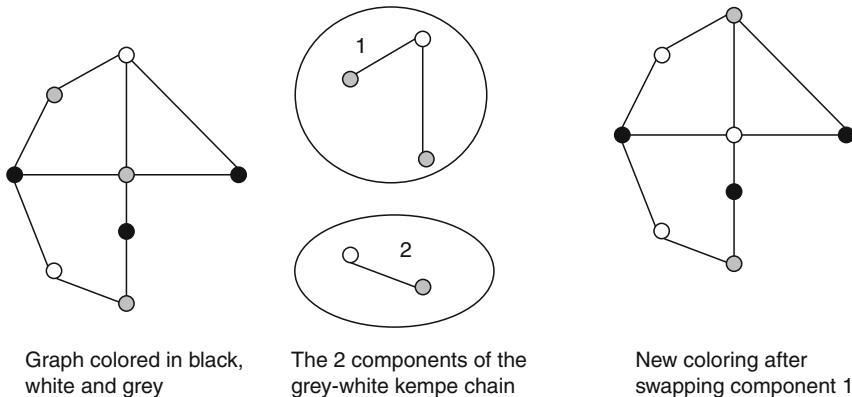
The graph coloring problem can be stated as follows. Given a graph $G(V, E)$ with vertex set V and edge set E find an allocation of colors to the vertices, such that adjacent vertices are given different colors and the number of colors used is minimized.

Unlike the traveling salesman problem, prior to the 1980s the graph coloring problem was not regarded as a natural candidate for local search. Nevertheless, there has been considerable interest in simulated annealing for graph coloring. The most obvious definition for the solution space is the set of all feasible colorings. These can be represented as partitions of the set of vertices into subsets, such that there are no edges between two vertices in the same subset. All vertices within a subset are considered to be of the same color, so the objective is to minimize the number of subsets. A natural definition for a neighborhood of a given solution is those partitions that can be obtained by moving a single vertex into a new subset. However, there is a problem with this, in that not all moves will result in a feasible solution as edges may be introduced into the subsets involved in the move. One can overcome this by limiting the moves to those subsets that are feasible. This could mean that some vertices will not have any feasible moves and also makes the generation of a neighbor more complex. A more fundamental problem is that it is easy to generate instances where the reachability condition is violated under this neighborhood. Even if the solution landscape remains connected, paths between different regions may be sparse thus making it difficult for the search to move away from the region containing the starting solution.

One way of overcoming this is to use a larger neighborhood. Morgenstern and Shapiro (1989) suggested a neighborhood based on Kempe chains. If we consider that part of the graph relating to just two colors or subsets, then it will consist of one or more components, each of which is completely disconnected from the others. Such components are known as Kempe chains. Swapping the colors (or subsets) of the vertices within one of these components will not introduce any edges between vertices in either subset, and thus the result is also a feasible coloring. This is illustrated in [Fig. 3](#). Morgenstern and Shapiro suggest defining the neighborhood as those solutions that can be reached by swapping the colors of a single Kempe chain. There is a computational overhead in generating such moves, and in calculating their effect on the cost function. Nevertheless, Kempe chain neighborhoods have been shown to outperform simple neighborhoods for some classes of graph. In addition, Thompson and

Fig. 3

Example of a Kempe chain neighborhood move for the graph coloring problem.



Dowsland (1998), Wright (1996), and Tuga et al. (2007) have all demonstrated their effectiveness in solving graph coloring-type problems arising in scheduling and timetabling.

An alternative to larger neighborhoods is to relax the definition of the solution space to include infeasible solutions and to penalize these with a suitable penalty term in the cost function. Examples of this approach will be given after a more general discussion of the cost function.

The natural definition of the cost function is the number of colors used. This is not suitable for a simulated annealing implementation as it will lead to large plateau-like areas in which the costs of neighboring solutions are identical and there is no information to guide the search toward lower cost solutions. For example, a solution in which there are k subsets each with a similar number of vertices will have the same cost as a solution with k subsets in which one subset has just one vertex. Clearly, the latter is preferable to the former as it requires just one move to reach an improved solution requiring $(k-1)$ subsets. Aarts and Korst (1989) suggest a

cost function of the form $\sum_{i=1}^q w_i(|V_i| - \lambda|E_i|)$ where V_i, E_i are the sets of vertices and edges in

the i th subset, λ is a weighting factor such that $\lambda > 1$ and q is an upper bound on the number of colors needed. w_1, \dots, w_q is a decreasing sequence of weights based on the number of vertices in subset i , and the subset index, designed to encourage lower indexed sets to be emptied in favor of those with higher indices. The optimal solution with respect to this cost function is guaranteed to be optimal in terms of the original objective, but the weights can be very large and therefore cause computational difficulties. For practical purposes, a relaxation of the condition on the weights is suggested, but this no longer guarantees that the optimal solution will correspond to an optimal coloring. Johnson et al. (1991) suggest a simpler solution based on a term to encourage large subsets and a penalty term for infeasible colorings. The form of their function is:

$$-\sum_{i=1}^q |V_i|^2 + \sum_{i=1}^q |V_i||E_i|.$$

Although optimal solutions using this function may not correspond to optimal colorings, they are guaranteed to be local optima.

An alternative way of dealing with the cost is to determine an upper bound, q , on the number of colors required, define the solution space as the set of all partitions of the vertices into q subsets, and simply minimize the sum of the $|E_i|$ values. Once this is reduced to zero, q can be reduced and the process repeated. The last value of q for which a zero-valued solution was achieved is taken as the final solution. The problem here is the requirement to find an optimal solution in a series of annealing runs. However, for practical problems, the objective is sometimes simply to find a coloring in q colors for a given q . Johnson et al. (1991) compare this approach with their penalty methods and a Kempe chain-based implementation. They conclude that for problems where the global optimum based on the penalty cost implies several more colors than the optimal coloring, the q partition method can often do better than the Kempe-chain approach. Their main conclusion is that none of the three implementations outperformed the others on all classes of graph, emphasizing the difficulty in finding the right combination of problem-specific definitions for a particular problem type.

7 Enhancements and Modifications

The previous sections have dealt with implementations of simulated annealing that adhere to the analogy with the physical cooling process. Thus the theoretical results on convergence hold for the optimization process. However, as we have seen, run times for cooling schedules that are able to give any performance guarantees tend to be too long to be of practical value. Thus, very few real implementations have the benefit of such theoretical backup. It is also the case that the theory behind many cooling schedules is based on the assumption that the returned solution is the final, and not necessarily the best solution. It is therefore arguable that there is no reason to stick rigidly to the original Metropolis simulation, and that any heuristic that searches the solution space using neighborhood moves and allows uphill moves that are controlled using a probability function based on temperature may be an equally good candidate for solving optimization problems in practice. For some generalizations the convergence proofs will still hold. Even if they do not, the heuristic may still perform well enough. This section examines a number of ways in which successful implementations have bent the rules of the classical Metropolis-based algorithm. In some cases, these modifications improve the performance of SA on problems for which it may not seem ideally suited, for example, those with spiky solution landscapes, or cost functions that are expensive to calculate. In others, they simply improve the efficiency of the search, thus allowing larger problems to be tackled.

7.1 Acceptance Probability

An obvious way of relaxing the conditions of the physical analogy is to move away from the Boltzmann distribution in determining whether or not an uphill move should be accepted. Its use in the original heuristic is entirely due to the laws of thermodynamics, which have no meaning in terms of an optimization problem. Nevertheless, it is still used by the majority of implementations. This is no doubt due to the fact that it does possess some desirable properties: solutions are accepted according to the magnitude of increase in the cost function, δ ; the probability of acceptance decreases monotonically as δ increases; and this probability tends to zero as $\delta \rightarrow \infty$. On the other hand there is some motivation for using a different distribution. Johnson et al. (1989) showed that in a straightforward implementation of annealing to a graph

partitioning problem, the calculation of $\exp(-\delta/t)$ took up approximately one-third of the execution time. They concluded that it may be beneficial to use a simpler function, thereby allowing time for additional iterations. They suggested using a linear function that approximated the exponential over an appropriate range of values but found that better results could be obtained if $\exp(-\delta/t)$ was calculated for all possible δ, t combinations beforehand and stored in a look-up table. This is obviously only possible if the number of different cost function values is reasonably small.

A very simple scheme suggested by Dueck and Sheuer (1990) and by Moscato and Fontanari (1990) is known as threshold accepting. The idea here is that t represents a threshold on the values of δ that will lead to acceptance and the probability is defined as $P(\delta) = 1$ if $\delta \leq t$, $= 0$ otherwise. Threshold accepting has proved successful for some problems and less so for others. More recently, Penna (1994) and Tsallis and Stariolo (1996) suggested a more complex acceptance probability of the form

$$P(\delta) = \begin{cases} 1 & \text{if } \delta \leq 0 \\ (1 - (1 - q)^{\frac{\delta}{t}})^{\frac{1}{1-q}} & \text{if } \delta > 0 \text{ and } (1 - q)^{\frac{\delta}{t}} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

where q is an additional parameter such that $q \neq 1$. This allows larger jumps to be accepted occasionally depending on the value of q . If $q \rightarrow 1$ then this is equivalent to the standard Metropolis algorithm. Franz and Hoffmann (2003) have since shown that a slight modification of the Tsallis acceptance probability produced by scaling t by a factor of $(2-q)$ not only tends to the Metropolis algorithm as $q \rightarrow 1$, but also tends to threshold accepting as $q \rightarrow -\infty$. They were able to show that given an ideal cooling schedule results improve as $q \rightarrow -\infty$. Given that threshold accepting was introduced as a simplification of SA designed purely for easy implementation this is a somewhat surprising result. However, it is worth noting that performance depends on an appropriate schedule and far more is known about good schedules for the Boltzmann acceptance rules than for threshold accepting.

Many other practical implementations have been shown to benefit from changes to the acceptance rules. For example, Dowsland (1993a) found that the solution landscape for a two-dimensional rectangle packing problem contained large peaks and troughs so that moderate increases in cost were needed to escape local optima. When the temperature was high enough to accept such moves with a reasonable probability, too many small uphill moves were made. This was overcome by adding a positive constant to the change in cost. This has the effect of moving to a flatter part of the exponential curve, thus decreasing the probability of accepting small moves relative to moderate ones, while still ensuring that very large uphill moves are accepted with extremely low probability. There has also been some success with simple functions. For example, Ogbu and Smith (1990) and Vakharia and Chang (1990) use probabilities that are independent of δ on different sequencing problems, while others have used a linear function such as δ/t (Brandimarte et al. 1987).

Wright (2001) suggests an alternative rule for problems in which the objective is made up of a number of different terms, which he refers to as subcosts. This is often the case for problems with hard and soft constraints, where the objective is often to minimize a weighted linear function based on the violated soft constraints. In such cases, the conflict between the constraints may mean that it is necessary to violate several constraints in different groups in order to gain any improvement in another. Therefore, if traditional acceptance probabilities are used, the search may get trapped in regions where certain constraints are never satisfied. In order to overcome this, Wright suggests subcost-guided simulated annealing, which uses the acceptance criterion

$$P(\delta) = \exp\left(\frac{-\delta \cdot \exp(-\frac{\Theta B}{\delta})}{t}\right)$$

where B is the best improvement in any of the subcosts and Θ is a problem-specific parameter. This has the effect of improving the chances of accepting a move that results in a large improvement in one of the subcosts at the expense of several smaller increases in the others.

We have seen that it may be beneficial to include infeasible solutions in the solution space and to penalize them in the cost function. However, as pointed out by Abramson (1991), finding the right weights for such penalties can be difficult. Weights that are high enough to ensure convergence to a feasible solution tend to mean that the search becomes trapped early in a sub-region of the solution space as infeasible solutions tend never to be accepted even at moderate temperatures. Pedamallu and Ozdamar (2008) suggest a novel acceptance criterion for this situation. Instead of considering a combined cost, they monitor the sequence of feasible and infeasible solutions separately and make acceptance decisions independently within each sequence. These two ideas are incorporated into two hybrid simulated annealing methods. They also reduce the probability of accepting an uphill move by only considering acceptance if the last n trial moves were also uphill moves.

7.2 Cooling

Most simulated annealing runs that start with a high enough temperature to allow free movement and cool until the system appears to be frozen do much of their useful work in the middle part of the run. When the temperature is high the search will accept almost any solution, each one of which could have been generated as the initial solution. This part of the search can also be computationally expensive as the high acceptance rate means that there are more solution updates to be carried out. In order to overcome this, some implementations may use rapid cooling at this stage, for example, reducing temperature after just a few moves have been accepted. Others suggest starting at a lower temperature where the rate of acceptance is much lower than that suggested by the theory. Conversely, at low temperatures when few moves are accepted, SA becomes a slower version of random descent in which equal cost moves as well as improving moves are accepted. This can lead to cycling around local optima without actually making any progress. This is sometimes avoided by replacing this phase of the algorithm by a random descent phase and stopping once a local optimum has been reached (e.g., Merlot et al. (2003)).

Many practical implementations set the temperature so that just the middle phase of the cooling schedule is used. However, as long as we return the best solution found during the search, it can be argued that overall cooling may not be necessary. Boese and Kahng (1994) show that under these conditions optimal schedules may involve heating, even at the end of the search. In a practical context, Connolly (1990) suggested using a constant temperature, as long as that temperature is appropriate for the problem at hand. After experimenting with temperatures that resulted in a high measure of specific heat and finding that they did not result in high quality solutions, he suggests starting the process with a fairly quick run using the Lundy and Mees schedule over a wide temperature range, and then using most of the available time using a fixed temperature, defined as that temperature where the minimum cost solution was located. Results on the quadratic assignment problem were better than those obtained with standard cooling.

Connolly reports that a constant temperature allowed a good balance between downhill and uphill moves, thus encouraging the search to get to the bottom of local minima, but also allowing it to escape again. For spiky landscapes this may not be sufficient, and the idea of reheating as a means of escape from deep valleys has been used in a range of different ways. Glover and Greenberg (1989) point out that this is perhaps a more intuitive approach as it corresponds to making a concerted effort to move uphill once it becomes clear that further progress in the current valley is unlikely. It is worth noting that Kirkpatrick et al. did use some reheating in their initial study of the TSP, using a graphical representation of the tours to identify when the search seemed to be stuck and using manual intervention to reheat. Such manual intervention is obviously not practical for most cases and so a number of ways of automating the reheating process have been suggested. We have already mentioned Triki et al. (2005) who reheat slightly when it appears that the latest cooling step may have lowered the temperature too far. Several practical implementations reheat once a frozen state has been reached. For example, Anagnostopoulos et al. (2006) raise the temperature to twice the value where the best solution was found. This is repeated until the number of reheats without an improvement reaches a given limit. Other examples include Azizi and Zolfaghari (2004) who use the cooling function $\alpha(t) = t_0 + \lambda \ln(1+r)$ where r is a count of the number of consecutive uphill moves up to the current iteration and λ is a control parameter. The initial temperature is set to a *low* value, which is equal to the minimum value that t can take. At the start, the low temperature means that the search should move rapidly downhill. Each time an uphill move is selected, the temperature increases, thereby giving it more impetus to climb out of local minima. As soon as a downhill move is accepted, the temperature reverts to its minimum value. Dowsland (1993a) also suggests an undulating schedule. In this case, the decision as to whether to raise or lower temperature is based on whether the last move was accepted or rejected. If a move is accepted, then temperature is reduced and vice versa. Both cooling and reheating are carried out using the Lundy and Mees formula with different values of β . If $\beta_c = k\beta_h$ where β_c is used for cooling and β_h is used for heating, then the ratio of rejected to accepted moves will be approximately k . Dowsland first used this schedule with constant k for a rectangle packing problem. If k is reduced periodically, then this schedule can be used to replace standard cooling. This has been used with some success for examination scheduling (Thompson and Dowsland 1996), and as a way of ensuring that the acceptance rate drops monotonically with temperature in an implementation of SA that involves periodically selecting a new neighborhood (Dowsland et al. 2007).

7.3 Neighborhoods

In the basic algorithm, it is assumed that neighborhoods are well defined and do not change at all during the search. There are many practical examples where nonconstant definitions of the neighborhood have been shown to lead to improvements in solution time or quality. One popular ploy is to adjust the neighborhood as the temperature changes. For example, for problems such as packing or placement problems, where neighborhood moves involve moving an element over a finite distance, it is common to reduce the maximum distance allowed as the temperature reduces. Sechen et al. (1988) applied this idea to a simulated annealing approach to a VLSI design problem in which rectangular blocks are to be placed on a chip in such a way as to minimize a combination of cost factors. Valid moves are a horizontal or vertical transition of a single block. As only small transitions tend to be accepted at low temperatures,

the authors argue that time is wasted sampling and rejecting such moves, and so the maximum transition length is decreased as temperature decreases. Similar strategies have been used in SA implementations for packing problems. Such strategies are not limited to problems where moves involve physical distances and are also applied where neighborhood moves involve changing the value of a single variable where an upper limit on the magnitude of the change is reduced gradually.

A reduction in the neighborhood may also save time in situations where not all entities in the solution are contributing to cost. For example, consider a scheduling problem in which the cost function represents a series of penalties for undesirable features in the schedule and neighborhood moves involve moving a single event. Not all events will contribute to the cost and so moving such an event will not lead to an improvement. Many implementations therefore exclude these moves from the neighborhood. For example, in solving the rectangle packing problem Dowsland (1993a) found it beneficial to move only those pieces causing overlap, and in solving an examination scheduling problem. Thompson and Dowsland (1998) move only those exams causing clashes. While this strategy is a good idea for many problems, it should be used with care because it may destroy the reachability property. In view of this, Tovey (1988) suggests that the full neighborhood should be used with a given probability and the reduced neighborhood used the rest of the time.

In the context of other local search algorithms, it has been shown that using different neighborhoods, either with a fixed probability throughout the search, or at different times in the search, can be very effective (Hansen and Mladenovic 2005). Such a strategy is not straightforward in a simulated annealing context, as using different neighborhoods at different temperatures will have a significant effect on the transition probabilities, and therefore the expected cost may no longer decrease monotonically with temperature. Nevertheless, neighborhoods with significant variations have been used in some implementations. For example, Dowsland et al. (2007) use several different neighborhoods in the solution of a shipper rationalization problem and attempts to overcome this problem by using the acceptance rate to set the temperature using the undulating schedule described in the last section.

7.4 Neighborhood Sampling

In addition to the definition of the neighborhood, the transition probabilities are also affected by the probability with which each neighbor is sampled. In [Sect. 4](#), it was assumed that every neighbor was generated with an equal probability. In practice, sampling may not be uniform. In some cases, the choice of nonuniform sampling is made to simplify the neighborhood generation procedure, while in others it is an attempt to guide the search more effectively.

A simple modification that has been used in a number of implementations is the use of cyclic, as opposed to random sampling. The rationale behind this is twofold. Close to local optima, there may only be one or two moves that lead to an improvement in cost. At low temperatures it is desirable that such moves be made. However, even if the number of trials is greater than the size of the neighborhood such moves may not be sampled and eventually an uphill move may be accepted and the search may move away from the region without the local optimum having been visited. Cyclic sampling ensures that every move is considered before any is considered for a second time. Cyclic sampling also has the advantage of saving time as it does not require the generation of one or more random numbers in order to select a neighbor. Nevertheless, it is important to impose a random ordering on the cycle at the start of the run in

order to avoid introducing any form of bias. This can be illustrated by considering an approach taken by Connolly (1990) for the quadratic assignment problem in which a set of n items are to be allocated to n locations so that each location contains a single item. The natural neighborhood for this problem is to swap two items. For any solution we can therefore consider the neighborhood moves as the set of (unordered) pairs of items. If we take the natural ordering then all swaps for item 1 would be considered before those for item 2, etc. This may introduce unwanted bias and so Connolly randomizes the list and then cycles through the list in this new order. This resulted in improved performance over random sampling in terms of both time and solution quality. This example also illustrates a third potential benefit of cyclic sampling. If the list of neighborhood moves can be represented in such a way that it is valid for all solutions, as in this case, then when a move is accepted, instead of starting from the beginning of the list it makes sense to carry on from the current position. In this case, cyclic sampling will help to avoid the search cycling through a small set of solutions as it behaves like a (long) tabu list. For example, if items a and b are swapped, all other swaps will be considered before a and b can be swapped back again. Others who have used cyclic sampling to ensure the search is sufficiently diverse include Osman (1993). It should be noted, however, that other researchers have found that cyclic sampling has an adverse effect on solution quality for their particular implementations.

At low temperatures, time may be wasted in sampling many solutions (often more than once) without an acceptance. Greene and Supowit (1986) suggest the alternative of determining the probability of acceptance for each neighborhood move and then using these to weight the sampling distribution, that is, they combine the two probabilities $A_{ij}(t)$ and $G_{ij}(t)$ into a single acceptance probability. As the transition probabilities remain the same, this will not change the behavior of the algorithm. Greene and Supowit apply their method to a graph partitioning problem and show that the required probabilities are easy to maintain. For other problems this may not be the case, but this approach can be used at a local minimum at low temperatures. In this situation, it may take a long time before an uphill move is accepted. If some form of sampling without replacement is used, as each neighbor is generated, its cost can be recorded. If no move is selected before the neighborhood is exhausted, the appropriate probabilities can be calculated and a move selected according to these values.

Perhaps the most intuitive use of nonuniform sampling is to bias the selection toward low cost neighbors. For example Lin et al. (2008) consider a truck and trailer routing problem and instead of making the accept/reject decision on a single neighbor, they generate several neighbors and then use the best of these as the trial solution. Ropke and Pisinger (2006) apply a very large neighborhood search within a simulated annealing framework for a pick up and delivery problem. VLNS is a local search method that uses neighborhoods that are very large. This obviously smoothes out the solution landscape, but means that it is not practical to evaluate the whole neighborhood or even a large proportion of it. Thus VLNS is only effective when there is some form of optimization algorithm (or good fast heuristic) that can find an optimal (good) member of the neighborhood. If the neighborhood is optimized, then simulated annealing may not work well as the idea behind SA is that the pressure to move downhill should increase slowly throughout the search, and the selection of a neighbor should therefore be sufficiently random to maintain an appropriate level of flexibility. Ropke and Pisinger try to avoid this problem by introducing randomization in two ways. First, one of a number of different heuristics can be used to select the “best” element to remove/insert. This choice is random, but is biased according to previous performance. Second noise is added to the calculated cost before applying the acceptance/rejection criterion. Computational tests

show that this latter strategy gave better results than a version without noise. Anagnostopoulos et al. (2006) compare a neighborhood that includes some very large moves with one that includes only simpler moves. They conclude that the former neighborhood gives the best results. In the case of differentiable real-valued functions the concept of large neighborhoods is very attractive as local optima can be found easily using gradient methods. Wales and Scheraga (1999) coined the term “basin hopping” for a simulated annealing solution to a clustering problem, in which each trial solution was a local optimum. One problem with this approach is that the jump between starting points needs to be large enough to converge to a different basin, but close enough to maintain the spirit of a local, rather than random, search.

In the case of real-valued functions, the simplest way of generating a neighbor is to select solutions from a hyper-sphere with a given radius centered at the current solution. Alternatively, the distance is sometimes selected according to the Boltzmann distribution. This may mean that it takes a long time for the search to reach new areas of the solution space. In order to avoid this, it has been suggested that the distance moved is generated from other distributions with fatter tails, thus increasing the probability of generating larger jumps. In variable step sized simulated annealing, the distance is adjusted according to the acceptance ratio. If it is high, the search is considered to be far from the optimal and the size is increased, and vice versa (Kalivas 1992). In fast simulated annealing (Szu and Hartley 1987), the Cauchy distribution is used for this purpose.

In all the above examples, the choice of nonuniform sampling was made to improve the search. For some problems, the most straightforward way(s) of neighbor generation may not correspond to uniform sampling. For example, timetabling problems can be regarded as graph coloring problems in which the feasible solutions are the feasible colorings in q colors, and the objective is to optimize a cost function relating to the quality of the timetable. Suppose we define a neighborhood move as changing the color of a single vertex. As seen in Sect. 6, if the solution is to remain feasible, not all colors will be feasible for a given vertex. Therefore, if a neighbor is generated by selecting a vertex and then a new color for that vertex, the selection policy will tend to move vertices from “fuller” colors to “emptier” ones. An alternative with different bias might be to select two colors and then select a vertex that is free to move from the first color to the second. Such bias can affect solution results. For example, Thompson and Dowsland (1998) used this approach in the solution for an examination scheduling problem. They found that when the objective was to even out the number of students sitting exams on each day, then the first strategy performed well, but when the objective was to minimize back-to-back exams, when solutions in which alternate full and empty sessions are best, the second strategy was better. Similar results were reported using two different generators for a Kempe chain neighborhood on the same problem. Bias was also found to be a problem in Dowsland (1993b). In this case, the two natural sampling strategies introduced different biases both of which adversely affected solution quality. The final implementation used them both in alternate iterations, which removed the bias and yielded good results.

7.5 Cost Function

The standard simulated annealing algorithm does not assume any restrictions on the cost function, except that it should be well defined and constant. In practice, there are situations in which it may be necessary or advantageous to break these rules. The most obvious scenario is the case of a stochastic cost function or one that is subject to noise. There is a growing body of

literature on meta-heuristics, including simulated annealing, for such problems. Gelfand and Mitter (1989) show that if the noise in the objective function is normally distributed, then the convergence properties of SA still hold for noisy functions, while Gutjahr and Pflug (1996) generalize these results to other distributions. Others consider the situation in which the cost of a solution is estimated by sampling either from an appropriate distribution or from a simulation run. Some such as Alrefaei and Andradottir (1999) suggest storing all feasible solutions visited during the search. They then suggest that, when suitably adjusted to take account of the number of neighbors, the number of times a solution is accepted is a good measure of solution quality. Whatever measure of cost or acceptance strategy is employed, storage of all solutions is only feasible if the solution space is small.

When dealing with stochastic costs, the use of statistical tests to aid acceptance/rejection decisions is very common. These may be used simply to determine whether a move is a downhill or an uphill move. In the latter case, mean costs are then used to calculate the change in cost δ . Alternatively a t-test can be used to determine whether additional sampling is required in order to make a decision with confidence. Much of the work on stochastic cost functions is either theoretical or involves empirical analysis of small unrealistic problems and there are relatively few papers showing that these ideas work well in practice. Exceptions include Bulgak and Sanders (1988). A survey of SA for stochastic combinatorial optimization can be found in Bianchi et al. (2008).

A second reason for using an estimated cost function is that it may be computationally expensive to calculate the true cost. In  Sect. 6 we stated that if possible this situation should be avoided, but in some cases it may be worth compromising on this in order to improve other problem-specific factors. Tovey (1988) suggests that in such situations it may be possible to use a simpler approximation function. Rather than using the approximation at every iteration he suggests occasionally taking the time to calculate the true cost value. The difference between the real and estimated values at these points are then used to adjust the acceptance probabilities to compensate for the error. For example, if the true value is calculated 10% of the time and it is found that the estimated value was too high, then 90% of the time this solution will have a lower probability of acceptance. Therefore, on this occasion, the probability is increased to compensate. A drawback of this approach is that if the optimal solution is visited but its cost is overestimated it may not be returned as the best solution. An alternative approach is to use an approximation for the trial solution, but once a solution is accepted, to calculate its true value. In this way there may be some errors in the decision as to whether or not to accept a solution, but the true value of all accepted solutions will be known and the minimum of these can be retained as the final solution.

Even if the cost function is well defined and the minimum cost solutions are true global minima, it may be worthwhile considering an adjustment. This is particularly true when the cost includes one or more penalty terms relating to hard or soft constraints. Although the penalty weights may be set so that the cost is a true reflection of the relative quality of the solutions, these may not be the best weights for the annealing algorithm. For example, Wright (1991) argues that weights should reflect not only the importance of each constraint but also its difficulty. Anagnostopoulos et al. (2006) suggest that the weights need not be constant throughout the search. They adjust the weight of a penalty term up or down whenever a new best solution is accepted depending on whether the solution is infeasible or feasible. Fleischer and Jacobson's (1999) study on the relationship between entropy and the expected cost of the final solution includes experiments with a weighted penalty function for the maximum clique problem. They conclude that results can be significantly improved if the penalty function is

weighted in such a way as to smooth the solution landscape, thereby increasing the entropy of the system.

Rather than smoothing the landscape at the start, Hamacher and Wenzel (1999) adjust the landscape through changes to the cost function as the search progresses. Their approach, named stochastic tunneling (STUN), is designed not only to make it easy to descend into local optima, but also to escape from them again. They do this by adjusting the cost function so that the part of the landscape that is higher than the best solution found so far is smoothed, while the troughs in lower areas are exaggerated. This is achieved by making the transformation $f(i) \rightarrow 1 - \exp(-\lambda(f(x) - f(x_{\min})))$ where x_{\min} is the best solution to date and λ is a scaling factor. However they note that close to the global minimum this function becomes too smooth and insufficient downhill moves are made. For this reason they recommend alternating STUN with low temperature annealing. A later paper (Hamacher 2006) details further improvements.

7.6 With Other Methods

There are a large number of empirical investigations and practical case studies that show that a stand alone simulated annealing heuristic can be a very good solution approach for a wide range of problems. Nevertheless, in some cases, performance may be improved by combining it with other methods. These may be used as preprocessors to find a good starting solution or to transform the problem in some way to make it easier to solve; as post processors to make further improvements to the final solution; or as ingredients of the search process itself. The latter case may simply involve alternating a simulated annealing phase with one or more other search algorithms, or may be a true hybrid in which another method is embedded within the simulated annealing process or vice versa. The volume of papers and different possibilities for combination and hybridization are too great to provide a complete overview, so this section cites a few examples to give a flavor of this type of work.

The most common form of preprocessing is to use a heuristic to find a good starting solution instead of generating a purely random starting point. In this case, it makes sense to start with a lower temperature than normal so that all the benefits of the good solution are not destroyed. This can save time but care must be taken that this does not make the search entirely dependent on the starting solution. This not only involves ensuring that the temperature is not too low, but also that the starting solution should not be too good – for example, by using some form of randomized greedy construction rather than a totally greedy one. Bonomi and Lutton (1984) suggest an interesting way of initializing a solution for the TSP. They cluster the cities into regions and then find a good tour through the centers of the clusters. Within each cluster, the ordering is random but the clusters are ordered according to the tour. A different form of preprocessing is suggested by Chams et al. (1987). They point out that for partitioning problems such as graph coloring where the objective is to minimize the number of partitions required, a greedy algorithm that builds a solution one partition at a time is able to make good decisions early on and it is only in the later part of the construction that the myopic nature of such an approach becomes a problem. They therefore suggest building a partial solution in this way and then using SA to minimize the number of partitions required by the remaining elements. They report promising results using this approach for the graph coloring problem. More recently, Burke et al. (2008) report success with a similar approach for a stock-cutting problem. In this case, the

simulated annealing phase is also hybridized with a construction algorithm and further details are given later in this section.

Another possibility is to transform the problem, either into an easier to solve instance of the same problem, or into an equivalent problem. The possibility of reducing the size of the solution space has already been mentioned. Fleischer and Jacobson (1999) suggest that more dramatic transformations may be beneficial and illustrate their ideas by comparing the performance of SA in the solution of a maximum clique problem using a direct model and after transforming the problem into a satisfiability problem. They conclude that the level of entropy differs between the two models and that better solutions tend to result from the model with higher entropy.

The most common form of post-processing is to apply a descent phase starting from the best simulated annealing solution. This ensures that at least a local minimum has been found. In some cases, the neighborhood used for such a search may be different to that used for the annealing phase.

By far the largest body of literature covers other forms of hybridization. An approach that is becoming increasingly common for problems such as cutting and packing is to hybridize simulated annealing with a construction heuristic. The solution space is defined as a permutation of the elements of the solution, and simulated annealing explores this space. Each permutation is transformed into a solution to the original problem by applying the construction heuristic to the pieces in the order given by the permutation. Early implementations of this strategy were often abandoned as the cost function calculation proved too expensive but this is now less of a problem. For example, Burke et al. (2008) use simulated annealing to order the pieces in a two-dimensional packing problem and then pack them using a commonly used heuristic placement policy.

Search heuristics other than simulated annealing also involve acceptance/rejection decisions. There are several examples in which such decisions are made using a simulated annealing acceptance criterion. For example, in some genetic algorithm implementations, a decision as to whether or not a child is to replace its parent is required. While standard practice is to select the fittest of the two, some implementations replace the parent with the child even if the child is worse using the Boltzmann distribution. Examples of this type of approach include Pakhira (2003). Another class of algorithms that need to make acceptance/rejection decisions are hyperheuristics. A hyperheuristic is essentially a high-level heuristic that selects from a number of low-level heuristics to use at different points of the search. This choice may be made in a number of ways but is usually based on some measure of recent performance. Bai et al. (2006) compare a simulated annealing acceptance rule with other options and show that the SA rule works well.

Local search is often used as an ingredient of other search heuristics. For example, in GRASP it is an integral part, while it is now almost always used in an ant colony optimization (ACO) algorithm, and is frequently included in a genetic algorithm, when it is known as a memetic algorithm. In the simplest cases, the local search is a simple downhill search, but some implementations use simulated annealing in this role. Examples include Liu et al. (2000) in a GRASP approach to the frequency assignment problem, Chen et al. (2005) in an ACO heuristic to solve a vehicle routing problem, Guo et al. (2006) in a memetic algorithm for a knapsack problem, and Ge et al. (2007) in a particle swarm optimization algorithm to solve the job shop scheduling problem. However, unless simulated annealing is to become the primary search mechanism the time allowed must be kept relatively low. As success depends on slow cooling, the use of SA in this context has received some criticism.

The above examples involve the use of SA within another algorithm. A further class of hybrids involve the embedding of other methods within SA. Mention has already been made of VLNS and basin hopping in which optimization techniques are used to search large neighborhoods. In the case of discrete problems, examples include the use of dynamic programming, network flow models, and shortest path algorithms, while for continuous differentiable landscapes gradient methods can be applied.

In other cases, simulated annealing implementations may be enhanced by the use of ingredients from other approaches. For example, a tabu list is sometimes applied to ensure that cycling is avoided (Altiparmak and Karaoglan 2008), or a population of simulated annealing runs is used, with the worst performing members being periodically killed off and replaced by solutions found by the better runs (van Hentenryck and Vergados 2007).

Finally, there are many examples of implementations that involve several different heuristics including SA that are run independently at different points in the search.

8 Applications

As illustrated by the examples in the previous sections, simulated annealing has been used across a broad spectrum of application areas. A comprehensive survey is beyond the scope of this chapter. Instead, a few key application areas are highlighted and some examples are cited to illustrate the diversity of problems where simulated annealing has been applied successfully.

One of the earliest practical applications was in electronic engineering where it became a popular approach for placement and routing problems in VLSI design (Sechen and Sangiovanni-Vincentelli 1988; Chandy and Banerjee 1996; Wong et al. 1998). Since then it has been applied across a wide range of engineering applications. Examples include reactive power planning (Chen and Ke 2004; Jwo et al. 1995), optimizing fuel cells (Outeiro et al. 2008; Wishart et al. 2006), designing building frames (Paya et al. 2008), concrete bridge frame construction (Perea et al. 2008), and the design of steel frame structures (Degertekin 2007).

The operational research community was also quick to embrace simulated annealing and in 1990 Eglese (1990) published a paper entitled “simulated annealing: A general tool for operational research” that was recently selected by the European Association of Operational Research Societies (EURO) as one of the 30 most influential papers published in the *European Journal of Operational Research* between 1975 and 2005. Initial interest was in benchmark instances of classical problems such as the traveling salesman problem, graph coloring, etc. but this soon shifted to real-life problems in a variety of application areas. These include the fields of scheduling (Sekiner and Kurt 2007; Thompson and Dowsland 1998), routing (Tavakkoli-Moghaddam et al. 2007; Van Breedam 1995), location problems (Erdemir et al. 2008), and packing problems (Egeblad and Pisinger 2009; Gomes and Oliveira 2006).

Due to its ability to solve messy problems without any restrictions on the form of the objective function or the structure of the constraints, SA remains a popular choice of optimizer for a variety of scientific applications. A particularly vibrant area of research is in medicine, where it has been used to plan dosages in radiotherapy treatment (Jacob et al. 2008; Kubicky et al. 2008; Morton et al. 2008) and to optimize parameters in medical simulation models, for example, Marsh et al. (2007) and Choi et al. (2004). It is also widely used in the field of earth sciences and agriculture. Applications include land use allocation (Sante-Riveira et al. 2008), the study of gravity and seismic data (Yu et al. 2008), estimating sea surface

temperature (Arai and Sakakibara 2006), optimizing water distribution in irrigation canals (Monem and Namdarian 2005) and determining harvest schedules (Crowe and Nelson 2005). Examples of other fields include astronomy (Cornish and Porter 2007), genetics (Tewari et al. 2008), chemistry (De Andrade et al. 2008), and economics (Chen and Yeh 2001).

9 Conclusions

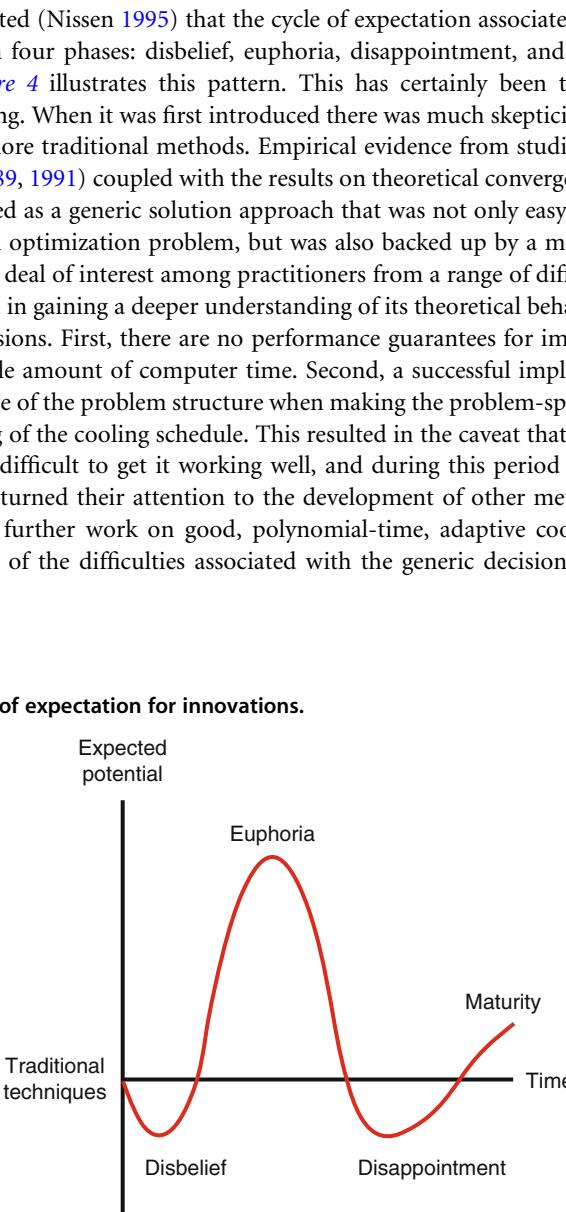
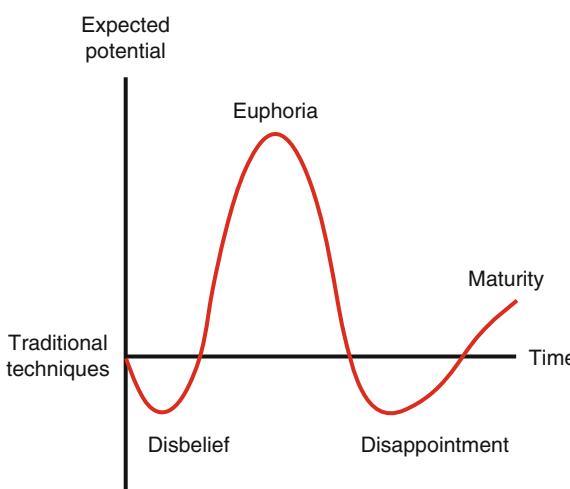
It has been suggested (Nissen 1995) that the cycle of expectation associated with any innovation goes through four phases: disbelief, euphoria, disappointment, and maturity or abandonment.  Figure 4 illustrates this pattern. This has certainly been true in the case of simulated annealing. When it was first introduced there was much skepticism about its ability to do as well as more traditional methods. Empirical evidence from studies such as those by Johnson et al. (1989, 1991) coupled with the results on theoretical convergence led to a period when SA was hailed as a generic solution approach that was not only easy to apply to almost any combinatorial optimization problem, but was also backed up by a mathematical theory. This led to a great deal of interest among practitioners from a range of different fields, as well as those interested in gaining a deeper understanding of its theoretical behavior. These studies led to two conclusions. First, there are no performance guarantees for implementations that run in a reasonable amount of computer time. Second, a successful implementation usually requires knowledge of the problem structure when making the problem-specific decisions and careful fine tuning of the cooling schedule. This resulted in the caveat that while SA is easy to get working, it is difficult to get it working well, and during this period of disillusionment, many researchers turned their attention to the development of other meta-heuristic techniques. Since then, further work on good, polynomial-time, adaptive cooling schedules has ameliorated some of the difficulties associated with the generic decisions, while a range of

 Fig. 4
The Nissen model of expectation for innovations.



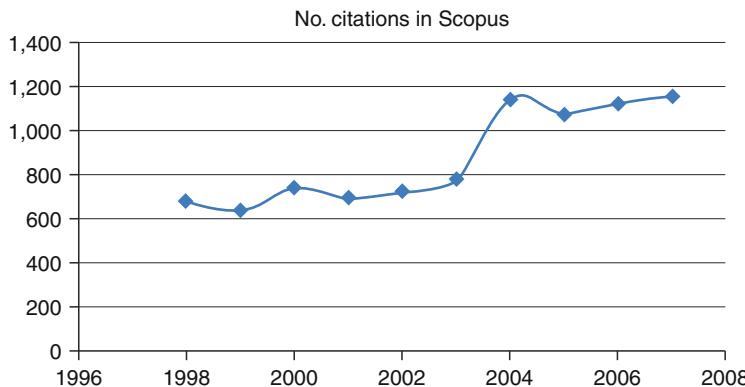
successful implementations on both real and artificial problems have been published. This has led to SA reaching maturity as a useful addition to the optimization toolkit.

As outlined in [Sect. 7](#), there has been a lot of research into ways of modifying the standard Metropolis algorithm in order to increase the range of problems that can be tackled successfully. Some of these are backed up by theory and/or are based on sound reasoning and have become common in many practical implementations. An obvious example is the use of reheating, which makes sense as the best, rather than the final, solution is returned. Others are ad hoc methods that have been shown to be successful on the problems to which they have been applied but have often not been taken up by others. Although these ideas may work for other problems, before bending the rules of SA too far, it is worth considering whether or not another heuristic, or some form of hybrid, may be more appropriate. Unfortunately there is little guidance on this. There have been a number of articles and larger projects comparing the performances of different heuristics on both benchmark and practical problems. For example, methods based on simulated annealing have performed strongly in two recent international timetabling competitions, with a simulated annealing implementation winning the 2003 competition (Kostuch [2004](#)) and a GRASP-simulated annealing hybrid coming second to a constraint-satisfaction approach in the examination scheduling track in 2008 (Gogos et al. [2008](#)). Other examples where simulated annealing has been shown to outperform other heuristics, or beat previous best-known results on benchmark problems include Egebald and Pisinger ([2009](#)), Rodriguez-Tello et al. ([2008](#)), and Zolfaghari and Liang ([2002](#)). Egebald and Pisinger consider two- and three-dimensional knapsack packing problems and produce competitive results on benchmark datasets, and Rodriguez-Tello et al. improve several best known results for benchmark problems for the minimum linear arrangement problem. Zolfaghari and Liang demonstrate that simulated annealing outperforms both genetic algorithm and tabu search methods on machine-grouping problems. However, the results from comparative studies need to be interpreted with care as the solution quality of any implementation depends as much on the skill of its designer as on the underlying heuristic. This point is clearly illustrated in Tiourine et al. ([1995](#)) in which different research groups applied one or more solution approach to the same problem with simulated annealing, tabu search, and genetic algorithms each being implemented by more than one group. The resulting implementations were then scored on a number of different factors. In all three cases, the implementations from different groups resulted in a different set of scores.

In the last 3 decades, there has been a range of competitive new heuristics, many also inspired by nature. In spite of this, an examination of the number of citations of the term “simulated annealing” in Scopus indicates that research is still prospering (see [Fig. 5](#)). The occurrence of simulated annealing-based solutions in the scientific literature increased between 2002 and 2004 and has remained steady for the last few years, suggesting that it is still regarded as a strong candidate for the solution of practical problems. In 2007, there were more than twice as many references for simulated annealing than there were for “tabu search” (including “taboo search”) or “ant colony.” Thus it remains a vibrant and exciting area of research. This is in part due to the fact that the claim of its early advocates remains true. It is very easy to get it working. For some problems a straightforward implementation using the most obvious definitions of solution space, neighborhood structure, and cost function, combined with a simple geometric or Lundy and Mees cooling schedule, may be all that is required. If this is not the case, then more sophisticated problem-specific decisions or a more complex cooling schedule can be the focus of further development. It is worth noting here that when undertaking such development, graphical information can be invaluable. If solutions are

Fig. 5

The number of citations of the term “simulated annealing” in Scopus.



easy to visualize (as with routing or packing problems) many researchers have found it worthwhile to build in routines that show how the current solution changes as the run progresses. Failing this, plots of statistics such as best cost, average cost, acceptance rate, entropy, specific heat, etc. often provide useful clues.

As shown in Fig. 5, simulated annealing continues to be of interest to researchers working in both practical and theoretical areas. This chapter has given a brief overview, with our main focus being the practical solution of combinatorial optimization problems. For a slightly different viewpoint, Aarts et al. (2005), Salamon et al. (2002), and Henderson et al. (2003) are suggested. Freely downloadable simulated annealing software is available from a variety of sources. Examples include a C language downloadable code of adaptive simulated annealing, where the algorithm parameters are automatically adjusted according to the progress of the algorithm (www.ingber.com), code for solving constrained, single-objective problems (<http://manip.crhc.uiuc.edu/CSA/>), a general purpose simulated annealing package designed for parallel machines (<http://wwwcs.uni-paderborn.de/fachbereich/AG/monien/SOFTWARE/PARSA>), and a framework for simulated annealing to be implemented within Matlab (<http://www.frostconcepts.com/software/index.html>).

References

- Aarts EHL, Korst JHM (1989) Simulated annealing and Boltzmann machines. Wiley, Chichester
- Aarts EHL, Van Laarhoven PJM (1985) Statistical cooling: a general approach to combinatorial optimisation problems. Philips J Res 40:193–226
- Aarts EHL, Korst JHM, Michiels W (2005) Simulated annealing. In: Burke EK, Kendall G (eds) Search methodologies. Springer, New York, pp 187–210
- Abramson D (1991) Constructing school timetables using simulated annealing: sequential and parallel algorithms. Manag Sci 37:98–113
- Alrefaei MH, Andradottir S (1999) A simulated annealing algorithm with constant temperature for discrete stochastic optimisation. Manag Sci 45:748–764
- Altıparmak F, Karaoglan I (2008) An adaptive tabu simulated annealing for concave cost transportation problems. J Operational Res Soc 59:331–341
- Anagnostopoulos A, Michel L, Van Hentenryck P, Vergados YA (2006) Simulated annealing approach to the traveling tournament problem. J Scheduling 9:177–193
- Arai K, Sakakibara J (2006) Estimation of sea surface temperature, wind speed and water vapour with

- microwave radiometer data based on simulated annealing. *Adv Space Res* 37(12):2202–2207
- Azizi N, Zolfaghari S (2004) Adaptive temperature control for simulated annealing: a comparative study. *Comput Operations Res* 31(4):2439–2451
- Bai R, Burke EK, Kendall G, McCollum B (2006) A simulated annealing hyper-heuristic for university course timetabling. In: Burke EK, Rudova H (eds) In: Proceedings of PATAT 2006, Brno, Czech Republic, August–September 2006. Lecture notes in computer science, vol 3867. Springer, Heidelberg
- Bianci L, Dorigo M, Gambardella LM, Gutjahr WJ (2008) A survey on metaheuristics for stochastic combinatorial optimisation. *Nat Comput.* DOI 101007, Online September 2008
- Boese KD, Kahng AB (1994) Best-so-far vs. where-you-are: implications for optimal finite time annealing. *Syst Control Lett* 22:71–78
- Bonomi E, Lutton JL (1984) The N-city travelling salesman problem: statistical mechanics and the Metropolis algorithm. *SIAM Rev* 26:551–568
- Brandimarte P, Conterno R, Laface P (1987) FMS production scheduling by simulated annealing. In: Micheletti GF (ed) Proceedings of the 3rd international conference on simulation in manufacturing, Turin, Italy, November 1987. Springer, Berlin, pp 235–245
- Bulgak AA, Sanders JL (1988) Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimise buffer sizes in automatic assembly systems. In: Abrams M, Haigh P, Comfort J (eds) Proceedings of the 1988 winter simulation conference, San Diego, CA, December 1988. IEEE Press, Piscataway, NJ, pp 684–690
- Burke EK, Kendall G, Whitwell G (2008) A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock cutting problem. *INFORMS J Comput* 21(3):505–516
- Černý V (1985) A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *J Optimization Theory Appl* 45:41–55
- Chams M, Hertz A, De Werra D (1987) Some experiments with simulated annealing for colouring graphs. *Eur J Operational Res* 32:260–266
- Chandy JA, Banerjee P (1996) Parallel simulated annealing strategies for VLSI cell placement. In: Proceedings of the 9th conference on VLSI design, Bangalore, India, January 1996. IEEE Computer Society, Washington, pp 37–42
- Chen YL, Ke YL (2004) Multi-objective VAr planning for large-scale power systems using projection-based two-layer simulated annealing algorithms. *IEE Proc Generation, Transm Distribution* 151(4):555–560
- Chen S-H, Yeh C-H (2001) Evolving traders and the business school with genetic programming: a new architecture of the agent based artificial stock market. *J Econ Dyn Control* 25 (3–4):363–393
- Chen C-H, Ting C-J, Chang P-C (2005) Applying a hybrid ant colony system to the vehicle routing problem. *Lect Notes Comput Sci* 3483:417–426
- Choi K-S, Sun H, Heng P-A (2004) An efficient and scalable deformable model for virtual reality-based medical applications. *Artif Intell Med* 32(1):51–69
- Connolly DT (1990) An improved annealing scheme for the QAP. *Eur J Operational Res* 46:93–100
- Cook SA (1971) The complexity of theorem procedures. In: Proceedings of 3rd ACM symposium on the theory of computing, Shaker Heights, OH, 1971. ACM, New York, pp 151–158
- Cook SA (1972) An overview of computational complexity. *Commun ACM* 26:400–408
- Cornish NJ, Porter EK (2007) The search for massive black hole binaries with LISA. *Classical Quantum Gravity* 24(23):5729–5755
- Crowe KA, Nelson JD (2005) An evaluation of the simulated annealing algorithm for solving the area-restricted harvest-scheduling model against optimal benchmarks. *Can J Forest Res* 35(10):2500–2509
- De Andrade MD, Nascimento MAC, Mundim KC, Sobrinho AMC, Malbouisson LAC (2008) Atomic basis sets optimization using the generalized simulated annealing approach: new basis sets for the first row elements. *Int J Quantum Chem* 108 (13):2486–2498
- Degertekin SO (2007) A comparison of simulated annealing and genetic algorithm for optimum design of nonlinear steel space frames. *Struct Multidisciplinary Optimization* 34(4):347–359
- Dowsland KA (1993a) Some experiments with simulated annealing techniques for packing problems. *Eur J Operational Res* 68:389–399
- Dowsland KA (1993b) Using simulated annealing for efficient allocation of students to practical classes. In: Vidal RVV (ed) Applied simulated annealing. Lecture notes in economics and mathematical systems, vol 396. Springer-Verlag, Berlin
- Dowsland KA, Thompson JM (1998) A robust simulated annealing based examination timetabling system. *Comput Oper Res* 25:637–648
- Dowsland KA, Soubeiga E, Burke EK (2007) A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *Eur J Oper Res* 179(3):759–774
- Dueck G, Sheuer T (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *J Comput Phys* 90:161–175
- Edmonds J (1965) Paths, trees and flowers. *Can J Maths* 17:449–467

- Egeblad J, Pisinger D (2009) Heuristic approaches for the two and three dimensional knapsack packing problem. *Comput Oper Res* 36(4):1026–1049
- Eglese RW (1990) Simulated annealing: a general tool for operational research. *Eur J Oper Res* 46(3):271–281
- Erdemir ET, Batta R, Spielman S, Rogerson PA, Blatt A, Flanigan M (2008) Location coverage models with demand originating from nodes and paths: application to cellular network design. *Eur J Oper Res* 190(3):610–632
- Feo TA, Resende MGC, Smith SH (1994) A greedy randomised adaptive search procedure for maximum independent set. *Oper Res* 42:860–878
- Fleischer M, Jacobson SH (1999) Information theory and the finite time behavior of the simulated annealing algorithm: experimental results. *INFORMS J Comput* 11:35–43
- Franz A, Hoffmann KH (2003) Threshold accepting as limit case for a modified Tsallis statistics. *Appl Math Lett* 16:27–31
- Garey MR, Johnson DS (1979) Computers and intractability. WH Freeman, San Francisco, CA
- Ge H, Du W, Qian F (2007) A hybrid algorithm based on particle swarm optimisation and simulated annealing for job shop scheduling. In: Proceedings of ICNC 2007. Third International Conference on Natural Computation, vol 3, Haikou, China, August 2007. IEEE Computer Society, Washington, pp 715–719
- Gelfand SB, Mitter SK (1989) Simulated annealing with noisy or imprecise measurements. *J Opt Theory Appl* 69:49–62
- Gendreau M, Potvin JY (2005) Tabu search. In: EK Burke, G Kendall (eds) Introductory tutorials in optimisation, decision support and search methodology. Springer, New York, pp 165–186
- Glover F (1989) Tabu search part 1. *ORSA J Comput* 1:190–206
- Glover F, Greenberg HJ (1989) New approaches for heuristic search: a bilateral link with artificial intelligence. *Eur J Oper Res* 39:119–130
- Gogos C, Alefragis P, Housos E (2008) A multi-staged algorithmic process for the solution of the examination timetabling problem. In: Burke EK, Gendreau M (eds) The 7th international conference on the practice and theory of automated timetabling, Montreal, Canada, August 2008
- Goldstein L, Waterman MS (1988) Neighbourhood size in the simulated annealing algorithm. *Am J Math Manag Sci* 8:409–423
- Gomes AM, Oliveira JF (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *Eur J Oper Res* 171(3):811–829
- Greene JW, Supowit KJ (1986) Simulated annealing without rejected moves. *IEEE Trans Comput Aided Des CAD* 5:221–228
- Guo XP, Yang GK, Zhiming W, Huang ZH (2006) A hybrid fine-tuned multi-objective memetic algorithm. *IEICE Trans Fundam Electron Commun Comput Sci* E89A(3):790–797
- Getjahr WJ, Pflug GCH (1996) Simulated annealing for noisy cost functions. *J Global Optimisation* 8(1):1–13
- Hajek B (1988) Cooling schedules for optimal annealing. *Math Oper Res* 13:311–329
- Hamacher K (2006) Adaptation in stochastic tunnelling global optimisation of complex potential energy landscapes. *Europhys Lett* 74:944–950
- Hamacher K, Wenzel W (1999) Scaling behaviour of stochastic minimisation algorithms in a perfect funnel landscape. *Phys Rev E* 59:938–941
- Hansen P, Mladenovic N (2005) Variable neighbourhood search. In: Burke EK, Kendall G (eds) Search methodologies. Springer, New York, pp 211–238
- Henderson D, Jacobson SH, Johnson AW (2003) The theory and practice of simulated annealing. In: Glover F, Kochenberger GA (eds) The handbook of metaheuristics, International series in operations research and management science, vol 57. Springer, New York
- Huang MD, Romeo F, Sangiovanni-Vincentelli AL (1986) An efficient general cooling schedule for simulated annealing. In: Proceedings of IEEE international conference on computer aided design, Santa Clara, CA, November 1986. IEEE Computer Society, Washington, pp 381–384
- Jacob D, Raben A, Sarkar A, Grimm J, Simpson L (2008) Anatomy-based inverse planning simulated annealing optimization in high-dose-rate prostate brachytherapy significant dosimetric advantage over other optimization techniques. *Int J Radiat Oncol Biol Phys* 72(3):820–827
- Johnson DS, Aragon CR, McGeoch LA, Schevon C (1989) Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Oper Res* 37:865–892
- Johnson DS, Aragon CR, McGeoch LA, Schevon C (1991) Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Oper Res* 39:378–406
- Jwo W-S, Liu C-W, Liu C-C, Hsiao Y-Y (1995) Hybrid expert system and simulated annealing approach to optimal reactive power planning. *IEE Proc Generation, Transm Distribution* 142(4):381–385
- Kalivas JH (1992) Optimization using variations of simulated annealing. *Chemometrics Intell Lab Syst* 15(1):1–12
- Karp RM (1972) Reducibility amongst combinatorial problems. In: Miller RE, Thatcher JW (eds)

- Complexity of computer computations. Plenum Press, New York, pp 85–103
- Kern W (1986) On the depth of combinatorial optimisation problems. University of Koln Technical Report 8633
- Kirkpatrick CD, Gellat CD, Vecchi MP (1983) Optimisation by simulated annealing. *Science* 220:671–680
- Kubicky CD, Yeh BM, Lessard E, Joe BN, Speight JL, Pouliot J, Hsu I-C (2008) Inverse planning simulated annealing for magnetic resonance imaging-based intracavitary high dose-rate brachytherapy for cervical cancer. *Brachytherapy* 7(3):242–247
- Kostuch PA (2004) The university course timetabling problem with a 3-phase method. In: Burke EK, Trick M (eds) The practice and theory of automated timetabling V. Lecture notes in computer science, vol 3616. Springer-Verlag, Berlin, pp 109–125
- Lin S, Yu VF, Chou S-Y (2008) Solving the truck and trailer problem based on a simulated annealing heuristics. *Comput Oper Res*, Available online 17-4-2008 (corrected proof)
- Liu X, Pardalos PM, Rajasekaran S, Resende MGC (2000) A GRASP for frequency assignment in mobile radio networks. In: Badrinath BR, Hsu F, Pardalos PM, Rajasekaran S (eds) Mobile networks and computing. DIMACS series on discrete mathematics and theoretical computer science, vol 52. American Mathematical Society, Providence, RI, pp 195–201
- Lundy M, Mees A (1986) Convergence of an annealing algorithm. *Math Programming* 34:111–124
- Marsh RE, Riauka TA, McQuarrie SA (2007) Use of a simulated annealing algorithm to fit compartmental models with an application to fractal pharmacokinetics. *J Pharm Pharm Sci* 10(2):167–178
- Merlot LTG, Boland N, Hughes BD, Stuckey PJ (2003) A hybrid algorithm for the examination timetabling problem. *Lect Notes Comput Sci* 2740:207–231
- Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculation by fast computing machines. *J Chem Phys* 21:1087–1091
- Mitra D, Romeo F, Sangiovanni-Vincentelli AL (1986) Convergence and finite time behaviour of simulated annealing. *Adv Appl Probability* 18:747–771
- Monem MJ, Namdarian R (2005) Application of simulated annealing (SA) techniques for optimal water distribution in irrigation canals. *Irrigation Drainage* 54(4):365–373
- Morgenstern C, Shapiro H (1989) Chromatic number approximation using simulated annealing. Technical Report CS86-1, Department of Computer Science, University of New Mexico
- Morton GC, Sangrecha R, Halina P, Loblaw A (2008) A comparison of anatomy-based inverse planning with simulated annealing and graphical optimization for high-dose-rate prostate brachytherapy. *Brachytherapy* 7(1):12–16
- Moscato P, Fontanari JF (1990) Stochastic versus deterministic update in simulated annealing. *Phys Lett A* 146:204–208
- Nissen V (1995) An overview of evolutionary algorithms in management applications. In: Biethahn J, Nissen V (eds) Evolutionary algorithms in management applications. Springer Verlag, New York, pp 44–97
- Ogbu FA, Smith DK (1990) The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem. *Comput Oper Res* 17:243–253
- Osman IH (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann Oper Res* 41:421–451
- Outeiro MT, Chibante R, Carvalho AS, de Almeida AT (2008) A parameter optimized model of a proton exchange membrane fuel cell including temperature effects. *J Power Sources* 185(2):952–960
- Pakhira MK (2003) A hybrid genetic algorithm using probabilistic selection. *J Inst Eng (India)* 84: 23–30
- Paya I, Yepes V, Gonzalez-Vidosa F, Hospitaler A (2008) Multiobjective optimization of concrete frames by simulated annealing. *Comput Aided Civil Infrastructure Eng* 23(8):596–610
- Pedamallu CS, Ozdamar L (2008) Comparison of simulated annealing, interval partitioning and hybrid algorithms in constrained global optimisation. In: Siarry P, Michalewicz Z (eds) Advances in metaheuristics for hard optimization 2008. Natural computing series. Springer, Berlin, pp 1–22
- Penna TJP (2008) Travelling salesman problem and Tsallis statistics. *Phys Rev E* 51:R1–R3
- Perea C, Alcaca J, Yepes V, Gonzalez-Vidosa F, Hospitaler A (2008) Design of reinforced concrete bridge frames by heuristic optimization. *Adv Eng Software* 39(8):676–688
- Rodriguez-Tello E, Hao J-K, Torres-Jimenez J (2008) An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. *Comput Oper Res* 35:3331–3346
- Ropke S, Pisinger D (2006) An adaptive large neighbourhood search heuristics for the pickup and delivery problem with time windows. *Transportation Sci* 40:455–472
- Salamon P, Suibani P, Frost R (2002) Facts, conjectures and improvements for simulated annealing. SIAM Monographs on Mathematical Modeling and Computation 7, Society for Industrial and Applied Mathematics
- Santé-Riveira I, Boullón-Magán M, Crecente-Maseda R, Miranda-Barrós D (2008) Algorithm based on simulated annealing for land-use allocation. *Comput Geosci* 34(3):259–268

- Sechen D, Braun D, Sangiovanni-Vincentelli A (1988) Thunderbird: a complete standard cell layout package. *IEEE J Solid State Circuits* 23:410–420
- Seçkiner SU, Kurt M (2007) A simulated annealing approach to the solution of job rotation scheduling problems. *Appl Math Comput* 188(1):31–45
- Szu H, Hartley R (1987) Fast simulated annealing. *Phys Lett A* 122:157–162
- Tavakkoli-Moghaddam R, Safaei N, Kah MMO, Rabbani M (2007) A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *J Franklin Inst* 344(5): 406–425
- Teitz MB, Bart P (1968) Heuristics methods for estimating the generalised vertex median of a weighted graph. *Oper Res* 16:955–961
- Tewari S, Arnold J, Bhandarkar SM (2008) Likelihood of a particular order of genetic markers and the construction of genetic maps. *J Bioinform Comput Biol* 6(1):125–162
- Thompson JM, Dowsland KA (1998) A robust simulated annealing based examination timetabling system. *Comput Oper Res* 25:637–648
- Thompson JM, Dowsland KA (1996) General cooling schedules for a simulated annealing based timetabling system. In: Burke EK, Ross P (eds) Practice and theory of automated timetabling. Lecture notes in computer science, vol 1153. Springer-Verlag, Berlin
- Tiourine S, Hurkens C, Lenstra JK (1995) An overview of algorithmic approaches to frequency assignment problems. Technical report, EUCLID CALMA project, Eindhoven University of Technology
- Tovey CA (1988) Simulated simulated annealing. *Am J Math Manag Sci* 8:389–407
- Triki E, Collette Y, Siarry P (2005) A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *Eur J OR* 166:77–92
- Tsallis C, Stariolo DA (1996) Generalized simulated annealing. *Phys A* 233:395–406
- Tuga M, Berretta R, Mendes A (2007) A hybrid simulated annealing with Kempe chain neighbourhood for the university timetabling problem. In: Lee R, Chowdhury M, Ray S, Lee T (eds) 6th IEEE/ACIS Conference Proceedings Computer and Information Science 2007, Melbourne, Australia, July 2007. IEEE Computer Society, Washington, pp 400–405
- Vakharia AJ, Chang Y-L (1990) A simulated annealing approach to scheduling a manufacturing cell. *Naval Res Logistics* 37:559–577
- Van Breedam A (1995) Improvement heuristics for the vehicle routing problem based on simulated annealing. *Eur J Operational Res* 86(3):480–490
- Van Hentenryck P, Vergados Y (2007) Population-based simulated annealing for traveling tournaments. Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Vancouver, Canada, AAAI Press, pp 267–271
- Van Laarhoven PJM, Aarts EHL (1987) Simulated annealing: theory and applications. Kluwer, Dordrecht, The Netherlands
- Wales DJ, Scheraga HA (1999) Chemistry: global optimisation of clusters, crystals and biomolecules. *Science* 285:1368–1372
- Wishart JD, Dong Z, Secanell MM (2006) Optimization of a PEM fuel cell system for low-speed hybrid electric vehicles. In: Proceedings of the ASME Design Engineering Technical Conference 2006, Philadelphia, PA, September 2006
- Wong DF, Leong HW, Liu HW (1998) Simulated annealing for VLSI design. The Springer International Series in Engineering and Computer Science, vol 42. Springer, Berlin
- Wright M (1991) Scheduling English cricket umpires. *J OR Soc* 42:447–452
- Wright M (1996) School timetabling using heuristic search. *J OR Soc* 47:347–357
- Wright M (2001) Subcost-guided search – experiments with timetabling problems. *J Heuristics* 7:251–260
- Yu P, Dai M-G, Wang J-L, Wu J-S (2008) Joint inversion of gravity and seismic data based on common gridded model with random density and velocity distributions. *Chinese J Geophys* 51(3):845–852
- Zolfaghari S, Liang M (2002) Comparative study of simulated annealing, genetic algorithms and tabu search for solving binary and comprehensive machine-grouping problems. *Int J Prod Res* 40:2141–2158

50 Evolvable Hardware

Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology, Brno,
Czech Republic
sekanina@fit.vutbr.cz

1	<i>Introduction</i>	1658
2	<i>Reconfigurable Technology</i>	1659
3	<i>First Approaches to Evolvable Hardware and Terminology</i>	1665
4	<i>Evolutionary Design Using Simulators</i>	1668
5	<i>Intrinsic Evolution</i>	1681
6	<i>Adaptive Hardware</i>	1692
7	<i>Other Areas</i>	1698
8	<i>Conclusions</i>	1699

Abstract

This chapter surveys the field of evolvable hardware. After a brief overview of the reconfigurable devices used in evolvable hardware, elementary principles of evolvable hardware, corresponding terminology, and open problems in the field are introduced. Then, the chapter is divided into three main parts: extrinsic evolution, intrinsic evolution, and adaptive hardware. Extrinsic evolution (i.e., evolution using simulators) covers evolutionary design of digital circuits, analog circuits, antennas, optical systems, and microelectromechanical systems (MEMS). Intrinsic evolution conducted in field programmable gate arrays (FPGAs), field programmable transistor arrays (FPTAs), field programmable analog arrays (FPAs), and some unconventional devices is discussed together with a description of the most successful applications. Examples of real-world adaptive hardware systems are also presented. Finally, an overview of major achievements and problems is given.

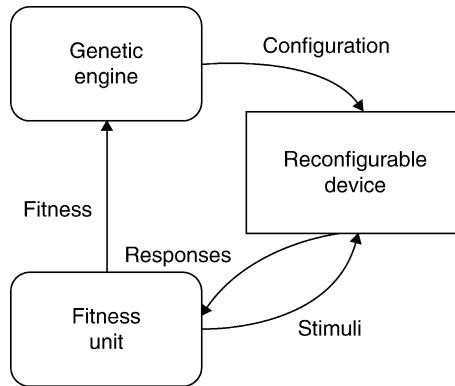
1 Introduction

At the beginning of the 1990s, researchers began using evolutionary algorithms (EAs) – population-based search algorithms (Eiben and Smith 2003) – to generate configurations for reconfigurable chips that could dynamically alter the functionality and physical connections of their circuits. This combination of EAs with reconfigurable devices spawned a new field called evolvable hardware (EHW) (Higuchi et al. 1993; de Garis 1993). Since that time, the EHW field has expanded and utilized many different combinations of EAs and biologically inspired algorithms with various reconfigurable devices including field programmable gate arrays (FPGAs), field programmable analog arrays (FPAs), reconfigurable antennas, mirrors, MEMS, and special reconfigurable materials. Research in the field of EHW can be split into the two related areas of evolutionary hardware design and adaptive hardware. While evolutionary hardware design is the use of EAs for creating innovative (and sometimes patentable) physical designs, the goal of adaptive hardware is to endow physical systems with some adaptive characteristics in order to allow them to operate successfully in a changing environment or under the presence of faults.

❶ *Figure 1* shows the basic principle of the evolvable hardware method: Electronic circuits that are encoded as bit strings (chromosomes, in the parlance of EAs) are constructed and optimized by the evolutionary algorithm in order to obtain the circuit implementation satisfying the specification given by the designer. In order to evaluate the candidate circuit, the new configuration of a reconfigurable device is created on the basis of the chromosome content. This configuration is uploaded into the reconfigurable device and evaluated for a chosen set of input stimuli. The fitness function, which reflects the problem specification, can include behavioral as well as nonbehavioral requirements. For example, the correct functionality is a typical behavioral requirement. As a nonbehavioral requirement, we can mention the requirement for minimum power consumption or minimum area occupied on the chip. Once the evaluation of the population of candidate circuits is complete, a new population can be produced. That is typically performed by applying genetic operators (such as mutation and crossover) on existing circuit configurations. High-scored candidate circuits have a higher probability that their genetic material (configuration bitstreams) will be selected for future generations. The process of evolution is terminated when a perfect solution is obtained or when a certain number of generations is evaluated.

Fig. 1

Evolvable hardware: Candidate configurations are generated by evolutionary algorithm, uploaded to a reconfigurable device and evaluated using the fitness function.



As the EA is a stochastic algorithm, the quality of resultant circuits is not guaranteed at the end of evolution. However, the method has two important advantages: (1) Artificial evolution can in principle produce intrinsic designs for electronic circuits, which lie outside the scope of circuits achievable by conventional design methods. (2) The challenge of conventional design is replaced by that of designing an evolutionary algorithm that automatically performs the design in a target place (e.g., in space). This may be harder than doing the design directly, but makes autonomy possible.

This chapter surveys the field of evolvable hardware. A more detailed explanation can be found in monographs (Zebulum et al. 2002; Thompson 1999; Greenwood and Tyrrell 2007; Higuchi et al. 2006; Sekanina 2004; Koza et al. 1999, 2003). The most important journals dealing with evolvable hardware are Genetic Programming and Evolvable Machines and IEEE Transactions on Evolutionary Computation. A lot of valuable material can be found in the conference proceedings: Evolvable Systems: From Biology to Hardware (Springer, 1996–), NASA/ESA Conference on Adaptive Hardware and Systems (IEEE, 2006–) and NASA/DoD Conference on Evolvable Hardware (IEEE, 1999–2005).

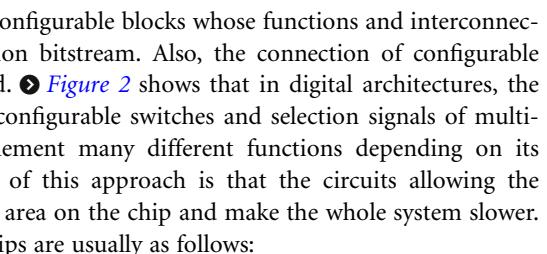
This chapter is organized as follows. [Section 2](#) presents reconfigurable devices that can be used for evolvable hardware. In [Sect. 3](#), elementary principles of evolvable hardware and corresponding terminology are introduced. [Section 4](#) surveys the area of extrinsic evolution, i.e., evolutionary hardware design using simulators. [Section 5](#) deals with the evolution conducted directly in physical hardware. Applications of evolutionary computing in the area of adaptive hardware are described in [Sect. 6](#). Some other relevant research dealing with bioinspired algorithms and hardware is discussed in [Sect. 7](#). Conclusions are given in [Sect. 8](#).

2 Reconfigurable Technology

Common electronic chips such as processors or application-specific integrated circuits have an architecture that remains fixed during the lifespan of a system. It is impossible to change the architecture at runtime when faults occur in circuit components or even when a little

modification introduced in a certain time could lead to a much better performance of the system. In recent years, we can observe a boom in the area of reconfigurable devices (Hauck and DeHon 2008). In comparison to fixed architectures, the structure and parameters of reconfigurable chips can be modified by writing configuration data to the configuration memory. While the first programmable devices (such as programmable logic arrays) used hundreds of bits to store the configuration, recent FPGAs require tens of megabytes to store their configuration data.

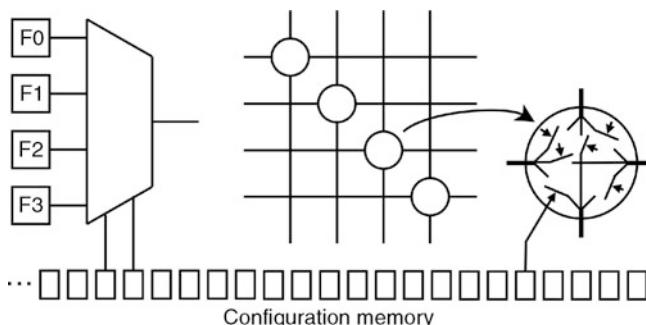
2.1 Why Reconfiguration?

Most reconfigurable devices consist of configurable blocks whose functions and interconnections are controlled by the configuration bitstream. Also, the connection of configurable blocks with I/O pins can be configured.  Figure 2 shows that in digital architectures, the configuration bits directly control the configurable switches and selection signals of multiplexers. Thus, a single chip can implement many different functions depending on its configuration. The main disadvantage of this approach is that the circuits allowing the “configurability” occupy a considerable area on the chip and make the whole system slower. The reasons for using reconfigurable chips are usually as follows:

- *Updating the firmware.* The reconfiguration extends the lifespan of a system. For example, when a new driver or peripheral device is introduced to a system, existing hardware could have a problem to communicate with it. However, if the system is implemented in a reconfigurable chip, the hardware can be updated by simply reprogramming the configuration memory. In this case, the reconfiguration is performed occasionally and only when the application is suspended.
- *Increasing the functional density.* The goal is to perform a complex task on a small chip and thus reduce the power consumption, size or weight of the application, even reduce the cost. The application has to be divided into modules whose configurations alternate on the chip. The reconfiguration is performed dynamically at runtime.
- *Increasing the reliability.* When a fault is detected and isolated, a system can be (in some cases) reconfigured to maintain its original function.

 Fig. 2

Multiplexer-based implementation of a reconfigurable functional unit (*left*) and a configurable interconnecting network implemented by configuration switches (*right*).



- *Adapting hardware.* The goal is to dynamically create electronic circuits that are optimized for a given task, time, and location of the chip.
- *Shortening the design time.* Creating a configuration for a reconfigurable device usually takes much less time than building a new application-specific chip.

The possibility of reconfiguration is typical for digital architectures. However, reconfigurable devices are now available in the areas of analog circuits, antennas, mirrors, molecular electronics, and others. Major digital and analog reconfigurable devices used in the evolvable hardware field will be briefly described.

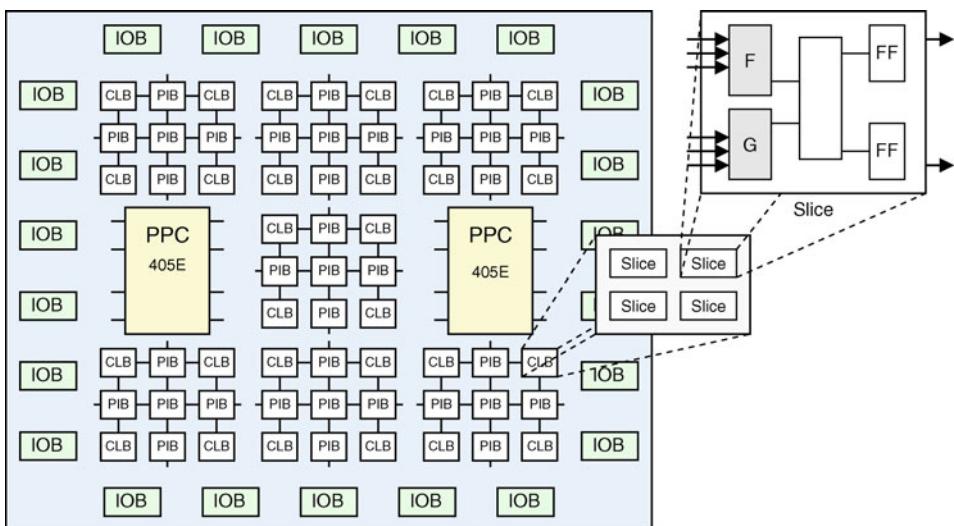
2.2 Field Programmable Gate Arrays

- ➊ *Figure 3* shows a typical architecture of a Xilinx FPGA (Xilinx Inc. 2009). It is a two-dimensional array of reconfigurable resources that include configurable logic blocks (CLB), programmable interconnect blocks (PIB), and reconfigurable I/O blocks (IOB). A CLB consists of so-called slices; each of them contains the function generators implemented using 3-, 4- or 6-input look-up tables, flip-flops, and some additional logic. The configuration bitstream is stored in the configuration SRAM memory.

The FPGA chips differ in the amount and type of resources available on the chip. The most advanced FPGAs contain more than 10 thousand CLBs and integrate, in addition to CLBs, various embedded hard cores such as SRAM memories, fast multipliers, processors, gigabit interfaces, and PCI interfaces (see PowerPC processors in ➋ *Fig. 3*). Because the existence of these cores has been identified as important to designers in the past, it is reasonable to integrate them as hard cores on the chip instead of implementing them using CLBs and other resources. Current FPGAs can compete with application-specific integrated circuits

➋ **Fig. 3**

FPGA Virtex II Pro architecture that contains two PowerPC processors.



(ASICs) in many domains, for example, in applications of advanced signal processing or embedded systems.

Some FPGAs support a dynamic partial reconfiguration, which means that some parts of the FPGA can be reconfigured while remaining parts of the FPGA perform computation. In the following sections it will be seen that the possibility of partial reconfiguration is crucial for evolvable hardware. FPGAs can be configured either externally or internally. In the case of external reconfiguration, the configuration bit stream is copied to the configuration memory from an external (flash) memory. The internal reconfiguration is available in Xilinx Virtex FPGAs via the internal access configuration port (ICAP), which allows for reading and modifying the FPGA configurations by circuits created directly in the same FPGA.

The goal of digital circuit design is to provide such an implementation of the circuit which satisfies the user specification and is available in a reasonable time. As the conventional circuit design process with FPGAs is very similar to programming, the resultant system can be obtained relatively quickly. The designer has to describe the circuit structure or behavior using a hardware description language (such as VHDL or Verilog). Then, the source code is almost automatically transformed into the configuration bit stream for a particular FPGA. The transformation, which includes the synthesis, placement and routing is performed by CAD tools. This process can be constrained using various requirements, for example, the maximum delay of the circuit can be specified. Also, it is possible to simulate intermediate results of the transformation, modify the original source code when needed, and optimize the design. In most cases, the format of the configuration bit stream is not documented for the designer (with the exception of the XC6200 family). In this way, manufacturers protect their know-how and prevent the designers from physically destroying the FPGA chip.

2.3 Field Programmable Transistor and Analog Arrays

Reconfigurable analog circuits allow, in fact, a software control of analog circuits. In comparison with FPGAs, reconfigurable analog circuits contain fewer configurable blocks and operate at lower frequencies. The reconfiguration is usually based either on configurable transistor switches, analog multiplexers, switching capacitors, or operational transconductance amplifiers (OTAs). Reconfigurable analog chips were introduced much later than FPGAs.

2.3.1 FPTA-2

The field programmable transistor array (FPTA-2) developed at NASA JPL uses transistor switches to implement the reconfiguration (Stoica et al. 2002). The FPTA-2 can implement analog, digital, and mixed signal circuits. The architecture of the FPTA consists of an 8×8 array of reconfigurable cells. Each cell has a transistor set as well as a set of programmable resources, including programmable resistors and static capacitors. The reconfigurable circuitry consists of 14 transistors connected through 44 switches in each cell. ◉ [Figure 4](#) provides a detailed view of the reconfigurable transistor array cell. A total of 5000 bits are used to program the whole chip. The pattern of interconnection between cells is similar to the one used in commercial FPGAs: each cell interconnects with its north, south, east, and west neighbors. Another FPTA was developed at the University of Heidelberg (Langeheine 2005). This chip enables developing circuits directly at the transistor level; the designer can select the transistor type (P/N), parameters (channel length and size), and interconnection.

2.3.2 Anadigm FPAA

The reconfiguration of field programmable analog arrays (FPAA) is typically based on either switched capacitors or OTAs. Switched capacitors perform the function of configurable resistors. [Figure 5](#) shows the principle: capacitor C is connected between two switches controlled by signals S_1 and S_2 . The switches are implemented using unipolar transistors and signals S_1 and S_2 are nonoverlapping clocks. The charge Q over one clock period transferred to the capacitor C is given by

$$Q = C(V_1 - V_2) \quad (1)$$

and the average current associated to this charge is

$$I_a = C(V_1 - V_2)/T \quad (2)$$

where T denotes the clock period. The value of the equivalent resistor can be calculated as

$$R = (V_1 - V_2)/I_a = T/C \quad (3)$$

Fig. 4

A single cell of the FPTA-2 chip (Stoica et al. 2002). Configuration switches are shown as circles.

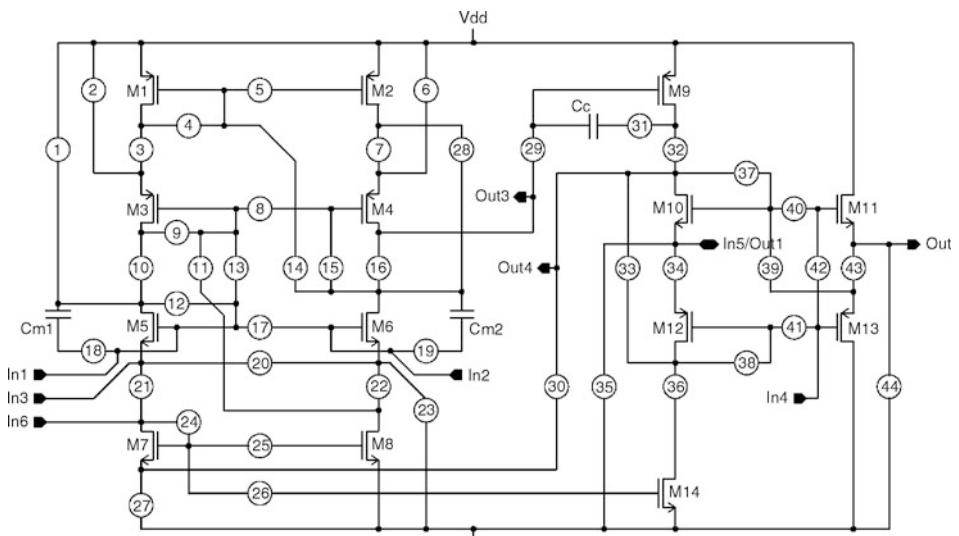
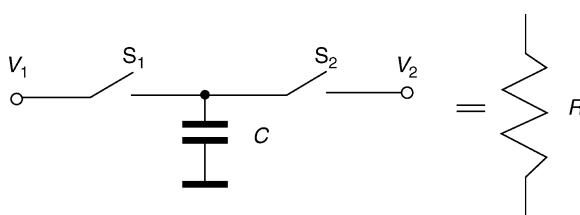


Fig. 5

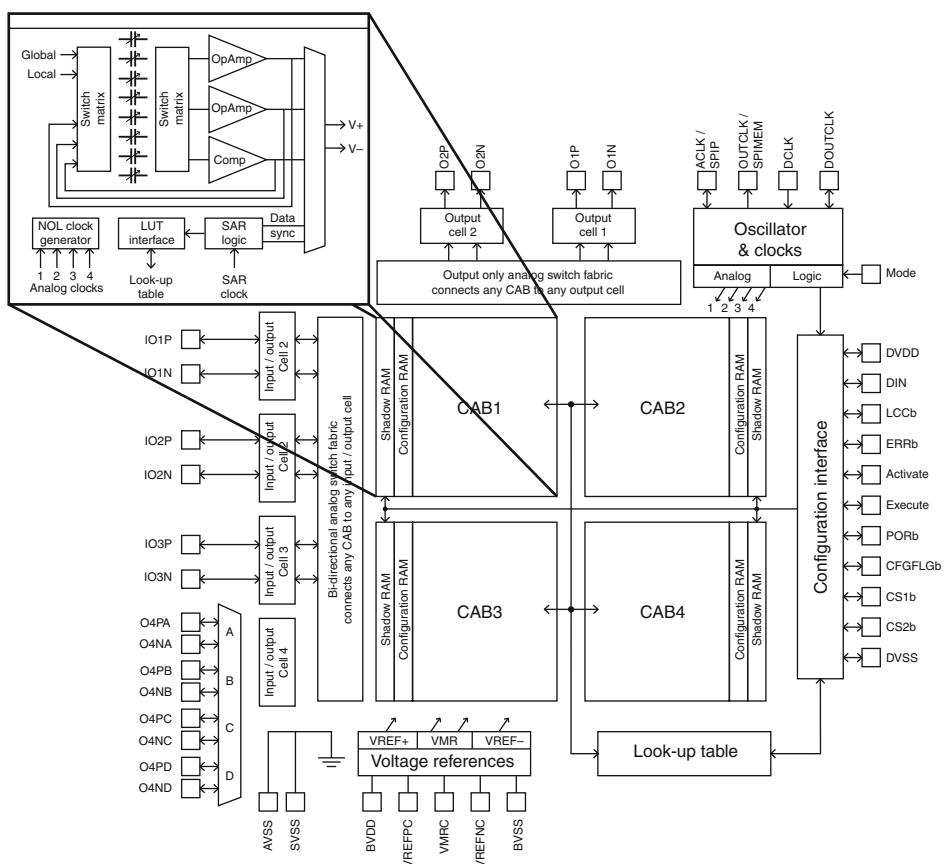
Switched capacitor and its equivalent resistor.



This resistor value can be controlled by the switching frequency $f = 1/T$. In comparison to conventional resistors, switching capacitors are advantageous in terms of linearity, dynamic range, precision, and size on the chip. As f can be controlled from software, analog circuits (such as filters and oscillators) can be easily tuned. A disadvantage might be that circuits containing switched capacitors operate in the discrete domain, that is, there is a limit on the possible operation frequency that is determined by f .

The commercially available Anadigm AN221E04 FPAA (Anadigm 2007), developed using switching capacitors, is an array of four configurable analog blocks (CAB), each of which contains two OpAmps, a comparator, and a successive approximation register (SAR) that performs 8-bit analog-to-digital conversion of signals. The device also contains one programmable look-up table that can be used to store information for the generation of arbitrary waveforms, and is shared among the CABs. The FPAA architecture is shown in Fig. 6. The configuration bit stream is stored in SRAM and the maximum switching frequency is 16 MHz.

Fig. 6
Anadigm AN221E04 FPAA (Anadigm 2007).



2.3.3 Operational Transconductance Amplifiers

Another approach to software control of analog circuits is based on operational transconductance amplifiers (OTAs). A typical OTA produces a current output I_0 that linearly depends on an input voltage present at both an inverting input (V_-) and non-inverting input (V_+):

$$I_0 = -g_m(V_+ - V_-) \quad (4)$$

where g_m is the transconductance of the circuit. The transconductance can be predefined or programmable using an external biasing current input I_{set} (see Fig. 7). Biasing currents for OTAs are generated using DACs. Ideally, the circuit has infinite values for both the input and output impedances. OTAs are the main building blocks of continuous time filters in which the transconductance (and thus the frequency characteristics) can be controlled externally. An example of an FPAA that utilizes configurable OTAs is that developed at the University of Freiburg (Henrici et al. 2007).

2.4 Other Reconfigurable Devices

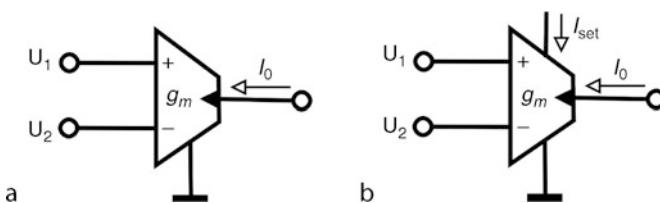
Evolutionary design has been performed for many other reconfigurable devices. Special ASICs include reconfigurable chips: POETic (Moreno et al. 2005), RISA (Greensted and Tyrrell 2007), PAMA (Zebulum et al. 2002), and REPOMO32 (Sekanina et al. 2009). Among more exotic examples, we find reconfigurable mirrors (Loktev et al. 2003), reconfigurable nanosystems (Tour 2003), reconfigurable antennas (Linden 2001) and reconfigurable liquid crystals (Harding et al. 2008). Whenever needed, detailed descriptions of these devices will be given in the sections below devoted to particular examples of evolvable systems.

3 First Approaches to Evolvable Hardware and Terminology

Practically, since the very beginning of the research in evolutionary computation, evolutionary algorithms have been applied in the area of hardware optimization. One of the earliest works was the optimization of wiring configurations for a subsystem on a ballistic missile, published in 1963 (Lohn and Hornby 2006). Various monographs (Drechsler 1998; Larsson 2005) summarize the applications from the areas of electronic circuits design, diagnostics, and testing. Later, evolutionary algorithms were applied to generate complete circuit structures, that is, not only to optimize parameters of existing circuits.

Fig. 7

Operational transconductance amplifiers with fixed transconductance (a), transconductance programmable by I_{set} (b).



3.1 Higuchi's Experiment

In their pioneering work from 1992, Higuchi et al. used a genetic algorithm to create configurations for a physical electronic chip, the GAL16Z8 – a predecessor of modern FPGAs (Higuchi et al. 1993). This chip uses a programmable AND/OR array to implement digital circuits in the form of “sum of products.” Only a part of the chip was utilized resulting in a chromosome size of 108 bits. A relatively simple combinational circuit (6-input/1-output multiplexer) was evolved only on the basis of its behavioral input/output specification. In order to obtain the fitness value, the candidate circuit response was measured for all possible input patterns (i.e., 2^6 combinations). For each correct output value, the fitness score was incremented, resulting in the maximum fitness value of 64 points. In this experiment, candidate configurations were evaluated using the GAL simulator. Only the resulting configuration was uploaded into the physical chip at the end of the evolution and tested physically. The scenario that employs a circuit simulator is known as *extrinsic evolution* (sometimes called offline evolution). This experiment is considered the birth of evolvable hardware.

3.2 Thompson's Experiment

In 1996 and the following years, Adrian Thompson performed a series of experiments, which clearly demonstrated that there can be a significant difference in the resulting behavior if candidate circuits are evaluated not in a circuit simulator but directly in a physical reconfigurable device (Thompson 1996). Thompson used an FPGA XC6216 chip to evolve a tone discriminator – a circuit discriminating between square waves of 1 and 10 kHz. With 10×10 configurable blocks of an XC6216 device, the circuit has to output 5 V for one of the frequencies and 0 V for the other. The problem is that the evolved circuit has to discriminate between input periods five orders of magnitude longer than the propagation time of each configurable block. No external components or synchronization signals were provided as is typical for conventional solutions to this problem.  [Figure 8](#) shows the experimental setup.

The goal of the evolution is to maximize the difference between the average output voltage when the 1 and 10 kHz input signals are applied:

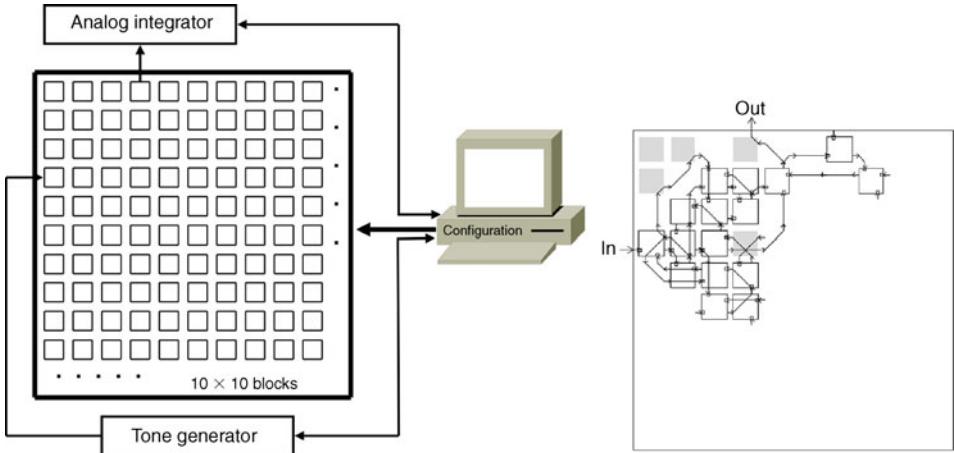
$$\text{fitness} = \frac{1}{5} |(k_1 \sum_{t \in S_1} i_t) - (k_2 \sum_{t \in S_{10}} i_t)| \quad (5)$$

where i_t represents the integrator output at the end of test tone t , S_1 is the set of 1 kHz tones, and S_{10} is the set of 10 kHz tones. k_1 and k_2 are experimentally determined constants.

Fortunately, the format of the configuration bit string of XC6216 was known to the users. Hence, Thompson could use a very efficient and fast dynamic reconfiguration to quickly generate new candidate circuits, though the experiment took 2–3 weeks because of time-consuming evaluation (around 5 s) of every individual.  [Figure 8](#) shows one of the evolved solutions. Only 21 out of 10×10 blocks contribute to the actual circuit behavior. In particular, gray-labeled blocks are very important in the solution. Although they are not connected, they also influence the circuit behavior. According to Thompson, these blocks interact with the circuit in some nonstandard ways. The evolved circuit was carefully analyzed using simulators, tested under different conditions, and on other XC6216 chips (Thompson et al. 1999). Surprisingly, Thompson was not able to create a reliable simulation model of the evolved circuit. In addition, he observed that the circuit works only in the chip used during evolution.

Fig. 8

Thompson's experiment setup (left) and utilization of logic blocks in evolved circuit (right, according to Thompson (1996)).



Although the circuit has a digital interface, its internal behavior is typical for analog circuits. It can be stated that the evolution was able to explore a full repertoire of behaviors available from the silicon resources provided to create the required behavior. This is an outstanding result, practically unreachable by means of conventional design methods. Hence, this experiment stimulated a lot of research in the evolvable hardware field.

The approach in which all candidate circuits are evaluated in a physical hardware is known as *intrinsic evolution* (sometimes called online evolution). The main advantage of the approach is that evolved circuits can be perfectly adapted to a given hardware substrate and external environment (temperature, radiation, ...) where the evolutionary design was carried out. On the other hand, intrinsically evolved solutions might not be robust and so portable to other reconfigurable devices. In order to avoid the portability problem, Stoica et al. introduced the so-called *mixtrinsic evolution*, in which some individuals are evaluated intrinsically and others extrinsically (Stoica et al. 2000).

3.3 Scalability Problems

The evolutionary circuit design is not competitive in all design problems because of the so-called *scalability problems*. From the viewpoint of the *scalability of representation*, the problem is that long chromosomes, which are usually required to represent complex solutions imply large search spaces that are typically difficult to search. In many cases, even a well-tuned parallel evolutionary algorithm running on a cluster of workstations fails to find an adequate solution in a reasonable time. In order to evolve large designs and simultaneously keep the size of the chromosome small, four main techniques have been developed:

- *Functional-level evolution* (Murakawa et al. 1996): Instead of gates and single-wire connections, the system is composed of complex application-specific functional blocks (such as adders, multipliers, and comparators) connected using multi-bit connections.

The selection of suitable functional blocks represents a domain knowledge that has to be included in the system.

- *Incremental evolution* (Torresen 1998, 2002): Target circuit is decomposed onto modules that are evolved separately. The decomposition strategy is a kind of domain knowledge that has to be supplied by the designer.
- *Development* (Kitano 1996; Koza et al. 1999): The above mentioned approaches employ a direct encoding of the target circuit (phenotype) in the chromosome (genotype). Hence, the size of the chromosome is proportional to the size of the circuit. Developmental approaches utilize indirect (generative) encodings that specify how to construct the target circuit. The phenotype is in fact constructed by a program that is encoded in the genotype. Designing these developmental encodings is not trivial and represents a domain knowledge that has to be supplied by the designer.
- *Modularization* (Walker and Miller 2008; Koza et al. 1999): Some EAs enable us to dynamically create and destroy reusable modules (subcircuits). The reuse of modules make the evolution easier even for large circuits.

Another problem is related to the fitness calculation time. In the case of the combinational circuit evolution, the evaluation time of a candidate circuit grows exponentially with the increasing number of inputs (assuming that all possible input combinations are tested in the fitness function). Hence, the evaluation time becomes the main bottleneck of the evolutionary approach when complex circuits with many inputs are evolved. This problem is known as the problem of *scalability of evaluation*. In order to reduce the time of evaluation, various techniques have been adopted:

- Only a subset of all possible input vectors is utilized. That is typical for the evolution of filters, classifiers, or robot controllers (Higuchi et al. 2006). Unfortunately, nobody can guarantee correct circuit behavior for those input combinations that were not used during evolution. Hence, evolved circuits have to be validated at the end of evolution using a test set – a representative set of input vectors that differ from the training set.
- In some cases, it is sufficient to evaluate only some structural properties (not the function) of candidate circuits that can be done with a reasonable time overhead. For example, because the testability of a candidate circuit can be calculated in a quadratic time, very large benchmark circuits (more than 1 million gates) with predefined testability properties were evolved (Pecenka et al. 2008).
- If the target system is linear, it is possible to perfectly evaluate a candidate circuit using a single input vector independently of the circuit complexity. Multiple-constant multipliers composed of adders, subtracters, and shifters were evolved for tens of outputs (Vasicek et al. 2008).

An obvious conclusion is that the perfect evaluation procedures are applicable only for small circuits or in very specific cases of large circuits. On the other hand, when more complex circuits have to be evolved, only an imperfect fitness calculation method may be employed due to time constraints.

4 Evolutionary Design Using Simulators

This section is devoted to extrinsic evolution of hardware. It covers the evolutionary design of digital circuits (at various levels), analog circuits, antennas, optical systems, and MEMS.

4.1 Gate-Level Evolution Using Cartesian Genetic Programming

Cartesian genetic programming (CGP), introduced by Miller and Thompson (2000), is a widely used method for extrinsic evolution of digital circuits and programs. CGP was originally defined for gate-level evolution; however, it can easily be extended for functional-level evolution. In its basic variant, candidate circuits are directly represented in the chromosome.

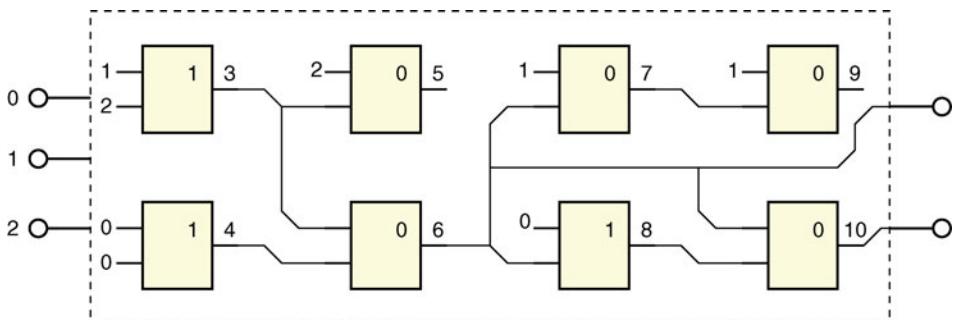
In CGP, a candidate entity (program or circuit) is modeled as an array of u (columns) \times v (rows) of programmable elements (gates). The number of inputs, n_i , and outputs, n_o , is fixed. Each node input can be connected either to the output of a node placed in the previous L columns or to some of the program inputs. The L -back parameter, in fact, defines the level of connectivity and thus reduces/extends the search space. For example, if $L = 1$ only neighboring columns may be connected; if $L = u$, full connectivity is enabled. Feedback is not allowed. Each node is programmed to perform one of n_n -input functions defined in the set Γ (n_f denotes $|\Gamma|$). As Fig. 9 shows, while the size of chromosome is fixed, the size of the phenotype is variable (i.e., some nodes are not used). Every individual is encoded using $u \times v \times (n_n + 1) + n_o$ integers. In order to make the search process easier, a common practice is to use some redundant elements in the CGP grid and a maximum value for the L -back parameter.

CGP operates with the population of $1 + \lambda$ individuals (typically, $\lambda = 5 - 20$). The initial population is randomly generated. Every new population consists of the best individual of the previous population and its λ offspring created by point mutation. If two or more individuals have received the same highest fitness score in the previous population, the individual that has not served as the parent in the previous population will be selected as the new parent. This strategy is used to ensure the diversity of population.

For the evolution of logic circuits, the fitness function is constructed in such a way that all possible input combinations are applied at the candidate circuit inputs, the outputs are collected and the goal is to minimize the difference between the obtained truth table and the required truth table. If evolution has found a solution that produces correct outputs for all possible input combinations, other parameters, such as the number of components or delay, can be optimized. The evolution is stopped when the best fitness value stagnates or the maximum number of generations is exhausted.

Fig. 9

An example of a candidate program. CGP parameters are as follows: $L = 3$, $u = 4$, $v = 2$, $\Gamma = \{\text{AND } (0), \text{OR } (1)\}$. Nodes 5 and 9 are not utilized. Chromosome: 1,2,1, 0,0,1, 2,3,0, 3,4,0, 1,6,0, 0,6,1, 1,7,0, 6,8,0, 6, 10. The last two integers indicate the outputs of the program.



The design of small multipliers is the most popular benchmark problem for gate-level evolution. Because the direct CGP approach is not scalable it works only for 4-bit multipliers and smaller multipliers.  [Table 1](#) summarizes the best-known results for various multipliers (Miller et al. 2000; Vassilev et al. 2000). For this class of circuits, CGP is able to create innovative designs, that is, circuits containing fewer gates than the best existing implementations, although only minimal domain knowledge has to be inserted into the representation.

CGP can be used to evolve compact implementations of such circuits whose efficient construction using conventional techniques is quite difficult. The construction of polymorphic circuits is a typical example (Stoica et al. 2001, 2002; Sekanina et al. 2008). Polymorphic circuits consist of conventional as well as polymorphic gates. Polymorphic gates are unconventional logic components that can switch their logic functions according to a changing environment. For example, a polymorphic gate controlled by the level of the power supply voltage exists, which operates as NAND when $V_{dd} = 3.3$ V and as NOR when $V_{dd} = 1.8$ V. Another gate operates as AND when the temperature is 27°C and as OR when the temperature is 125°C. Consider that the objective is to design a circuit that operates as a multiplier (the function will be denoted f_1) when polymorphic gates perform the NAND function and as a sorting network (the function will be denoted f_2) when polymorphic gates perform the NOR function. In order to use CGP for the design of polymorphic circuits with the minimum number of gates, the fitness function has to be modified so that a candidate circuit is evaluated in both modes. The new fitness value is defined as follows:

$$\text{fitness} = B_1 + B_2 + (u \cdot v - z) \quad (6)$$

where B_1 (resp. B_2) is the number of correct output bits for f_1 (resp. f_2) obtained as the response for all possible input combinations, z denotes the number of gates utilized in a particular candidate circuit, and $u \cdot v$ is the total number of programmable gates available. The last term is considered only if the circuit behavior is perfect in both modes; otherwise $uv - z = 0$.  [Table 2](#) summarizes the results obtained for various instances of the problem. In most cases, evolved circuits exhibit fewer gates than circuits created by multiplexing conventional implementations using polymorphic multiplexers (Sekanina et al. 2008).

4.2 Sequential Circuits

The evolution of sequential circuits is not as popular as the evolution of combinational circuits. However, evolutionary algorithms were successfully applied in the so-called state

 **Table 1**

The number of two-input gates in the best implementations of multipliers according to Miller et al. (2000) and Vassilev et al. (2000). CGP used with two-input gates AND, XOR, and AND with one input inverted, L -back is max., $\lambda = 4$

Multiplier	Best conventional	Best evolved	CGP array	Max. generations
2b × 2b	8	7	7 × 1	10k
3b × 2b	17	13	17 × 1	200k
3b × 3b	30	23	35 × 1	20M
4b × 3b	47	37	56 × 1	200M
4b × 4b	64	57	67 × 1	700M

assignment problem, the goal of which is to compute optimal binary codes for each symbolic state and construct the state transition table of a finite state machine (FSM). CGP is also used to implement combinational circuits computing the next state of the FSM. An improvement in the gate count (with respect to conventional techniques) was obtained for small circuits having up to 30 states (Nedjah and de Macedo 2005; Ali et al. 2004).

4.3 Incremental Evolution

The evolutionary design of larger gate-level circuits is usually performed using modular CGP (Walker and Miller 2008) or incremental evolution (Torresen 1998, 2002; Stomeo et al. 2006). A typical feature of circuits evolved modularly or incrementally is that they are fully functional, but no innovation is usually visible in their implementation. [Figure 10](#) shows basic approaches to the incremental evolution. The n -input/ m -output circuit can be

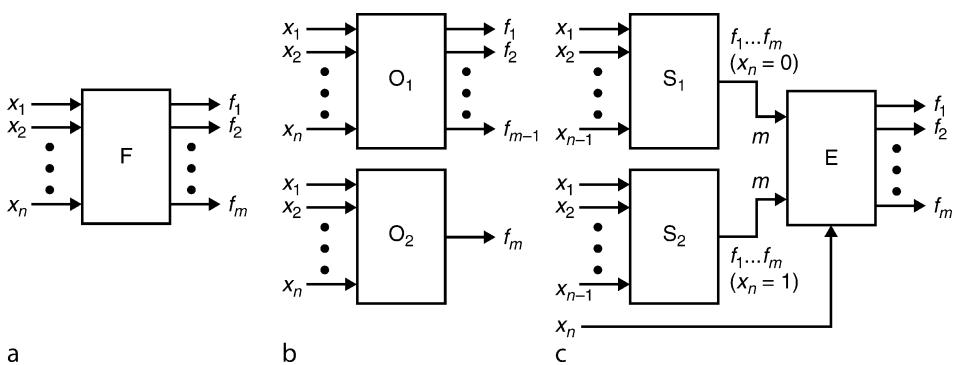
Table 2

Parameters of evolved polymorphic multipliers/sorters. Gates in “Gate set” are numbered as (1) NAND/NOR, (2) AND, (3) OR, (4) XOR, (5) NAND, (6) NOR, (7) NOT A, (8) NOT B, (9) MOV A, and (10) MOV B, where MOV denotes the identity operation. Population size is 15

Multiplier/sorter	$2b \times 2b/4b$	$3b \times 2b/5b$	$3b \times 3b/6b$	$4b \times 3b/7b$
$u \times v$	10×12	100×1	120×1	16×16
L-back	1	100	120	16
Mutation (genes)	1	2	4	4
Gate set	1, 2, 9, 10	1–4, 9, 10	1–10	1, 2, 9, 10
Runs	10	10	10	10
Successful runs	10	10	9	3
Generations (average)	52,580	854,900	26,972,648	62,617,151
Min. # of gates	23	30	52	113

Fig. 10

(a) Original circuit, (b) input-oriented decomposition, (c) output-oriented decomposition.



decomposed either according to the inputs or outputs. Both decompositions can be applied recursively and thus they can yield subcircuits that can be evolved directly using CGP, for example. Advanced variants of the incremental evolution further optimize evolved modules in order to maximize the number of gates shared among them and so to reduce the total number of gates. The most complex circuits evolved by the gate-level incremental evolution were reported in Stomeo et al. (2006): 6-bit multipliers, 17-input parity circuits, and some MCNC benchmarks (such as the alu4 with 14 inputs and 8 outputs). Incremental evolution was also used to design classifiers directly in the FPGA (see ➤ Sect. 5.1.2).

4.4 Developmental Approaches

As developmental approaches employ indirect encoding, the chromosome contains a program which is executed in order to construct the target circuit. The program can be implemented as an L-system, cellular automaton, if-then-else rules or, in general, as a program.

Arbitrarily large multipliers were constructed using evolved programs working in a grid of programmable nodes (Bidlo and Skarvada 2008). However, no innovation is observable in evolved multipliers in comparison to conventional multipliers. A similar approach was proposed to create arbitrarily large sorting networks, which exhibit slightly better properties (in the number of components and delay) than sorting networks created using conventional construction algorithms (Sekanina and Bidlo 2005). Papers of Gordon and Bentley (2002), Tufte and Haddow (2005), Tempesti et al. (2007), and Zhan et al. (2008) present more biologically plausible models of development for digital circuits. Although these approaches do not lead to innovative circuit designs, they are useful for investigation of principles of development, genetic regulatory networks, environmental interactions, and fault tolerance that could be useful for evolution and adaptation of large-scale digital circuits in future.

4.5 Functional-Level Evolution

CGP can easily be modified to evolve larger circuits. Instead of gates and single-wire connections, application-specific functions and multiple-bit connections are employed. The advantage is that while the size of the chromosome is similar to the gate-level evolution, the size of the phenotype can be arbitrarily large, depending on the building blocks used. Filters, predictors, and classifiers are typical circuits designed at this level. Because of the size of the candidate circuits, it is impossible to evaluate circuit responses for all possible input vectors. Hence, candidate circuits are evaluated using a training set during evolution and validated using a test set at the end of evolution. The goal of the evolution is to minimize the difference between the response of a candidate circuit and the target response. ➤ Section 5.1 will demonstrate real-world applications of functional-level evolution accelerated in FPGAs.

4.5.1 Digital Filter Evolution

Digital filter optimization and design is one of the most common applications of evolutionary computation in the hardware field. Digital filters are traditionally modeled using discrete-time systems theory (Ifeachor and Jervis 2002). A *discrete-time* system is essentially a mathematical

algorithm that takes an input sequence, $x(n)$, and produces an output sequence, $y(n)$. A *digital filter* is an example of a discrete-time system. A discrete-time system may be linear or nonlinear, time invariant or time varying. *Linear time-invariant* (LTI) systems form an important class of systems used in digital signal processing. The input–output relationship of an LTI system is given by the convolution sum

$$y(n) = \sum_{k=-\infty}^{+\infty} h(k)x(n-k) \quad (7)$$

where $h(k)$ is the *impulse response* of the system. The values of $h(k)$ completely define the discrete-time system in the time domain. A general infinite impulse response (IIR) digital filter is described by the equation

$$y(n) = \sum_{k=0}^N b_k x(n-k) - \sum_{k=1}^M a_k y(n-k) \quad (8)$$

The output samples, $y(n)$, are derived from current and past input samples, $x(n)$, as well as from current and past output samples. The task of the designer is to propose the values of the coefficients a_k and b_k and the sizes of vectors N and M . In finite impulse response (FIR) filters (☞ Fig. 11), the current output sample, $y(n)$, is a function only of past and present values of the input, that is,

$$y(n) = \sum_{k=0}^N b_k x(n-k) \quad (9)$$

The stability and linear phase are the main advantages of FIR filters. On the other hand, in order to get a good filter, many coefficients have to be considered in contrast to IIR filters. In general, IIR filters are not stable (because of feedback). FIR filters are algebraically more difficult to synthesize.

Conventional design methods are well developed and represent the approach to digital filter design widely adopted by industry. Digital filters are usually implemented either on DSPs or as custom circuits. Their implementation is based on multiply-and-accumulate structures. The multiplier is the primary performance bottleneck when implementing filters in hardware as it is costly in terms of area, power, and signal delay. Hence, multiplierless filters were introduced in which multiplication is reduced to a series of bitshifts, additions, and subtractions.

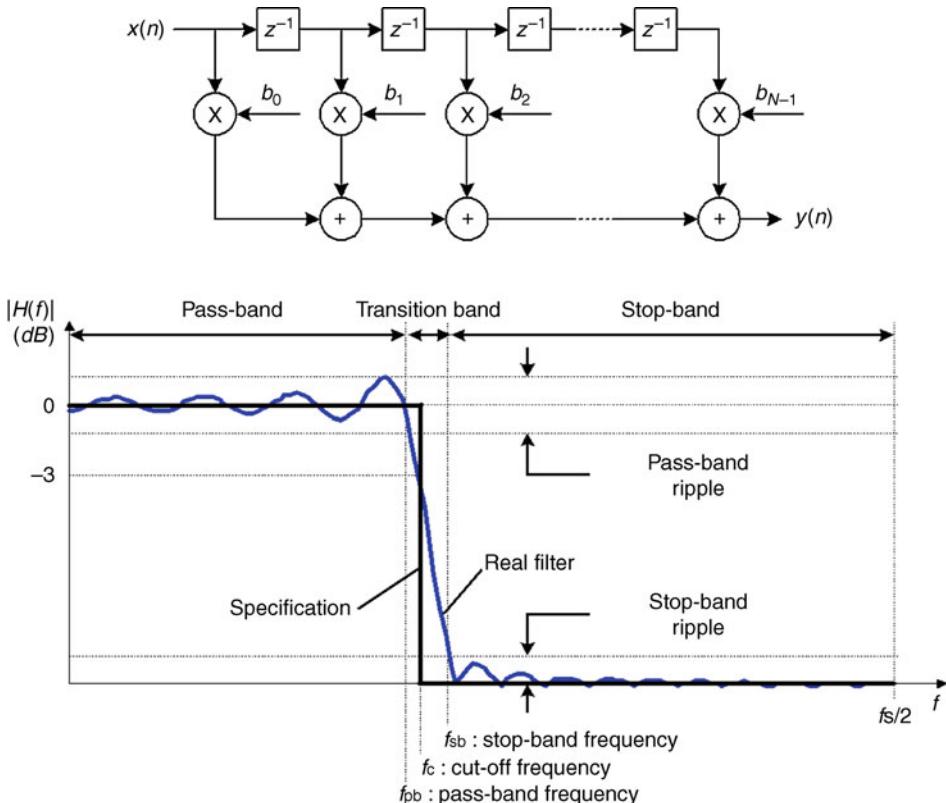
Evolutionary algorithms have been utilized either to optimize filter coefficients (Harris and Ifeachor 1995) or to design complete filter structures. In particular, structures of multiplierless filters were sought by many authors (Wade et al. 1994; Hounsell et al. 2004; Erba et al. 2001; Gwaltney and Dutton 2005). Regardless of filter representation, the fitness function is usually constructed in the frequency domain. Typically, a candidate's filter frequency characteristics are measured at k frequencies f_i (distributed logarithmically) and compared with target frequency characteristics that are derived from specification (☞ Fig. 11). The fitness value is calculated as

$$\text{fitness} = \sum_{i=0}^{k-1} [W(d(f_i), f_i) * d(f_i)] \quad (10)$$

where $d(x)$ is the absolute value of the difference between the target and observed values at frequency x , and $W(y, x)$ is the weighting for difference y at frequency x . The goal is to approximate the target characteristics as accurately as possible and better-than-conventional

Fig. 11

FIR filter structure (top) and frequency characteristics of a low pass filter: ideal vs. real (bottom).

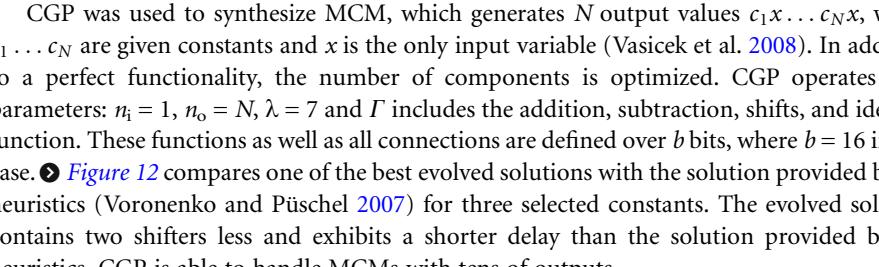


methods allow using the same resources. Additional objectives can be considered, for example, the circuit size or power consumption. The fitness calculation procedure is not trivial and can be time consuming.

4.5.2 Multiple Constant Multipliers

In digital filters, it is often required to multiply a single value x by N constants and so to generate N output products, that is, to implement a multiple constant multiplier (MCM). A multiplierless variant of MCM is such that it is composed of adders, subtractors, and shifters. Finding the optimal MCM, that is, the one with the fewest number of components is known to be NP-complete. A very efficient heuristic algorithm for the MCM problem was recently published (Voronenko and Püschel 2007).

Although MCMs are relatively complex circuits composed of high-level components, their interesting feature from the point of view of evolutionary design is that a candidate MCM can be perfectly evaluated by applying only one input vector (e.g., $x = 1$). The reason is that MCMs implement linear transforms. This feature is unique because it enables us to evolve very large circuits that are simultaneously perfectly evaluated. No training and test data sets are needed.

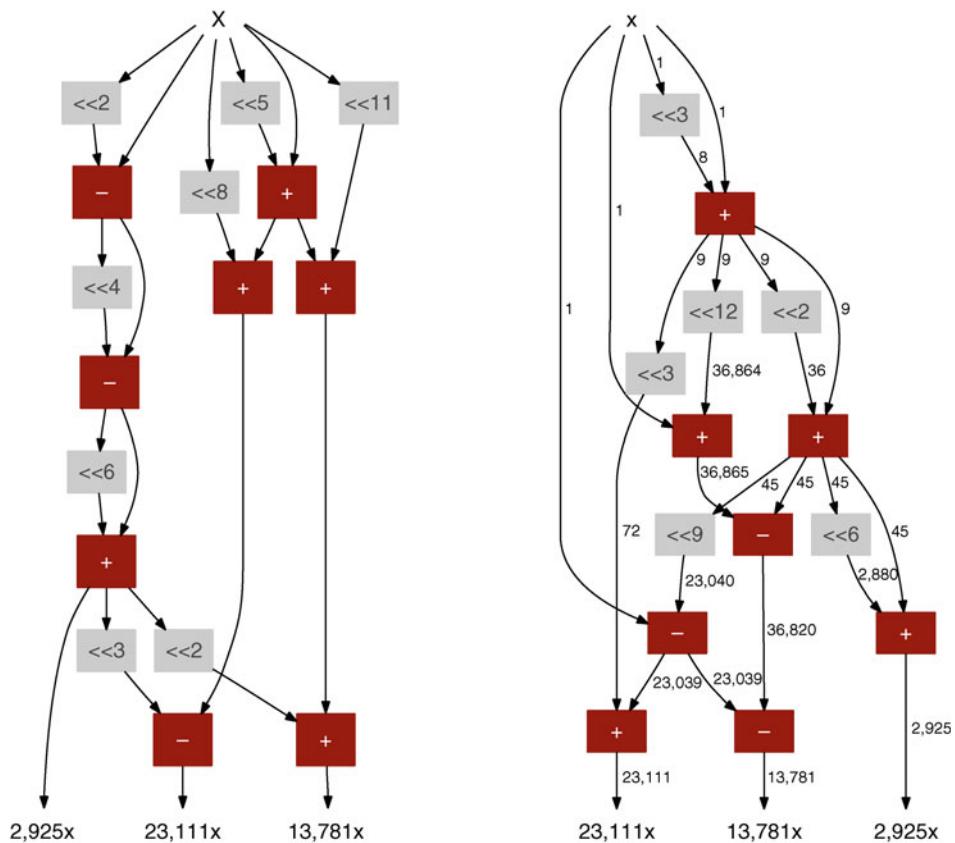
CGP was used to synthesize MCM, which generates N output values $c_1x \dots c_Nx$, where $c_1 \dots c_N$ are given constants and x is the only input variable (Vasicek et al. 2008). In addition to a perfect functionality, the number of components is optimized. CGP operates with parameters: $n_i = 1$, $n_o = N$, $\lambda = 7$ and Γ includes the addition, subtraction, shifts, and identity function. These functions as well as all connections are defined over b bits, where $b = 16$ in this case.  (Figure 12) compares one of the best evolved solutions with the solution provided by the heuristics (Voronenko and Püschel 2007) for three selected constants. The evolved solution contains two shifters less and exhibits a shorter delay than the solution provided by the heuristics. CGP is able to handle MCMs with tens of outputs.

4.6 Evolutionary Approaches in Diagnostics and Testing

Similarly to the circuit design, digital circuit diagnostics and testing have profited from *evolutionary optimization* for a long time. Among typical problems that were attempted in this area, we mention test vectors selection, test scheduling, scan-chain optimization, and test

 Fig. 12

MCM with three coefficients (2,925, 23,111, 13,781): according to Voronenko and Püschel (2007) (left), the best evolved solution (Vasicek et al. 2008) (right).



power consumption optimization under various constraints (Larsson 2005). Evolutionary design is applied in this area rarely. Two examples are briefly described below.

A method was described in Pecenka et al. (2008), which enables us to evolve large synthetic RTL benchmark circuits with a predefined structure and testability. These benchmark circuits are useful in a validation process of novel algorithms and tools in the area of digital-circuits testing. More specifically, as benchmark circuits are designed by means of the EA, they can contain constructions that do not usually appear in the circuits designed by classical design techniques. Thus, the use of evolved benchmark circuits in the process of evaluating new testability analysis tools can reveal problems of the tools that remain hidden when conventional benchmark circuits are used. With sizes up to 1.2 million gates, evolved benchmark circuits currently represent the most complex benchmark circuits with a known level of testability. Furthermore, these circuits are the largest that have ever been designed by means of evolutionary algorithms. This complexity was achieved because no requirement on the functionality was specified. Only some structural properties were measured in the fitness function that could be done in polynomial time.

Bernardi et al. presented a genetic programming-based approach to the construction of diagnosis-oriented software-based test sets for microprocessors (Bernardi et al. 2008). The methodology exploits existing manufacturing test sets designed for software-based self-test and improves them by using evolved test sets. Experimental results showed the feasibility, robustness, and effectiveness of the approach for diagnosing stuck-at faults on an Intel i8051 processor core. A similar approach was used to test peripheral cores.

4.7 Analog Circuits

Evolutionary algorithms can be used to optimize parameter values of circuit components in order to maximize circuit performance, reduce power consumption, or increase fabrication yield. However, in recent years, a lot of attention has been devoted in the evolvable hardware field to the evolutionary synthesis of complete structures of analog circuits. Analog circuits are evolved at various levels of description including passive circuits (composed of resistors, capacitors, and inductors), transistor-level circuits and circuits with operational amplifiers.

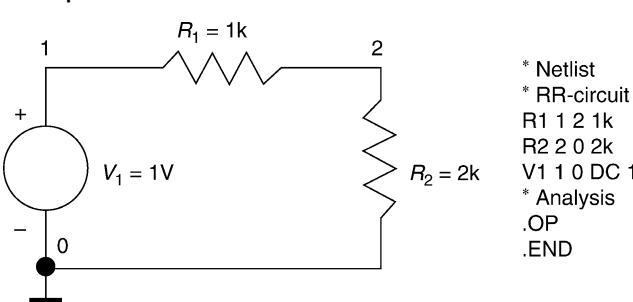
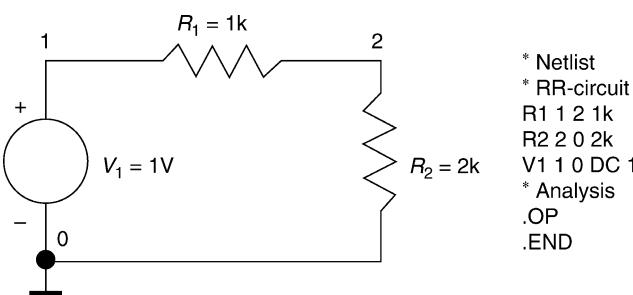
Candidate circuits are usually evaluated using the SPICE simulator, which is the most popular simulator for analog circuit designers.  Figure 13 shows a simple candidate circuit and its representation in a netlist – the input data format for SPICE. Before the circuit

Fig. 13

A circuit and its description in netlist.



evaluation can begin, the chromosome has to be transformed to a netlist and the input stimuli have to be specified. Then, the circuit is simulated using methods supported by SPICE, for example, DC analysis, AC analysis, transient analysis, or noise analysis can be performed. The results of the simulation are available in a text file. A typical fitness function takes the form of [Eq. 10](#), that is, the goal is to minimize the difference between the obtained signal and the target signal. The main disadvantage of the use of SPICE is that the fitness evaluation is relatively slow in comparison with measuring real circuits (several orders of magnitude reported in Stoica et al. (2002)).

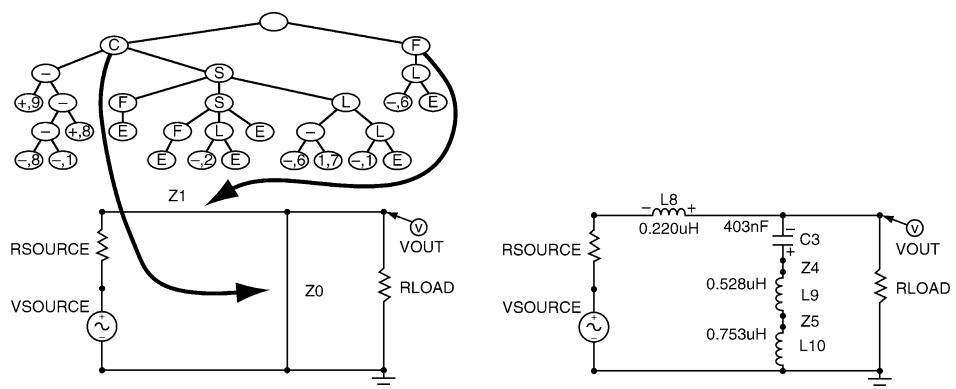
Analog circuits can be encoded in the chromosome either directly or indirectly. Direct encoding is similar to the structure of the netlist. For each component, parameters (e.g., resistances of the resistors) are given together with the description of its terminal connections. The chromosome is represented as a linear variable-length array of integers or real values (for parameter values).

Innovative implementations of analog circuits were obtained using indirect encoding, so-called *developmental genetic programming* introduced by Koza et al. (1999). In his method, a target circuit is subsequently constructed from a simple initial circuit (called the embryo) according to a program that is encoded in the chromosome. [Fig. 14](#) shows a typical embryonic circuit that contains fixed components (power supply, source resistor, and load resistor) and two modifiable wires Z0 and Z1. The developmental program ([Fig. 14](#)), which is encoded as a tree, contains two subtrees, each of them specifying the construction process for one of the modifiable wires. The functions used in the circuit-constructing program tree can be divided into five categories (Koza et al. 2003):

- Topology-modifying functions (e.g., series division (S), parallel division (P), and flip (F)) that modify the topology of the developing circuit
- Component-creating functions that insert components into the developing circuit (insert a capacitor (C), insert an inductor (L))
- Development-controlling functions that control the developmental process by which the embryo and its successors are converted into a fully developed circuit (e.g., the no-operation function, the end of construction function (E))

Fig. 14

Example of an embryonic circuit and a candidate program (left) used to create target circuit (right) (Koza et al. 1999).



- Arithmetic-performing functions that may appear in a value-setting subtree
- Automatically defined functions that enable certain substructures to be reused (including parameterized reuse)

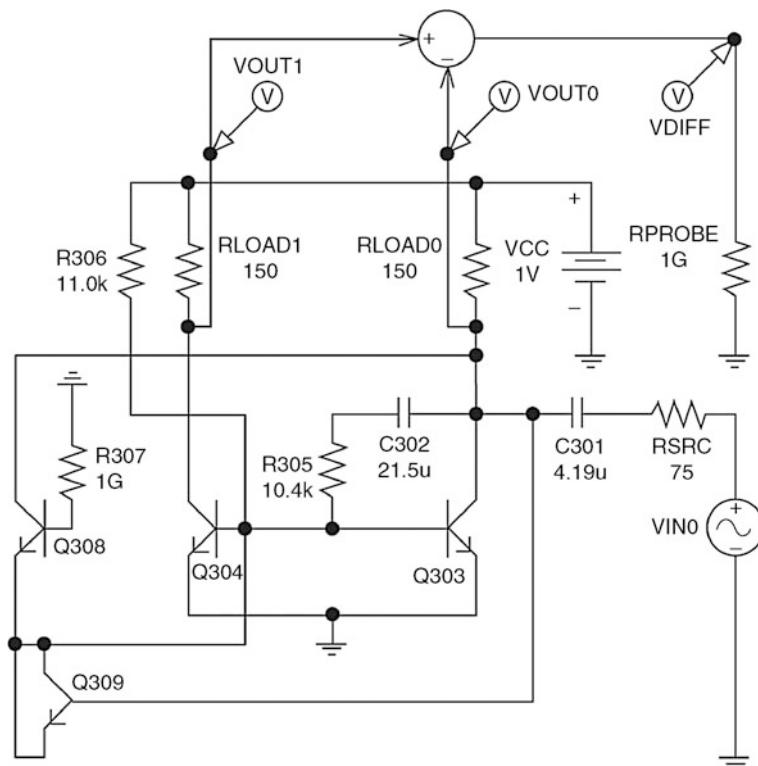
☞ *Figure 14* shows a fully developed circuit that was created from the embryo according to the given program. In order to obtain the fitness value of the circuit, this circuit has to be converted into a netlist and simulated.

Various books (Koza et al. 1999, 2003) present tens of circuits (passive analog circuits, circuits with transistors, circuits with operational amplifiers, controllers) evolved using the developmental genetic programming that can be classified as human-competitive designs, many times leading to rediscovering or even improving patented inventions. For example, an evolved analog circuit that duplicates the functionality of the low-voltage balun circuit (patented by Sang Gug Lee in 2001) is shown in ☞ *Fig. 15*. This design contains the key part of Lee's invention: a coupling capacitor that blocks DC (see capacitor C302 in ☞ *Fig. 15*).

Very high computational requirements are considered as the main disadvantage of the method. In contrast to CGP, it is typical for Koza's genetic programming that very large populations are used and only a few generations are produced. For example, Koza's team utilized two clusters of workstations, 1000 x Pentium II/350 MHz processor and 70 x DEC

▀ **Fig. 15**

Evolved analog circuit that duplicates the functionality of the low-voltage balun circuit that was patented by Sang Gug Lee in 2001 (according to Koza et al. 2003).



Alpha/533 MHz processor. For 36 tasks solved using GP on the clusters, the average population size is 3,350,000 individuals, 128.7 generations are produced on average and the average time to reaching a solution is 81.9 h (Koza et al. 2003).

Another criticism of Koza's genetic programming method for analog circuit synthesis comes from analog circuit designers who do not find evolved circuits trustworthy and suitable for fabrication. A solution to this problem was proposed by Gao, McConaghay, and Gielen who invented an analog circuit topology synthesis method called importance sampled circuit learning ensembles (ISCLEs). ISCLEs is a method that synthesizes circuit topologies that are novel in both functionality and topology, yet trustworthy, within reasonable CPU effort (Gao et al. 2008).

4.8 Antennas, Optical Systems, and MEMS

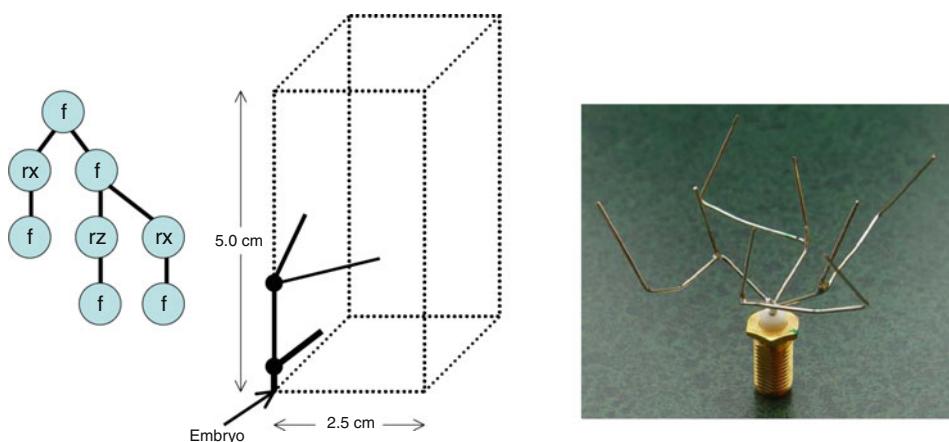
The evolutionary design of antennas, optical systems, MEMS, and other physical objects is strongly dependent on the existence of reliable simulators. With the increasing computer performance, sufficiently fast simulators have become available. These objects are usually evolved using generative encodings, similar to Koza's developmental genetic programming.

4.8.1 Antennas

Evolutionary optimization has been used in antenna design since the early 1990s (Linden 1997; Haupt and Werner 2007). Most of the work has been focused on optimizing parameters of a predetermined design as opposed to evolutionary design of the antenna's topology. One of the most famous examples of evolutionary design is the evolved antenna design for the flight hardware deployed on a NASA spacecraft for the Space Technology 5 (ST5) mission in 2006 (see Fig. 16). The antenna was evolved using generative encoding. The antenna's chromosome contains the instructions that control the construction of target shape. The process starts

Fig. 16

One of the candidate trees (left) that defines the process of antenna construction in a 3D space (middle) and evolved antenna ST5-3-10 (Hornby et al. 2006) (right).



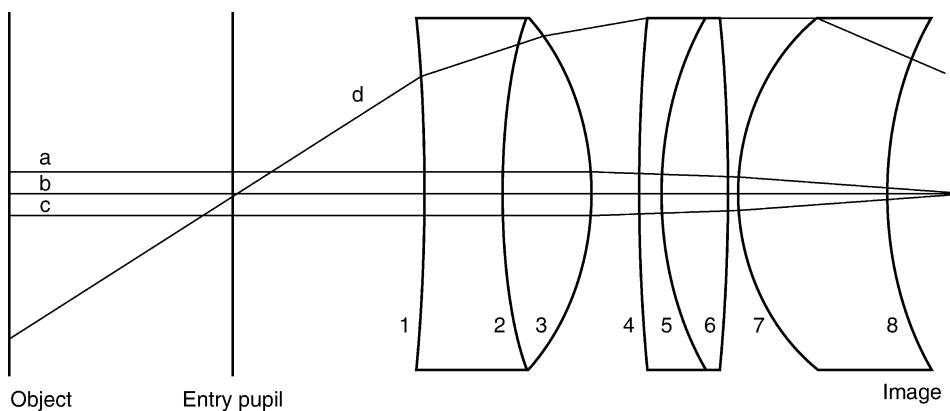
from a simple embryo – a small wire in this case. The instructions (such as F – move forward, R_i - rotate in direction i) can move and rotate the construction head in a 3D space of possible wire antennas. The paper by Hornby et al. (2006) explains that the fitness value of a candidate antenna is “the product of three scores involving the gain pattern, the VSWR, and pattern outliers. The gain pattern score compares the measured gain values across a range of elevation angles to the desired gain values and rewards antennas that exceed the requirements. VSWR is the ratio between the highest voltage and the lowest voltage in the signal envelope along a transmission line, with a ratio of 1 being ideal. The outlier score rewards antenna designs for having sample points with gains greater than zero.” Candidate antennas were simulated using NEC software. The evolved antenna has a number of advantages, in particular, low power consumption, short fabrication time, and good performance.

4.8.2 Optical Systems

Evolutionary algorithms have been used to optimize the parameters of optical systems with a prespecified layout and prespecified number of components. Koza has applied developmental genetic programming to create novel topologies for optical systems.  [Figure 17](#) shows one of the lens systems evolved using genetic programming that exhibits the same function as the patented solution of Koizumi and Watanabe, however its topology is different (Koza et al. 2005).

As the micron-meter resolution alignment of optical components usually takes a long time, to overcome this time problem the National Institute of Advanced Industrial Science and Technology in Japan (AIST) team proposed five systems that can automatically align the positions of optical components by evolutionary algorithms in very short times compared to conventional systems (Nosato et al. 2006): (1) an evolvable fiber alignment system, (2) an evolvable interferometer system, (3) an evolvable femtosecond laser system, (4) a wave-front correction system, and (5) a multiobjective adjustment system. In particular, the evolutionary algorithm was used to find positions of optical components that can be controlled by step motors. Proposed solutions have led to compacting the implementation of optical systems, reducing cost and lowering maintenance operations.

 **Fig. 17**
Evolved lens system according to Koza et al. (2005).



4.8.3 MEMS

Similarly to optical systems, evolutionary algorithms have been used to optimize parameters of MEMS, including the size, orientation, number of segments, etc. (Li and Antonsson 1998; Kamalian et al. 2002). Lohn, Kraus, and Hornby evolved a complete MEMS meandering resonator (i.e., topology and parameters) using a generative representation (Lohn et al. 2007). The JPL team has developed a tuning method for MEMS gyroscopes based on evolutionary computing to increase the accuracy of MEMS gyroscopes through electrostatic tuning. The method was successfully tested for the second generation of the JPL/Boeing post-resonator MEMS gyroscope (Keymeulen et al. 2006).

5 Intrinsic Evolution

This section surveys approaches to intrinsic evolution conducted directly in FPGAs, FPTAs, and FPAs. Evolution on unconventional devices is also discussed.

5.1 Evolvable FPGAs

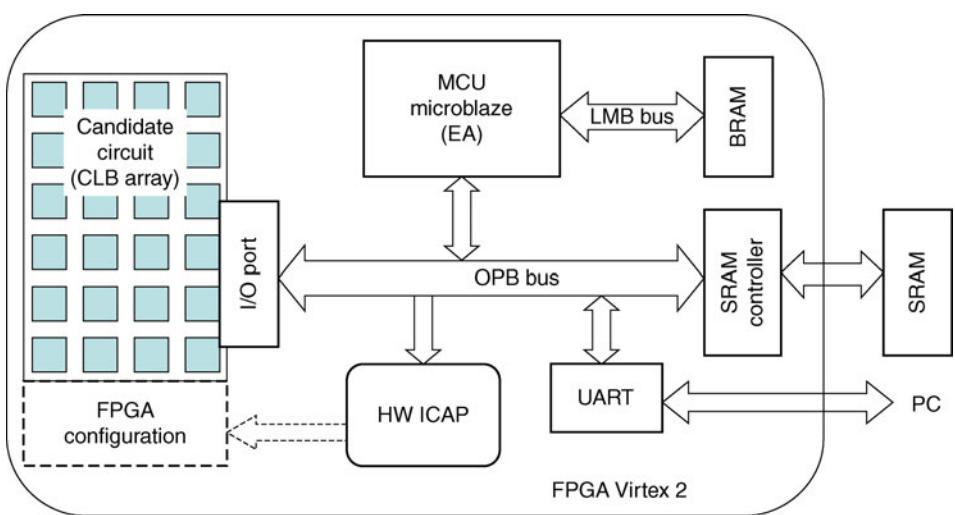
Nowadays, FPGAs are primarily used in the evolvable hardware field to accelerate circuit evolution. Most FPGA families can be configured only *externally*. The *internal reconfiguration* means that a circuit placed *inside* the FPGA can configure the programmable elements of the same FPGA. Although the internal configuration access port (ICAP) has been integrated into the recent Xilinx Virtex families (Virtex II, Virtex 4, Virtex 5) (Blodget et al. 2003), it is still too slow for many evolvable hardware applications.  Figure 18 shows a block diagram of an evolvable system implemented in an FPGA Virtex 2. The system includes an MCU microblaze (EA) connected to a BRAM via an LMB bus. The MCU is connected to an OPB bus. The OPB bus connects to an SRAM controller, which is connected to an SRAM. The SRAM controller is also connected to a PC. An I/O port is connected to the MCU and to an HW ICAP. The HW ICAP is connected to the FPGA configuration area of the Virtex 2. A Candidate circuit (CLB array) is also connected to the I/O port. The PC is connected to a UART, which is connected to the HW ICAP.

 Fig. 18

An evolvable system implemented in the FPGA using ICAP interface (Upegui 2006).

processor, reads a section of the configuration bitstream through ICAP, modifies the bitstream according to the genome currently evaluated in the MicroBlaze, sends the bitstream back through the ICAP for the partial reconfiguration of the FPGA, and evaluates the fitness of the current individual by interacting with the reconfigurable evolvable section through the standard OPB bus. The system was used to evolve the topology of Boolean networks (Upogui 2006).

In order to overcome the problem of slow reconfiguration, the concept of *virtual reconfigurable circuits* (VRC) was developed (Sekanina 2003, 2004). The VRC is, in fact, a second reconfiguration layer developed on top of the FPGA in order to obtain fast reconfiguration and application-specific programmable elements.  [Figure 19](#) shows that the VRC consists of a 2D array of programmable elements E_i . The routing circuits are created using multiplexers. The configuration memory of the VRC is typically implemented as a register array. All bits of the configuration memory are connected to multiplexers that control the routing and selecting of functions in programmable elements. Because the array of programmable elements, routing circuits, configuration memory, style of reconfiguration, and granularity of VRC can be designed exactly according to the requirements of a given application, designers can create an optimized application-specific reconfigurable device. If the structure of the chromosome corresponds to the configuration interface of the VRC then a very fast reconfiguration can be achieved (e.g., consuming a few clock cycles only).

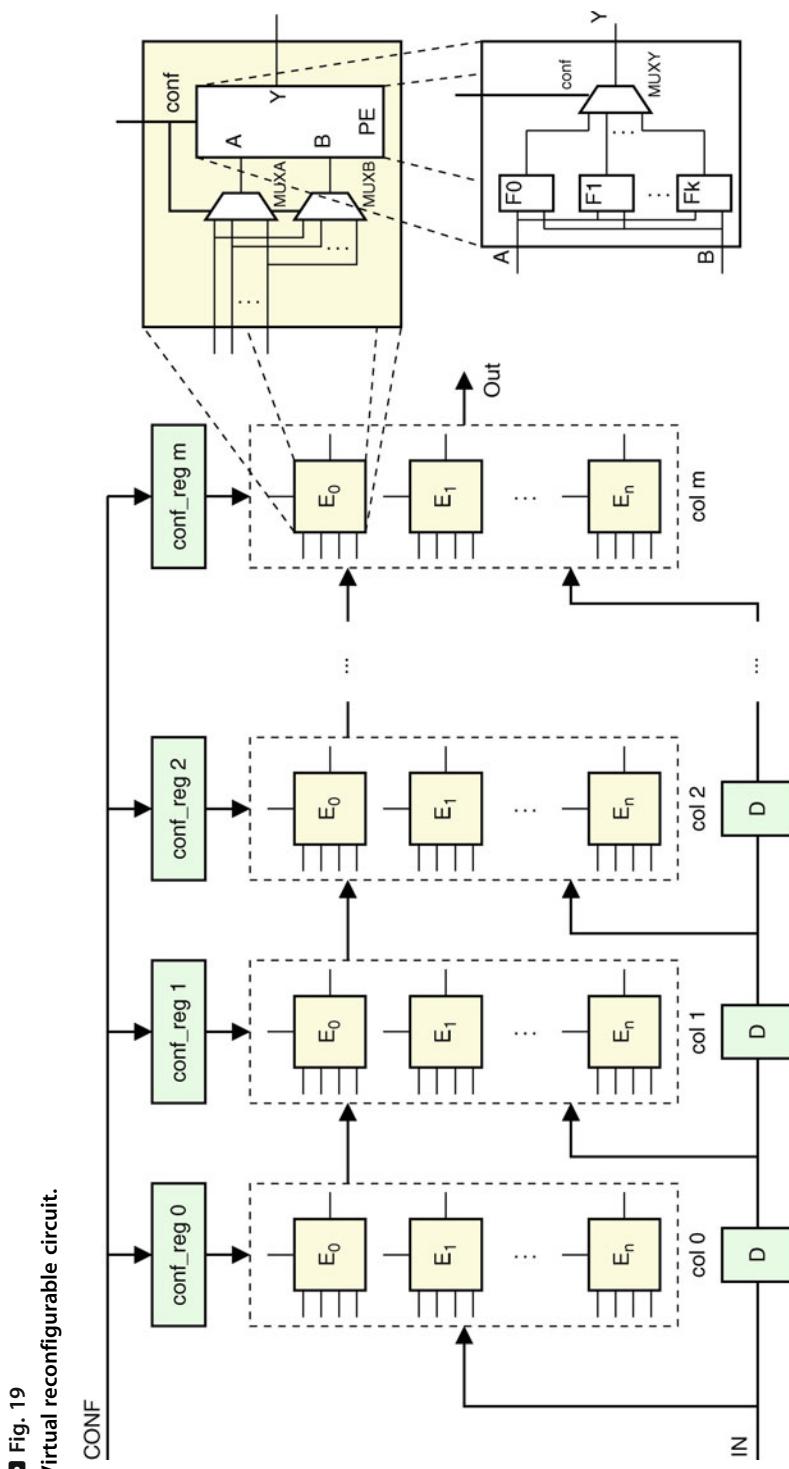
The FPGA-based implementations of evolvable hardware systems can be divided into two classes:

- *The FPGA serves in the fitness calculation only.* The evolutionary algorithm (which is usually executed on a personal computer) sends the configuration bitstreams representing candidate circuits to the FPGA in order to obtain their fitness values. The FPGA is configured externally. A typical example of this scenario is Thompson's experiment (see  [Sect. 3.2](#)).
- *The entire evolvable system is implemented in the FPGA.* The idea of the complete hardware evolution for FPGAs was initially demonstrated in Tufte and Haddow (2000); however, the paper provides only a simple example of the optimization of a few FIR filter coefficients stored in a register. Current FPGA implementations are capable of evolving circuit structures. They utilize either ICAP or VRC.

 [Table 3](#) surveys examples of FPGA implementations of digital evolvable systems. One can identify the following components in all systems: the array of reconfigurable elements, evolutionary algorithm, fitness calculation unit, and controller. The problem domain determines the type of reconfigurable elements. In some cases, the evolution is performed directly with reconfigurable cells of the FPGA; in other cases, a kind of VRC is utilized. An evolutionary optimization of coefficients stored in registers represents the simplest example. The EA and fitness calculation unit can be implemented either as an application-specific circuit or as a program. The program is running either in a personal computer or in an embedded processor, which is integrated into the FPGA.

5.1.1 Evolution of Image Filters

An FPGA-based evolvable image filter architecture was proposed to create novel implementations of image filters (Sekanina 2004; Vasicek and Sekanina 2007). The system consists of the



genetic unit, an array of reconfigurable elements (implemented as VRC), and a fitness calculation unit. While candidate filters are created and evaluated using user logic available on the FPGA, the genetic operations are carried out in the on-chip PowerPC processor. Every image operator is considered as a digital circuit of nine 8-bit inputs and a single 8-bit output, which processes gray-scaled (8-bit/pixel) images (see [Fig. 20](#)).

Table 3

Examples of FPGA implementations of evolvable digital systems

Reference	Application	Platform	EA	Fitness
External reconfiguration				
Thompson et al. (1999)	Tone discriminator	XC6216	PC	PC
Huelsbergen et al. (1999)	Oscillators	XC6216	PC	PC
Zhang et al. (2004)	Image filters	VRC	PC	PC
Gordon (2005)	Arithmetic circuits	Virtex CLB	PC	PC
Gwaltney and Gutton (2005)	IIR filters	VRC	DSP	DSP
Internal reconfiguration				
Tufte and Haddow (2000)	FIR filters	Register values	HW	HW
Martinek and Sekanina (2005)	Image filters	VRC	HW	HW
Vasicek and Sekanina (2007)	Image filters	VRC	PowerPC	HW
Sekanina and Friedl (2004)	Logic circuits	VRC	HW	HW
Vasicek and Sekanina (2008)	CGP accelerator	VRC	PowerPC	HW
Salomon et al. (2006)	Hash functions	VRC	HW	HW
Glette (2008)	Face recognition	VRC	MicroBlaze	HW
Glette et al. (2007)	Sonar spectrum class.	VRC	PowerPC	HW
Upogui and Sanchez (2006)	Cellular automaton	Virtex CLB	MicroBlaze	HW
Vasicek et al. (2008)	MCM	VRC	HW	HW

Fig. 20

Image filter design problem.

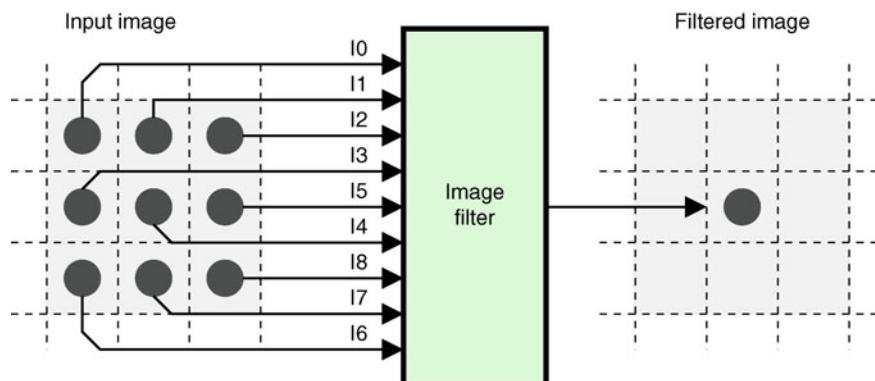


Figure 19 also shows a corresponding VRC, which consists of 2-input configurable functional blocks (CFBs), denoted as E_i , placed in a grid of eight columns and four rows. Any input of each CFB may be connected either to a primary circuit input or to the output of a CFB, which is placed anywhere in a preceding column. Every CFB can be programmed to implement one of the functions given in Table 4. All these functions operate with 8-bit operands and produce 8-bit results. The reconfiguration is performed column by column. The computation is pipelined; a column of CFBs represents a stage of the pipeline. The configuration bitstream of VRC, which is stored in a register array *conf_reg* consists of 384 bits. A single CFB is configured by 12 bits. The evolutionary algorithm directly operates with the configurations of the VRC.

The fitness calculation is carried out by the fitness unit (FU). The corrupted image, original image, and filtered image are stored in external SRAM memories, which are accessible via memory controllers implemented in the FPGA. The pixels of the corrupted image u are loaded from external SRAM1 memory and forwarded to the inputs of the VRC. The pixels of the filtered image v are sent back to the fitness unit, where they are compared with the pixels of the original image w that is stored in another external memory, SRAM2. The filtered image is simultaneously stored into the third external memory, SRAM3. The design objective is to minimize the difference between the filtered image and the original image, that is, the fitness value is calculated for an $M \times N$ -pixel image as

$$\text{fitness} = \sum_{i=1}^{M-2} \sum_{j=1}^{N-2} |v(i, j) - w(i, j)| \quad (11)$$

As Fig. 21 shows, the system is completely implemented in a single FPGA (except the SRAM memories). The program memory of the PowerPC as well as the population memory is implemented using on-chip Block RAM (BRAM) memories. The evaluation of the candidate configurations is pipelined in such manner as there are no idle clock cycles. Therefore, the time of evolution can be expressed as

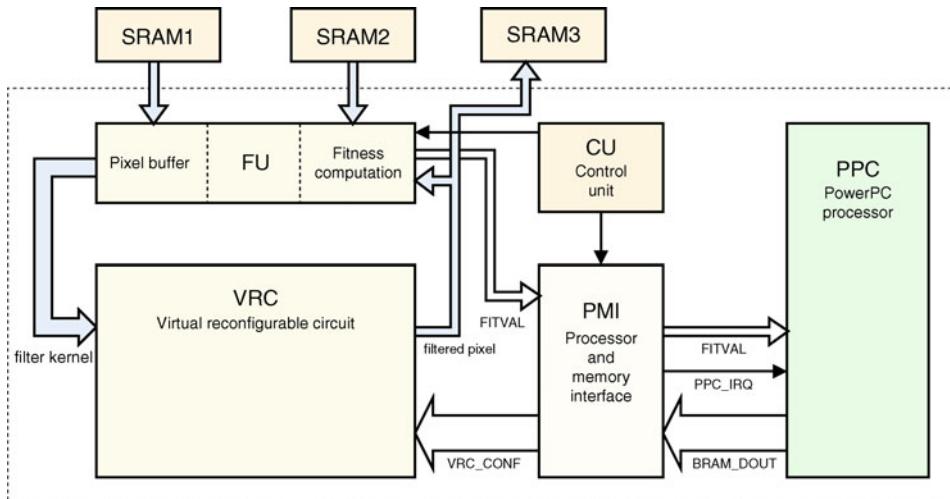
$$t_{\text{evol}} = Q(M - 2)(N - 2) \frac{1}{f}$$

where Q is the number of evaluations, $N \times M$ is the number of pixels (note that border pixels are ignored), and f is the operation frequency. The system was implemented in a COMBO6X

Table 4
List of functions implemented in each CFB

Code	Function	Description	Code	Function	Description
0	255	Constant	8	$x \gg 1$	Right shift by 1
1	x	Identity	9	$x \gg 2$	Right shift by 2
2	$255 - x$	Inversion	A	$\text{swap}(x, y)$	Swap nibbles
3	$x \vee y$	Bitwise OR	B	$x + y$	+ (addition)
4	$\bar{x} \vee y$	Bitwise \bar{x} OR y	C	$x +^S y$	+ with saturation
5	$x \wedge y$	Bitwise AND	D	$(x + y) \gg 1$	Average
6	$\bar{x} \wedge \bar{y}$	Bitwise NAND	E	$\max(x, y)$	Maximum
7	$x \oplus y$	Bitwise XOR	F	$\min(x, y)$	Minimum

Fig. 21
System for image filter evolution in FPGA.



card, which is equipped with a Virtex-II Pro FPGA. It utilizes 20% slices. Experimental results show that approximately 6,000 candidate filters can be evaluated per second ($N = M = 128$), which is 44 times faster than the same algorithm running on a Celeron@2.4 GHz processor.

This platform was used to obtain the salt-and-pepper noise filters, Gaussian noise filters, edge detectors, and other image operators. By combining three evolved filters in a bank of filters, even better filtering properties can be obtained (Vasicek and Sekanina 2007).

Figure 22 compares evolved filters (a single filter and 3-band filter) with conventional solutions (median filters (MF) and adaptive median filters (AMF)) for 25 images corrupted by the salt-and-pepper noise of various intensities. In comparison to conventional filters, evolved filters exhibit better filtering quality and lower implementation cost.

Recent work of Harding (2008) shows how the same application can be accelerated using a modern graphical processing unit that comes as a standard component of conventional PCs.

5.1.2 Incremental and Run-Time Evolution of Classifiers

Glette and Torresen proposed an application-specific architecture for online incremental evolution of classifiers (Glette 2008; Glette et al. 2007). The architecture consists of three main parts – the classification module, the evaluation module, and the CPU (PowerPC or MicroBlaze IP core) where the genetic algorithm is carried out. The reconfigurable part of the system is implemented as a VRC.

The classifier system consists of K category detection modules (CDMs), one for each category C_i to be classified (see Fig. 23). The CDM with the highest output indicates the resulting class. The input data to be classified are presented to each CDM concurrently. A CDM consists of M “rules,” each of them is implemented using N functional units (FU) working concurrently over the input pattern (Fig. 24 left). The rule is activated when all outputs from the FUs are 1. The counter is used to count the number of activated FU rows. Figure 24

Fig. 22 Comparison of standard image filters (MF, AMF) and evolved filters (single filter, 3-bank) for the salt-and-pepper noise of various intensity. The implementation cost is given in FPGA slices.

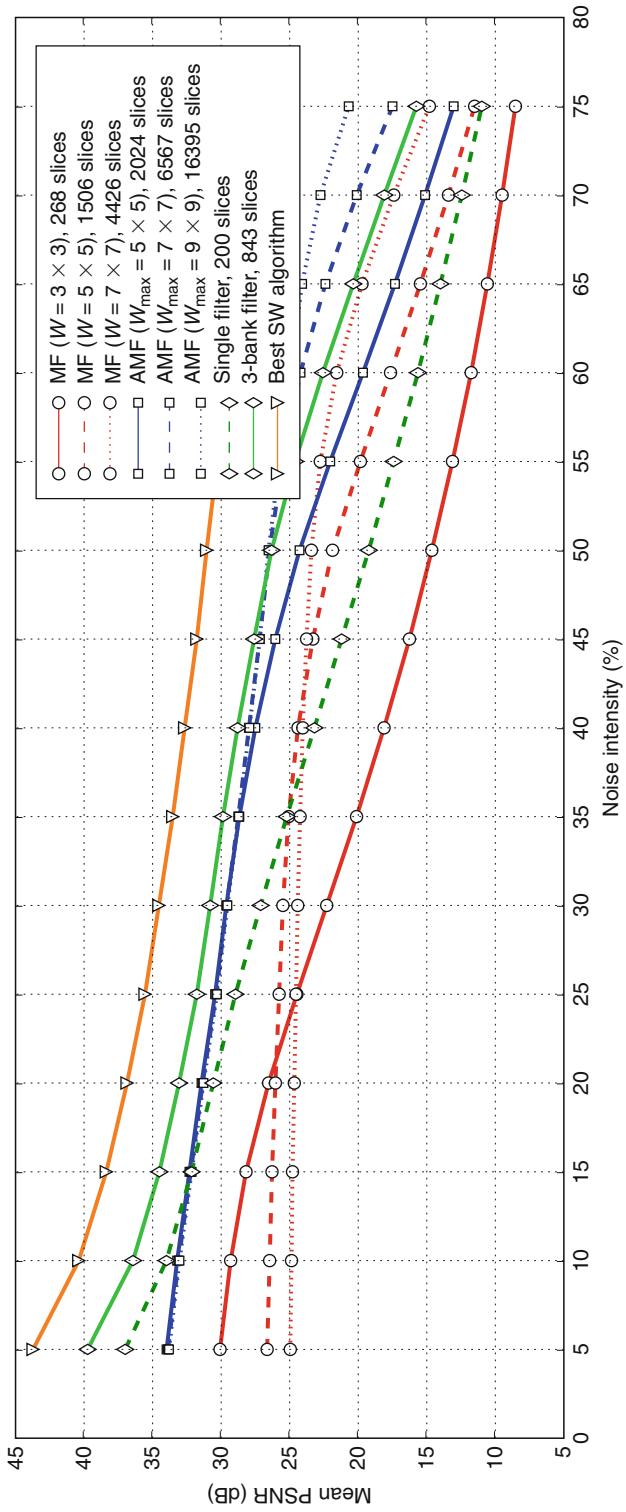
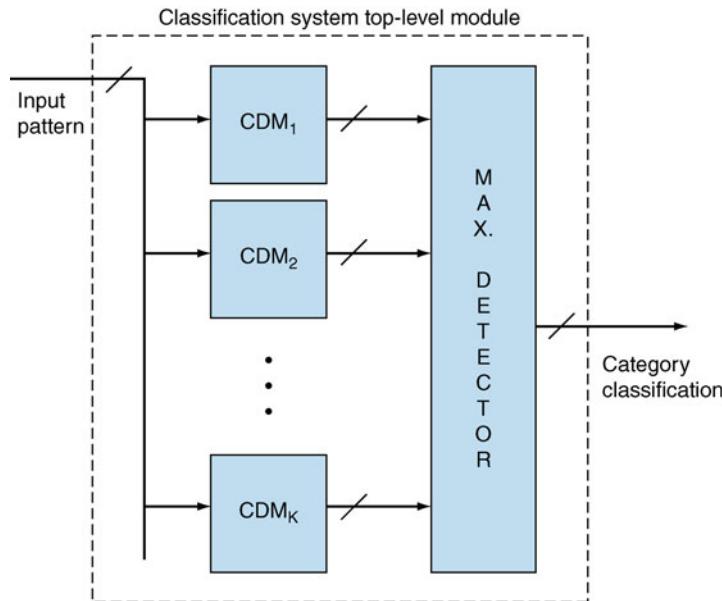
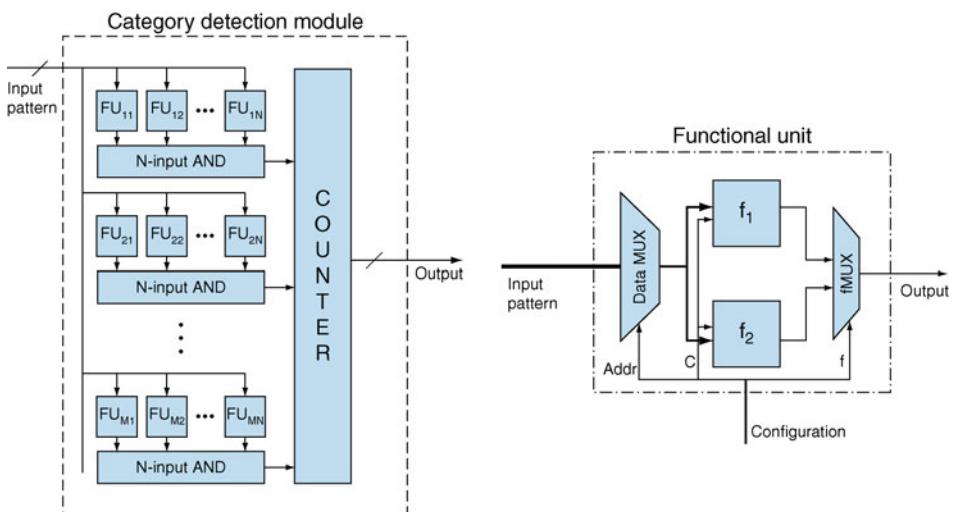


Fig. 23

Evolvable classifier system. The pattern to be classified is input to all of the category detection modules (according to Glette (2008)).

**Fig. 24**

Category detection module (left) and function unit (right) according to Glette (2008).



shows that each FU is controlled by configuration bits that determine some part of the input pattern to be processed by the FU, selection of function, and a constant value, C , which can be utilized for computations. This architecture is suitable for incremental evolution because each category detector, CDM_i , which is encoded using 100–200 bits, can be evolved separately.

According to Glette (2008), the fitness calculation is performed in the following way: Let V_t denote training vectors and V_v validation vectors utilized at the end of evolution. Each row of FUs is fed with the training vectors ($v \in V_t$), and fitness is based on the row's ability to give a positive (1) output for vectors v belonging to its own category ($C_v = C_i$), while giving a negative (0) output for the rest of the vectors ($C_v \neq C_i$). In the case of a positive output when $C_v = C_i$, the value A is added to the fitness sum. When $C_v \neq C_i$ and the row gives a negative output (value 0), 1 is added to the fitness sum. The other cases do not contribute to the fitness value. The fitness function F for a row can then be expressed in the following way, where o is the output of the FU row:

$$F = \sum_{v \in V_t} x_v \text{ where } x_v = \begin{cases} A \times o & \text{if } C_v = C_i \\ 1 - o & \text{if } C_v \neq C_i \end{cases} \quad (12)$$

The architecture was evaluated using nontrivial benchmark problems – face image recognition (Glette 2008), sonar spectrum classification (Glette et al. 2007) and electromyographic prosthetic hand control (Glette et al. 2008) – and compared with conventional as well as evolvable hardware approaches. The proposed solution exhibits fast classification and a short adaptation time. In comparison to existing evolvable hardware approaches, the quality of the classification is higher. In comparison to the state-of-the-art classification algorithms (such as support vector machines), the quality of classification is slightly worse.

5.2 Evolutionary Design in FPAA and FPTA

Although the previous section has dealt with the intrinsic approach, the goal of the presented applications was not to exploit the physical characteristics of FPGAs (as, e.g., Thompson et al. (1999)). The main objective was to make the evolutionary design faster. However, this is not always the case for analog circuits.

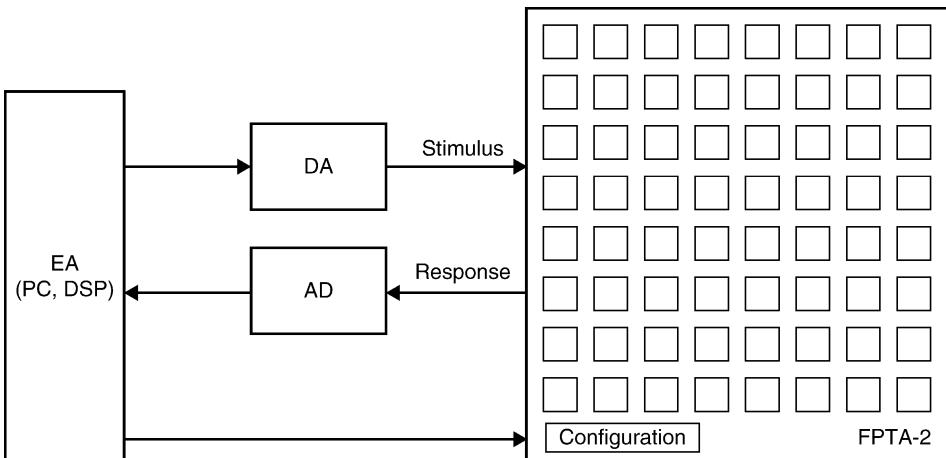
Full-custom design of analog circuits is time consuming and expensive. The aim of evolutionary design using FPTAs or FPAs is to obtain circuits working directly in a real environment. The intrinsic evolution should help to overcome the “simulator vs reality” gap, enable designers to immediately verify the structure and parameters setting of the circuit under design, and provide a working prototype in a relatively short time.

❸ *Figure 25* shows a typical setup for analog circuit evolution in FPTA (or FPAA). The EA runs in a PC (or DSP) that generates candidate circuits (configurations) for FPTA. The PC also generates test stimuli that are converted to analog signals using a DAC. Circuit responses are converted using an ADC on digital data that are compared with the required target signals. The objective is to minimize the differences between the obtained signals and target signals. Alternatively, the fitness calculation can be based on a comparison of the measured frequency characteristics and target frequency characteristics. This approach was used to evolve various circuits including rectifiers, amplifiers, oscillators, filters, DA converters, controllers, regulators, and primitive gates (Aggarwal et al. 2006; Greenwood and Tyrrell 2007; Henrici et al. 2007; Stoica et al. 2002, 2006).

As far as the reconfigurable device is analog, it is natural for evolution to exploit physical properties of transistors, configurable switches, and other components and the environment to find the required behavior. The evolutionary process can escape the space of conventional implementations (which designers are able to represent in advance) and produce circuits that are difficult to understand or even reconstruct in a circuit simulator.

Fig. 25

A typical setup for circuit evolution in FPTA.



It was recognized that intrinsic evolution of analog circuits can lead to some serious problems. Circuits highly ranked during evolution may obtain low fitness when reevaluated individually at the end of evolution. It can be due to the transient behavior, which describes a configuration that is not stable as a function of time. Or, the circuit can be dependent on a previous configuration of the chip because of various parasitic capacitors in the chip. In practice, the transient behavior can be resolved by reevaluating the individuals for a longer time period. The operational range of the evolved circuit in the frequency domain is another potential pitfall, since in principle the circuit behavior should be evaluated for the overall frequency domain in which it is expected to work. Similar problems may appear when loads different from those used during evaluation are connected. Various techniques were proposed to eliminate these undesirable effects (Stoica et al. 2000, 2004; Thompson et al. 1999). In summary, evolved circuits may not be stable after some time, they may not work correctly when uploaded on another chip (or even to another area of the same chip) or used in a different environment.

A promising application for the evolution of analog circuits is adaptation in extreme environments. Future NASA missions will face extreme environments, including environments with large temperature differences (from -233°C to 460°C) and high radiation levels, upto a 5 Mrad total ionizing dose (TID) (Stoica et al. 2006). Conventional approaches to extreme environment electronics include *hardening-by-process* (i.e., fabricating devices using materials and device designs with higher tolerance to extreme environment) and *hardening-by-design* (i.e., the use of special design/compensation schemes). Both of these hardening approaches are limited, in particular for analog electronics, by the fact that current designs are fixed and, as components are affected by extreme environments, these drifts alter functionality. A recent approach pioneered by JPL is to mitigate drifts, degradation, or damage on electronic devices in extreme environments by using reconfigurable devices and an adaptive self-reconfiguration of circuit topology. This new approach is referred to as *hardening-by-reconfiguration*. As the only investigator in this area, JPL's evolvable hardware group has demonstrated for simple circuits created in the FPTA-2 that evolutionary approach can

recover functionality lost (1) after a fault artificially injected into FPTA-2, (2) in extreme high and low temperatures (from -196°C to 320°C), and (3) in high radiation environments (up to 250 krad TID) (Keymeulen et al. 2000; Stoica et al. 2004, 2005, 2006; Zebulum et al. 2006).

5.3 Evolution on Unconventional Platforms

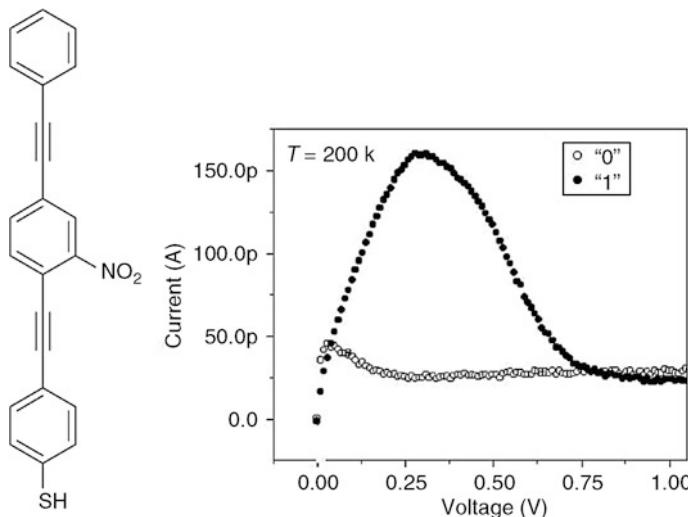
Inspired by Thompson's experiment, researchers began to use EAs to exploit physical properties of reconfigurable devices for computing. The aim was to obtain useful computations directly in materials, without formulating a mathematical model of the problem and without understanding "the rules of the game." In addition to common FPGAs and reconfigurable analog devices mentioned in previous sections, several platforms were proposed for these experiments, for example, an evolvable motherboard (Layzell 1998), RISA (Greensted and Tyrrell 2007) and REPOMO32 (Sekanina et al. 2009).

Recently, Miller has used the more general term, *evolvable matter*, to address the use of evolutionary algorithms for the design on any physical reconfigurable platform (e.g., chemical) in a real environment. The idea behind the concept is that applied voltages may induce physical changes that interact in unexpected ways with other distant voltage-induced configurations in a rich physical substrate. In other words, it should theoretically be possible to perform the evolution directly *in materio* if the platform is configurable in some way. Promising technology for evolution in materio was reviewed in (Harding et al. 2008). In particular, liquid crystals were used as a platform to evolve various circuits including primitive logic functions, signal discriminators, and robot controllers (Harding 2006).

Tour's group has presented another example – the Nanocell (Tour 2003). The nanocell is a 2D network of self-assembled metallic particles connected by molecular switches (Fig. 26). The nanocell is surrounded by a small number of lithographically defined access leads. The nanocell is not constructed as a specific logic gate – the logic is created in the nanocell by

Fig. 26

Molecular switch used in NanoCell and its V/I characteristics (from Tour 2003).



training it postfabrication by changing the states of the molecular switches. The training algorithm does not know the connections within the nanocell or the locations of the switches. However, the configuration can be changed by voltage pulses applied to the I/O pins. A genetic algorithm was utilized to generate these pulses in order to form the required logic gates.

Bartels et al. (2004) applied the evolutionary algorithm to configure a reconfigurable mirror used in a laser. The aim was to shape laser pulses in order to distort molecules in specific ways to catalyze chemical reactions, with the ultimate goal of manipulating large molecules, for example, proteins and enzymes. The method has also been used to create new quantum behaviors at the atomic level. Results completely unexpected from a physical point of view were reported. Evolved behaviors (such as anti-correlated attosecond harmonics in quantum systems) were not known to be physically possible. The ability to move beyond the nanoscale to the attoscale is considered as a major breakthrough. Potential applications of controlling the behavior of materials at the atomic level are enormous.

6 Adaptive Hardware

A promising approach to building adaptive hardware is to develop systems that combine a reconfigurable device with an intelligent mechanism for its reconfiguration. The reconfiguration mechanism is activated when the adaptation is needed. A straightforward solution for the adaptive system is to maintain a set of configurations designed a priori and use the most suitable one when the current configuration becomes unsatisfactory. An obvious weakness of this approach is that a suitable configuration may not be prepared. A more advanced approach, developed in the evolvable hardware field, tries to generate new configurations at run time using an evolutionary algorithm. A potential problem is that the EA is a stochastic algorithm. Nobody can guarantee that the resulting configuration really solves the problem. In addition, nobody can guarantee that the EA will provide an acceptable configuration on time. However, in many applications, a partial solution provided by the EA may be a very good result.

Examples of adaptive hardware systems will be presented according to the types of changes that usually lead to the requirement of system adaptation. These changes can be classified either as changes

- in the hardware platform itself (i.e., faults)
- in the input data characteristics, or
- in the specification

6.1 Self-repairing Hardware

The EA is used to recover the system function that was degraded/lost due to failure in the hardware platform (such as being stuck at zero, changes in the transistors' characteristics in extreme environments, etc.). Note that the requirement on the function remains unchanged and the input data are as expected. First of all, the failure has to be detected. Then, if no other mechanisms are capable of repairing the system, the EA is activated. In some cases, it is sufficient to tune only some circuit parameters (e.g., bias voltages) in order to repair the circuit. In more complicated cases, a new topology must be evolved.

6.1.1 SRAA

The self-reconfigurable analog array (SRAA) developed at NASA JPL provides the capability of *continual temperature compensation* (Stoica et al. 2008). The SRAA contains an array of 4×6 analog cells. Various specific circuits can be implemented in the SRAA for which mission-oriented ASICs were designed in the past. The SRAA cells are based on tunable OTAs used to compensate for the effects that come from temperature variations. A subset of cells, called reference cells, are continually compensated at the same time as other functional cells perform user operations. The result of compensation is then transferred to the main array. Elementary cells can be compensated for temperature-induced deviations within a -180°C to 120°C range. The EA is implemented as a digital circuit either in an FPGA or as ASIC.

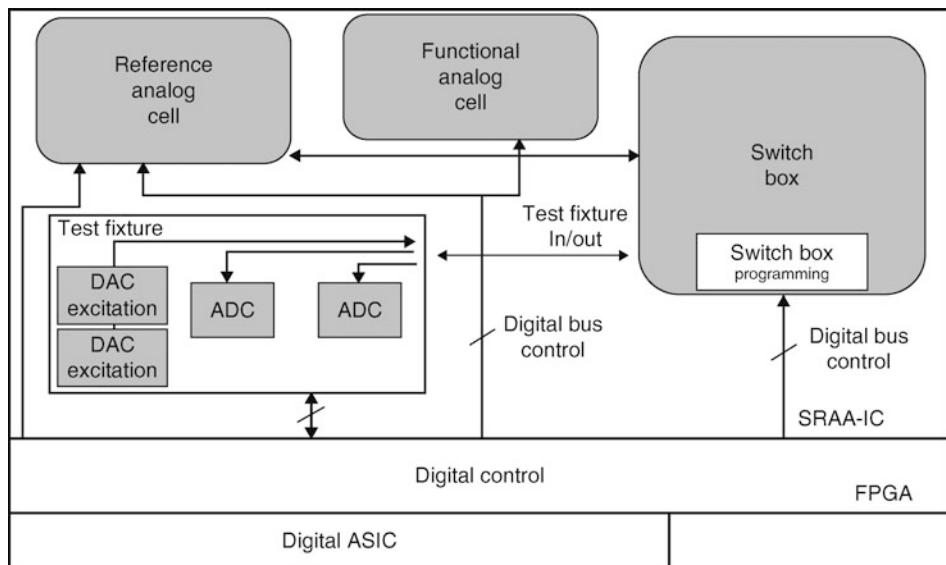
Figure 27 shows the SRAA chip that contains the array of cells divided into the functional and the reference analog arrays. The reference analog cells are individually probed to check for degradation. Both reference and functional cells include digital-to-analog converters (DACs) used to provide bias voltages or currents to tune the analog cell response. The switch box allows for the configuration of different functions in the functional analog array, as well as providing an interface between the reference analog cells and the test fixture. The SRAA contains 40 reconfiguration DACs for the 6×4 functional array.

The SRAA utilizes a hierarchical approach to calibration/compensation. First, a faulty configuration is replaced by a priori prepared configuration. If it fails, a second level takes place, which involves a less computationally expensive search algorithm. Finally, a third level is employed in case the previous levels do not succeed, and a global search is conducted using evolutionary algorithms.

Once an analog cell response goes out of specification (e.g., due to temperature or radiation effects), the FPGA will recalibrate the corresponding reference cell by reprogramming its

Fig. 27

Self-reconfigurable analog array (SRAA) according to Stoica et al. (2008).



respective reconfiguration DACs. Once new values are found that recover the behavior of the reference cell, the same are used to reprogram the configuration DACs of the associated functional cells. This tuning process assumes that for each type of cell, the functional and reference versions will exhibit the same behavior. This assumption usually holds when the SRAA chip is exposed to extreme temperatures. Since the chip is fabricated in a rad-hard process, the SRAA will tolerate permanent radiation effects (total ionization dose [TID]) up to 300 krad. The SRAA is designed to maintain a stable operation over a wide temperature range of more than 300°C, from -180°C to 125°C. Over this range, parameters of interest may vary with < 1–5% deviation from their 27°C value, depending on the circuit.

6.1.2 Self-repairing FPGAs

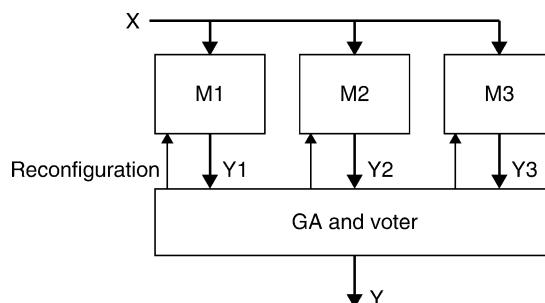
An interesting evolutionary-repair approach for FPGAs was introduced by Garvie (2005). He employed triple module redundancy (TMR), which is a conventional scheme used to mitigate faults. A TMR system has three copies of a module and uses a voting circuit on their outputs so that the final output is an agreement between at least two modules. A TMR extension, TMR+Scrubbing, in addition, provides fault tolerance to single event upsets (SEUs) in the FPGA configuration data by regular reprogramming of the FPGA. Garvie extended TMR +Scrubbing by “Jiggling” to repair also permanent damage by using the two healthy modules to repair the faulty one (Garvie 2005). He recognized that the two healthy modules can provide a reference signal (i.e., training vectors) for evolutionary design of the configuration of the third module. Once a Jiggling repair is complete three healthy modules are again available (☞ Fig. 28). Permanent faults can be repaired until spare resources are exhausted. The Jiggling repair evolutionary module is implemented on the same FPGA and requires a small additional area on the chip. Hence, it can be itself repairable by another Jiggler. Reliability analysis for small benchmark circuits shows the Jiggling system using 2.8 times the overhead per module can survive 17 times longer than a TMR/Simplex approach.

6.2 Changes in the Specification or Input Data Characteristics

Consider an image-recognition system in which an image filter is used to enhance the input images coming from a camera. If the type of noise is variable, the filter may have problems

Fig. 28

Triple module redundancy extended by a module repairing mechanism (Garvie 2005).



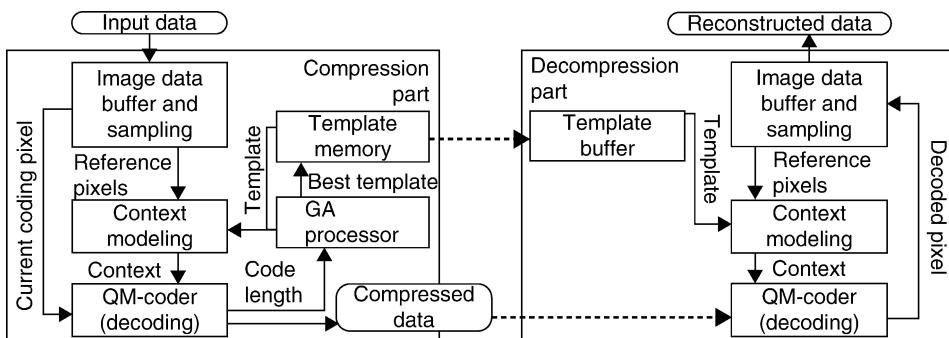
enhancing the images and the quality of recognition may decrease. Hence, the EA is employed to optimize the filter for a particular type of noise. In this case, the specification of the function remains unchanged; however, the system has to adapt because of changing environments. Also the specification can be changed either by an operator or autonomously, by the system itself which can, for example, continuously monitor its history and plan new actions (considered as circuit configurations). If the configuration that implements the required action is unknown in advance, the EA is activated to perform the adaptation. Several real-world applications developed in the AIST laboratory in Japan (Higuchi et al. 1999, 2006) in which this kind of adaptation has been employed will be shown.

6.2.1 Image Compression Chip

Adaptive image compression methods allow for modifying the compression algorithm according to the particular data that has to be compressed. In consequence, a higher compression ratio can be achieved. The AIST team has developed an evolvable chip for lossless image compression, which is suitable for electrophotographic printers (Fig. 29) (Sakanashi et al. 2001). As the size of a typical image printed using this printer goes from tens to hundreds of MBs, a compression algorithm must be used to reduce the image size as well as the time needed to read the images from a hard disk. The conventional algorithm (JBIG) is based on predicting the next pixel value using an MQ coder according to its (typically 16) neighbors. The positions of the neighbors are determined by a so-called template, which is invariable in the conventional algorithm. Since the evolvable chip is able to find the optimal positions of pixels in the template for each block of the image, the predictions are better than in the conventional implementation and hence the compression rate is higher. The EA is executed for each image block separately with the aim of maximizing the compression ratio for the block. The incoming blocks can in fact be viewed as a changing environment for the prediction algorithm. The developed chip operates at 133 MHz and is 7 times faster than the reference implementation running on a Pentium 4.2 GHz processor. The compression rate is 1.7 times better than that achieved by the standard JBIG2 algorithm (Sakanashi et al. 2001, 2006). This is a very good result for lossless compression.

□ Fig. 29

Image compression chip developed at AIST that adapts pixel positions for prediction (Sakanashi et al. 2001).



6.2.2 Adaptive Controllers

Another evolvable hardware chip from AIST was designed for a myoelectric prosthetic hand. The myoelectric hand is controlled by the signals generated with muscular movements (electromyography, EMG signals). It takes a long time, usually almost 1 month, before a disabled person is able to control a multifunction prosthetic hand freely using a fixed controller. This chip allows the myoelectric hand to adapt itself to the disabled person and thus significantly reduce the training period to several minutes (☞ Fig. 30).

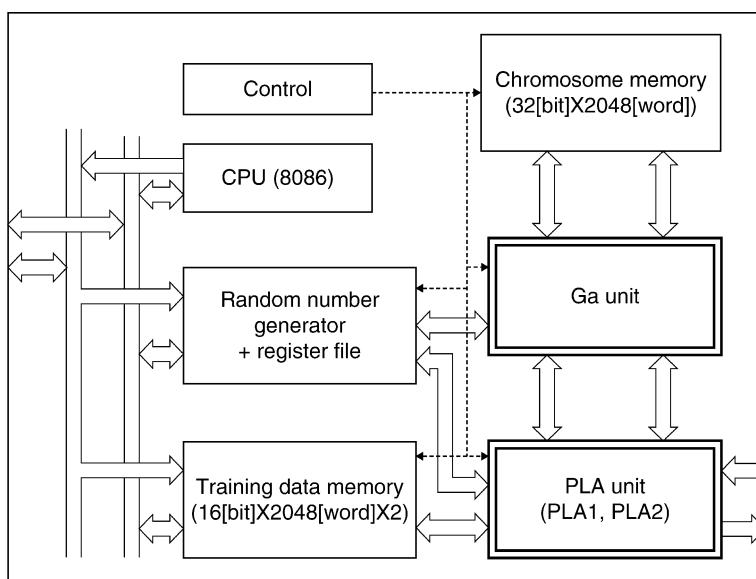
The myoelectric hand described in Kajitani et al. (2001, 2006) is able to perform the six actions, which are paired (open–grasp, supination–pronation, and flexion–extension), with a separate motor control for each pair. The task for the evolvable hardware is to synthesize a circuit that can map input patterns (feature vectors extracted from EMG signals) to the desired actions of the hand.

The training procedure is as follows: First, training pattern data is obtained in an online mode, where the hand user makes one of the six actions from which ten EMG signal patterns are measured. For all six actions, a set of 60 signals is obtained as training data. Then, based on this obtained training data, a classifier is evolved in the PLA (16-bit input/6-bit output), which is available in the chip. GA is implemented on the same chip (partly in an on-chip processor and partly as a special circuit). The same chip was also used for an adaptive robot controller (Keymeulen et al. 1997).

First implementations of the chip achieved 81% classification accuracy. An experienced user is able to achieve 98% accuracy on the recent version of the chip, which uses more complex data preprocessing. The classifier evolution in hardware is 40 times faster than a 1.2 GHz Athlon processor (Kajitani et al. 2006). An incremental evolution-based scheme for

■ Fig. 30

AIST evolvable hardware chip used to implement adaptive controllers and classifiers (according to Kajitani et al. 1998).



EMG signal classification achieving even a higher classification accuracy was also proposed (Glette et al. 2008).

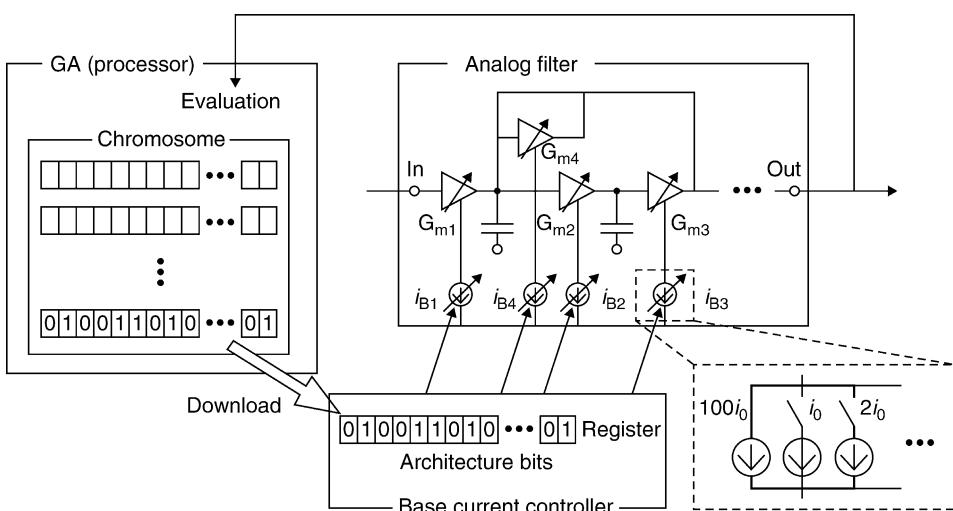
6.2.3 Post-fabrication Adaptation

Post-fabrication calibration of ASICs can also be considered as a kind of hardware adaptation. Fabrication variance in the manufacture of chips can cause problems in some high-performance applications, especially when a cutting-edge manufacturing technology is employed. In other words, only some of the fabricated chips really meet the specification. Hence, an EA is used to tune selected circuit parameters with the aim of obtaining required circuit function even for the chips that do not work because of variance of the fabrication process. Although some area of the chip has to be used for implementing the reconfiguration logic and controller, the overall benefits are significant: It is possible to fabricate simpler, smaller, and low-power designs that do not contain large circuits needed to compensate fabrication variances, and thus increase the yield.

A typical chip in which the post-fabrication tuning was successfully applied is an intermediate filter (IF), commonly found in mobile telephones (Murakawa et al. 1998). In this case, a 1% discrepancy from the center frequency (455 kHz) is unacceptable. Chips that do not meet this requirement must be discarded lowering the yield. The AIST team used GA to control 18 OTAs that properly tune currents to compensate for the errors introduced by fabrication variances (see Fig. 31). The GA operates with a 20-member population and 234-bit chromosomes. The tuning is usually complete after 100 generations, which takes a few seconds. In the experiment, 29 out of 30 fabricated chips were successfully tuned. The method allowed us to reduce the area by 9% and the power consumption by 26%. The recent generation of the chip reduces the area by 49% (with respect to existing commercial designs) (Murakawa et al. 2006).

Fig. 31

Tunable intermediate filter composed of programmable OTAs (from Murakawa et al. 1998).



The same concept was used to compensate for out-of-specification timing delays between circuits, or “clock skew.” AIST devised a clock timing adjustment architecture to eliminate the clock skew problem by using a genetic algorithm to make the clock timings perform within the intended design specifications. Simulation results showed yields rising from 2.9% to 51.1% using evolved clock-timing circuits (Takahashi et al. 2006).

A similar approach was applied to tune signal transmitters (of USB/IEEE 1394 standards) in order to compensate signal distortions at the end of the cable (Kasai et al. 2005). For a particular cable and its placement, the GA-based tuning of the transmitter has successfully achieved a transmission speed of 1.6 GHz, which is four times faster than the current IEEE1394 standard (400 MHz), and a cable length of 21 m, which is four times longer than the current USB standard (5 m).

The evolvable data-compression chip, evolvable controller, as well as the chips that enable post-fabrication parameters tuning were commercialized in Japan. The post-fabrication tuning seems to be the most attractive application of evolvable hardware from the commercial point of view.

7 Other Areas

Evolutionary design has been applied in many hardware-oriented areas, significantly extending the original intention of the evolvable hardware research – electronic circuit evolution and adaptation. For example, in evolutionary robotics, robot controllers are evolved together with robot physical bodies (Lipson and Pollack 2000). The specialized evolutionary robotics literature provides more details, for example, Nofli and Floreano (2000) and Gross et al. (2006). Evolving physical designs have become popular in the aerospace industry (Terrile et al. 2005). Other evolutionary design applications are well covered in Bentley (1999) and Bentley and Corne (2002).

Several research teams have developed adaptive and fault-tolerant research platforms inspired by embryonic processes (Durbeck and Macias 2001; Mange et al. 2000; Moreno et al. 2005; Tufte and Haddow 2005). These platforms implement arrays of artificial cells, which are able to divide (i.e., copy the configuration to neighboring cells) and differentiate (i.e., interpret the configuration according to, for example, location of the cell). These platforms contain various mechanisms for fault detection, localization, and repair. Circuit recovery is implemented using reconfiguration of spare cells and bridging over faulty cells. Fault-tolerance mechanisms are implemented hierarchically providing thus a really robust solution for digital circuits. These approaches will be useful for the massively parallel architectures that we expect in future with the progress in nanotechnology. However, the circuit overhead connected with the implementation of fault-tolerant mechanisms is impractically huge for current technology.

Finally, research is concentrated on the biological aspects and possibilities offered by modeling biological systems in hardware. This includes building biologically inspired structures such as virtual cells, artificial immune systems, and artificial brains using evolvable hardware. Research in this area focuses primarily on biological mechanisms, which can eventually lead to techniques for improved hardware design. Haddow speculates that modeling biological networks can be a killer-application for evolvable hardware (Haddow 2008).

8 Conclusions

The field of evolvable hardware has been surveyed. The status of its main subareas can be now summarized.

Digital circuits evolved so far contain from several gates to millions of gates. It is typical for the evolution of small circuits that all possible input vectors are used in the fitness function and the aim is to improve circuit parameters (area, delay) with respect to existing designs. Evolved mid-size circuits such as image filters or classifiers contain thousands of gates. As the training set has to be used in the fitness function, candidate circuits are evaluated imperfectly, that is, they may not work perfectly for some of the unseen input patterns. For million-component circuits evolved so far, requirements neither on function nor on size were specified; only some structural properties of circuits were evaluated. It can be stated that in all the circuit classes, the chromosomes have similar size, typically around one or two thousand bits. It seems that larger search spaces cannot be sought effectively using existing search algorithms and computational resources. In order to evolve a circuit larger than a small multiplier, a domain knowledge method has to be employed to define suitable components, possibilities of interconnection, and fitness function. Real-world circuits are usually evolved using function-level representation or incremental evolution. Although many developmental encodings were proposed and investigated to overcome the scalability problem, no really innovative designs have been obtained using developmental schemes so far.

In contrast to digital circuit design, analog circuit design requires more experience and intuition. Despite this fact, John Koza and his team have demonstrated high competitiveness of evolutionary algorithms (in particular, genetic programming) in the area of analog circuit synthesis. An important step toward evolving really trustworthy analog designs was done by McConaghy. Evolutionary design is particularly good at exploiting properties of physical systems in order to achieve target behaviors. This unique feature is important for fast tuning of analog circuits that are implemented in reconfigurable devices and operated in a real environment. Successful results were demonstrated for FPTAs running in various environments and FPAs working as controllers. Similarly, it was demonstrated that design for other reconfigurable devices (e.g., antennas, optical systems, and nanosystems, including unconventional platforms such as liquid crystals and nanoparticles) can significantly benefit from the evolutionary approach.

The main contribution of the evolutionary hardware design can be seen in the areas where it is impossible to provide a perfect specification for target implementations, and conventional design methods are based on experience and intuition rather than on a fully automated methodology. Then, the EA can explore the “dark corners” of design spaces that humans have left unexplored. Also, the existence of a fast simulator for a particular domain is important for successful evolutionary designs.

Only few systems exist in which autonomous hardware adaptation is achieved using evolutionary computation. The reason is that there are some crucial problems associated with the use of EAs in these applications, namely, a solution is not always obtained in real-time, the quality of a solution may not be acceptable, and it is difficult/impossible for many applications to obtain training data during system operation. A typical adaptive/evolvable system uses an EA to optimize only a few parameters. In comparison to evolutionary circuit design, chromosomes are relatively short and the task is not difficult for EA (since the EA has to provide a sufficient solution almost anytime). However, with the development of

reconfigurable technology, we can expect a boom in adaptive systems that will be tuned for a particular location, user, or time of operation. The evolutionary approach seems to be very promising in achieving the adaptive behavior.

Finally, evolvable hardware has an impact on the theory of computation. The evolved computational devices represent a distinct class of devices that exhibits a specific combination of properties, not visible and studied in the scope of all computational devices till now. Devices that belong to this class show the required behavior; however, in general, we do not understand how and why they perform the required computation. The reason is that the evolution can utilize, in addition to the “understandable composition of elementary components,” material-dependent constructions and properties of environment (such as temperature, electromagnetic field, etc.) and, furthermore, unknown physical behaviors to establish the required functionality. Therefore, nothing is known about the mapping between an abstract computational model and its physical implementation. The standard notion of computation and implementation developed in computer science as well as in cognitive science has become very problematic with the existence of evolved computational devices (Sekanina 2007).

It can be concluded that evolvable hardware as a field is maturing. Real-world applications of evolutionary hardware synthesis as well as evolutionary-based adaptive hardware exist and bring advantages when compared to conventional solutions.

Acknowledgments

This work was partially supported by the Grant Agency of the Czech Republic under No. 102/07/0850 *Design and hardware implementation of a patent-invention machine* and the Research Plan No. MSM 0021630528 *Security-Oriented Research in Information Technology*.

References

- Aggarwal V, Mao M, O'Reilly UM (2006) A self-tuning analog proportional-integral-derivative (PID) controller. In: AHS '06: Proceedings of the first NASA/ESA conference on adaptive hardware and systems. IEEE Computer Society, Washington, DC, USA, pp 12–19
- Ali B, Almaini AEA, Kalganova T (2004) Evolutionary algorithms and their use in the design of sequential logic circuits. *Genet Programming Evol Mach* 5(1):11–29
- Anadigm (2007) Anadigm, AN221E04 – field programmable analog arrays – user manual. URL <http://www.anadigm.com/doc/UM021200-U007.pdf>
- Bartels RA, Murnane MM, Kapteyn HC, Christov I, Rabitz H (2004) Learning from learning algorithms: applications to attosecond dynamics of high-harmonic generation. *Phys Rev A* 70(1):1–5
- Bentley PJ (ed) (1999) Evolutionary design by computers. Morgan Kaufmann, San Francisco, CA
- Bentley PJ, Corne DW (2002) Creative evolutionary systems. Morgan Kaufmann, San Francisco, CA
- Bernardi P, Sanchez E, Schillaci M, Squillero G, Reorda MS (2008) An effective technique for the automatic generation of diagnosis-oriented programs for processor cores. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 27(3):570–574
- Bidlo M, Skarvada J (2008) Instruction-based development: from evolution to generic structures of digital circuits. *Int J Knowl-Based Intell Eng Syst* 12(3): 221–236
- Blodget B, James-Roxby P, Keller E, McMillan S, Sundararajan P (2003) A self-reconfiguring platform. In: Proceedings of the 13th conference on field programmable logic and applications FPL'03, Lisbon, Portugal, LNCS, vol 2778. Springer Verlag, pp 565–574
- Drechsler R (1998) Evolutionary algorithms for VLSI CAD. Kluwer Academic Publishers, Boston
- Durbeck L, Macias N (2001) The cell matrix: an architecture for nanocomputing. *Nanotechnology* 12(3): 217–230
- Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, Berlin

- Erba M, Rossi R, Liberali V, Tettamanzi A (2001) An evolutionary approach to automatic generation of VHDL code for low-power digital filters. In: Proceedings of the 4th European conference on genetic programming EuroGP2001, LNCS, vol 2038. Springer Verlag, Berlin, pp 36–50
- Gao P, McComagh T, Gielen G (2008) ISCLES: importance sampled circuit learning ensembles for trustworthy analog circuit topology synthesis. In: Proceedings of the 8th international conference on evolvable systems: from biology to hardware. LNCS, vol 5216. Springer Verlag, Berlin, pp 11–21
- de Garis H (1993) Evolvable hardware – genetic programming of a Darwin Machine. In: International conference on artificial neural networks and genetic algorithms, Innsbruck, Austria. Springer Verlag
- Garvie M (2005) Reliable electronics through artificial evolution. PhD thesis, University of Sussex
- Glette K (2008) Design and implementation of scalable online evolvable hardware pattern recognition systems. PhD thesis, University of Oslo
- Glette K, Torresen J, Yasunaga M (2007) An online EHW pattern recognition system applied to sonar spectrum classification. In: Evolvable systems: from biology to hardware, LNCS, vol 4684. Springer Verlag, pp 1–12
- Glette K, Torresen J, Gruber T, Sick B, Kaufmann P, Platzner M (2008) Comparing evolvable hardware to conventional classifiers for electromyographic prosthetic hand control. In: Proceedings of the 2008 NASA/ESA conference on adaptive hardware and systems, Noordwijk. IEEE Computer Society, pp 32–39
- Gordon T (2005) Exploiting development to enhance the scalability of hardware evolution. PhD thesis, Department of Computer Science, University College, London
- Gordon TGW, Bentley PJ (2002) Towards development in evolvable hardware. In: Proceedings of the 2002 NASA/DoD conference on evolvable hardware. IEEE Computer Society Press, Washington, DC, pp 241–250
- Greensted A, Tyrrell A (2007) RISA: a hardware platform for evolutionary design. In: Proceedings of 2007 IEEE workshop on evolvable and adaptive hardware, Long Beach, CA. IEEE, pp 1–7
- Greenwood G, Tyrrell AM (2007) Introduction to evolvable hardware. IEEE Press, Los Alamitos, CA
- Gross R, Bonani M, Mondada F, Dorigo M (2006) Autonomous self-assembly in Swarm-Bots. *IEEE Trans Robot* 22(6):1115–1130
- Gwaltney D, Dutton K (2005) A VHDL core for intrinsic evolution of discrete time filters with signal feedback. In: Proceedings of the 2005 NASA/DoD conference on evolvable hardware. IEEE Computer Society, Washington, DC, USA, pp 43–50
- Haddow PC (2008) Evolvable hardware: a tool for reverse engineering of biological systems. In: Proc. of the 8th int. conference on evolvable systems: from biology to hardware. LNCS, vol 5216. Springer Verlag, Berlin, pp 342–351
- Harding S (2008) Evolution of image filters on graphics processor units using Cartesian genetic programming. In: 2008 IEEE world congress on computational intelligence. IEEE CIS, Hong Kong, pp 1921–1928
- Harding SL (2006) Evolution in materio. Ph.D. thesis, University of York
- Harding SL, Miller JF, Rietman EA (2008) Evolution in materio: exploiting the physics of materials for computation. *J Unconventional Comput* 4(2):155–194
- Harris SP, Ifeachor EC (1995) Automating IIR filter design by genetic algorithm. In: Proceedings of the first IEE/IEEE international conference on genetic algorithms in engineering systems: innovations and applications (GALESIA'95), vol 414. IEE, London, pp 271–275
- Hauck S, DeHon A (2008) Reconfigurable computing: the theory and practice of FPGA-based computation. Morgan Kaufmann, Seattle, WA
- Haupt RL, Werner DH (2007) Genetic algorithms in electromagnetics. Wiley-IEEE Press, Hoboken, NJ
- Henrici F, Becker J, Buhmann A, Ortmanns M, Manoli Y (2007) A continuous-time field programmable analog array using parasitic capacitance GM-C filters. In: Proceedings of the IEEE international symposium on circuits and systems. IEEE New Orleans, LA, pp 2236–2239
- Higuchi T, Iwata M, Keymeulen D, Sakanashi H, Murakawa M, Kajitani I, Takahashi E, Toda K, Salami M, Kajihara N, Otsu N (1999) Real-world applications of analog and digital evolvable hardware. *IEEE Trans Evolut Comput* 3(3):220–235
- Higuchi T, Liu Y, Yao X (2006) Evolvable hardware. Springer, Berlin
- Higuchi T, Niwa T, Tanaka T, Iba H, de Garis H, Furuya T (1993) Evolving hardware with genetic learning: a first step towards building a Darwin machine. In: Proceedings of the 2nd international conference on simulated adaptive behavior. MIT Press, Cambridge, MA, pp 417–424
- Hornby G, Globus A, Linden D, Lohn J (2006) Automated antenna design with evolutionary algorithms. In: Proceedings 2006 AIAA Space Conference. AIAA, San Jose, CA, pp 1–8
- Hounsell BI, Arslan T, Thomson R (2004) Evolutionary design and adaptation of high performance digital filters within an embedded reconfigurable fault tolerant hardware platform. *Soft Comput* 8(5): 307–317
- Huelsbergen L, Rietman E, Slous R (1999) Evolving oscillators in silico. *IEEE Trans Evolut Comput* 3(3):197–204

- Ifeachor E, Jervis B (2002) Digital signal processing: a practical approach (2nd edn). Prentice-Hall Upper Saddle River, NJ
- Kajitani I, Hoshino T, Nishikawa D, Yokoi H, Nakaya S, Yamauchi T, Inuo T, Kajihara N, Iwata M, Keymeulen D, Higuchi T (1998) A gate-level EHW chip: implementing GA operations and reconfigurable hardware on a single LSI. In: Proceedings of the 2nd International conference on evolvable systems: from biology to hardware ICES' 98, Lausanne, Switzerland, LNCS, vol 1478. Springer, pp 1–12
- Kajitani I, Iwata M, Higuchi T (2006) A GA hardware engine and its applications. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware, Springer, Heidelberg, pp 41–63
- Kajitani I, Sekita I, Otsu N, Higuchi T (2001) Improvements to the action decision rate for a multi-function prosthetic hand. In: The first international symposium on measurement, analysis and modeling of human functions. Sapporo, pp 84–89
- Kamalian R, Zhou N, Agogino M (2002) A comparison of MEMS synthesis techniques. In: Proceedings of the 1st Pacific Rim Workshop on Transducers and Micro/Nano Technologies. Xiamen, China, pp 239–242
- Kasai Y, Takahashi E, Iwata M, Iijima Y, Sakanashi H, Murakawa M, Higuchi T (2005) Adaptive waveform control in a data transceiver for multi-speed IEEE 1394 and USB communication. In: Evolvable systems: from biology to hardware, 6th International conference, ICES 2005, Sitges, Spain, LNCS, vol 3637. Springer, Berlin, 198–204
- Keymeulen D, Durantez M, Konaka K, Kuniyoshi Y, Higuchi T (1997) An evolutionary robot navigation system using a gate-level evolvable hardware. In: Proceedings of the 1st International conference on evolvable systems: from biology to hardware ICES'96, LNCS, vol 1259. Tsukuba, Japan, Springer, Berlin, pp 195–209
- Keymeulen D, Ferguson MI, Breuer L, Fink W, Oks B, Peay C, Terrile R, Kim Y-CD, MacDonald E, Foor D (2006) Hardware platforms for electrostatic tuning of MEMS gyroscope using nature-inspired computation. In: Higuchi T, Liu Y, Yao X (eds) Evolvable Hardware. Springer, Berlin, pp 209–222
- Keymeulen D, Zebulum R, Jin Y, Stoica A (2000) Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Trans Reliability* 49(3):305–316
- Kitano H (1999) Morphogenesis for evolvable systems. In: Towards evolvable hardware: the evolutionary engineering approach. LNCS, vol 1062. Springer, Berlin, pp 99–117
- Koza JR, Al-Sakran SH, Jones LW (2005) Automated re-invention of six patented optical lens systems using genetic programming. In: GECCO'05: Proceedings of the 2005 conference on genetic and evolutionary computation. ACM, New York, NY, USA, pp 1953–1960
- Koza JR, Bennett FH, Andre D, Keane MA (1999) Genetic programming III: Darwinian invention and problem solving. Morgan Kaufmann Publishers, San Francisco, CA
- Koza JR, Keane MA, Streeter MJ, Mydlowec W, Yu J, Lanza G (2003) Genetic programming IV: routine human-competitive machine intelligence. Kluwer Academic Publishers, Norwell, MA
- Langeheine J (2005) Intrinsic hardware evolution on the transistor level. Ph.D. thesis, Rupertus Carola University of Heidelberg
- Larsson E (2005) Introduction to advanced system-on-chip test design and optimization. Springer, Dordrecht
- Layzell PJ (1998) A new research tool for intrinsic hardware evolution. In: Proceedings of the evolvable systems: from biology to hardware conference. LNCS, vol 1478. Springer, Lausanne, Switzerland pp 47–56
- Li H, Antonsson EK (1998) Genetic algorithms in MEMS synthesis. In: Proceedings of IMECE'98 1998 ASME International mechanical engineering congress and expositions, Anaheim, CA
- Linden D (1997) Automated design and optimization of antennas using genetic algorithms. PhD thesis, MIT Cambridge
- Linden DS (2001) A system for evolving antennas in-situ. In: EH'01: Proceedings of the 3rd NASA/DoD workshop on evolvable hardware, IEEE Computer Society, Washington, DC, USA, pp 249–255
- Lipson H, Pollack JB (2000) Automatic design and manufacture of robotic lifeforms. *Nature* 406:974–978
- Lohn JD, Hornby GS (2006) Evolvable hardware: using evolutionary computation to design and optimize hardware systems. *IEEE Computat Intell Mag* 1(1):19–27
- Lohn JD, Kraus WF, Hornby GS (2007) Automated design of a MEMS resonator. In: Proceedings of the IEEE congress on evolutionary computation, Singapore, pp 3486–3491
- Loktev M, Soloviev O, Vdovin G (2003) Adaptive optics – product guide. OKO Technologies, Delft
- Mange D, Sipper M, Stauffer A, Tempesti G (2000) Towards robust integrated circuits: the embryonics approach. *Proc IEEE* 88(4):516–541
- Martinek T, Sekanina L (2005) An evolvable image filter: experimental evaluation of a complete hardware implementation in FPGA. In: Evolvable systems: from biology to hardware, LNCS, vol 3637. Springer Verlag, Sitges, Spain, pp 76–85
- Miller J, Job D, Vassilev V (2008) Principles in the evolutionary design of digital circuits – Part I. Genet programming evol Mach 1(1):8–35

- Miller J, Thomson P (2000) Cartesian genetic programming. In: Proceedings of the 3rd European conference on genetic programming EuroGP2000, LNCS, vol 1802. Springer, Edinburgh, Scotland, pp 121–132
- Moreno JM, Eriksson J, Iglesias J, Villa AEP (2005) Implementation of biologically plausible spiking neural networks models on the poetic tissue. In: Proceedings of evolvable systems: from biology to hardware, Sitges, Spain, LNCS, vol 3637. Springer, pp 188–197
- Murakawa M, Kasai Y, Sakanashi H, Higuchi T (2006) Evolvable analog ISI. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware. Springer, Berlin, pp 121–143
- Murakawa M, Yoshizawa S, Kajitani I, Furuya T, Iwata M, Higuchi T (1996) Evolvable hardware at function level. In: Parallel problem solving from nature PPSN IV. LNCS, vol 1141. Springer, Berlin, pp 62–71
- Murakawa M, Yoshizawa S, Adachi T, Suzuki S, Takasuka K, Iwata M, Higuchi T (1998) Analogue EHW chip for intermediate frequency filters. In: Evolvable systems: from biology to hardware, second International conference, ICES 98, Lausanne, Switzerland, LNCS, vol 1478. Springer, Heidelberg, pp 134–143
- Nedjah N, de Macedo Mourelle L (2005) Evolutionary synthesis of synchronous finite state machines. In: Nedjah N, de Macedo Mourelle L (eds) Evolvable machines: theory and practice. Springer, Berlin, pp 103–127
- Nolfi S, Floreano D (2000) Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines. MIT Press/Bradford Books, Cambridge, MA
- Nosato H, Murakawa M, Kasai Y, Higuchi T (2006) Evolvable optical systems. In: Higuchi T, Liu Y, Yao X (eds) Evolvable Hardware. Springer, Heidelberg, pp 200–207
- Pecenka T, Sekanina L, Kotasek Z (2008) Evolution of synthetic RTL benchmark circuits with predefined testability. ACM Trans Des Autom Electron Syst 13(3):1–21
- Sakanashi H, Iwata M, Higuchi T (2001) A lossless compression method for halftone images using evolvable hardware. In: Evolvable systems: from biology to hardware, 4th International conference, ICES 2001 Tokyo, Japan. LNCS, vol 2210. Springer, Berlin, pp 314–326
- Sakanashi H, Iwata M, Higuchi T (2006) EHW applied to image data compression. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware. Springer, Berlin, pp 19–40
- Salomon R, Widiger H, Tockhorn A (2006) Rapid evolution of time-efficient packet classifiers. In: IEEE congress on evolutionary computation, IEEE CIS, Vancouver, Canada, pp 2793–2799
- Sekanina L (2003) Virtual reconfigurable circuits for real-world applications of evolvable hardware. In: Evolvable systems: from biology to hardware, fifth international conference, ICES 2003. LNCS, vol 2606. Springer, Trondheim, Norway, pp 186–197
- Sekanina L (2004) Evolvable components: from theory to hardware implementations. Natural Computing, Springer Verlag, Berlin
- Sekanina L (2007) Evolved computing devices and the implementation problem. Minds Mach 17(3):311–329
- Sekanina L, Bidlo M (2005) Evolutionary design of arbitrarily large sorting networks using development. Genet Programming Evol Mach 6(3):319–347
- Sekanina L, Friedl S (2004) An evolvable combinational unit for FPGAs. Comput Informatics 23(5):461–486
- Sekanina L, Ruzicka R, Vasicek Z, Prokop R, Fujcik L (2009) Repomo32 – new reconfigurable polymorphic integrated circuit for adaptive hardware. In: Proceedings of 2009 IEEE workshop on evolvable and adaptive hardware. IEEE CIS, Nashville, TN, pp 39–46
- Sekanina L, Starecek L, Kotasek Z, Gajda Z (2008) Polymorphic gates in design and test of digital circuits. Int J Unconventional Comput 4(2):125–142
- Stoica A, Keymeulen D, Zebulum RS, Guo X (2006) Reconfigurable electronics for extreme environments. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware. Springer, Heidelberg, pp 145–160
- Stoica A, Keymeulen D, Zebulum RS, Katkoori S, Fernando P, Sankaran H, Mojarradi M, Daud T (2008) Self-reconfigurable mixed-signal integrated circuits architecture comprising a field programmable analog array and a general purpose genetic algorithm IP core. In: Evolvable systems: from biology to hardware, 8th International conference, ICES 2008. LNCS, vol 5216. Springer, Prague, pp 225–236
- Stoica A, Wang X, Keymeulen D, Zebulum RS, Ferguson MI, Guo X (2005) Characterization and recovery of deep sub micron (DSM) technologies behavior under radiation. In: 2005 IEEE Aerospace Conference. IEEE, Montana, pp 1–9
- Stoica A, Zebulum R, Keymeulen D (2000) Mixtrinsic evolution. In: Proceedings of the 3rd International conference on evolvable systems: from biology to hardware ICES'00, Edinburgh, Scotland, UK, LNCS, vol 1801. Springer, pp 208–217
- Stoica A, Zebulum RS, Keymeulen D (2001) Polymorphic electronics. In: Proceedings of evolvable systems: from biology to hardware conference, LNCS, vol 2210. Springer, Tokyo, Japan, pp 291–302
- Stoica A, Zebulum RS, Ferguson MI, Keymeulen D, Duong V (2002a) Evolving circuits in seconds: experiments with a stand-alone board-level evolvable system. In: Proceedings of the 2002 NASA/DoD conference on evolvable hardware (EH'02). IEEE Computer Society, Washington, DC, USA, pp 67–64

- Stoica A, Zebulum RS, Keymeulen D, Lohn J (2002b) On polymorphic circuits and their design using evolutionary algorithms. In: Proceedings of IASTED international conference on applied informatics AI2002. Innsbruck, Austria
- Stoica A, Zebulum R, Guo X, Keymeulen D, Ferguson I, Duong V (2004a) Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration. IEE Proc Comp Digit Technol 151(4):295–300
- Stoica A, Zebulum RS, Keymeulen D, Ferguson MI, Duong V, Guo X (2004b) Evolvable hardware techniques for on-chip automated reconfiguration of programmable devices. Soft Comput 8(5):354–365
- Stomeo E, Kalganova T, Lambert C (2006) Generalized disjunction decomposition for evolvable hardware. IEEE Trans Syst Man Cybern Part B 36(5):1024–1043
- Takahashi E, Kasai Y, Murakawa M, Higuchi T (2006) Post-fabrication clock-timing adjustment using genetic algorithms. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware, Springer, Heidelberg, pp 65–84
- Tempesti G, Mange D, Mudry PA, Rossier J, Stauffer A (2007) Self-replicating hardware for reliability: The embryonics project. JETC 3(2):1–21
- Terrire R, Aghazarian H, Ferguson MI, Fink W, Huntsberger TL, Keymeulen D, Klimeck G, Kordon MA, Lee S, von Allmen P (2005) Evolutionary computation technologies for the automated design of space systems. In: 2005 NASA / DoD conference on evolvable hardware (EH 2005). IEEE Computer Society, Washington, DC, pp 131–138
- Thompson A (1996) Silicon evolution. In: Proceedings of genetic programming GP'96. MIT Press, Cambridge, MA, pp 444–452
- Thompson A (1999) Hardware evolution: automatic design of electronic circuits in reconfigurable hardware by artificial evolution. Springer, London
- Thompson A, Layzell P, Zebulum S (1999) Explorations in design space: unconventional electronics design through artificial evolution. IEEE Trans Evolut Comput 3(3):167–196
- Torresen J (1998) A divide-and-conquer approach to evolvable hardware. In: Proceedings of the 2nd International conference on evolvable systems: from biology to hardware ICES'98. LNCS, vol 1478. Springer, Lausanne, Switzerland, pp 57–65
- Torresen J (2002) A scalable approach to evolvable hardware. Genetic programming and evolvable machines 3(3):259–282
- Tour JM (2003) Molecular electronics. World Scientific, Singapore
- Tufte G, Haddow P (2000) Evolving an adaptive digital filter. In: The second NASA/DoD workshop on evolvable hardware. IEEE Computer Society, Palo Alto, CA, pp 143–150
- Tufte G, Haddow PC (2005) Towards development on a silicon-based cellular computing machine. Nat Comput 4(4):387–416
- Upegui A (2006) Dynamically reconfigurable bio-inspired hardware. Ph.D. thesis, EPFL
- Upegui A, Sanchez E (2006) Evolving hardware with self-reconfigurable connectivity in Xilinx FPGAs. In: The 1st NASA/ESA conference on adaptive hardware and systems (AHS-2006), IEEE Computer Society, Los Alamitos, CA, USA, pp 153–160
- Vasicek Z, Sekanina L (2007a) An evolvable hardware system in Xilinx Virtex II Pro FPGA. Int J Innovative Comput Appl 1(1):63–73
- Vasicek Z, Sekanina L (2007b) An area-efficient alternative to adaptive median filtering in FPGAs. In: Proceedings of 2007 conference on field programmable logic and applications. IEEE Computer Society, Los Alamitos, CA, pp 216–221
- Vasicek Z, Sekanina L (2008) Hardware accelerators for Cartesian genetic programming. In: Proceedings of the 12th European conference on genetic programming, Naples, Italy. LNCS, vol 4971, pp 230–241
- Vasicek Z, Zadnik M, Sekanina L, Tobola J (2008) On evolutionary synthesis of linear transforms in FPGA. In: Proceedings of the 8th International conference on evolvable systems: from biology to hardware. LNCS, vol 5216. Springer Verlag, Berlin, pp 141–152
- Vassilev V, Job D, Miller J (2000) Towards the automatic design of more efficient digital circuits. In: Lohn J, Stoica A, Keymeulen D, Colombano S (eds) Proceedings of the 2nd NASA/DoD workshop on evolvable hardware. IEEE Computer Society, Los Alamitos, CA, USA, pp 151–160
- Voronenco Y, Püschel M (2007) Multiplierless multiple constant multiplication. ACM Trans Algorithms 3(2):1–282
- Wade G, Roberts A, Williams G (1994) Multiplier-less FIR filter design using a genetic algorithm. IEE Proc Vis Image Signal Process 141(3):175–180
- Walker JA, Miller J (2008) The automatic acquisition, evolution and re-use of modules in Cartesian genetic programming. IEEE Trans Evolut Comput 12(4):397–417
- Xilinx Inc. (2009) URL: <http://www.xilinx.com>
- Zebulum R, Keymeulen D, Ramesham R, Sekanina L, Mao J, Kumar N, Stoica A (2006) Characterization and synthesis of circuits at extreme low temperatures. In: Higuchi T, Liu Y, Yao X (eds) Evolvable hardware, Springer, Berlin, pp 161–172
- Zebulum R, Pacheco M, Vellasco M (2002) Evolutionary electronics – automatic design of electronic circuits and systems by genetic algorithms. The CRC Press International Series on Computational Intelligence. Boca Raton, FL

Zhan S, Miller JF, Tyrrell AM (2008) A developmental gene regulation network for constructing electronic circuits. In: Proceedings of the 8th international conference on evolvable systems: from biology to hardware. LNCS, vol 5216. Springer Verlag, Berlin, pp 177–188

Zhang Y, Smith S, Tyrrell A (2004) Intrinsic evolvable hardware in digital filter design. In: Applications of Evolutionary Computing, Coimbra, Portugal. LNCS, vol 3005. Springer Verlag, pp 389–398

51 Natural Computing in Finance – A Review

Anthony Brabazon¹ · Jing Dang² · Ian Dempsey³ · Michael O'Neill⁴ · David Edelman⁵

¹Natural Computing Research and Applications Group, University College Dublin, Ireland
anthony.brabazon@ucd.ie

²Natural Computing Research and Applications Group, University College Dublin, Ireland
jing.dang@ucd.ie

³Pipeline Financial Group, Inc., New York, NY, USA
ian.dempsey@pipelinefinancial.com

⁴Natural Computing Research and Applications Group, University College Dublin, Ireland
m.oneill@ucd.ie

⁵School of Business, UCD Michael Smurfit Graduate Business School, Dublin, Ireland
david.edelman@ucd.ie

1	<i>Introduction</i>	1708
2	<i>Natural Computing</i>	1709
3	<i>Financial Applications</i>	1711
4	<i>The Future</i>	1729

Abstract

The field of natural computing (NC) has advanced rapidly over the past decade. One significant offshoot of this progress has been the application of NC methods in finance. This chapter provides an introduction to a wide range of financial problems to which NC methods have been usefully applied. The chapter also identifies open issues and suggests future directions for the application of NC methods in finance.

1 Introduction

Recent years have seen the application of multiple natural computing (NC) algorithms (defined in this chapter as computer algorithms whose design draws inspiration from phenomena in the natural world) for the purposes of financial modeling (Brabazon and O'Neill 2006). Particular features of financial markets including their dynamic and interconnected characteristics bear parallel with processes in the natural world and, *prima facie*, this makes NC methods “interesting” for financial modeling applications. Another feature of both natural and financial environments is the phenomenon of emergence, or the activities of multiple individual agents combining to coevolve their own environment.

The scale of NC applications in finance is illustrated by Chen and Kuo (2002) who list nearly 400 papers that had been published by 2001 on the use of evolutionary computation alone in computational economics and finance. Since then several hundred additional papers have been published underscoring the continued growth in this application area (see also Barbazon and O'Neill 2008, 2009; Chen 2002; Tsang and Martinez-Jaramillo 2004 and Wong et al. 2000 for additional examples of NC applications in finance).

Some of the major areas of financial applications using NC methods are: forecasting, algorithmic trading, portfolio management, risk management, derivatives modeling and market modeling. This chapter describes the utility of NC methods within each of these areas wherein their usage can be broadly categorized as optimization, model induction, and agent-based modeling.

1.1 Optimization

A wide variety of NC methodologies including genetic algorithms, evolutionary strategies, differential evolution, and particle swarm optimization have been applied for optimization purposes in finance. A particular advantage of these methodologies is that, if applied properly, they can cope with “difficult” search spaces. Examples of the use of optimization techniques in finance include optimal asset allocation, stock selection, risk management, pricing and hedging of options, and asset liability management (Zenios 2008).

1.2 Model Induction

While optimization applications of natural computing are important, the underlying model or data-generating process is not known in many real-world financial applications. Hence, the task is often to “recover” or discover an underlying model from a dataset. This is usually a difficult task as both the model structure and associated parameters must be uncovered.

Financial markets are affected by a myriad of interacting economic, political, and social events. The relationship between these factors and financial asset prices is not well understood and, moreover, is not stationary over time. Most theoretical financial asset pricing models are based on strong assumptions which are often not met in real-world asset markets. This offers opportunities for the application of model induction methodologies in order to recover the underlying data-generating processes. These methods can be applied, for example, to financial forecasting, credit risk assessment, and derivatives pricing.

1.3 Agent-Based Modeling

Agent-based modeling (ABM) has become a fruitful area of financial and economic research in recent years. ABM allows the simulation of markets which consist of heterogeneous agents, with differing risk attitudes and differing expectations to future outcomes, in contrast to traditional assumptions of investor homogeneity and rational expectations. ABM attempts to explain market behavior, replicate documented features of real-world markets, and allows one to gain insight into the likely outcomes of different regulatory policy choices.

A growing community of researchers are engaged in the application of natural computing methodologies in finance as illustrated by the number of conferences, workshops, and special sessions in this area. Examples of these include the annual track on Evolutionary Computation in Finance and Economics at the IEEE Congress on Evolutionary Computation, the IEEE Symposium on Computational Intelligence for Financial Engineering (CIFEr), the annual International Conference on Computational Intelligence in Economics & Finance (CIEF), and the European Workshop on Evolutionary Computation in Finance (EvoFIN) held annually as part of Evo*.

1.4 Structure of the Chapter

The rest of this chapter is organized as follows: [Section 2](#) provides a concise overview of a number of key families of natural computing methods. [Section 3](#) introduces various financial applications of natural computing methods and shows how these methodologies can add value in those applications. [Section 4](#) concludes this chapter, suggesting multiple avenues of future work at the intersection of finance and natural computing.

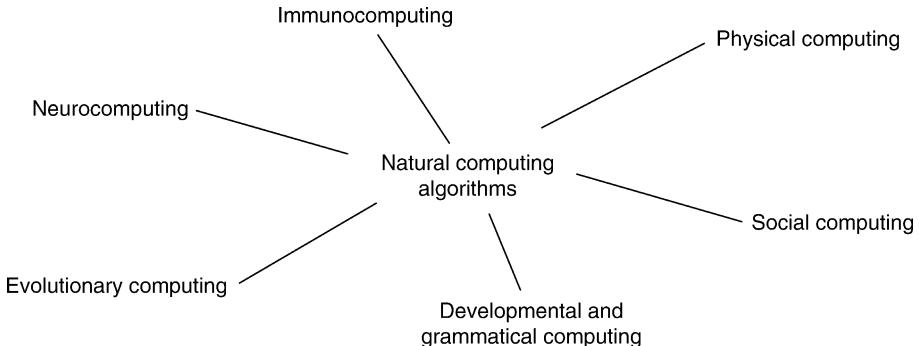
2 Natural Computing

NC algorithms can be clustered into different groups depending on the aspects of the natural world upon which they are based. The main clusters that are relevant for finance applications illustrated in this chapter are neurocomputing, evolutionary computing, social computing, immunocomputing, physical computing, and developmental and grammatical computing (see [Fig. 1](#)).

Neurocomputing (or neural networks, NNs) typically draws inspiration from simplified models of the workings of the human brain or the nervous system. From a design perspective, neural networks can be characterized by a set of neurons (or nodes), the network structure which describes the pattern of connectivity between neurons, and the learning (or training)

Fig. 1

An overview of natural computing algorithms.



approach used. The predominant neurocomputing paradigms include *feedforward networks*, *recurrent networks*, *self-organizing networks*, *radial basis function networks*, *support vector machines* (Vapnik 1995; Cristianini and Shawe-Taylor 2000), etc. Financial firms worldwide are employing NNs to tackle difficult tasks involving intuitive judgment or requiring the detection of data patterns which elude conventional analytic techniques. For example, NNs are already being used to trade the securities markets, to forecast the economy, and to analyze credit risk. NNs were among the earliest NC methodologies to see widespread applications in finance (see, for example, Trippi and Turban 1993) but they do suffer from the practical drawback that their black-box nature makes their internal workings opaque to the user.

Evolutionary Computation (EC) is based upon neo-Darwinian principles of evolution. A population-based search process is used, whereby better (fitter) members of the population are preferentially selected for reproduction and modification, leading to a new population of individuals increasingly adapted to their environment. The main streams of EC are *genetic algorithms* (GA), *evolution strategies* (ES), *evolutionary programming* (EP), and *genetic programming* (GP). These methods have broad applications for optimization and model induction purposes. Recent extensions of the literature on EC to encompass dynamic, multi-objective, and constrained optimization problems has greatly increased the practical utility of these algorithms in finance.

Social Computing adopts a swarm metaphor and includes algorithms inspired by the flocking and schooling behavior of birds and fish. It also includes algorithms inspired by behaviors observed in social insects such as ants. These social systems exhibit a number of characteristics facilitating self-organization, flexibility, robustness, and direct or indirect communication among members of the population. Some examples of social computing include *ant colony*, *particle swarm*, and *bacterial foraging* algorithms. These algorithms are population-based like their evolutionary computation counterparts, and they operate by allowing the population of problem-solvers to communicate their relative success in solving the problem to each other, thereby biasing the future actions of the individuals in the population.

Immunocomputing encompasses a family of algorithms, which turn to the complex and adaptive biological immune system of vertebrates to inspire their design. The natural immune system represents an intricate network of specialized chemicals, cells, tissues, and organs with the ability to recognize, destroy, and remember an almost unlimited number of foreign bodies,

and to protect the organism from misbehaving cells in the body. These properties are especially useful for tasks such as classification and optimization. Practical applications of immuno-computing include financial pattern-recognition such as the identification of potentially fraudulent credit card transactions, the identification of financially at-risk companies, and the identification of market “state.”

Physical Computing draws inspiration from the physical processes of the natural world to design computational algorithms. These algorithms draw inspiration from phenomena such as simulated annealing and quantum mechanics. A claimed benefit of the quantum-inspired algorithms is that, because they use a quantum representation, they can maintain a good balance between exploration and exploitation. It is also suggested that they offer computational efficiencies, as use of a quantum representation can allow the use of smaller population sizes than typical evolutionary algorithms. Computational efficiency is important in many financial applications such as real-time trading where systems have to deal with large data flows and a dynamic environment. Consequently, there is a continuing demand for optimization algorithms, which can potentially offer efficiency gains.

Developmental and Grammatical Computing borrows from both a developmental and a grammar metaphor. Grammatical computing refers to algorithms, which adopt concepts from linguistic grammars and are dominated by the generative form of grammars. Generative grammars are used to construct a “sentence” in the language specified by the grammar, and this generative process is metaphorically similar to the developmental process in biology in which “rules” govern the production of a complex, multicellular organism from a single embryonic cell. Generative grammars have been used in natural computing as a convenient representation by which developmental systems can be realized *in-silico*. The implementations of developmental and grammatical computing, such as *grammatical evolution* (GE) (O’Neill and Ryan 2001, 2003) (a grammatical variant of GP) may also embed an evolutionary algorithm typically used to drive the search process. GE has been already successfully applied to financial forecasting, credit rating assessment, and other financial applications.

These families of NC algorithms provide a rich set of tools for the development of quality optimization, model induction, and agent-based modeling applications, and all have seen application in finance. Readers requiring detailed information on these algorithms are referred to earlier chapters in this book. A review of these methods can also be found in Brabazon and O’Neill (2006), de Castro (2007), and Kari and Rozenberg (2008).

3 Financial Applications

This section introduces the application of NC methods across a range of financial areas including forecasting, algorithmic trading, portfolio management, risk management, derivatives modeling, and agent-based market modeling.

3.1 Forecasting

Financial forecasting applications may involve the prediction of future values of macroeconomic variables, individual stock market indices, commodity futures, the volatility of some financial products, etc. From an optimization perspective, NC methods can be applied either for variable identification or for parameter estimation (optimization). For example, NC

methods can be used to select the explanatory variables from a large pool of candidates, which are then incorporated into the forecasting model. Even where the modeler knows the appropriate set of explanatory variables and model form, the selection of appropriate model parameters (coefficients) can be a difficult task, particularly for complex, nonlinear model structures. In the more difficult problem where the model form is not known, model induction methodologies such as GP or NNs can be used. This is potentially of considerable importance in financial applications, as many theoretical forecasting models are based on assumptions which are not met in real-world financial markets. This offers opportunities for the application of NC methods as model induction tools in order to “recover” the underlying data-generating processes directly from the data.

One family of NC methods which has seen extensive application for financial forecasting is NNs (e.g., Wong and Selvi 1998; Kaastra and Boyd 1996; Aiken 2000; Cao and Tay 2003a Edelman 2007). They offer particular advantages due to their ability to identify nonlinear models, handle noisy data, and embed a memory (recurrent NNs). A simple case of index prediction using a basic feedforward multilayer perceptrons (MLP) to construct a financial prediction model is illustrated in Brabazon and O’Neill (2006), where the MLP model is employed to predict the 5-day percentage change in the value of the FTSE 100 index. Ten inputs selected from a range of technical, fundamental, and intermarket data were included in the final model:

1. 5-Day lagged percentage change of the FTSE 100 index
2. 20-Day lagged percentage change of the FTSE 100 index
3. Ratio of the 10 vs 5-day moving average of the FTSE 100 index
4. Ratio of the 20 vs 10-day moving average of the FTSE 100 index
5. Bank of England Sterling index
6. S&P 500 composite index_{(t) – (t-5)}
7. LIBOR 1-month deposit rate
8. LIBOR 1-year deposit rate
9. Aluminum (\$ per tonne)
10. Oil (\$ per barrel)

In developing the final MLP models, a 11:6:1 structure was utilized, as illustrated in Fig. 2

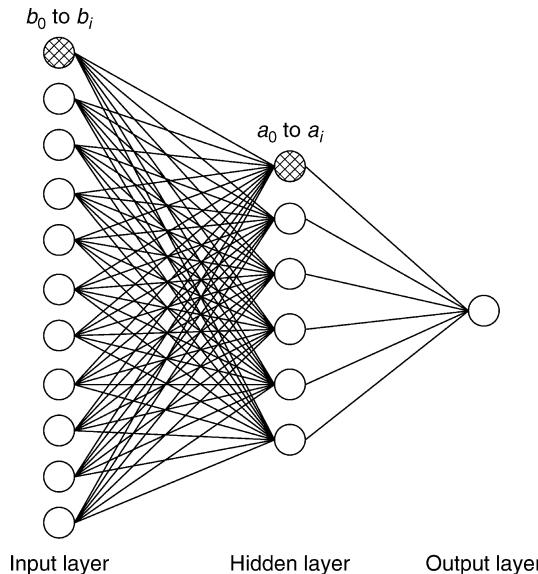
$$y_t = L \left(\sum_{j=0}^5 w_j L \left(\sum_{i=0}^{10} b_i w_{ij} \right) \right)$$

where b_i represents *input_i* (b_0 is a bias node), w_{ij} represents the weight between input node_i and hidden node_j, w_j represents the weight between hidden node_j and the output node, and L represents the hyperbolic tangent function. Generally, NNs are developed through a trial-and-error approach guided by heuristics, the process is time consuming, and there is no guarantee that the final network structure is optimal. One approach is to automate the construction of the NN using evolutionary approaches (e.g., Aiken and Bsat 1999; Armano et al. 2005; and Cao and Tay 2003b).

GP can also be applied for forecasting. One of the best-known examples, EDDIE (which stands for *Evolutionary Dynamic Data Investment Evaluator*), was developed as an interactive decision tool (Tsang and Lee (2002); Li (2001)). It is a *genetic-programming*-based system for channeling expert knowledge into forecasting. Given a set of variables, EDDIE attempts to find interactions among variables and discover nonlinear functions.

Fig. 2

Index prediction using MLP model, with 11 input nodes, six hidden nodes, and a single output node.



Other examples of forecasting in the financial literature include the prediction of takeover targets (Harris et al. 1982; Belkaoui 1978; Palepu 1986; Rege 1984; Hickey et al. 2006), the prediction of auditor qualification of the financial statements of a company (Mutchler 1985; Dopuch et al. 1987; Thompson et al. 2007), the prediction of earnings and the prediction of IPO underpricing. In earnings prediction, Trigueros (1999) uses a GA to select explanatory variables from financial statements in order to predict corporate earnings. In the prediction of IPO underpricing, Quintana et al. (2005) employs GA for rule-based prediction. A less common, but nonetheless important, application in the literature is the prediction of volatility. Neely and Weller (2002) use GP to predict exchange rate volatility, where GP is applied for producing forecasting rules of the out-of-sample daily volatility in the foreign exchange market.

Typically, studies applying NNs, GA, or GP methods, for financial time series forecasting, use measures of goodness of fit drawn from statistics such as *mean squared error*, *sum of squared error*, *mean absolute percentage error*, etc., as their error (or fitness) function. The aim is to uncover or train a model using historical data, which “fits” that data well. Unsurprisingly, the choice of fitness function usually has a critical impact on the behavior of resulting model, hence a model constructed using one fitness metric would not necessarily perform well on another. While many forecasting studies applying NC methods to financial data indicate that models could be constructed to fit historic data fairly well, a common finding is that the quality of the forecasts diminishes, over time, out of sample.

3.2 Algorithmic Trading

Algorithmic trading is defined here as the use of computer programs to assist with any aspect of the trading of financial assets. It can therefore encompass systems which decide on certain

aspects of the order such as the timing, price, or even the final quantity of the order. Hence, algorithmic trading can be combined with any investment strategy. A few related processes of algorithmic trading where NC methods can be applied, namely, investment analysis, arbitrage, and trade execution, are illustrated below.

3.2.1 Investment Analysis

Financial trading has seen a large number of applications of NC methods. Typically, these studies take one of two approaches of investment analysis, using either fundamental data (*fundamental analysis*) or market data, primarily price and volume (*technical analysis*).

Fundamental Analysis

Taking the example of investing in stocks, fundamental investment concentrates on the use of accounting information about the company, as well as industry and macroeconomic data, in order to identify companies which are mispriced by the market. In other words, the objective is to identify stocks which are good value (underpriced by the market), or stocks which are overpriced by the market (and therefore are candidates for “shorting”). In this approach, the investor needs to develop stock screening rules in order to decide which stocks to invest in. These rules were formulated manually in decades before computers. With a natural computing algorithm such as the GA, a large range of stock filter rules can be searched efficiently in order to find the highest-quality rules. In this approach, each individual in the population corresponds to a potential stock filter rule. The utility of these rules are tested using historical data, with the best rule (or set of rules) then being used for investment purposes (☞ Fig. 3). More generally, GP methods can be applied to evolve the structure of the filter rules.

Technical Analysis

In contrast to investors using a fundamental investment approach, technical analysts attempt to identify imbalances in the supply and demand for a financial asset using information from the time-series of the asset’s trading (such as historical price, volume, and volatility data). Usually, investors who adopt a technical analysis approach look to combine technical indicators (preprocessed price and volume time series data about a financial asset), in order to produce a “trading signal”. For example, a “technical indicator” could be the *moving average convergence-divergence* (MACD) oscillator, calculated by taking the difference of a short-run and a long-run moving average. If the difference is positive, it may indicate that the market is

Fig. 3

String encoding of a number of fundamental indicators. Each indicator can be coded as a 0 (no) or 1 (yes).

High sales growth relative to industry average?	High debt level relative to industry average?	High level of cash flow from operations relative to industry average?	High level of liquidity relative to industry average?	High profit level relative to industry average?
---	---	---	---	---

trending upward. For example, a buy signal could be generated when the shorter moving average crosses the longer moving average in an upward direction. A sell signal could be generated in a reverse case. Hence, a sample MACD trading rule could be:

IF x -day MA of price \geq y -day MA of price
THEN Go Long ELSE Go Short

where $x < y$. The optimal value of x and y can be evolved through a genetic algorithm, in order to maximize the trading profit (the fitness measure). A candidate solution encoded as a binary string of length 8 is illustrated below.

$$\begin{array}{r} x: \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \\ \hline y: \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \end{array}$$

This solution indicates that $x = 10$ and $y = 50$. The MACD oscillator is a crude band-pass filter, removing both high-frequency price movements and certain low-frequency price movements, depending on the precise moving average lags selected. In essence, the choice of the two lags produces a filter, which is sensitive to particular price-change frequencies. In a recursive fashion, more complex combinations of moving averages of values calculated from an MACD oscillator can themselves be used to generate trading rules. In the past decades, the search for apparently useful technical indicators (or combinations of these) was undertaken manually by investors who back tested various indicators on historical financial data. GP allows the automation of this process with the concurrent vast expansion of the search space, which can be feasibly searched (Dempster and Jones 2001).

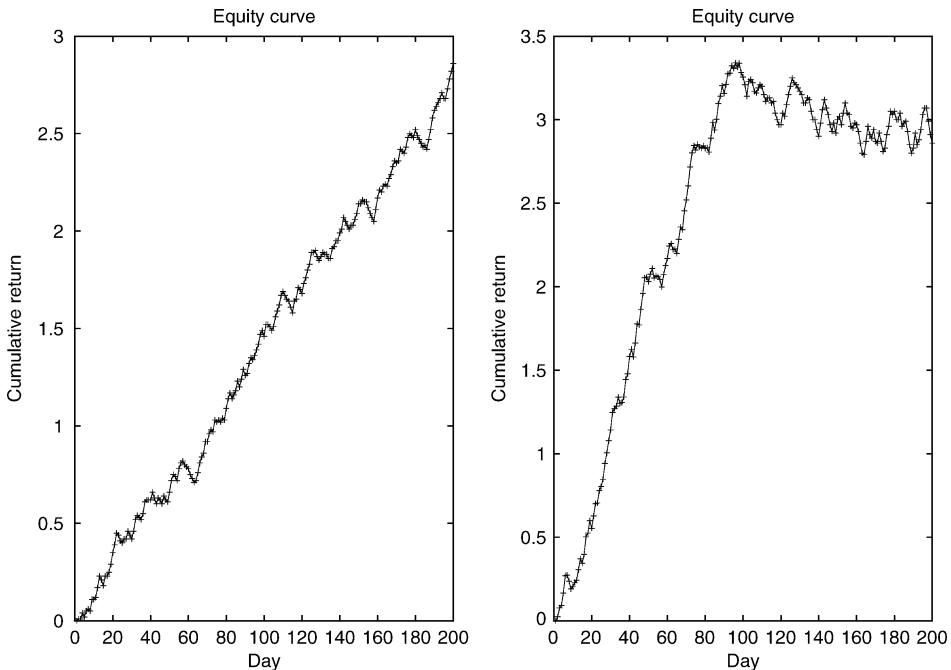
A trading rule should specify the entry, profit-taking, and stop-loss or exit strategy. NC methods can be used for rule optimization (to find optimal parameters for a fixed rule) or rule induction (to find optimal combination of diversified rules). Early applications of EC to uncover trading rules include Bauer (1994), Mutchler (1985), Allen and Karjalainen (1999), and Fyfe et al. (1999). The flexibility to implement different fitness functions is one particular advantage of EC approaches. However, there are also some issues related to the fitness functions as to include transaction cost and to consider different types of risks (Pavlidis et al. 2007). While markets exhibit periods in which a static trading rule can work, it is hard to find evidence of rules which are successful over long time periods. Of course, as financial markets comprise a dynamic system, the utility of any static trading system can be expected to degrade over time (Dempsey et al. 2009). One basic way of examining the characteristics of a trading system is to use an equity curve (Fig. 4). The use of an adaptive trading strategy seems more plausible (Ghandar et al. 2008; da Costa Pereira and Tettamanzi 2008).

Recent work has seen a broadening of the information sources used as inputs in trading models. Instead of typical data drawn from the market, financial statements or macroeconomic data, for example, Thomas and Sycara (2002) and Larkin and Ryan (2008), used text data drawn from either internet message boards/the financial press in the creation of trading rules.

A wide range of forecasting approaches have also been employed to support making trade decisions, such as support vector machines (Edelman and Davy 2004), and hybrid methods like neuro-fuzzy hybrids (Zaiyi et al. 2006), neuro-genetic hybrids, geno-fuzzy (Ghandar et al. 2008) and ensemble methods (combining multiple models) (Kwon and Moon 2004).

Fig. 4

Sample equity curves showing cumulative returns on the y axis and time on the x axis. The left-hand graph exhibits gradual return accumulation, whereas the right-hand graph suggests that the model is working less well on the second half of the time period.



3.2.2 Arbitrage

Arbitrage trading can be defined in a variety of ways, but, broadly speaking, these trades seek to make profits by exploiting price differences of identical or similar financial instruments, on different markets or in different forms (e.g., buying a share at \$23.78 on one exchange, and selling it immediately on another for \$23.82, thereby exploiting a pricing difference for the same asset between the two exchanges). As would be expected, arbitrage opportunities tend to be closed very quickly and transaction cost can negate apparent arbitrage possibilities.

A simple example of an arbitrage play based on *Put-Call Parity* is illustrated in Tung and Quek (2005). In essence, the concept underlying this trade is that the price of a “long” position on an asset and an associated put option (the right to sell that asset in the future at a specified price, see [Sect. 3.5](#) for a detailed description of options) must be equal to the price of a long call option on the same asset and a long position in a risk-free bond. Specifically, for example, for a European option:

$$S_0 + P_E(S_0, T, K) = C_E(S_0, T, K) + K(1 + r)^{-T}$$

where S_0 is the current price of the underlying asset, P_E is the current price of a European put, C_E is the current price of a European call, r is the risk free rate of return, K is the strike price for both options, and T is the time to maturity of the options. If either the put or call option are

mispriced, the investor can, in theory, make a risk-free gain by constructing a portfolio of the four financial instruments.

The above example describes an arbitrage opportunity between the cash market (for the asset) and the option market. More generally, arbitrage opportunities can also exist between cash and future markets and between future and options markets. In its purest form, arbitrage is a risk-free transaction but in reality, most arbitrage trades are exposed to some risk such as liquidity risk or credit risk, and more significantly, execution risk where prices move before all elements of the trade can be completed.

An alternative approach that wait for arbitrage opportunities to emerge and then try to trade on them, is to anticipate or forecast opportunities in advance of the actual mispricing occurring. Markose et al. (2002) adopted this approach and developed a GP model to predict arbitrage opportunities between the FTSE 100 index futures and options market up to 10 min in advance of the arbitrage opportunity arising. Another example of the application of computational intelligence for uncovering arbitrage opportunities is provided in Tung and Quek (2005) who used self-organising, fuzzy NNs to identify mispriced American style currency options between the USD and the GBP, and then use a Delta hedge trading strategy to execute the arbitrage play.

3.2.3 Trade Execution

An important issue in trading financial assets is the efficient execution of large institutional size orders, and applications of EC for this task have begun to emerge in recent years. Typically, orders to buy or sell a stock can be either *market orders* (the transaction is undertaken immediately in the market at current prices) or limit orders (the purchase/sale must occur at a price which is no greater than/or less than a specific price). When a market order is placed, the customer does not have control over the final price, and in a limit order, while the customer has some price control, there is no guarantee that the order will actually be executed. So for example, if a customer places a limit order to buy a stock at \$25 per share, the transaction will only take place if the market price falls to \$25 or less.

Over the last number of years, trading algorithms have been executing an ever-increasing number of trades on markets. In the USA their rise has been brought about through a series of technological and regulatory changes. Since 2001, with the move to decimalization of the US equity markets and the widespread acceptance of electronic market places, the average trade size has declined from 1,200 shares per transaction in 2000 to 300 shares in 2008 (NYSE Euronext). This in turn has led to an explosion in the number of trades executed and a narrowing of spreads, with large institutional orders taking longer to execute. As a result, investors wishing to trade large blocks face trade-offs in balancing the risk of tipping their hand and providing information to the marketplace, thereby suffering market risk as the trade is executed. Trading algorithms seek to optimally execute these orders, using the vast amounts of data produced by the market place and submitting appropriately sized smaller orders to various destinations with the aim of achieving best execution. In reaching this goal, an entire ecology of different trading algorithms have been designed to perform under different market conditions, with recent innovations intelligently switching between these algorithms depending on current market conditions.

When trading shares, particularly when an investor is looking to buy or sell a large quantity of stock, the problem of *market impact* arises. Market impact occurs when the actions of an investor start to move the price adversely against themselves. Hence, market impact is the

difference between a transaction price and what the market price would have been in the absence of the transaction. For example, the order may be executed as quickly as possible through sweeping any orders posted to the limit-order book, however, this would incur significant cost and drive the price of the stock against the investor. In this case, the investor avoids market risk but, by demanding instantaneous liquidity, incurs significant market impact costs. The obvious strategy to minimize market impact is to break the order into smaller lots and spread it over several purchases. While this will reduce the market impact, it incurs the risk of suffering *opportunity cost*, that market prices may start moving against investor during the multiple purchases. Added to this, the steady flow of small orders over time will inform other market participants of the presence of a large institutional order and so encourage competitors to run ahead of the investor. Hence, the design of trade execution strategies is intended to balance out the total cost of market impact and opportunity cost while maintaining a tight control over information leakage.

In selecting a trade execution strategy, the investor must not only balance her preferences but must also be prepared to adapt and change strategy as market conditions evolve. NC techniques provide ample scope to assist in uncovering information that can help optimize trading algorithm selection and/or adaptation. Various rules and heuristics can be evolved and adapted using (for example) GAs that can provide a trading strategy with predictive capability (Stephens and Sukumar 2006) with the aim of selecting best trading tactic under current market conditions. For institutional sized orders, which can be on the order of millions of shares, the reduction in average price by a couple of pennies can lead to significant savings.

Despite the importance of optimizing trade execution, there has been relatively little attention paid in the literature to the application of evolutionary methodologies for this task. One interesting exception is Lim and Coggins (2005) who used a GA to evolve a trading strategy in order to optimize trade execution performance using order book data from the Australian Stock Exchange. In this study, the approach taken was to initially split each trade into a series of N equal sized orders, and the objective was to evolve the timing strategy for the execution of each of these N orders during a single trading day. Each order was submitted as a limit order at the best ask or bid prevailing at the time the order was submitted, depending on whether the investor was seeking to buy or sell shares. A simple traditional trading strategy could be to submit one of these orders every 10 min. However, there is no guarantee that a 10-min order spacing would produce good results in terms of minimizing market impact. Lim and Coggins (2005) used a GA to uncover good quality timings for each order by evolving a chromosome of N genes, where each gene encoded the maximum lifetime that the order would remain on the order book (if it had not already been executed) before it was automatically ticked over the spread (e.g., a limit buy order being repriced to the current ask) to close out the trade. Any uncompleted trade at the end of the day were closed out the same way. Hence, the GA evolved the maximum time that each order would be exposed to the market before being crossed over the spread.

A variety of fitness functions could be designed to drive the evolution of the trading strategy but a common metric of trade execution performance is its *volume weighted average price* (VWAP):

$$\text{VWAP} = \frac{\sum(\text{Price} \cdot \text{Volume})}{\sum(\text{Volume})}$$

The VWAP of a strategy can be calculated and benchmarked against (for example) the overall VWAP for that share during the period of the trading strategy's execution. The aim is to evolve a strategy which produces as competitive a VWAP as possible.

In the above approach, the basic structure of the execution rule is determined in advance (number of trades, etc.) and the task of GA is to parameterize the rule. Another approach which could be applied is to use GP to evolve the structure of the execution rule, as well as its parameters.

In real-world trading, a number of interesting additional issues arise. While the order book provides an indication of the current state of supply and demand for a share, it does not always present a true reflection of the investor's trading intentions. For example, market participants can attempt to "game" the order book by placing limit orders which are subsequently canceled or amended, and (as described above) the order book may contain *iceberg orders* (a large order which has been split into several smaller orders in order to disguise the investor's trading intent). The dynamic nature of the order book suggests that an agent-based modeling approach could be used in order to uncover robust trade execution strategies.

3.3 Portfolio Management

In finance, a *portfolio* refers to a basket of financial assets such as stocks, bonds, and cash equivalents. Portfolio management involves the art and science of making decisions about investment mix and policy, matching investments to objectives, asset allocation, and balancing risk and return. An overview of the portfolio management process is illustrated in Fig. 5. For a portfolio manager, the first and foremost part of the investment process is understanding the client's needs, the client's tax status and his or her risk preferences. The next part of the process is the actual construction of the portfolio, which involves asset class allocation and security selection decisions. Asset class allocation refers to the allocation of the portfolio across different asset classes defined broadly as equities, fixed income securities, and real assets (such as real estate, commodities, and other assets). The security selection decision refers to the selection of specific securities under each asset class. The final component of portfolio management is trade execution, the efficient purchase or sale of the relevant assets in the marketplace. An important and open question is how best to measure the performance of the resulting portfolio.

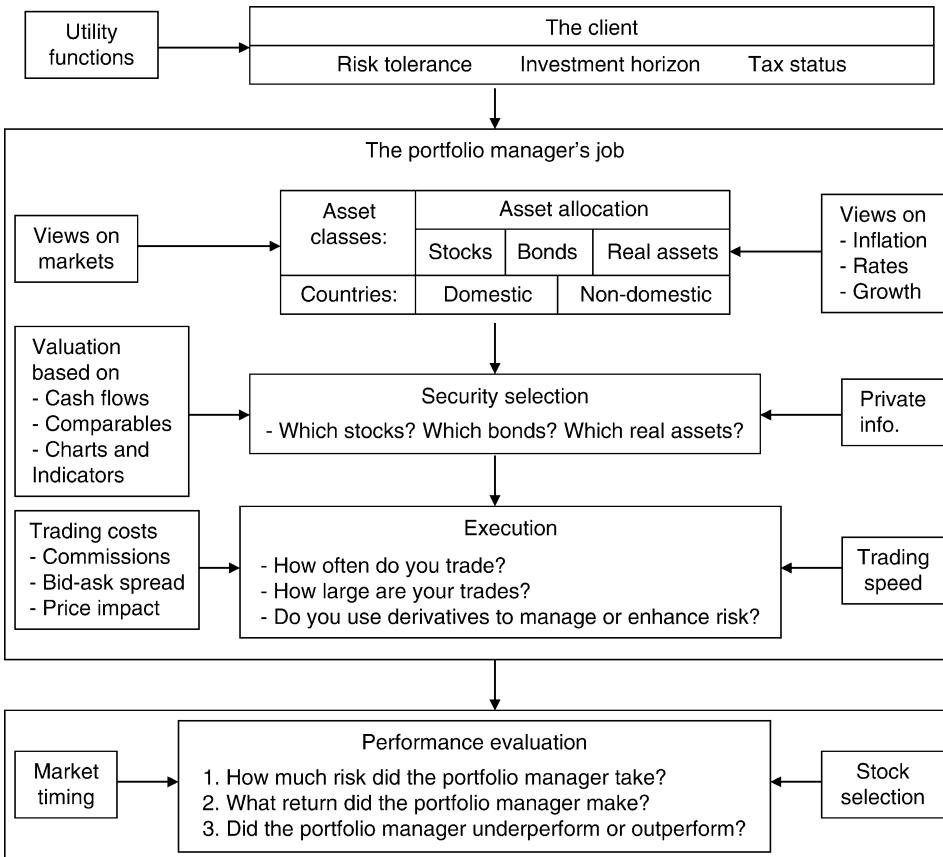
"Optimization is the engineering part of portfolio construction," as mentioned by Fabozzi et al. (2007) in their recent survey for quantitative equity portfolio management: "*Most portfolio construction problems can be cast in an optimization framework, where optimization is applied to obtain the desired optimal risk-return profile.*" Multiple elements of the portfolio management process, such as asset allocation, security selection, index tracking, etc., where optimization is crucial, are amenable to NC methodologies. Applications of NC methods for the purpose of asset allocation and index tracking are illustrated below.

3.3.1 Asset Allocation

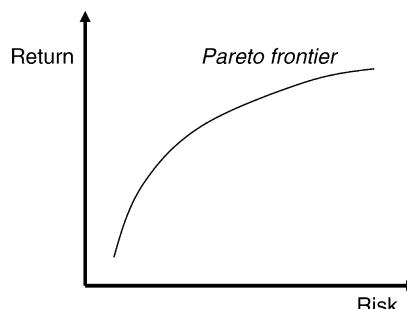
Asset allocation is the selection of a portfolio of investments where each component is an asset class rather than an individual security. The aim of asset allocation is to invest a fixed amount of money in a diverse set of assets so as to maximize return while minimizing a risk measure. The solution to this is a Pareto frontier (usually referred to as the *efficient frontier*) shown in Fig. 6, as for a given level of risk, there should not be a portfolio with a higher rate of return, or for a given level of return, there should not be a portfolio with a lower level of risk.

Fig. 5

An overview of the portfolio management process (from Damodaran (2003)).

**Fig. 6**

Pareto frontier. The points that correspond to the risk-return of the set of portfolios that are Pareto optimal.



Once the frontier is uncovered, the final choice of portfolio is determined by the individual investor's risk preference.

A classical approach used for asset allocation is the Markowitz mean-variance model (Markowitz 1952, 1959). It assumes that investors wish to maximize their return (measured as mean or expected return) and minimize their risk (measured as variance or the standard deviation of their return). This produces the risk-return trade-off. The goal of the Markowitz model is therefore to find an optimal portfolio p of N assets, each with a weighting w_i (in percentage), such that the return E_p is maximized:

$$E_p = \sum_{i=1}^N w_i \cdot \mu_i$$

while minimizing the variance of the return:

$$V_p = \sum_{i=1}^N \sum_{\substack{j=1 \\ i \neq j}}^N w_i \cdot w_j \cdot \sigma_{ij}$$

subject to

$$\begin{aligned} \sum_{j=1}^N w_i &= 1 \\ w_i &\geq 0; i = 1, \dots, N \end{aligned}$$

where σ_{ij} is the covariance of return between asset i and j , the constraints are used to ensure that all the money is invested and all investments are positive (assuming that short selling is not allowed). It produces a multi-objective optimization problem (maximize return, minimize risk) and there are two equivalent formulations which are the dual of each other (first, fix a value of expected return and find the portfolio that minimizes the risk, second, select a level of risk and find the portfolio that maximizes the expected return). Either of the formulations produces a quadratic programming problem and several algorithms exist, which can be applied to uncover good-quality portfolios.

Adding Real-World Constraints

Quadratic programming relies on a number of assumptions, including a single quadratic objective function, linear constraints, and the existence of a positive definite covariance matrix between the asset returns. However, these assumptions are typically breached in the real world. For example, the constraints can include *cardinality constraints* (a limit on the number of assets which can be held in the portfolio), i.e., $\sum_{i=1}^N sign(w_i) = K$, or there may be *threshold limits* on the amount of investment in any single asset, i.e., $w_i \leq m_i$, $i = 1, \dots, N$, where m_i is the threshold amount for asset i . Other constraints may include industry or sector (or *concentration*) holding constraints, round lot constraints, and transaction cost may have both fixed and nonlinear variable cost elements. These constraints can lead to non-convex, non-differential models. In addition, some constraints may be hard and others may be soft. Hence, real-world portfolio selection can present a difficult, high-dimensional, constrained optimization problem, which is beyond the capabilities of traditional optimization methods. In this setting, heuristic approaches such as evolutionary computing methods are of particular interest, because of their ability to find good solutions, even if optimality is not assured.

MOEA and Portfolio Selection

An extensive literature on multi-objective evolutionary algorithms (MOEA) has developed over the past 20 years (see Schaffer 1984; Deb 2001; Coello et al. 2002 for a detailed review). MOEA have an advantage of maintaining a population of solutions and therefore offer the potential to uncover multiple points on the Pareto frontier. A wide range of approaches have been offered to deal with different types of constraints including penalty function approaches, repair mechanisms, the design of appropriate representations, and diversity-generation operators. A stream of literature also exists which has used hybrid evolutionary algorithms or local search techniques for MOEA. Many of these approaches have been applied for portfolio selection.

The earliest papers to apply EAs for portfolio selection include Arone et al. (1993), Loraschi et al. (1995), Shoaf and Foster (1998), and Vedarajan et al. (1997). In the case of Arone et al. (1993) multiple GA populations were used to identify the Pareto frontier. Rather than use the standard Markowitz model, the authors used a downside risk measure. The multi-objective problem was converted into a single objective using a trade-off function, with each population using a different trade-off coefficient and therefore producing a different portion of the Pareto frontier. The formulation of the portfolio problem includes cardinality and buy-in constraints, and a repair mechanism was applied in order to ensure that generated solutions were feasible. The utility of differing crossover operators and differing genotypic representations for the portfolio selection problem was examined by Streichert et al. (2004a, b), and the application of EC hybrids was examined by Subbu et al. (2005) and Streichert et al. (2006). The impact of cardinality constraints was examined in Fieldsend et al. (2004) and Moral-Escudero et al. (2006) (the latter also adopted an EC hybrid approach). A number of other applications including stock ranking, and credit portfolio optimization have been reported in the literature (see Castillo Tapia and Coella (2007) and Schlottmann and Seese (2004) for a review of MOEA applications in finance).

In practical settings, investment managers are concerned with a variety of risk and return measures, not just expected return and its variance. For example, *value at risk* (VaR), the risk that a portfolio could lose a significant amount of its value over a defined time window (more precisely, VaR at level $(1-\alpha)$ is the α -quantile of the loss distribution), has gained importance especially for regulatory purposes. VaR is typically nonlinear and non-convex, making optimization in models, which use this metric difficult. More generally, a portfolio manager may be concerned with more than one risk constraint. A variety of papers have applied MOEA to non-Markowitz risk metrics, including Lipski (2008) which uses a compound risk metric. Hochreiter (2008) introduces an evolutionary stochastic portfolio optimization methodology and illustrates its application using a set of structurally different risk measures, which include, standard deviation, mean-absolute downside semi deviation, value-at-risk, and expected shortfall. Recent work has also seen the application of coevolutionary MOEAs for portfolio optimization (Dreżewski and Siwik 2008).

3.3.2 Index Tracking

There are two common types of portfolio management strategies: passive and active. Active portfolio management consists of picking assets which are expected to outperform the market. In contrast, passive management simply tracks a market index, where the objective is to form a portfolio that replicates the performance of an index as closely as possible.

Passive portfolio management strategies have become very common in recent decades. Just like general portfolio optimization, the construction of an index tracking portfolio is a constrained optimization problem, where the objective is to minimize a measure of *tracking error* (or difference between the return to the portfolio and the return to the index), subject to a variety of constraints, similar to those in the general portfolio optimization problem. The solution space is non-convex suggesting a useful role for population-based, global optimization heuristics.

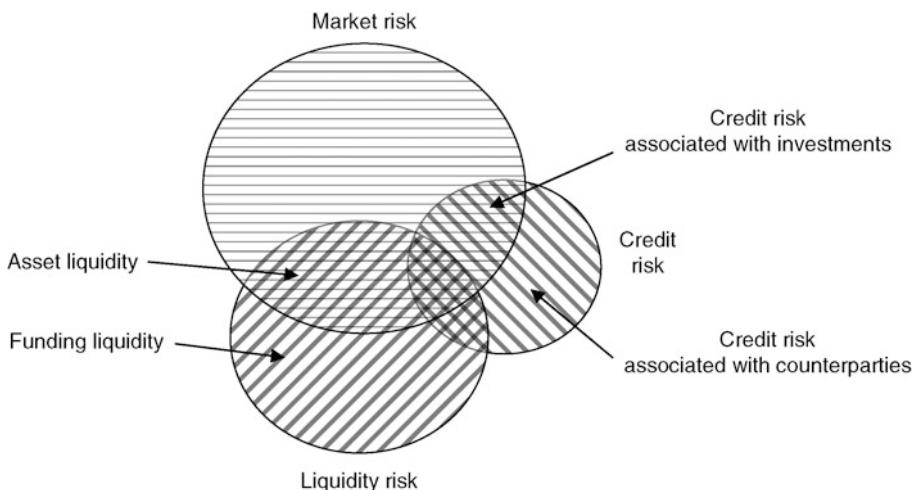
At first glance, the construction of an index tracking portfolio appears trivial, merely requiring the purchase of the same basket of assets that make up the index, using the same weight that each asset has in the index. However, the creation of a perfect replica portfolio is difficult for several reasons including a requirement for frequent portfolio rebalancing (with associated transactions costs), integer constraints on asset purchases, mandate limits on the maximum holding in any individual asset, etc. Another practical issue is that not all assets making up market indices have equal liquidity. Hence, there may be good reason to seek to track the performance of a broad market index using a portfolio which comprises of a subset (rather than all) of the assets making up the index. Another feature of this problem is that the investor's risk attitudes to tracking errors are not symmetric. While investors will not wish to underperform the index, they will not object if the portfolio outperforms the index. EC applications to the index tracking problem include Shapcott (1992), Streichert and Tanaka-Yamawaki (2006), Orito et al. (2007), and Maringer (2008) who also examine the impact of investor loss-aversion preferences on tracking portfolio construction.

3.4 Risk Management

Risk management is a critical aspect of investing. Some of the main types of risks faced by investors are illustrated in Fig. 7 below.

Fig. 7

Risk illustration (from Sound Practices for Hedge Fund Managers (2000)).



3.4.1 Market Risk Computation

Market risk refers to the risk faced by an investor arising from changes in financial market prices. The degree of risk of loss faced by an investor will vary depending on the price volatility of the assets they hold. Not all assets will have the same degree of volatility. There are various techniques to measure market risk (Marrison 2002), applications of NC methodology include the calculation of value-at-risk (VaR) and the sensitivities. The VaR approach measures the worst expected loss under normal market conditions over a specific time interval. The loss distribution is usually assumed to be normal when calculating VaR. Evolutionary algorithms do not need to embed this assumption and can incorporate any preferred loss distribution type (Uludag et al. 2007). NNs have also been used for other market risk measures, such as conditional VaR estimates (Lee et al. 2005), and expected shortfall (Diagne 2002). The sensitivity analysis approach measures how much the portfolio's (or a specific financial instrument's) value is expected to change, if there is a small change in one of the market-risk factors, such as interest rates, equity prices, commodity prices, etc. The sensitivities are the so-called *greeks* for derivatives; *duration* and *convexity* for fixed income products. Keber and Schuster (2001) has used GP to calculate greeks of the American-type put options (see  Sect. 3.5 for an explanation of derivative products).

3.4.2 Credit Risk Assessment

Credit risk assessment is an important component of the lending decision of financial institutions and other commercial companies. Examples of decisions of where credit scoring could be useful include, decisions such as should a loan be extended to a firm or to an individual, should a customer be allowed to purchase goods on credit, or what credit limit should be offered to a customer on their credit card?

Over the past several decades, an extensive literature has amassed on modeling creditworthiness and default risk. Traditional statistical methods including linear discriminant analysis (Altman 1968) and logit (Ohlson 1980) have been applied. In all of these applications, the objective is to develop a model, which will provide a metric of creditworthiness from a series of explanatory variables. Typically, in assessing corporate creditworthiness, explanatory variables can include numbers drawn from the financial statements of the firm, from financial markets, general macroeconomic variables, and nonfinancial, firm-specific information). In assessing personal consumer creditworthiness, explanatory variables can include income, age, occupation, current employment status, past borrowing record, etc. (West 2000; Yobas et al. 2000; Leung et al. 2007). Closely associated streams of academic literature include corporate failure or bankruptcy prediction (Altman 1968), and the reverse engineering of the bond-rating models used by rating firms such as Standard & Poor's (S&P), Moody's, Fitch's, or Dominion Bond Rating Service (Ederington 1985; Dutta and Shekhar 1988; Gentry et al. 1988). Assessments of credit default probability could also form a useful input into a stock- or bond-trading model. A practical problem in constructing risk-assessment models is that there is no clear theoretical framework for guiding the choice of explanatory variables or model form. In the absence of an underlying theory, most published work on credit rating employs a data-inductive modeling approach. This produces a high-dimensional combinatorial

problem, as the modeler is attempting to uncover a good set of explanatory variables and model form.

An illustration of an early credit risk assessment model is provided by Altman's (1968) classic study in which five ratios were combined to produce a linear discriminant classification model for corporate bankruptcy. A Z score was calculated for each company, and this value determined whether the company was classified as bankrupt or solvent:

$$Z = 0.012X_1 + 0.014X_2 + 0.033X_3 + 0.006X_4 + 0.999X_5$$

where

X_1 = working capital to total assets

X_2 = retained earnings to total assets

X_3 = earnings before interest and taxes to total assets

X_4 = market value of equity to book value of total debt

X_5 = sales to total assets

As the range of NC techniques have expanded over the past 20 years, each new technique has been applied to credit scoring and corporate failure prediction. Examples include feedforward NNs (Wilson et al. 1995; Atiya 2001), self-organizing maps (Serrano-Cina 1996; Kiviluoto and Bergius 1998), GAs (Kumar et al. 1997; Varetto 1998), Ant models (Wang et al. 2004; Brabazon and O'Neill 2006), GP and GE (McKee and Lensberg 2002; Brabazon and O'Neill 2006; O'Neill and Ryan 2001; Alfaro-Cid et al. 2008). The domain offers particular potential for evolutionary automatic programing methodologies such as GP or GE as these methods can produce human-readable credit decision rules. This can be important in some countries where lenders can be required to justify decisions not to grant loans. Another advantage of GP and GE is that the rule-evolution process can be seeded using domain knowledge.

Another closely related application is the prediction of bank failure (Lee et al. 2006), with many regulatory authorities using risk models in order to assess which financial institutions require the closest scrutiny. Obviously, for these applications it is important that the regulatory authority can verify the correctness of the underlying prediction model, hence, methodologies which can incorporate expert knowledge and produce interpretable decision rules, such as fuzzy systems and GP, are of particular interest.

Of course, there are other types of risks which need to be quantified in practice, such as liquidity risk and operational risk (arising due to poor or inadequate management control systems or due to human error). However, as yet, there is little literature concerning the application of NC methods in these areas.

3.5 Derivatives Modeling

Derivatives are contracts whose value is derived from the value of the underlying assets, such as equities, interest rates, currencies, market indices, commodities, etc. Two of the best known forms of derivatives are futures and options. A futures contract is an agreement to buy or sell goods, currency or securities on an agreed future date and for a price fixed in advance. An option is a financial instrument that simply gives the holder (buyer) the right, but not the obligation, to buy (a call option), or sell (a put option), a specified underlying asset at

a pre-agreed price on or before a given date. A European style option refers to an option that may only be exercised on expiration; while an American style option can be exercised on any trading day on or before expiration.

The key issue for investors wishing to trade in derivatives is the determination of the fair price for the derivative. For some standard derivatives (based on specific assumptions such as continuous time finance theory), closed-form pricing equations have been determined (e.g., the Black–Scholes model (Black and Scholes 1973; Merton 1973) for pricing European options, the Cox et al. (1979) binomial model, etc.). The traditional approach to pricing a derivative is (Noe and Wang 2002):

- Specify a stochastic process for the underlying asset(s)
- Derive the pricing equation for the derivative (using a no-arbitrage argument) and
- Price the derivative by solving the pricing equation

Of course, this approach can be difficult to implement, as the relevant stochastic process may be imperfectly understood, and the pricing equation may be too difficult to solve analytically. In the latter case, there is scope to use tools such as Monte Carlo (MC) simulation to estimate the expected payoff and the associated payoff risk for the derivative. In valuing a complex derivative using MC, the typical approach is to randomly generate a set of independent price paths for each security underpinning the derivative, then compute the present value of the payoff to the derivative under each set of these price paths. The simulation process is repeated multiple times and the distribution of the payoffs is considered to characterize the derivative. An example of this approach is illustrated in Kim and Byun (2005). A critical issue in applying an MC approach is the correct design of the theoretical pricing model.

There have been two main avenues of application of NC methods in pricing financial derivatives, namely,

- Model calibration and
- Model induction

In model calibration, the objective is to estimate the parameters of (or “calibrate”) a theoretical pricing model. The parameters are estimated by fitting the model to the relevant return time series. Typically, the pricing model will have a complex, nonlinear structure with multiple parameters. Hence, global search heuristics such as the genetic algorithm can have utility in uncovering a high-quality set of parameters. Examples of the use of NC algorithms for model calibration include Dang et al. (2008) and Fan et al. (2007).

In model induction, NC approaches such as GP would have particular utility when little is known about the underlying asset-pricing dynamics as both the structure and the parameters of the pricing model are estimated directly from the data, thereby extracting the pricing model implicitly. Even where theory does exist, model induction methodologies allow one to investigate whether other plausible theories may exist to explain observed prices. Applications of such methods include NNs (Malliaris and Salchenberger 1993; Hutchinson et al. 1994; White 1998), self-organizing, fuzzy NNs (Tung and Quek 2005), or GP (Chen et al. 1999; Chidambaran et al. 1998; Keber 2000; Chidambaran 2003; Yin et al. 2007) to recover a proxy for the price-generating model directly from the data.

An example of using GP to generate an option pricing model is illustrated below. One advantage noted by Chidambaran (2003) is that the GP process can be seeded with the Black–Scholes equation, with the final resulting model being an adaptation of the Black–Scholes equation for conditions which violate its underlying assumptions. For example, in the

Black–Scholes setting for a non-dividend-paying European call option, there are five factors that affect the price of the option (assuming no dividends):

1. S_0 – the underlying asset price
2. K – the exercise price of the option
3. T – the time to maturity (of the option)
4. r – the risk free rate of return (of the underlying asset)
5. σ – the expected volatility of the asset price

Items 1 and 2 can be combined to give ($S_0 - K$) or a measure of the *moneyness* of the option. An option is said to be “in the money” when this value is greater than zero, and “out of the money” when it is less than zero. In developing a pricing model for options from these factors, the Black–Scholes model embeds several critical assumptions. It is assumed that the stock price undergoes a diffusion process that is lognormally distributed, with an instantaneous drift and volatility given by μ and σ , respectively. The volatility σ and the risk-free rate r are also assumed to be constant during option’s life. This implies that:

$$\ln \frac{S_T}{S_0} \sim N\left(\left(\mu - \frac{\sigma^2}{2}\right)T, \sigma\sqrt{T}\right)$$

and in turn this leads to:

$$S_T = S_0 e^{\eta T}, \eta \sim N\left(\left(\mu - \frac{\sigma^2}{2}\right), \frac{\sigma}{\sqrt{T}}\right)$$

where μ is the instantaneous expected return on the stock, σ is the instantaneous volatility of stock price return, S_T is the stock price at a future time T , S_0 is the stock price at time zero, $N(m, s)$ denotes the normal density function with mean m and standard deviation s and η is defined as the continuously compounded rate of return per annum realized between time zero and T . The Black–Scholes formula for the price at time zero of an European call option on a non-dividend-paying stock is therefore:

$$C_0 = S_0 N(d_1) - K e^{-rT} N(d_2)$$

where

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

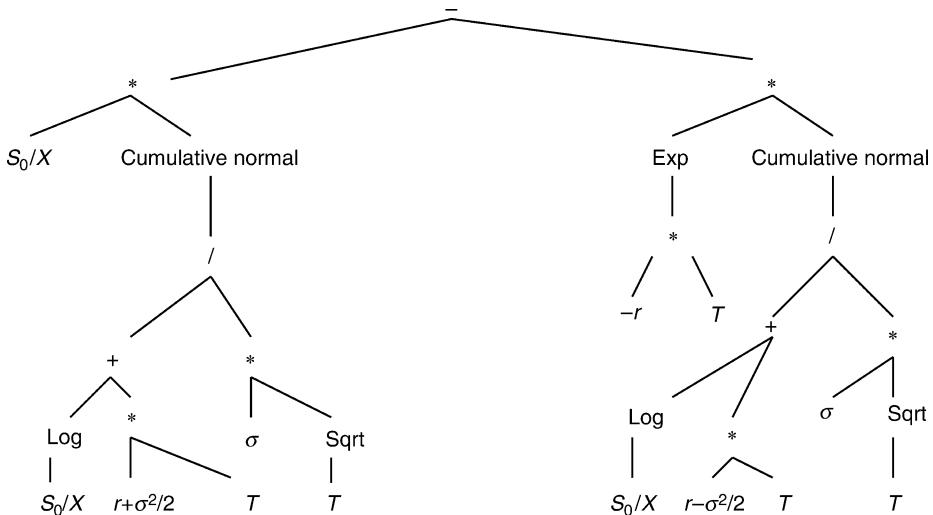
$N(\chi)$ is the cumulative probability distribution function for a standardized normal distribution, C_0 is the price of the European call option at time 0.

Of course, assuming that the Black–Scholes model did precisely value options, model induction techniques such as neural networks or GP could be used to recover the structure of the option pricing model directly from a historical time series of option prices (C_0) and the five factors that influence option prices.  [Figure 8](#) illustrates a tree representation of the Black–Scholes Model which could (potentially) be recovered by GP.

In reality, some of the key assumptions in the Black–Scholes model do not hold in real-world option markets, and hence the model does not explain observed option prices correctly. For example, prices can experience discontinuous jumps, the distribution of price changes has fatter tails than those implied by a lognormal distribution, and asset price volatility changes over time. (There is a long line of literature which examines alternative (non-normal) stock

Fig. 8

Stylized illustration of a tree representation of the Black–Scholes model (not all sub-trees are shown).



return models including Poisson jump-diffusion return processes and GARCH processes. However, closed-form solutions for the option price cannot be obtained for all these models.) The latter issue can be easily seen if market prices are substituted into the Black–Scholes model, in order to calculate the “implied volatility” for that option. If the assumptions underlying the Black–Scholes option pricing model were correct, the implied volatilities for options on the same underlying asset would be constant for different strike prices and maturities. However, in practice, the Black–Scholes implied volatilities are varying over strike price and maturity. Keber (2002) has used GP to generate formulae for determining the implied volatility based on American put options.

3.6 Agent-Based Market Modeling

The essence of agent-based modeling (ABM) lies in the notion of autonomous agents whose behavior evolves endogenously leading to complex, emergent, system dynamics, which are not predictable from the properties of the individual agents. ABM is an exciting new tool for exploring behavior in financial markets that are far from traditional notions of equilibrium, and where agents exhibit behavior that is less than fully rational at times. Financial markets are particularly appealing applications for agent-based methods especially considering the following: issues of price and information aggregation tend to be sharper in financial settings where agent objectives tend to be clearer; financial markets are rich in data sets (such as price and volume data) at many different frequencies, that can be used for testing and calibrating agent-based models; financial markets are well organized, centralized, and trade homogeneous products in a generally efficient fashion relative to markets for other goods and services; there are continuing developments in the area of experimental financial markets, which give carefully controlled environments that can be compared with agent-based experiments; the

key debates in finance about market efficiency and rationality are still unresolved; many puzzles of the financial time series (such as the volatility persistence) are still not well understood. In designing ABMs of financial markets, modelers face a daunting list of design choices which can critically impact on the system's behavior. Important design questions include:

- Representation and structure of the actual trading agents. Agents can vary from simple budget constrained *zero intelligence agents* (refers to a type of trader that randomly makes price bids (offers to buy) and/or price asks (offers to sell) subject only to a budget constraint), as in Gode and Sunder (1993), to sophisticated learning agents as in Chen and Yeh (2001).
- The actual mechanism that governs the trading of assets. Ways of designing this include assuming a simple price response to excess demand, building the market, such that a kind of local equilibrium price can be found easily, or explicitly modeling the dynamics of trading to mimic the continuous trading of real-world markets.
- Types of securities to be incorporated into the agent-based market model, where typically simple securities (such as stocks) are considered.

In designing agent-based models (ABMs) of financial markets, NC methods can be used to model the information processing and storage by agents, the process of adaptive learning by agents, or to model the trading mechanism. One example of the use of ABM to simulate a financial market is provided by LeBaron (2002) in building the well-known Santa Fe Artificial Stock Market. This model simulates price generation and trading in a market made up of artificial adaptive agents. This is a prototype example of a complex system and is thought to illustrate the benefits of simulation modeling. The Santa Fe Artificial Stock Market consists of a central computational market and a number of artificially intelligent agents. The agents choose between investing in a stock and leaving their money in the bank, which pays a fixed interest rate. The stock pays a stochastic dividend and has a price which fluctuates according to agent demand. The agents make their investment decisions by attempting to forecast the future return on the stock, using GA to generate, test, and evolve predictive rules. Other applications of ABM include the simulation of a foreign exchange market (Izumi 1999), the modeling of an artificial stock option market (Ecca et al. 2008) and the modeling of an artificial payment card market (Alexandrova-Kabadjova 2008).

A key output from the ABM literature on financial markets is that it illustrates that complex market behavior can arise from the interaction of quite simple agents. Carefully constructed, ABM can help increase one's understanding of market processes and can potentially provide insights for policy makers and regulators, where, unlike laboratory sciences, one cannot rerun a real market under different regulations "to see what would happen." Of course, issues of model validation are important in all ABM applications, including those in financial markets.

4 The Future

Though a plethora of academic literature on NC applications in finance exists, it is notable that many papers have been concerned with proof of concept rather than robust, industry-strength, applications. While NC methods offer potential in multiple areas in finance, the maturing of their application requires future work focusing on complex real-world problems. This will require the construction of multidisciplinary research teams, drawing academic

expertise as necessary from finance, computer science, mathematics, biology, etc., and combining this with industrial collaborators. For example, quality work on trading systems requires deep knowledge of market microstructure, the regulatory environment, available financial instruments, and the technology available to traders. Indicated below are some promising future directions for research at the nexus of natural computing and finance.

- **Forecasting:** While forecasting models applying NC methods can typically be constructed to fit historical data fairly well, a common finding is that the quality of the out-of-sample forecasts degrades over time. Hence, one can expect to see increased use of sophisticated methods for preprocessing the raw time-series inputs, and for the adaptation of the resulting models in response to changing environmental conditions. Another area of growing interest is the incorporation of information from text mining (e.g., from the financial press) into forecasting models.
- **Algorithmic Trading:** While many published works have focused on the development of simplified trading systems, successful real-world applications have often focused on the support of specific elements of the investment process. In the medium term, one can expect to see a notable increase in the rigor of the published works in this area as computer scientists form teams with finance academics and practitioners, incorporating realistic models of market microstructure. The area of trade execution has seen relatively little published application of NC methodologies, despite its real-world significance. Model induction tools such as GP offer interesting potential here, as do agent-based modeling approaches. The latter could be used to uncover robust trade execution strategies.
- **Portfolio Optimization:** There has already been an extensive application of NC methods for portfolio optimization, but there is still a need for further systematic investigation of portfolio constraints for special sub-application areas including hedge funds, pension funds, and insurance. The extension of dynamic portfolio optimization needs further development, which may involve the simulation or forecasting of extreme market situations, and the solution of multistage constraint optimization problems.
- **Risk Management:** Recent events on the financial markets have underscored the importance of risk management, and the weakness of existing theoretical models in this area. It is interesting to note that applications of NC methods in risk management have not attracted as much attention as might be expected in the literature and this remains an open research area. For example, NC methods could be applied to assist in the development of enterprise-wide risk management systems, and to improve the flexibility and efficiency of large-scale multistage *asset liability management* (ALM) models.
- **Derivatives Modeling:** In spite of the vast array of derivative products available, and the weakness of financial theory, once one moves beyond vanilla products there have only been a relatively limited number of applications of NC for model calibration or induction in this area. Possibilities also exist to hybridize NC methods with traditional numerical methods, and to develop dynamic derivative pricing models.
- **Agent-Based Market Modeling:** The field of ABM is attracting significant attention with the increasing questioning of agent homogeneity which underlies classical financial economics. ABM allows one to examine the effect of the differing forms of market structure on market behavior. Doubtless, the next few years will see increased focus on this, given the failures of market regulation during the recent financial crisis. The coevolutionary element of markets lends itself well to NC approaches in terms of modeling of agent behavior and strategy adaptation.

A practical issue that arises in the application of NC to finance is that the underlying algorithms are themselves undergoing a process of maturation. Recent years have seen extensive research in order to extend canonical NC algorithms into high-dimensional environments (enhancing algorithmic scalability), to develop efficient algorithms for constrained optimization, and to develop practical application of the algorithms in dynamic problem environments. Meanwhile, we have also seen developments in computer hardware, and in our ability to implement parallel versions of NC algorithms (e.g., using *graphics processing unit* (GPU) implementations). These two strands of development are creating an ever more powerful toolbox of NC algorithms for financial modelers.

Acknowledgments

This chapter has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.

References

- Aiken M (2000) Forecasting the United States gross domestic product with a neural network. *J Int Inform Manag* 9(1):11–21
- Aiken M, Bsat M (1999) Forecasting market trends with neural networks. *Inform Syst Manag* 16(4):42–48
- Alexandrova-Kabadjova B, Tsang E, Krause A (2008) Evolutionary learning of the optimal pricing strategy in an artificial payment card market. In: Brabazon A, O'Neill M (eds) *Natural computing in computational finance*. Springer, Berlin, pp 233–251
- Alfaro-Cid E, Cuesta-Canada A, Sharman K, Esparcia-Alcazar A (2008) Strong typing, variable reduction and bloat control for solving the bankruptcy prediction problem using genetic programming. In: Brabazon A, O'Neill M (eds) *Natural computing in computational finance*. Springer, Berlin, pp 161–186
- Allen F, Karjalainen R (1999) Using genetic algorithms to find technical trading rules. *J Financ Econ* 51:245–271
- Altman E (1968) Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *J Financ* 23:589–609
- Armano G, Marchesi M, Murru A (2005) A hybrid genetic-neural architecture for stock indexes forecasting. *Inform Sci* 170:3–33
- Arone S, Loraschi A, Tettamanzi A (1993) A genetic approach to portfolio selection. *Neural Netw World Int J Neural Mass-Parallel Comput Inform Syst* 3:597–604
- Atiya A (2001) Bankruptcy prediction for credit risk using neural networks: a survey and new results. *IEEE Trans Neural Netw* 12(4):929–935
- Bauer R (1994) *Genetic algorithms and investment strategies*. Wiley, New York
- Belkaoui A (1978) Financial ratios as predictors of Canadian takeovers. *J Bus Financ Account* 5 (5):93–108
- Black F, Scholes M (1973) The pricing of options and corporate liabilities. *J Polit Econ* 81:637–659
- Brabazon A, O'Neill M (2006) *Biologically inspired algorithms for financial modelling*. Springer, Berlin
- Brabazon A, O'Neill M (eds) (2008) *Natural computing in computational finance*. Springer, Berlin
- Brabazon A, O'Neill M (eds) (2009) *Natural computing in computational finance*, vol II. Springer, Berlin
- Castillo Tapia MG, Coello Coello C (2007) Applications of multi-objective evolutionary algorithms in economics and finance: a survey. In: CEC 2007: Proceedings of the IEEE international conference on evolutionary computation, Singapore, September 2007. IEEE Press, Piscataway, NJ, pp 532–539
- de Castro LN (2007) Fundamentals of natural computing: an overview. *Phys Life Rev* 4(1):1–36
- Cao LJ, Tay FEH (2003a) Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Trans Neural Netw* 14(6):1506–1518
- Cao LJ, Tay FEH (2003b) A hybrid neurogenetic approach for stock forecasting. *IEEE Trans Neural Netw* 18(3):851–864
- Chen S-H (eds) (2002) *Genetic algorithms and genetic programming in computational finance*. Kluwer Academic, Norwell, MA
- Chen S-H, Kuo T-W (2002) Evolutionary computation in economics and finance: a bibliography. In: Chen S-H (ed) *Evolutionary computation in economics and finance*. Physica-Verlag, Heidelberg
- Chen SH, Yeh CH (2001) Evolving traders and the business school with genetic programming: a new

- architecture of the agent-based stock market. *J Econ Dyn Control* 25(3–4):363–393
- Chen S-H, Lee W-C, Yeh C-H (1999) Hedging derivative securities with genetic programming. *Int J Intell Syst Account Financ Manag* 8(4):237–251
- Chidambaran N (2003) Genetic programming with Monte Carlo simulation for option pricing. In: Proceedings of the 2003 IEEE winter simulation conference, New Orleans, LA, December 2003. IEEE Press, Piscataway, NJ, pp 285–292
- Chidambaran N, Lee C, Trigueros J (1998) Adapting Black-Scholes to a non-Black-Scholes environment via genetic programming. In: CIFEr: Proceedings of the IEEE/IAFE/INFORMS 1998 conference on computational intelligence for financial engineering, New York, March 1998. IEEE Press, Piscataway, NJ, pp 197–211
- Coello C, Van Veldhuizen D, Lamont G (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer Academic, New York
- da Costa Pereira C, Tettamanzi A (2008) Fuzzy-evolutionary modeling for single-position day trading. In: Brabazon A, O'Neill M (eds) Natural computing in computational finance. Springer, Berlin, pp 131–159
- Cox J, Ross S, Rubinstein M (1979) Option pricing: a simplified approach. *J Financ Econ* 7:229–264
- Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, New York
- Damodaran A (2003) Investment philosophies: successful investment philosophies and the greatest investors who made them work. Wiley, Hoboken, NJ, p 8
- Dang J, Brabazon A, O'Neill M, Edelman D (2008) Option model calibration using a bacterial foraging optimisation algorithm. In: EvoFin 2008: Proceedings of the 2nd European workshop on evolutionary computation in finance and economics, Napoli, Italy, March 2008. Lecture notes in computer science, vol 4974. Springer, New York, pp 133–143
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK
- Dempsey I, O'Neill M, Brabazon A (2009) Foundations in grammatical evolution for dynamic environments. Springer, Berlin
- Dempster M, Jones C (2001) A real-time adaptive trading system using genetic programming. *Quant Financ* 1:397–413
- Diagne M (2002) Financial risk management and portfolio optimization using neural networks and extreme value theory. Ph.D. thesis, University of Kaiserslautern
- Dopuch N, Holthausen RW, Leftwich RW (1987) Predicting audit qualifications with financial and market variables. *Account Rev* LXII(3):431–454
- Dreżewski R, Siwik L (2008) Co-evolutionary multi-agent system for portfolio optimization. In: Brabazon A, O'Neill M (eds) Natural computing in computational finance. Springer, Berlin, pp 271–299
- Dutta S, Shekhar S (1988) Bond rating: a non-conservative application of neural networks. In: Proceedings of IEEE international conference on neural networks, II, San Diego, CA, July 1988. IEEE Press, Piscataway, NJ, pp 443–450
- Ecca S, Marchesi M, Setzu A (2008) Modeling and simulation of an artificial stock option market. *Comput Econ* 32(1):37–53
- Edelman D (2007) Adapting support vector machine methods for horse race odds prediction. *Ann OR* 151(1):325–336
- Edelman D, Davy P (2004) Adaptive technical analysis in the financial markets using machine learning: a statistical view. In: Fulcher J, Jain LC (eds) Applied intelligent systems new directions series studies in fuzziness and soft computing. Springer, Berlin, pp 1–15
- Ederington H (1985) Classification models and bond ratings. *Financ Rev* 20(4):237–262
- Fabozzi FJ et al. (2007) Trends in quantitative equity management: survey results. *Quant Financ* 7(2):115–122
- Fan K, Brabazon A, O'Sullivan C, O'Neill M (2007) Quantum-inspired evolutionary algorithms for calibration of the VG option pricing model. In: EvoFin 2007: Proceedings of the 1st European workshop on evolutionary computation in finance and economics, Valencia, Spain, April 2007. Lecture notes in computer science, vol 4447. Springer, Berlin, pp 186–195
- Fieldsend J, Matatko J, Peng M (2004) Cardinality constrained portfolio optimisation. In: IDEAL 2004: Intelligent data engineering and automated learning, Exeter, UK, August 2004. Lecture notes in computer science, vol 3177. Springer, New York, pp 788–793
- Fyfe C, Marney J, Tarbert H (1999) Technical analysis versus market efficiency – a genetic programming approach. *Appl Financ Econ* 9(2):183–191
- Gentry J, Whitford D, Newbold P (1988) Predicting industrial bond ratings with a probit model and funds flow components. *Financ Rev* 23(3):269–286
- Ghanda A, Michalewicz Z, Schmidt M, Tö D-T, Zurbrugg R (2008) Computational intelligence for evolving trading rules. *IEEE Trans Evol Comput* 13(1):71–86
- Gode DK, Sunder S (1993) Allocative efficiency of markets with zero intelligence traders. *J Polit Economy* 101:119–137
- Harris R, Stewart J, Guilkey D, Carleton W (1982) Characteristics of acquired firms: fixed and random coefficients probit analyses. *South Econ J* 49(1):164–184

- Hickey R, Little E, Brabazon A (2006) Identifying merger and takeover targets using a self-organising map. In: ICAI '06: Proceedings of the 2006 international conference on artificial intelligence, Las Vegas, NV, June 2006. CSEA Press
- Hochreiter R (2008) Evolutionary stochastic portfolio optimization. In: Brabazon A, O'Neill M (eds) Natural computing in computational finance. Springer, Berlin, pp 67–87
- Hutchinson J, Lo A, Poggio T (1994) A non-parametric approach to pricing and hedging derivative securities via learning networks. *J Financ* 49:851–889
- Izumi K (1999) An artificial market model of a foreign exchange market. Ph.D. Dissertation, Tokyo University
- Kastra I, Boyd M (1996) Designing a neural network for forecasting financial and economic time series. *Neurocomputing* 10:215–236
- Kari L, Rozenberg G (2008) The many facets of natural computing. *Commun ACM* 51(10):72–83
- Keber C (2000) Option valuation with the genetic programming approach. In: Proceedings of the sixth international conference. MIT Press, Cambridge, MA, pp 689–703
- Keber C (2002) Evolutionary computation in option pricing: determining implied volatilities based on American put options. In: Chen S-H (eds) Evolutionary computation in economics and finance. Physica-Verlag, New York, pp 399–415
- Keber C, Schuster M (2001) Evolutionary computation and the Vega risk of American put options. *IEEE Trans Neural Netw* 12(4):704–715
- Kim J, Byun S (2005) A parallel Monte Carlo simulation on cluster systems for financial derivatives pricing. In: CEC 2005: Proceedings of the IEEE international conference on evolutionary computation, Edinburgh, UK, September 2005. IEEE Press Piscataway, NJ
- Kiviluoto K, Bergius P (1998) Maps for analysing failures of small and medium-sized enterprises. In: Deboeck G, Kohonen T (eds) Visual explorations in finance with self-organizing maps. Springer-Verlag, Berlin, pp 59–71
- Kumar N, Krovi R, Rajagopalan B (1997) Financial decision support with hybrid genetic and neural based modeling tools. *Eur J Oper Res* 103(2):339–349
- Kwon Y-K, Moon B-R (2004) Evolutionary ensemble for stock prediction. In: GECCO 2004: Proceedings of the genetic and evolutionary computation conference, Seattle, WA, June 2004. Lecture notes in computer science, vol 3103. Springer, New York, pp 1120–1113
- Larkin F, Ryan C (2008) Good news: using news feeds with genetic programming to predict stock prices. In: O'Neill M et al. (eds) EuroGP 2008: Proceedings of the 11th European conference on genetic programming, Napoli, Italy, March 2008. Lecture notes in computer science, vol 4971. Springer, Berlin, pp 49–60
- LeBaron B (2002) Building the Santa Fe artificial stock market. Working Paper, Brandeis University, June 2002
- Lee H, Lee J, Yoon Y, Kim S (2005) Coherent risk measure using feedforward neural networks. In: ISNN 2005, Chongqing, China, May–June 2005. Lecture notes in computer science, vol 3497. Springer, Berlin, pp 904–909
- Lee C, Quek C, Maskell D (2006) A brain-inspired fuzzy neuro-predictor for bank failure analysis. In: CEC 2006: Proceedings of the IEEE international conference on evolutionary computation, Vancouver, BC, Canada, July 2006. IEEE Press, Piscataway, NJ, pp 7927–7934
- Leung K, Cheong F, Cheong C (2007) Consumer credit scoring using an artificial immune system algorithm. In: CEC 2007: Proceedings of the IEEE international conference on evolutionary computation, Singapore, September 2007. IEEE Press, Piscataway, NJ, pp 3377–3384
- Li J (2001) FGP: a genetic programming based tool for financial forecasting. Ph.D. Thesis, University of Essex
- Lim M, Coggins R (2005) Optimal trade execution: an evolutionary approach. In: CEC 2005: Proceedings of the IEEE international conference on evolutionary computation, Edinburgh, UK, September 2005. IEEE Press, Piscataway, NJ, pp 1045–1052
- Lipinski P (2008) Evolutionary strategies for building risk-optimal portfolios. In: Brabazon A, O'Neill M (eds) Natural computing in computational finance. Springer, Berlin, pp 53–65
- Loraschi A, Tettamanzi A, Tomassini M, Verda P (1995) Distributed genetic algorithms with an application to portfolio selection problems. In: Pearson D, Steele N, Albrecht R (eds) Artificial neural networks and genetic algorithms. Springer, Berlin, pp 384–387
- Malliaris M, Salchenberger L (1993) A neural network model for estimating option prices. *Appl Intell* 3(3):193–206
- Maringer D (2008) Constrained index tracking under loss aversion using differential evolution. In: Brabazon A, O'Neill M (eds) Natural computing in computational finance. Springer, Berlin, pp 7–24
- Markose S, Tsang E, Er H (2002) Evolutionary decision trees for stock index options and futures arbitrage. In: Chen S-H (eds) Genetic algorithms and genetic programming in computational finance. Kluwer Academic, Norwell, MA, pp 281–308
- Markowitz H (1952) Portfolio selection. *J Financ* 1(7):77–91
- Markowitz H (1959) Portfolio selection: efficient diversification of investments. Wiley, New York
- Marrison C (2002) The fundamentals of risk measurement. McGraw-Hill, New York
- McKee T, Lensberg T (2002) Genetic programming and rough sets: a hybrid approach to bankruptcy classification. *Eur J Oper Res* 138:436–451

- Merton R (1973) Rational theory of option pricing. *Bell J Econ Manag Sci* 4:141–183
- Moral-Escudero R, Ruiz-Torrbiano R, Suarez A (2006) Selection of optimal investment portfolios with cardinality constraints. In: CEC 2006: Proceedings of the IEEE international conference on evolutionary computation, Vancouver, BC, Canada, July 2006. IEEE Press, Piscataway, NJ, pp 8551–8557
- Mutchler JF (1985) A multivariate analysis of the auditor's going-concern opinion decision. *J Account Res* 23(2):668–682
- Neely C, Weller P (2002) Using a genetic program to predict exchange rate volatility. In: Chen S-H (eds) Genetic algorithms and genetic programming in computational finance. Kluwer Academic, Norwell, MA, pp 263–278
- Neely C, Weller P, Dittmar R (1997) Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *J Financ Quant Anal* 32(4):405–428
- Noe T, Wang J (2002) The self-evolving logic of financial claim prices. In: Chen S-H (eds) Genetic algorithms and genetic programming in computational finance. Kluwer, Norwell, MA, pp 249–262
- Ohlson J (1980) Financial ratios and the probabilistic prediction of bankruptcy. *J Account Res* 18 (1):109–131
- O'Neill M, Ryan C (2001) Grammatical evolution. *IEEE Trans Evol Comput* 5(4):349–358
- O'Neill M, Ryan C (2003) Grammatical evolution: evolutionary automatic programming in an arbitrary language. Kluwer, Boston, MA
- Orito Y, Takeda M, Iimura K, Yamazaki G (2007) Evaluating the efficiency if index fund selections over the fund's future period. In: Chen S-H, Wang P, Kuo T-W (eds) Computational intelligence in economics and finance. Springer, New York, pp 157–168
- Palepu K (1986) Predicting takeover targets: a methodological and empirical analysis. *J Account Econ* 8:3–25
- Pavlidis N, Pavlidis E, Epitropakis M, Plagianakos V, Vrahatis M (2007) Computational intelligence algorithms for risk-adjusted trading strategies. In: CEC 2007: Proceedings of the 2007 congress on evolutionary computation, Singapore, September 2007. IEEE Press, Piscataway, NJ, pp 540–547
- Quintana D, Luque C, Isasi P (2005) Evolutionary rule-based system for IPO underpricing prediction. In: GECCO 2005: Proceedings of the genetic and evolutionary computation conference, Washington DC, June 2005. ACM, New York, pp 983–989
- Rege U (1984) Accounting ratios to locate take-over targets. *J Bus Financ Account* 11(3):301–311
- Schaffer J (1984) Multiple objective optimization with vector evaluated genetic algorithms. Ph.D. Thesis, Vanderbilt University
- Schlottmann F, Seese D (2004) Financial applications of multi-objective evolutionary algorithms: recent developments and future research directions. In: Coello Coello C, Lamont G (eds) Applications of multi-objective evolutionary algorithms. World Scientific, Singapore, pp 627–652
- Serrano-Cina C (1996) Self organizing neural networks for financial diagnosis. *Decis Support Syst* 17(3): 227–238
- Shapcott J (1992) Index tracking: genetic algorithms for investment portfolio selection. Technical report, EPCC-SS92-24, Edinburgh Parallel Computing Centre
- Shoaf J, Foster J (1998) The efficient set GA for stock portfolios. In: CEC 98: Proceedings of the IEEE international conference on evolutionary computation, Anchorage, AK, May 1998. IEEE Press, Piscataway, NJ, pp 354–359
- Sound Practices for Hedge Fund Managers (2000) the Managed Funds Association (MFA), New York, p 16
- Stephens CR, Sukumar R (2006) An introduction to datamining. In: Grover R, Vriens M (eds) The handbook of market research do's and don'ts. Sage, Thousand Oaks, CA
- Streichert F, Tanaka-Yamawaki M (2006) The effect of local search on the constrained portfolio selection problem. In: CEC 2006: Proceedings of the IEEE international conference on evolutionary computation, Vancouver, BC, Canada, July 2006. IEEE Press Piscataway, NJ, pp 8537–8543
- Streichert F, Ulmer H, Zell A (2004a) Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem. In: CEC 2004: Proceedings of the IEEE international conference on evolutionary computation, Portland, OR, June 2004. IEEE Press, Piscataway, NJ, pp 932–939
- Streichert F, Ulmer H, Zell A (2004b) Comparing discrete and continuous genotypes on the constrained portfolio selection problem. In: GECCO 2004: Proceedings of the genetic and evolutionary computation conference, Seattle, WA, June 2004. Springer, Berlin, pp 1239–1250
- Subbu R, Bonissone P, Eklund N, Bollapragada S, Chalermkraivuth K (2005) Multiobjective financial portfolio design: a hybrid evolutionary approach. In: CEC 2005: Proceedings of the IEEE international conference on evolutionary computation, Edinburgh, UK, September 2005. IEEE Press, Piscataway, NJ, pp 2429–2436
- Thomas J, Sycara K (2002) GP and the predictive power of internet message traffic. In: Chen S-H (eds) Genetic algorithms and genetic programming in computational finance. Kluwer Academic, Norwell, MA, pp 81–102
- Thompson D, Thompson S, Brabazon A (2007) Predicting going concern audit qualification using neural networks. In: ICAI'07: Proceedings of the 2007

- international conference on artificial intelligence. CSEA Press, Las Vegas, NV, June 2007
- Trigueros J (1999) Extracting earnings information from financial statements via genetic algorithms. In: Proceedings of the 1999 IEEE international conference on computational intelligence for financial engineering, New York, March 1999. IEEE Press, Piscataway, NJ, pp 281–296
- Trippi RR, Turban E (1993) Neural networks in finance and investing. McGraw-Hill, New York
- Tsang E, Li J (2002) EDDIE for financial forecasting. In: Chen S-H (ed) Genetic algorithms and genetic programming in computational finance. Kluwer Academic, Norwell, MA, pp 161–174
- Tsang E, Martinez-Jaramillo S (2004) Computational finance. IEEE Computational Intelligence Society Newsletter, August 8–13, 2004
- Tung W, Quek C (2005) GenSoOPATS: a brain-inspired dynamically evolving option pricing model and arbitrage system. In: CEC 2005: Proceedings of the IEEE international conference on evolutionary computation, Edinburgh, UK, September 2005. IEEE Press, Piscataway, NJ, pp 1722–1729
- Uludag G et al. (2007) Comparison of evolutionary techniques for value-at-risk calculation. In: Proceedings of EvoWorkshops 2007, Valencia, Spain, April 2007. Lecture notes in computer science, vol 4448, pp 218–227
- Vapnik V (1995) The nature of statistical learning theory. Springer-Verlag, New York
- Varetto F (1998) Genetic algorithms in the analysis of insolvency risk. *J Bank Financ* 22(10):1421–1439
- Vedarajan G, Chan L, Goldberg D (1997) Investment portfolio optimization using genetic algorithms. In: Koza J (ed) Late breaking papers at the genetic programming 1997, Stanford, CA, July 1997, pp 256–263
- Wang C, Zhao X, Kang Li (2004) Business failure prediction using modified ants algorithm. In: Chen S-H, Wang P, Kuo T-W (eds) Computational intelligence in economics and finance. Springer, Berlin
- West D (2000) Neural network credit scoring models. *Comput Oper Res* 27:1131–1152
- White A (1998) A genetic adaptive neural network approach to pricing options: a simulation analysis. *J Comput Intell Financ* 6(2):13–23
- Wilson N, Chong K, Peel M (1995) Neural network simulation and the prediction of corporate outcomes: some empirical findings. *Int J Econ Bus* 2(1):31–50
- Wong BK, Selvi Y (1998) Neural network applications in finance: a review and analysis of literature. *Inform Manag* 34(3):129–140
- Wong B, Lai V, Lam J (2000) A bibliography of neural network business applications research: 1994–1998. *Comput Oper Res* 27:1045–1076
- Yin Z, Brabazon A, O'Sullivan C (2007) Adaptive genetic programming for option pricing. In: GECCO 2007: Proceedings of the genetic and evolutionary computation conference, London, England, July 2007. ACM, New York, pp 2588–2594
- Yobas M, Crook J, Ross P (2000) Credit scoring using neural and evolutionary techniques. *IMA J Math Appl Bus Ind* 11:111–125
- Zaiyi G, Quek C, Maskell D (2006) FCMAC-AARS: a novel FNN architecture for stock market prediction and trading. In: CEC 2006: Proceedings of the IEEE international conference on evolutionary computation, Vancouver, BC, Canada, July 2006. IEEE Press, Piscataway, NJ, pp 8544–8550
- Zenios SA (2008) Practical financial optimization. Wiley-Blackwell, Chichester, UK

52 Selected Aspects of Natural Computing

David W. Corne¹ · Kalyanmoy Deb² · Joshua Knowles³ · Xin Yao⁴

¹School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
dwcorne@macs.hw.ac.uk

²Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India
deb@iitk.ac.in

³School of Computer Science and Manchester Interdisciplinary Biocentre (MIB), University of Manchester, UK
j.knowles@manchester.ac.uk

⁴Natural Computation Group, School of Computer Science, University of Birmingham, UK
x.yao@cs.bham.ac.uk

1	<i>Introduction</i>	1738
2	<i>Strategies: Generating Expert Pilots, Players, and Traders</i>	1739
3	<i>Examples of Natural Computing’s “Outreach” Elsewhere in Science and Engineering</i>	1757
4	<i>Logistics and Combinatorics Made Easy: Robust Solutions and New Algorithms via Natural Computation</i>	1771
5	<i>Design: Art, Engineering, and Software</i>	1780
6	<i>Concluding Notes</i>	1795

Abstract

In this chapter we will discuss a selection of application areas in which natural computation shows its value in real-world enterprises. For the purposes of demonstrating the significant impact and potential of natural computation in practice, there is certainly no shortage of documented examples that could be selected. We present just ten applications, ranging from specific problems to specific domains, and ranging from cases familiar to the authors to highlights known well in the general natural computation community. Each displays the proven promise or great potential of nature-inspired computation in high-profile and important real-world applications, and we hope that these applications inspire both students and practitioners.

1 Introduction

The study of natural computation has borne several fruits for science, industry, and commerce. By providing exemplary strategies for designing complex biological organisms, nature has suggested ways in which design spaces can be explored and developed into innovative new products. By exhibiting examples of effective cooperation among organisms, nature has hinted at new ideas for search and control engineering. By showing how highly interconnected networks of simple biological processing units can learn and adapt, nature has paved the way for the development of computational systems that can discriminate between complex patterns and improve their abilities over time. And the list goes on.

It is instructive to note that the methods used that have been inspired by nature are far more than simply “alternative approaches” to the problems and applications that they address. In many domains, nature-inspired methods have broken through barriers in the erstwhile achievements and capabilities of “classical” computing. In many cases, the role of natural inspiration in such breakthroughs can be viewed as that of a strategic pointer, or a kind of “tiebreaker.” For example, there are many, many ways that one might build complex multiparameter statistical models for general use in classification or prediction; however, nature has extensive experience in a particular area of this design space, namely neural networks – this inspiration has guided much of the machine learning and pattern recognition community toward exploiting a particular style of statistical approach that has proved extremely successful. Similar can be said of the use of immune system metaphors to underpin the design of techniques that detect anomalous patterns in systems, or of evolutionary methods for design.

Moreover, it seems clear that natural inspiration has in some cases led to the exploration of algorithms that would not necessarily have been adopted, but have nevertheless proven significantly more successful than alternative techniques. Particle swarm optimization, for example, has been found enormously successful on a range of optimization problems, despite its natural inspiration having little to do with solving an optimization problem. Meanwhile, evolutionary computation, in its earliest days, was subjected to much skepticism and general lack of attention – why should a method be viable for real-world problems when that method, in nature, seems to take millions of years to achieve its ends? What need is there for slow methods that rely on random mutation, when classical optimization has a mature battery of sophisticated techniques with sound mathematical bases? Nevertheless, evolutionary methods are now firmly established, thanks to a long series of successful applications in which their performance is unmatched by classical techniques.

The idea of this chapter is to present and discuss a collection of exemplars of the claims made in this introduction. A handful of selected applications of natural computation will be looked at, each chosen for a subset of reasons, such as level of general interest, or impact. Some classic applications will be considered, which still serve as inspirational to current practitioners, and some newer areas will be looked at, with exciting or profound prospects for the future.

The applications are loosely clustered into four themes as follows. The first theme deals with applications under the banner of “Strategies,” where three examples in which natural computing methods have been used to produce novel and useful strategies for different enterprises are studied in detail. These include an evolutionary/neural hybrid method that led to the generation of an expert checkers player, the use of genetic programming (GP) to discover rules for financial trading, and the exploitation of a learning classifier system to generate novel strategies for fighter pilots. The next theme is “science and engineering” where applications that have wider significance for progress in one or more areas of science and engineering, in areas (or in ways) that may not be traditionally associated with natural computing, are considered. The two exemplars in this area are the use of multi-objective evolutionary computation for a range of areas (often in the bio and analytical sciences) for *closed-loop* optimization, and the concept of *innovization*, which exploits multi-objective evolutionary computation in a way that leads to generic design insights for mechanical engineering (and other) problems. The next theme, “logistics,” exemplifies how natural computing (largely, learning classifier systems and evolutionary computation) has provided us with successful ways to address difficult logistics problems (the case of a real-world truck scheduling problem is looked at), as well as a way to design new fast *algorithms* for a range of logistics and combinatorial problems, via approaches referred to as “super-heuristics” and/or “hyper-heuristics.” Finally, the theme of “Design” is considered and three quite contrasting examples are discussed. These are, in turn, antenna design, Batik pattern design, and the emerging area of software design using natural computing methods.

2 Strategies: Generating Expert Pilots, Players, and Traders

Many problems in science and industry can be formulated as an attempt to find a good strategy. A strategy is, for practical purposes, a set of rules (or an algorithm, or a decision tree, and so forth) that sets out what to do in a variety of situations. Expert game players are experts, presumably, because they use good strategies. Similar is true for good pilots, and successful stock market traders, as well as myriad other professionals who are experts in their particular domains. It may well come as a surprise to some that humans do not have the last word on good strategy – strategies can be discovered by software, which, in some cases, can outperform most or even all human experts in particular fields. In this section, three examples of applications in which strategies have been developed via natural computing techniques, respectively, for piloting fighter aircraft, for playing expert-level checkers, and for trading on the stock market will be looked at.

2.1 Discovery of Novel Combat Maneuvers

In the early 1990s and beyond, building on funding support from NASA and the USAF, a diverse group of academics and engineers collaborated to explore the automated development of

strategies for piloting fighter aircraft. A broad account of this work (as well as herein) is available in Smith et al. (2002). The natural computing technology employed is termed “genetics-based machine learning” (GBML), the most common manifestation of which is the *learning classifier system* – essentially a rule-based system that adapts over time, with an evolutionary process central to the rule adaptation strategy. In this work, such an adaptive rule-based system takes the role of a test pilot. In the remainder of this section, some of the background and motivation for this application are covered, the computational techniques used are explained, and some of the interesting and novel results that emerged from this work are presented.

2.1.1 Background: New Aircraft and Novel Maneuvers

As explained in Smith et al. (2002), a standard approach, when developing a new fighter aircraft, is to make a prototype for experimentation by test pilots, who then explore the performance of the new aircraft and, importantly, are then able to develop combat maneuver strategies in simulated combat scenarios. Without such testing, it is almost impossible to understand how a new aircraft will actually perform in action, which in turn depends, of course, on how it will be flown by experienced pilots. In particular, it is very important that pilots are able to develop effective and innovative combat strategies that exploit the technology in the new craft. Following such testing, issues in performance are then fed back into the design process, and perhaps the prototype will need to be reengineered, and so forth. This testing process is obviously very expensive – costing the price of at least one prototype craft, and the time of highly skilled pilots. One way to cut this cost includes using a real pilot, but to “fly” a simulation of the new craft; another is to resort to entirely analytical methods; however, both of these approaches are problematic for different reasons (Smith et al. 2002). The idea of Smith et al.’s research is to explore a third approach, in which a machine learning system takes the place of a test pilot and operates in the context of sophisticated flight simulations.

To help better understand the motivation for this work and grasp the importance of developing novel maneuvers, it will be useful to recount some background in fighter aircraft piloting. This is adapted next from Smith et al. (2002), while a comprehensive account is in Shaw (1998). A relatively new aspect of modern fighter aircraft is the use of post-stall technology (PST). This refers to systems that enable the pilot to fly at extremely high angles of attack (the angle between the aircraft’s velocity and its nose–tail axis). Pilots have developed a range of combat maneuvers associated with PST flight, including, for example, the Herbst Maneuver, in which the aircraft quickly reverses direction via a combination of rolling and a high angle of attack. In another example, the Cobra Maneuver, the aircraft makes a very quick pitch-up from horizontal to 30° past vertical; the pilot then pitches the aircraft’s nose down and resumes normal flight angles. This causes dramatic deceleration, meaning that a pursuing fighter will overshoot. The technologies that allow PST flight have led to the invention of these and several other maneuvers, as well as the prevalence of tactics that involve “out-of-plane” maneuvering, where the attacking aircraft flies in a continually changing maneuvering plane, invariably different from the plane of the target craft. This link between new technologies and new maneuvers is critical in the design and deployment of new aircraft and is the focus of Smith et al.’s work. The results that are described later involve experiments in which the attacking craft was an X-31 experimental fighter plane, with sophisticated PST capability, and where the target craft in the simulations was an F-18.

2.1.2 Learning Classifier Systems

Learning classifier systems (LCSs: Holland et al. 1986; Grefenstette 1988; Goldberg 1989; Holland 1992) use a collection of rules called *classifiers* in the form of state/action pairs. Each such pair indicates an action to take if the environment currently matches the “state” part of the rule. An LCS operates in an environment according to its current set of classifiers, and uses reinforcement learning and other adaptation methods, in particular, including genetic algorithms (GAs) to gradually adapt the rules over time. Classically, a classifier in an LCS represents states and actions as binary strings, but states may also contain “don’t care” characters (#s). For example, the string: “0 1 0 # 1/0 1 0” is a classifier with the meaning: “If the environment is in state 0 1 0 0 1 or state 0 1 0 1 1, then perform action 0 1 0.”

In a typical LCS, each cycle begins with a message representing the state of the environment (as will be seen, the environment in the fighter plane combat context is simply a characterization of the relative positions and velocities of the aircraft in the simulation). The LCS then sees which of its classifiers match this environmental message. There may of course be several, and some form of conflict resolution method must then be invoked to decide which classifier’s action will be executed. The action eventually chosen is then performed. This action may lead to a reward – that is, some aspect of the environment becomes (more) favorable, and the classifier that led to this action receives an increment to its “fitness” score. In some classifier systems, sophisticated credit allocation systems are in place to ensure that the most recent action does not necessarily receive all of the credit. After some specified number of cycles, the genetic algorithm is invoked. The GA population is formed from a subset of the classifiers focusing on those with higher fitness. New classifiers are produced by standard genetic operations on selected classifiers (where, naturally, selection is biased by fitness), and these are then incorporated into the LCS, overwriting some of the less fit existing classifiers. Clearly, an LCS operates in a way that attempts to find – via the GA, which is in turn informed by the fitnesses of classifiers, which in turn are informed by experience in the environment – a good set of classifiers that achieves continual rewards in its environment.

2.1.3 Implementation, Experimentation Details, and Results

The way that LCS technology has been used, with considerable and long-established success in the domain of combat maneuver discovery (Smith and Dike 1995; Smith et al. 2002), is basically as follows. The task faced by the LCS is (typically) a one-on-one engagement for a specific amount of time, such as 30 s. There is a specified initial configuration of positions and velocities, and the period is divided into periods of 0.1 s (i.e., each action of the classifier pilot must last for at least 0.1 s before another action can be performed). At each of the (typically) 300 timesteps during a simulation, each aircraft observes the current configuration, and decides on an action. At the end of an engagement, a score can be calculated based on the relative probabilities of the two aircrafts having damaged their opponents.

All Smith et al.’s experiments employed AASPEM, the Air-to-Air System Performance Evaluation Model developed by the U.S. Government for computer simulation of air-to-air combat. The encoding of the state/action parts of a classifier were as follows. The state part of a classifier comprised 20 bits: six bits were used to encode the two “aspect angles” that gave the current relative positions of the aircraft in terms of their lines of sight. The remaining 14 bits were used to encode seven parameters (hence, each discretized into four bins), namely: range,

speed, delta speed, altitude, delta altitude, climb angle, and opponent's climb angle. The action part of a classifier comprised eight bits, encoding three parameters: a relative bank angle (three bits), an angle of attack (three bits), and a speed (two bits). Speed, for example, was either 100 knots (00), 200 knots (01), 350 knots (10), or 480 knots (11). The meaning of an action that specified, for example, relative bank angle of 30° and speed of 200 knots, was to aim for these as desired targets. In all cases, the simulation environment (i.e., the AASPEM system) would automatically interpret these aims into realistic actions.

A set of such classifiers, therefore, represents a general strategy, and subsets of related classifiers can potentially encode entire novel maneuvers. During a simulated engagement, the classifiers are run for 300 cycles, as described; when no classifier matches the current environmental configuration, a default action for straight, level flight is used. When more than one classifier is matched, the fittest one is chosen as the provider of the action. At the end of the 300 cycles, a fitness measure is calculated. Following experiment with various approaches, the most promising fitness measurement was found to be based on the difference between the self and the opponents' "aspect angles." This basically gives a score that is a linear function along the continuum from: *self is aiming directly at opponent's tail* to *opponent is aiming directly at self's tail*, with the former obviously preferred. The fitness assigned after an engagement was based on the average of this value over the entire engagement, and is assigned to every individual classifier that was active at any point during the engagement.

Following a full engagement, the GA then operates over the whole population of classifiers. Using a moderate selection pressure for parents, and standard crossover and mutation operations, a collection of new classifiers is generated. The fitness assigned to a new classifier is simply the average of its parents' fitnesses. Typically, about half of the classifiers in a population were replaced with new classifiers. A learning run would continue with repeated such engagements (perhaps ~ 500), each resulting in fitness assignment and operation of the GA, leading to a revised set of classifiers for use in the next engagement. The starting configuration for all engagements in a single run was always one from the small set of tactically interesting situations shown in [Fig. 1](#).

The conditions in [Fig. 1](#) were designed to generate X-31 tactics results for a balanced set of relevant situations.

The early experiments described in Smith and Dike (1995) were quite similar, involving one-on-one combat, in which the LCS attempted to find novel maneuvers for the X-31, but the opponent F/A-18 aircraft used a fixed (although suitably reactive and challenging) set of standard maneuvers embedded in AASPEM. In short, the opponent would always attempt to execute the fastest possible turn that would leave it pointing directly at its opponent, at the same time attempting to match the opponent's altitude.

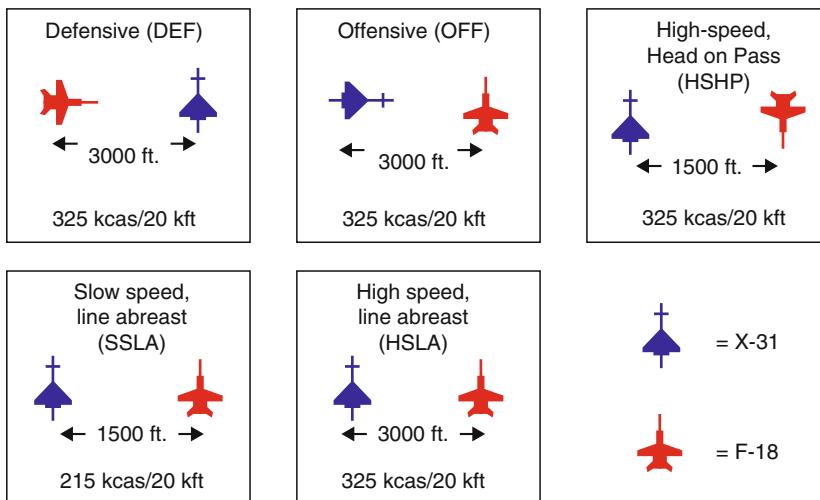
As reported, in considerably more detail, in Smith and Dike (1995), this setup led to the discovery of a wide variety of new and novel fighter maneuvers, which were evaluated in positive terms by real fighter test pilots. An example of such a maneuver is shown in [Fig. 2](#).

The strategy discovered by the LCS in [Fig. 2](#) involves pitching upward sharply, stalling, tipping over, and then engaging the opponent with a favorable relative position. This turns out to be a variation on the "Herbst maneuver" mentioned earlier – in fact it was common for the LCS to rediscover existing maneuvers, as well as discover novel variations.

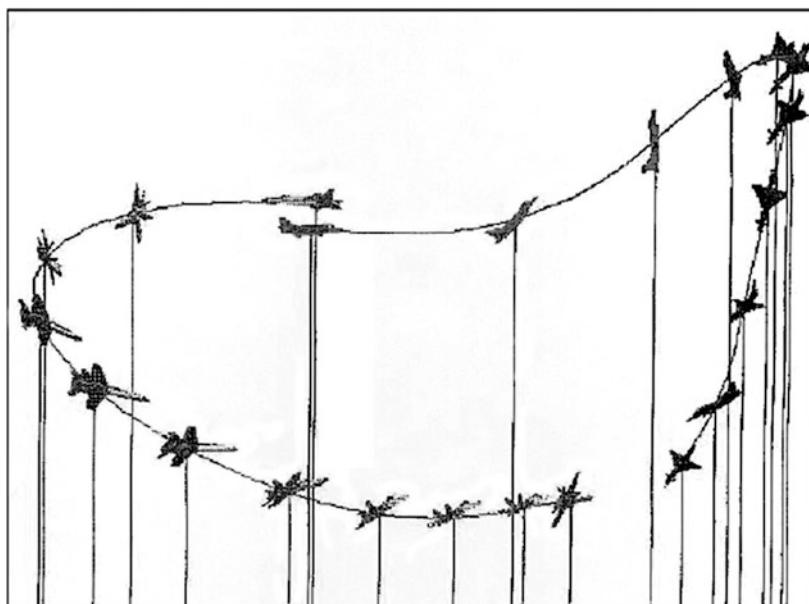
In a later work (Smith et al. 2000), both opponents were controlled by a separate LCS. As Smith et al. (2002) describe, in this scenario, reminiscent of the continuous iterated prisoners' dilemma (IPD), the resulting dynamic system has four potential attractors, the most attractive of which is an "arms race" dynamic, in which each pilot continuously improves his or her

Fig. 1

The matrix of initial conditions for the combat simulations. Reprinted from Smith et al. (2002), with permission from Elsevier.

**Fig. 2**

An example of a novel maneuver evolved by the learning classifier system under the HSHP starting condition (see Fig. 1). The aircraft on the *left* is following a fixed, but reactive strategy; the aircraft on the *right* is following a strategy evolved by the LCS, which in turn is a new variation on the Herbst maneuver. Reprinted from Smith et al. (2002), with permission from Elsevier.



strategies. Smith et al. (2000) explored various setups and indeed found that an arms race effect reliably occurs under some conditions.

2.1.4 Findings and Impact

Smith and Dike (1995) and Smith et al. (2000) contain and discuss several more examples of discovered maneuvers, including some revealing expositions of arms races that develop under the conditions of two LCSs in combat with each other. One clear result of this (still ongoing) work is the real impact it has had on its industrial collaborators. In general, the aerodynamics of a new aircraft can be understood before the first prototype is flown; but, the complexities of piloting and combat, and consequently any real knowledge about the potential combat performance with skilled pilots, are much more difficult to predict. Discovering successful combat maneuvers in the way described has many advantages – in particular, without the cost of test pilot time or prototype construction, LCS experiments generate rich sources of information on combat advantages (or disadvantages) that can be fed back to designers, pilots, and customers. The system described briefly here, and more fully in Smith et al. (2000, 2002), and Smith and Dike (1995), has resulted in several novel strategies that have been approved by test fighter pilots and continue to provide useful results in a highly complex, real-world domain.

2.2 Developing an Expert Checkers Player

The next example comes from the area of “computational intelligence.” The term “computational intelligence” has come to be associated largely with the major fruits of nature-inspired computing, particularly, evolutionary, neural, and fuzzy techniques. This is not to be confused with the older, more well-established term “artificial intelligence,” which stands for the much wider enterprise of, by whatever means, designing algorithms and systems that perform functions that can be called “intelligent.” Artificial intelligence (AI) includes classic areas and techniques such as expert systems, heuristic tree search, machine vision, natural language processing, planning, and so forth, as well as the growing range of nature-inspired techniques. AI is concerned with everything from full-scale intelligent systems, through to the details of appropriate heuristics for edge detection in images from a narrow domain.

Basically, almost any activity, other than those that are “easy” for computers to handle with standard techniques, can be labeled with the adjective “intelligent.” However, via natural computing, achievements have been made that will seem genuinely surprising to many people. It is no great surprise, for example, that computers can design, more successfully than humans, effective production schedules for factories with thousands of jobs per day. However, it perhaps is surprising that one can produce a software that plays checkers at the level of an expert, without encoding any expert knowledge of the game.

2.2.1 Blondie24

In 1999, on an Internet gaming site called “The Zone,” an online checkers player with the screen name *Blondie24* regularly played against a pool of 165 human opponents, and achieved

a rating of 2,048, placing it well into the top half a percent of checkers players using that site. Blondie24 learned to play well at checkers, as did all of the good human players using that site (or otherwise). However, “she” was (and still is) a computer program.

In common with many successful artificial intelligence game playing programs, Blondie24 (Chellapilla and Fogel 1999a, b, 2001; Fogel 2002) incorporates a minimax algorithm (Russell and Norvig 2003) to traverse the game tree induced by the available moves from the current position. However, individual nodes in the tree are evaluated by an artificial neural network (ANN). The input to this ANN is a specialized representation of the current state of the game, and the output is a single value that is then used by the minimax algorithm. So far so clear – one can perhaps imagine that a well-trained or well-designed ANN could be capable of returning values in this context that would translate to competent checkers playing. But how can one design, or train, such an ANN? In Blondie24’s case, training was accomplished by using an evolutionary algorithm. A population of such ANNs played against each other, accumulating points over many games. The result of a game between two such ANNs comes down to a single value (per ANN) – either 1 (win), 0 (draw), or -2 (lose) – and the overarching evolutionary algorithm operates by regarding the fitness of an ANN as its total score after a number of games. In each “generation” of this evolutionary algorithm, the ANNs with the lowest scores are eliminated, and new ones are generated by making mutant copies of the better performers, and so it continues.

For several reasons, Blondie24’s design and its success are both surprising and significant. Its prowess at checkers does not rely on tuition by human experts. Instead, it emerges from the evolutionary algorithm process, guided only by the bare, raw total of points earned after playing several games. If an individual had a fitness of six, for example, it was considered better (and hence had more chance for selection as a parent) than an individual with fitness four. However, this takes no account of the distribution of wins, losses, and draws. The individual with fitness six may have won six games and drawn four, while the individual with fitness four may have won eight games and lost two.

Guided only by this summary measure of performance, an evolutionary algorithm was able to traverse the space of checkers-playing ANNs (or, more correctly, ANNs for evaluating game positions in the context of minimax search) and emerge with expert-level players. It is worth covering in more detail the approach taken to generate Blondie24, which is done next, following the treatment provided in Chellapilla and Fogel (2001).

2.2.2 Checkers: The Game

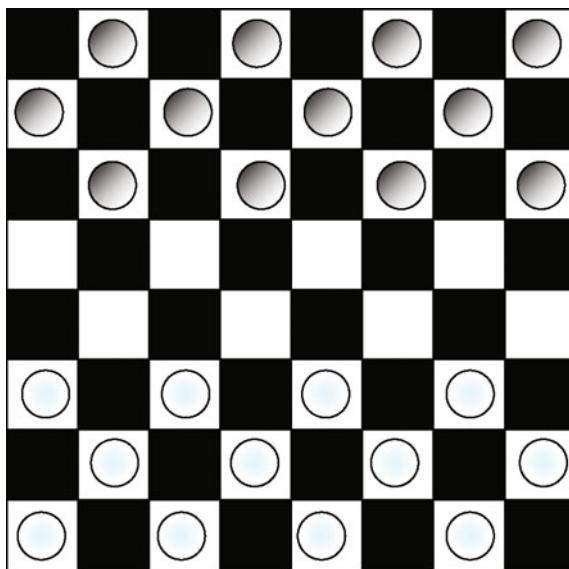
Checkers, known in some countries as “draughts,” involves an eight-by-eight board with squares of alternating colors, equivalent to a chessboard. Each player has 12 identical pieces, and the initial game position is as detailed in Fig. 3.

When it is a player’s turn to move, the allowed moves are as follows: an individual piece can move diagonally forward by one square; or an individual may jump over an opponent’s checker into an empty square. Such a “jump” is only allowed if it takes two diagonal steps in the same direction, the first such step is occupied by an opponent’s piece, and the second step is currently empty. After a jump, the opponent’s piece is removed from play.

If one or more jump moves are available, then it is mandatory for the player to make such a move. If an opponent manages to find itself in the final row (from their side’s viewpoint), it becomes a “king” piece. It is then able to move either forward or backward, but otherwise

Fig. 3

The initial position in a game of checkers. The White player moves upward, and the Black player moves downward.



follow the same rules. The object of the game is to reach a position in which the opponent has no possible moves – a common way in which this happens is for the opponent’s pieces to all have been removed.

2.2.3 Representing the Board and Evaluating Moves

Chellapilla and Fogel (2001) used a straightforward and sensible approach to encoding a board position. The current state of the game is simply represented by a vector of 32 numbers, one for each board position. The numbers in a position are either $-K$, -1 , 0 , 1 , or K , where K represents a value assigned to a king. From the viewpoint of a given player, a 1 or a K at a given board position represent, respectively, either a standard piece or a king at that position, while the negative values are used to represent the opponent’s pieces, and 0 indicates an empty position. In Chellapilla and Fogel’s work, K was not preset. Rather than bias the process toward giving a king any particular relative value over an ordinary piece, the value of K was itself subject to evolution.

When a move is to be made, Blondie24 operates by evaluating, in turn, each of its possible moves. Any such move leads to a future board position, and this future board position is evaluated by the ANN. The input to the ANN is therefore this 32-dimensional vector. As is well known from ANN theory, any reasonable ANN architecture (in terms of the number of hidden layers and the numbers of nodes in each layer) might suffice in being capable of then performing the appropriate mapping from input vector to appropriate, useful output. The difficulty, as ever, is in choosing an appropriate training regime, which promotes learning of suitable features and components of the problem state that are useful guides toward a proper

evaluation. After initial experiments with a more straightforward neural network architecture (which did not encapsulate the spatial information that human players take for granted), Chellapilla and Fogel designed the architecture of the ANN in a way that highlighted potentially appropriate features. This was done as follows.

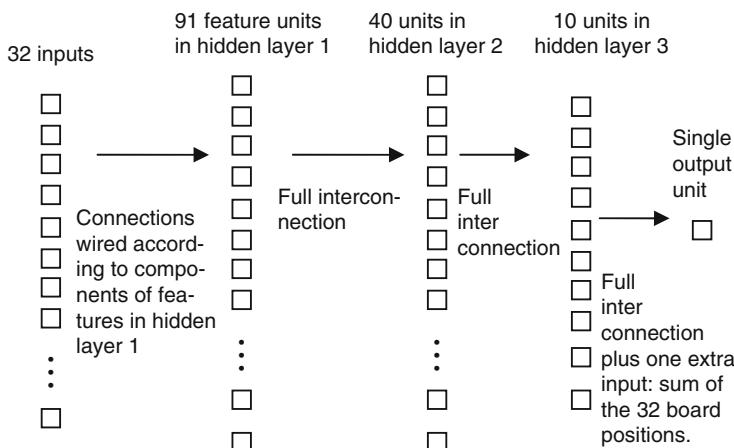
Each 3×3 block on the board was represented by its own unit in the first hidden layer. That is, given any specific 3×3 block, one of the units in the first hidden layer received incoming connections from that specific set of nine inputs from the input layer (from the nine parts of the vector corresponding to the component positions of that 3×3 block), and had no incoming connections from any of the other units. In this way, a specific signal emerging from this unit, for later processing in subsequent layers, summarizes the state of play in that specific 3×3 block. The first hidden layer contained such a unit for each of the 36 different 3×3 blocks on the board. In just the same way, each 4×4 block, 5×5 block, 6×6 block, 7×7 block, and 8×8 block (of which there was of course just one) was represented in the first hidden layer by its own unit. This resulted in a set of 91 units which comprised the first hidden layer.

The complete picture of the ANN's architecture is given in [Fig. 4](#). Between the input layer, which simply carries 32 units, one per board position, and the first hidden layer, the connections are arranged according to the specific feature encoded by each of the units in hidden layer 1. Between the pairs of layers, the connections are all complete – for example, each unit in hidden layer 1 has a feedforward connection to each unit in hidden layer 2, and similarly for hidden layers 2 and 3, while every unit in hidden layer 3 is connected to the single output unit. The output unit receives an additional input, which is the sum of the 32 board positions.

In total, including bias weights, there are 5,046 connections in this network, each of which is a real-valued weight subject to the evolution process. In addition, every hidden layer unit has a bias input, which means an additional weight to be evolved. Each unit in the hidden layers operates in the standard way, common in most ANN applications, by calculating the weighted sum of its inputs and applying the hyperbolic tangent function, resulting in an output signal

Fig. 4

The architecture of the Blondie24 artificial neural network.



strictly between -1 and 1 . From the perspective of the ANN “player,” this ultimate scalar value is directly used as an estimate of the value of this board position: the closer it is to 1 , the better for the ANN. However, where the board position was actually a win for the ANN, the value was taken to be precisely 1 , and if it was a win for the opponent, the value was taken to be -1 .

2.2.4 Evolving Checkers Players

The process begins with a population of 15 such ANNs, which are initialized randomly. Every connection weight and bias value is given a value chosen uniformly at random from the interval $[-0.2, 0.2]$, and with K set initially at 2.0 . In common with the practice of evolutionary programming and evolution strategies, each individual in the population also contained a vector of step-size parameters. For every connection weight, and every corresponding bias unit, there was also a step-size parameter governing the range of mutations that would be applied to that parameter. That is, when a weight or bias parameter was mutated, this was done by adding a Gaussian perturbation whose mean was 0 and whose variance was provided by the associated step-size parameter in the chromosome. The step-sizes were initially all set at 0.05 , and then subject to evolution along with the other parameters.

Whenever an ANN was selected as a parent, its offspring was generated as follows: first, each of the step-size parameters was mutated, by multiplying it with a random number from a specific exponential distribution, and every weight and bias parameter was mutated by adding a Gaussian perturbation whose step size was the associated step-size parameter, as indicated. Finally, recall that each individual also carries its own value for K , which is also subject to evolution. This was mutated by adding a perturbation chosen uniformly at random from the set $\{-0.1, 0, 0.1\}$, but was protected from moving below 1 or above 3 .

During the evolution process, each ANN played one game each against five opponents, selected uniformly from the population. With the scoring for individual games as indicated, the ANN would therefore accumulate a score over these five games ranging from ten (all losses) to five (all wins). A game was declared as a draw (zero points) if it lasted for 100 moves. Essentially, in each generation, each ANN took part in around ten games, and the top 15 (in terms of points received) became parents for the next generation. Each individual game was played using a minimax alpha–beta search set to four-ply (with extended ply in a number of special cases). After 840 generations in which evolving ANNs played against each other, the best-resulting ANN was then harvested and recruited to play against human opponents on the Internet gaming site “The Zone.” The next subsection summarizes the surprising and remarkable resulting performance of this ANN.

2.2.5 Humans Versus Evolved ANNs

Over a 2-month period, the evolved ANN, eventually named “Blondie24” (which was successful in attracting opponents) played 165 games against human opponents at “The Zone,” although opponents were not aware they were playing against a computer program. In these games, the ANN used an eight-ply search, and faced a variety of opponents. The ANN’s performance placed it at better than 99.6% of all the (rated) players using the site. On one occasion, the ANN beat an expert-level player (with a rating of 2,173, just below the master level of 2,200) who was ranked 98th of over 80,000 registered players.

Chellapilla and Fogel (2001) performed some comprehensive control experiments, which showed that the evolved ANN operated with a clear advantage over a system that simply used the piece differential as the basis for choosing moves in an eight-play approach. In particular, they compared the ANN with a piece-differential-based player, on the basis of using equal CPU time in their look-ahead search at each move; this disadvantages the ANN, since it has over 5,000 weight parameters involved in its heuristic calculation, so the piece-differential player can look further ahead in the time available. These experiments showed conclusively that the evolved ANN was a significantly better player in both equal-ply and equal CPU-time conditions.

The achievement of Blondie24 is remarkable from many viewpoints: particularly the essential simplicity of its approach, the fact that the search landscape for the evolutionary algorithm was so huge, and the fact that fitness assessment was a relatively coarse measure of a network's performance. A straightforward assessment of Blondie24's "message" is that it exemplifies the flexibility and potential of evolutionary search, even when this is recruited to search a coarse-grained 10,000-dimensional landscape (the evolution strategy that was employed optimized both a weight and a step-size parameter for each connection). Achieving expert-level performance (over 2,000 points) is considerably superior to most humans. Perhaps not surprisingly, this is also certainly superior to a simpler (but seminal) approach in this area by Samuel (1959), which attempted to derive, by an iterative learning process, a polynomial board rating function. Chellapilla and Fogel (2001) note that this was considered to rate below 1,600 in the opinion of the American Checkers Federation games editor.

The world champion checkers program, Chinook (Schaeffer et al. 1996), is rated at over 2,800, over 100 points above its closest human competitors (Schaeffer 1996). In fact, it is now known that Chinook can never be defeated in "go-as-you-please" checkers, in which there are no restrictions on the initial moves. The chief difference between Blondie24 and Chinook is the amount of built-in specialized knowledge. In Chinook, the level of such knowledge is very substantial indeed; in Blondie24, it is virtually none. Along with many other elements informed by careful expert knowledge and tuning, Chinook incorporates a database of games from previous grand masters and a complete endgame database for *all* cases that start from ten pieces or fewer. Blondie24 and Chinook represent entirely different artificial intelligence approaches to designing a game-playing program. It is not difficult to argue that the approach taken by Blondie24 is the more interesting and impactful – from no prior knowledge, other than a built-in awareness of the rules of the game, an expert-level player emerged from the evolutionary process, providing a very tough, usually insurmountable challenge to all but a very small percentage of human players.

Finally, since the checkers research, Chellapilla and Fogel's approach was extended to address chess, by combining the coevolutionary spatial neural network approach with domain-specific knowledge (Fogel et al. 2004, 2006). The result was an evolved chess player that earned wins over Fritz 8, which was the fifth best computer program in the world at that time.

2.3 Discovering Financial Trading Rules

Financial markets are complex and ever-changing environments in which groups of individuals, companies, and other investors are always competing for profit. There are many

opportunities in this area for machine learning and optimization methods, and consequently a variety of natural computation approaches to be exploited, and a chapter in this volume is indeed devoted to this topic. This section focuses on one specific thread of research in this area – which has a simply grasped approach and a straightforward task to solve. This is the use of GP to discover new and valuable rules for financial trading.

It is now common to see applications of evolutionary computation applied to the financial markets (Brabazon and O'Neill 2005, this volume). GP (Koza 1992; Angeline 1996; Banzhaf et al. 1998) is particularly prominent in terms of the degree to which it has recently been applied in finance (Chen and Yeh 1996; Fyfe et al. 1999; Allen and Karjalainen 1999; Marney et al. 2001; Chen 2002; Cheng and Khai 2002; Farnsworth et al. 2004; Potvin et al. 2004). This section focuses on the specific area in finance known as *technical analysis* (Pring 1980; Ruggiero 1997; Murphy 1999; Lo et al. 2000). Technical analysis is a set of techniques that forecast the future direction of stock prices via the study of historical data. Many different methods and tools are used, all of which rely on the principle that price patterns and trends exist in markets, and that these can be identified and exploited.

Common tools in technical analysis include indicators such as moving averages (the mean value of the price for a given stock or index over a given recent time period), relative strength indicators (a function of the ratio of recent upward movements to recent downward movements). There have been a number of attempts to use GP in technical analysis for learning technical trading rules, and a typical strategy is for such a GP-produced rule to be a combination of technical indicator “primitives” with other mathematical operations. Such a rule is often called a “signal.” For example, GP may be employed to find both a good buy signal and a good sell signal – that is, one rule that, if its output is above 0, indicates that it is a good time to buy, and a different rule indicates when it is a good time to sell.

Early attempts to use GP in technical trading analysis were by Chen and Yeh (1996) and Allen and Karjalainen (1999). However, although GP could produce profitable rules for the stock exchange markets, their performance did not show any benefit when compared to the standard buy-and-hold approach. “Buy-and-hold” simply means, for a given period, buying the stock at the beginning of the period, and selling at the end – hence, always a good idea in a market that generally moves up during the period.

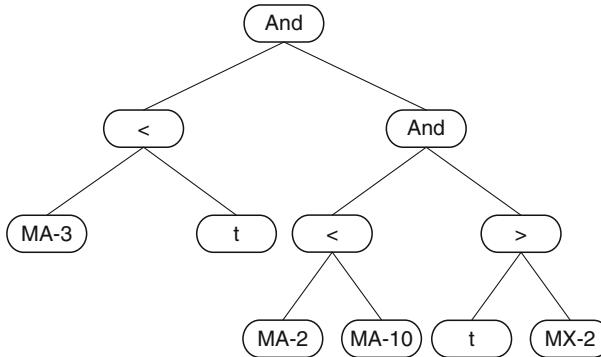
More recent applications of GP in this context have been more encouraging (Marney et al. 2000, 2001; Neely 2001). Becker and Seshadri’s (2003a, b, c) work is looked at in particular, which found GP-evolved technical trading rules that outperformed buy-and-hold (at least if dividends are excluded from stock returns). In turn, their approach was founded in Allen and Karjalainen (1999), with various modifications. After giving some detail of the overall approach, further experiments from Lohpetchi and Corne (2009, 2010), which probed certain boundaries of the technique and examined its robustness, are summarized.

2.3.1 Becker and Seshadri’s Approach to Evolving Trading Rules

Becker and Seshadri (2003a, b, c), based on Allen and Karjelainen (1999), used a fairly standard GP approach and found rules that significantly outperformed buy-and-hold on average over a 12-year test period of trading with the Standard & Poors (S&P 500) index. Their GP’s function set contained the standard arithmetic, Boolean, and relational operators, and the terminal set included some basic technical indicators. An example of a specific rule found by their method is in  Fig. 5.

Fig. 5

Example of a trading rule found by Becker and Seshadri's GP approach.



The rule in Fig. 5 has the following basic interpretation “the 3-month moving average (MA-3) is less than the lower trend line (t) and the 2-month moving average (MA-2) is less than the 10-month moving average (MA-10) and the lower trend line (t) is greater than the second previous 3-month moving average maxima (MX-2).” This signals trading behavior in the following way: If the trader is currently out of the market (no stocks invested in the S&P 500), and the rule evaluates to *true*, then *buy*; if the trader is currently in the market, and the rule becomes *false*, then *sell*. This procedure assumes a fixed amount to invest (e.g., \$1,000) whenever there is a buy signal.

The remainder of this subsection explains the approach in more detail, and tries to emphasize the key points that are necessary to replicate similar performance. In passing, the main ways in which Becker and Seshadri modified the original approach of Allen and Karjelainen are noted. These were: monthly trading decisions rather than daily trading; a reduced function set in the GP approach; a larger terminal set in the GP approach (with more technical indicators); the use of a complexity-penalizing element to avoid over-fitting; and finally, modifying fitness function to consider the number of periods with well-performing returns, rather than just the total return over the test period. In combination, these methods enabled Becker and Seshadri to find rules that outperformed buy and hold for the period they tested, when trading on the S&P 500 index. It is an open question as to which modifications were most important to this achievement; however, Lohpetchi and Corne (2010) begin to answer that question, as will be seen, by showing (as is intuitively the case) that it is increasingly easier to find good rules as one changes the trading interval from daily to weekly, and then to monthly.

In the following, we exclusively use S&P 500 data (as did Allen and Karjelainen 1999 and Becker and Seshadri 2003a, b, c); so, the “portfolio” is the fixed set of 500 stocks in the S&P 500 index, which aggregate to provide daily price indicators.

2.3.2 Function and Terminal Sets Used by Becker and Seshadri

In Becker and Seshadri's GP approach, the function set comprises simply the Boolean operators and, or, and not, and the relational operators $>$ and $<$. The terminal set comprises

the following, in which “period” was always *month* in Becker and Seshadri’s work, but later Lohpetch and Corne (2010) will be discussed in which it could be day, week, or month in different experiments.

- Opening, closing, high and low prices for the current period;
- 2-, 3-, 5-, and 10-period moving averages;
- Rate of change indicator: 3-period and 12-period;
- Price Resistance indicators: the two previous 3-period moving average minima, and the two previous 3-period moving average maxima;
- Trend Line indicators: a lower resistance line based on the slope of the two previous minima; an upper resistance line based on the slope of the two previous maxima.

The n -period moving average at period m is the mean of the closing prices of the n previous months (included m). The n -period rate of change indicator measured at period m is: $(c(m) - c(m - (n - 1))) \times 100 / c(m - (n - 1))$, where $c(x)$ indicates the closing price for period x . Previous maxima MX1 and MX2 are obtained by considering the 3-period moving averages at each point in the previous 12 periods. Of the two highest values, that closest in time to the current period is MX1, and the other is MX2. The two previous minima are similarly defined. Finally, to identify trend line indicators, the two previous maxima are used to define a line in the obvious way, and the extrapolation of that line from the current period becomes the upper trend line indicator; the lower trend line indicator is defined similarly, using the two previous minima.

2.3.3 Becker and Seshadri’s Fitness Function

The fitness function has three main elements. First is the so-called “excess return,” indicating how much would have been earned by using the trading rule, in excess of the return that would have been obtained from a buy-and-hold strategy. The other elements of the fitness function were introduced by Becker and Seshadri to avoid over-fitting. These were as follows: a factor that promoted fitness for trading rules that were less complex (e.g., with reference to [Fig. 5](#), a less complex rule is one in which the tree has smaller depth); and, a factor that considered “performance consistency” (PC), favoring rules that generally were used often, each time providing a good return, rather than rules that were very fortunate in only brief periods.

In more detail, the excess return is simply $E = r - r_{bh}$, where r is the return on an investment of \$1,000, and r_{bh} is the corresponding return that would have been achieved using a buy-and-hold strategy. To calculate r , Allen and Karjeleinen (1999) and Becker and Seshadri (2003a, b, c) used:

$$r = \sum_{t=1}^T r_t I_b(t) + \sum_{t=1}^T r_f(t) I_s(t) + n \ln \left(\frac{1-c}{1+c} \right)$$

in which: $r_t = \log P_t - \log P_{t-1}$, indicating the continuously compounded return, where P_t is the price at time t . The term $I_b(t)$ is the buy signal, either 1 (the rule indicates *buy* at time t) or 0. The sell signal, $I_s(t)$, is analogously defined. So, the first component gives the return on investment from the times when the investor is in the market, and the second component, $r_f(t)$, indicates the risk-free return which would otherwise be available, which is taken for any

particular day t from the published US Treasury bill data (these data are available from <http://research.stlouisfed.org/fred/data/irates/tb3ms>). The second component therefore represents time out of the market, in which it is assumed that the investor's funds are earning a standard risk-free interest. Finally, the third component is a correction for transaction costs, estimating the compounded loss from the expenditure on transactions; a single transaction is assumed to cost 0.25% of the traded volume – for example, \$2.50 for a transaction of volume \$1,000. The number of transactions sanctioned during the period by the rule is n .

The second main part of the fitness function, r_{bh} , is calculated as:

$$r_{bh} = \sum r_t + \ln\left(\frac{1-c}{1+c}\right)$$

in which r_t is as indicated above. This calculates the return over the period from risk-free investment in US Treasury bills, involving a single buy transaction.

Becker and Seshadri's complexity-penalizing adjustment works as follows: Given a rule that has *depth* and fitness value (excess return) f , the adjusted fitness becomes $5f/\max(5, \text{depth})$. This involves the constant 5 as a relatively arbitrary desired maximal depth, and in the trading rule evolution context, there has been little parametric investigation around this value so far. The other of Becker and Seshadri's modification to the excess return fitness function, PC works as follows. The excess return E is calculated for each successive group of K windows of a certain length covering the entire test period.

The value returned is simply the number of these periods for which E was greater than both the corresponding buy-and-hold return (from investing in the index over that period) and the risk-free return during that period. For example, if the rule is evaluated over a 5-year test period, the PC version of the fitness function might use 12-month windows. Clearly, there are five such windows in the test period, and the fitness value returned will simply be an integer between 0 (the rule did not outperform buy-and-hold and risk-free investment in any of the five windows) and 5 (the rule was more successful than both buy-and-hold and risk-free returns in all of the windows).

At last, the above background enables one to state the fitness function used (with minor variations in each case) in Becker and Seshadri (2003a, b, c) and Lohpetch and Corne (2009, 2010). The fitness of a GP tree of depth d in these studies was the performance-consistency-based fitness (i.e., a number from 0 to X , where there were X windows covering the test data period) adjusted to penalize undue complexity by $5f/\max(5, d)$, in which f is the number of the X windows in which both the corresponding buy-and-hold return and the risk-free return were outperformed by the rule.

2.3.4 Some Illustrative Results

Reported here are some results that show how this approach performs on various windows of time when trading with the S&P 500 index. The results shown are some of those from Lohpetch and Corne (2010), and the subset of those that were obtained under the same test conditions as used by Becker and Seshadri (i.e., monthly trading for a specific training and test window) are quite indicative of Becker and Seshadri's own results. However, it is worth first discussing some further details of the way that the GP method was set up for the experiments.

Although perhaps not always the case, it seems that the precise choice of mutation and crossover methods makes little real difference in this application; the chance of evolving

effective trading rules seems clearly related to a good choice of function and terminal sets for the expression trees, as well as a wise choice of fitness function. Although, as will be seen later, the frequency of trading is a significant factor. Meanwhile, Lohpetch and Corne (2009, 2010) used standard mutation operators, as described by Angeline (1996), namely, *grow*, *switch*, *shrink*, and *cycle* mutation, and used standard subtree-swap crossover (Koza 1992). Finally, it is noted that, in the experiments whose results are summarized next, the population was initialized by growing trees to a maximum depth of 5; however, no constraint was placed on tree size beyond the initial generation, other than the pressure toward less complex trees which is a part of the fitness function.

Presented now are some results that indicate the performance achievable by such a GP system as described in the last section. In the experiments summarized here, from Lohpetch and Corne (2009, 2010), a population size of 500 was used, and other relatively standard GP settings, with a run continued for 50 generations. Here results for each of the daily, weekly, and monthly trading are shown, and it is found that outperformance of buy-and-hold can indeed be achieved even for daily trading, but moving from monthly to daily trading, the performance of evolved rules becomes increasingly dependent on prevailing market conditions. The data used is the S&P 500 index from 1960 onward. In Becker and Seshadri's demonstration of outperforming buy-and-hold, only monthly trading was used, and their results arise from training the rules over the period 1960–1991, and evaluating them on a test period spanning 1992–2003. This corresponds to "MonthlySplit1" in the following; however, it is clear from Lohpetch and Corne (2009) that more robust performance is obtained when a validation period is used. The following illustrative results therefore reflect a training/validation/test regime in which the GP training run evaluated fitness on the training period only, but the rule that achieved the best performance on a validation period was harvested, and this was the rule evaluated on the test period.

Results for four different monthly trading data splits are summarized below. The splits themselves are as follows, in which N gives the length of the validation period in years, immediately following the training period, and K gives the length of the test period in years, again immediately following the validation period.

- MonthlySplit1: 31 years training; $N = 12$, $K = 5$
- MonthlySplit2: 31 years training, $N = 8$, $K = 8$
- MonthlySplit3: 31 years training, $N = 9$, $K = 9$
- MonthlySplit4: 25 years training, $N = 12$, $K = 12$

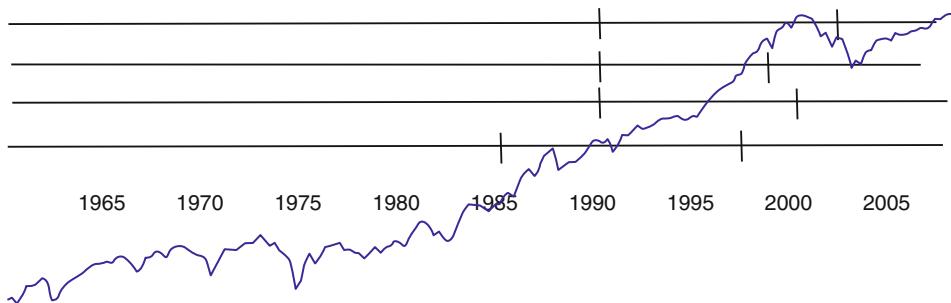
Corresponding splits for the weekly and daily trading experiments are also summarized here (for details see Lohpetch and Corne 2010). Four different weekly trading and daily trading data splits were also investigated, roughly corresponding to the monthly data splits in terms of the number of data points in each split. For example, WeeklySplit1 involved 366 weeks trading, 158 weeks validation, and 157 weeks testing. Similarly, the training periods for the daily splits were approximately 1 year in length. The four different weekly and daily splits started at different times, spread evenly between 1960 and 1996.

Figure 6 shows the four monthly data splits aligned against the S&P 500 index for the period 1960–2008. Note that the market movements were net positive in each part of each split, indicating that outperforming buy-and-hold was in all cases a challenge.

In Lohpetch and Corne's (2010) experiments, they also explored different lengths of window for the PC element of the fitness function. In Becker and Seshadri's work, the PC approach clearly results in improved performance, however, they only reported on the use of

Fig. 6

The S&P 500 index over the period 1960–2008, illustrating the four data splits for the case of monthly trading.

**Table 1**

Summary of results for monthly trading

Data split	PC period	Trials outperforming buy-and-hold	PC period	Trials outperforming buy-and-hold
Monthly Split1	6	10 out of 10	12	10 out of 10
Monthly Split2	6	9 out of 10	12	10 out of 10
Monthly Split3	6	10 out of 10	12	9 out of 10
Monthly Split4	6	10 out of 10	12	10 out of 10
Monthly Split1	18	10 out of 10	24	10 out of 10
Monthly Split2	18	8 out of 10	24	10 out of 10
Monthly Split3	18	8 out of 10	24	7 out of 10
Monthly Split4	18	10 out of 10	24	10 out of 10

12-month windows. Lohpetch and Corne experimented with different lengths for these windows for each trading situation, namely: 6-, 12-, 18-, and 24-month periods for monthly trading; 12 and 24 weeks for weekly trading, and 12 and 24 days for daily trading.

For each trading period (monthly, weekly, daily), Lohpetch and Corne did ten runs for each combination of data split and consistency of performance period. The outcome of the ten runs is summarized in [Tables 1–3](#), simply as the number of times that the result outperformed buy-and-hold.

As [Table 1](#) shows, monthly splits 1 and 4 were clearly well disposed to good performance, but performance was also rather robust on the other monthly splits. Note that outperforming buy-and-hold would seem to be more likely, according to a priori intuition, when the performance of buy-and-hold in the test period is relatively weak, but this is not the case for monthly splits 1 and 4 (see [Fig. 6](#)). The results are quite impressive from many points of view. In many cases, ten tests out of ten showed that a simple trading rule evolved by GP was able to outperform buy-and-hold in an upwardly moving market.

[Table 2](#) shows the results, summarized in the same way, for the case of weekly trading, and [Table 3](#) presents the corresponding results for the case of daily trading. These clearly

Table 2

Summary results for weekly trading

Data split	PC period	Trials outperforming buy-and-hold	PC period	Trials outperforming buy-and-hold
Weekly Split1	12	2 out of 10	24	7 out of 10
Weekly Split2	12	10 out of 10	24	5 out of 10
Weekly Split3	12	4 out of 10	24	4 out of 10
Weekly Split4	12	10 out of 10	24	10 out of 10

Table 3

Summary of results for daily trading

Data split	PC period	Trials outperforming buy-and-hold	PC period	Trials outperforming buy-and-hold
Daily Split1	12	0 out of 10	24	0 out of 10
Daily Split2	12	0 out of 10	24	0 out of 10
Daily Split3	12	10 out of 10	24	9 out of 10
Daily Split4	12	2 out of 10	24	4 out of 10

show increasingly less robust results. It certainly seems that this relatively straightforward GP method can find robust rules for weekly trading that outperform buy-and-hold in some circumstances (splits 2 and 4), with less reliable performance in other cases. However, Lohpetch and Corne (2009, 2010) were not able to discern any pattern that explains this from analyses of the data splits. Finally, for daily trading, [Table 3](#) shows that outperforming buy-and-hold is less likely, with strong performance in only one of the four data splits, and very poor performance in two of the data splits.

2.3.5 A Brief Discussion

The investigation of GP in financial applications and, in particular, the use of it to discover technical trading rules, remains an active thread of research in both industry and academia. In the published academic research, it was commonly found in earlier studies that rules found by GP were profitable, but usually not competitive with straightforward “buy-and-hold” strategies. However, as seen earlier, the situation is changing and it now seems that progress is being made in finding ways to use GP to produce effective and interesting rules that might be used by individual traders. There are several caveats, and of course this enterprise is only one thread of work in a wide area that also involves natural language understanding and many other areas of machine learning (e.g., to spot ideal trading opportunities based on the latest online news). However, this work represents another example of the way in which natural computation can help generate strategies for complex situations which are competitive with those that are self-designed.

It should also be noted that the approach described in this section is far from the last word in the application of GP to the specific area of technical trading. Pains have been taken to describe a

classic approach, and it has been shown that it can indeed find robustly profitable trading rules under a range of conditions – however, several more sophisticated ways to use GP in this area also exist. For example, rather than simply evolve a single rule that encapsulates both a buy and a sell signal, different rules can be evolved separately for buying and selling. Also, it is noted that interested researchers may pick up code for evolving technical trading rules (written by Dome Lohpetch) from the following site: <http://www.macs.hw.ac.uk/~dwcorne/gptrcode/>.

It is also worth mentioning alternative directions which attempt to gain on buy-and-hold by including risk metrics in the rules (or in their evaluation). Typically, a risk measure such as the Sharpe ratio (Sharpe 1966) is used to normalize the estimate of financial return, effectively downgrading the performance of rules that promote trading in volatile conditions, promoting rules more likely to be applied by investors. For example, in attempting to build on work by Fyfe et al. (1999), Marney et al. (2000, 2001) included the use of metrics for calculating risk, although still did not outperform buy-and-hold. More recently, Marney et al. (2005) used the Sharpe ratio and found that a technical trading rule easily outperformed simple buy-and-hold in terms of unadjusted returns, but not in terms of risk-adjusted returns. There is clearly much work still to do until techniques exist in the research literature that can robustly outperform buy-and-hold in a way that satisfies risk-conscious traders, although the progress and effort in this direction makes it clear that this will be achieved, as well as suggesting that private and unpublished research in commercial organizations has almost certainly achieved this already with appropriate use of GP and similar technologies.

3 Examples of Natural Computing’s “Outreach” Elsewhere in Science and Engineering

In this section, two areas of natural computation are selected that have wider implication for significant areas of science and technology. Mostly, an application of a natural computing technique may produce excellent results in its domain, and the impact of those results, though potentially significant, tend to remain solidly within that domain. Progress in general financial mathematics, for example, will not be revolutionized by the trading application discussed in [Sect. 2](#). However, sometimes an exemplar application will open up previously unconsidered possibilities in a whole subfield of science. In this section, two examples are discussed in which such broader consequences can be seen. The first is the use of (mostly) multi-objective evolutionary computation in the area of *closed-loop* optimization, in order to optimize a range of processes and products in the biosciences, process industries, and other areas. In this arena, evolutionary computing was never an “obvious” technique to try, given the potential cost in time; however, its use has time and again proven worthy, and this in turn leads directly to better and faster processes and products emerging from, for example, the use of the instruments that have been configured via evolutionary techniques. The second example area looked at in this section is the concept of *innovation*, which exploits multi-objective evolutionary computation in a way that leads to generic design insights for mechanical engineering (and other) problems. In multi-objective problems (see Deb 2001; Corne et al. 2003a), the result of solving the problem is (usually) a large collection of diverse solutions, each optimal in a sense, but traversing a Pareto surface of optimal from (for example) highly reliable and high-cost solutions to exceptionally cheap but less reliable ones. The notion of innovation is to exploit the prowess of evolutionary computing in obtaining such a diverse set, by further analyzing this collection of designs to

find, as it turns out, previously unknown generic design rules, which seem to be true of all “optimal” designs, wherever they sit on this Pareto surface. A well-designed natural computing approach to a specific problem in mechanical engineering, for example, thereby leads to new design principles that can have much wider impact than simply solving the given problem.

3.1 Applications in Analytical Science: Closed-Loop Evolutionary Multi-objective Optimization

Knowles (2009) provides a detailed and comprehensive summary of historical origins and current work in the broad area of closed-loop optimization using evolutionary multi-objective algorithms. A similar but briefer treatment is provided here, including a summary of two of the several interesting modern case studies covered in Knowles (2009).

As Knowles (2009) points out, the idea of using an evolution-inspired technique for producing solutions to optimization problems has been explored for around 60 years so far, starting in the 1950s. The celebrated British statistician George E.P. Box used the term “closed-loop” in describing the kind of evolution experiments that were first investigated, while Ingo Rechenberg (a pioneer in evolutionary computation) used the phrase “evolutionary experimentation.” In closed-loop evolution-inspired optimization, the evolution process is a combination of computation and physical experiment. The evaluation of candidate design solutions is done in the real world by conducting physical experiments. Much of the pioneering work in evolutionary computation (by Rechenberg and his team) was of this kind. In much more recent times, the closed-loop approach has been used, commonly with much success, in evolvable hardware research (see the chapter [• Evolvable Hardware](#) in this volume), in evolutionary robotics research, as well as in microbiology and biochemistry. In this section, some brief example case studies are described to illustrate the increasingly wide emerging impact of this technique at the evolution/engineering interface.

With a focus on closed-loop evolutionary multi-objective optimization (CL-EMO) in particular, two cases are looked at: (1) instrument optimization in analytical biochemistry and (2) on-chip synthetic biomolecule design; these are described in greater detail in Knowles (2009) as well as further references detailed later, along with other quite different examples. However, before these case notes, historical development and fundamental concepts in closed-loop optimization and CL-EMO will briefly be looked at.

3.1.1 Historic Highlights in Closed-Loop Optimization

In Berlin in the 1960s, Rechenberg, Schwefel, and Bienert conducted a series of studies in engineering and fluid dynamics, in which they tested the idea of using a process inspired by evolution to search for new and successful designs. Their work clearly demonstrated that complex design engineering problems (including: the optimal shape of a fluid-bearing pipe, and the design of a supersonic jet nozzle) could be addressed in this way with rampant success (see Chap. 8 of Fogel 1998, as well as Rechenberg 1965, 2000). The design process itself was found to be efficient and scalable, and the results were highly effective. Rechenberg and his team were using an early example of an evolutionary algorithm, but in which only the selection and variation steps were done by a microprocessor; the rest, the evaluation of candidate

designs, was done by constructing prototype designs and performing experiments to test their properties. Innovative solutions were found to all of the engineering design problems that they studied.

Predating Rechenberg's work, a similar principle was used by George Box, who introduced "evolutionary operation" (EVOP) in 1957. This was also an experimental method of optimization, which Box (1957) envisaged being used regularly in factories and similar processing facilities. Box's "closed-loop" scheme involved some human input, and was somewhat more deterministic than the approach taken in Berlin, but, just as Rechenberg's work, was inspired by principles from natural evolution. Box's methods were both successful and very influential (Hunter and Kittrell 1966), remaining in use today. Meanwhile, the work of Rechenberg's team was the beginning of the field of evolution strategies, one of the foundation stones of the current field of evolutionary computation.

Since these early studies, however, evolutionary computation as a whole has largely been concerned with entirely in silico optimization. The great majority of growth in this research area, as well as in industrial practice, concerns applications that involve convenient and entirely computational estimations of the fitness of computational abstractions of solutions. This is fine for a vast collection of scenarios, but there remains a need – in fact a quickly expanding one – for applications in which it makes sense for designs to be realized and evaluated physically throughout the simulated evolution process.

Research in evolvable hardware shows that, if the evolution process is given direct access to a complex physical structure, designs can be evolved that use entirely different principles than would be used by human designers, often exploiting aspects of the physics of the structures involved that are unfamiliar to human experts, or simply too difficult to use as part of the design process. Thompson and Layzell's work with field programmable gate arrays is exemplary of this. Meanwhile, evolutionary robotics projects have often relied upon the controllers being evolved in real time within physical robots, while they are performing real tasks in a real environment (Nolfi and Floreano 2004; Trianni et al. 2006). The benefits of such evolution experiments, exposed to and exploiting the true physics of the designs being evolved, are not just confined to evolutionary robotics (e.g., Davies et al. 2000; Evans et al. 2001).

Later, three further and recent uses of closed-loop evolutionary optimization are described, from recent work in which the third author (JDK) has been involved. Each of them is a scenario in which direct experimental evaluation of solutions is either the only option or is clearly preferable to simulation. Also each of them involves multi-objective evolution, a notable advance of the last 20 years (Fonseca and Fleming 1995; Coello 2000, 2006; Deb 2001; Corne et al. 2003a) which was not available to Box or Rechenberg. One of the several benefits of a multi-objective approach in these scenarios is that the different design objectives may simply be stated, without any need to define normalizations, weights, or priorities that mangle them into a single scalar (and usually misleading) measure of quality.

At this point, it is worth noting that there is widespread use of certain statistical methods in industry, for the types of problems that are considered in the "closed-loop" setting. The techniques employed are referred to as design of experiments (DoE) approaches, or sometimes experimental design (ED)-based approaches (Fisher 1971; Chernoff 1972; Myers and Montgomery 1995; Box et al. 2005). Such methods emphasize rational reasoning from all the information obtained so far, as opposed to more randomized exploration. Standard DoE is typically used for probing low-dimensional parameter spaces using few experiments, while evolutionary algorithms are typically used for optimization in high-dimensional spaces, using many evaluations, and optimizing many different types of structure, including permutations,

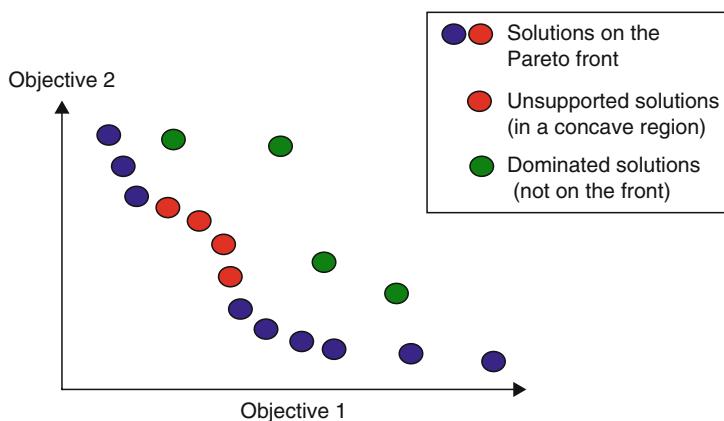
graphs, networks, and so on. However, there is an increasingly disappearing divide between the two types of approach, especially since the advent of sequential DoE, which incorporates aspects of evolutionary computing. The closed-loop optimization scenarios considered in this section lie between these niches, and benefit from aspects of both approaches.

3.1.2 Fundamentals of Closed-Loop Evolutionary Multi-objective Optimization

In closed-loop EMO, candidate solutions to a problem are generated by an algorithm in computer simulation, but their evaluation is achieved by physical experiment. Evaluations are fed back to the algorithm and its generation of subsequent solutions is a function of these. Thus, the process has the form of a closed loop, being at least partially sequential. Closed-loop problems can be defined generally as multi-objective optimization problems in which, essentially, one needs to find some ideal solution vector \mathbf{x} , which simultaneously minimizes each of a collection of k objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})$. Typically, a single physical experiment $g(\mathbf{x})$ yields the k measurements $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})$. That is, the k objectives are k different measurements that are made as the result of a single experiment, all of which need to be optimized in some way. Typically, at least some of the objectives will be in conflict (Brockhoff and Zitzler 2006), and no single solution is a minimizer of all functions. Rather, the improvement of one objective is only possible by sacrificing, or trading off, quality in some other objective. The solutions corresponding to optimal values of the k objectives are known as the Pareto set, and when plotted in objective space, form the Pareto front (see [Fig. 7](#)).

Fig. 7

An illustration of a Pareto front for a typical optimization problem with two objectives, both of which have to be minimized. Each of the solutions on the Pareto Front (PF) are optimal in the Pareto trade-off sense. For example, for any solution on the PF, no solution exists that is improved in one objective without being degraded on another objective. Often, some solutions on PFs are “unsupported” – these are valid optimal solutions in the Pareto trade-off sense, but for any linear combination of the objectives that might be used in a single-objective simplification of the problem, they would not be optimal.



Since solving such a vector optimization problem usually leads to a set of solutions, rather than a single one, there is, in most practical applications, a need for decision making to select one solution from this set. This aspect of multi-objective optimization is important and well studied (Fonseca and Fleming 1998; Miettinen 1999; Branke and Deb 2005), and the various alternative approaches will not be covered here. Suffice to say that in the experiences detailed later, the EMO algorithms were designed to find whole Pareto fronts, with the expectation that a human decision maker would make the final decision using the information incorporated in the output Pareto front.

3.1.3 Example 1: Instrument Optimization in Bioanalytical Chemistry

Modern biotechnology and bioanalytics often involve large-scale experiments which impose heavy demands on sophisticated laboratory instruments. To achieve timely throughput, these experiments often necessitate using configurations of instruments that go beyond the manufacturer's recommended settings. This situation happened in the "HUSERMET" project, which was a collaboration between several UK health authorities, two pharmaceutical companies, and the University of Manchester, undertaken between 2006 and 2009 (www.husermet.org). In this project, human blood samples were collected from around 2,000 people over a 3 year period, with the aim of understanding ovarian cancer and Alzheimer's disease in terms of the variations in metabolites (the chemical products of metabolism) present in patients suffering from, or free from, these diseases. The samples were analyzed with the help of various modern technologies for characterizing complex samples, including laboratory instruments that performed gas chromatography mass spectrometry. The configuration of such instruments is always subject to a degree of optimization in order to ensure that the analytes being detected can be seen, with maximal sharpness and minimal noise. This optimization is usually (though not always) *ad hoc*, subject to much domain knowledge.

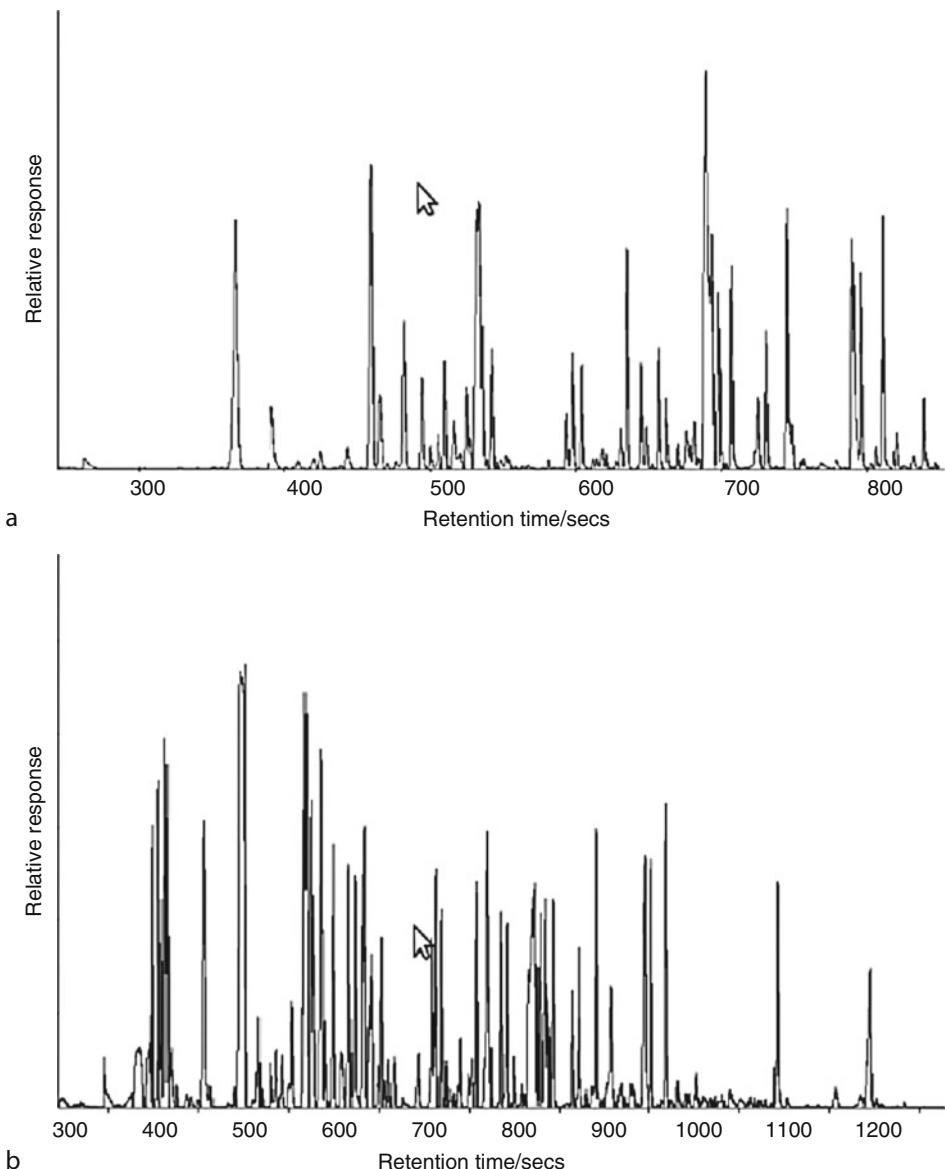
In the HUSERMET project, the need for a better instrument configuration optimization process arises from: the unusually large number and diversity of metabolites to be detected (around 2,000), the potential to vary around ten interacting instrument parameters, and the significantly conflicting nature of the optimization objectives. Instrument settings were needed that allowed fast processing of samples (preferably well under an hour), which conflicts with the desire to maximize the detection of the full complement of metabolites at low noise.

Optimizations of two instruments have been reported in detail in O'Hagan et al. (2005) and O'Hagan et al. (2007), respectively. The former study successfully used the evolutionary multi-objective algorithm PESA-II (Corne et al. 2001), but that study also directly inspired the development of the ParEGO algorithm (Knowles 2006), a multi-objective algorithm that is a hybrid of a surrogate modeling approach and an evolutionary algorithm. ParEGO was then used in the second study successfully, and settings derived from the evolutionary algorithms in both cases were subsequently used for the instruments to process the tasks of the HUSERMET project.

The major challenge in that project was the limited number of function evaluations that could be done. A function evaluation ties up an expensive instrument for an appreciable time, when it could otherwise be used more directly furthering the project's needs. This was even more bothersome, given the need to try to optimize three objectives simultaneously (chromatogram peaks, signal/noise ratio, and sample throughput). Only one instrument was available, and a single analysis of a serum sample takes between 15 min and over 1 h. The optimization process used 400 evaluations in total, with the EAs controlling the instrument

Fig. 8

Chromatograms indicating detection performance of the instrument optimized in the HUSERMET project: (a) from the initial generation of search; (b) toward the end of the search process. In (b), both the number of peaks and the range of retention times over which peaks are detected have improved, while maintaining noise at low levels.



settings and loading samples through a robotic interface that was designed especially for the optimization process. In **Fig. 8** some chromatograms can be seen, which indicate the instrument's performance characteristics before (*upper*) and after (*lower*) optimization. The optimized result achieved approximately threefold increases in the quantity of peaks

visible, while at the same time maintaining the signal/noise ratio at low levels, and achieving throughput of samples in around 20 min.

3.1.4 Example 2: Evolving Real DNA on Custom Microarrays

Another example covered in more detail in Knowles (2009) concerns the design of pharmaceutically active, highly targeted macromolecules. This is a significant goal in modern medicine, especially in the context of *ab initio* design, where one seeks a molecule with specific properties and activity, but has little or nothing to go on (in the sense of existing molecules with similar properties). In recent research, novel microarray-based technology has been used in the automation of such *ab initio* molecule design. Experimental biotechnology platforms are now available, which can synthesize, and then experimentally test in a variety of ways, any specified DNA sequence. Being able to synthesize any given sequence, and subject it various tests, means there is far less need for computational models which, in the current state of the art, are far from good enough (or fast enough) to support such a process.

The microarray used in the work described next (and more fully in Knowles (2009) and references therein) is the so-called CustomArray technology, available from Combinatrix Corp, which can be used to synthesize up to 90,000 specific bespoke DNA sequences of up to 40 bases long in a single experiment. Once the sequences have been synthesized, they can be tested for a variety of properties, but usually the main property of interest is the ability to bind them to a particular target molecule. In the testing (or assay) process for binding ability, the chip holding the sequences is “washed” with a solution containing the target molecule, and some form of fluorescent tagging is used so that binding can be observed; further automated processes can then estimate the strength of binding.

Short strands of DNA (or RNA) that bind strongly to specific targets are called *aptamers*, and hundreds of these have been developed for a wide variety of applications. Before the recent microarray-based work at Manchester (which is what is discussed here, with full details in Knight et al. 2008), new aptamers were almost always discovered by a method called SELEX (Tuerk and Gold 1990), or in vivo selection, in which the DNA strands are evolved in a test tube by repeated rounds of high-pressure selection and random mutation. As indicated, however, in the microarray approach one knows precisely the sequence information for every sequence tested, and can even exactly specify mutations or other variations to perform. This is not the case in SELEX, and one of the many benefits for the microarray approach is that it allows extremely richer possibilities for borrowing and exploiting algorithms from evolutionary computation, machine learning, and statistics.

Knight et al. (2008) reports the first use of an evolutionary algorithm to produce a DNA aptamer on the B3 Combinatrix platform. This happened after ten generations of evolution, eventually discovering several 30-base long strands that bound very strongly to the target molecule, allophycocyanin. The work in Knight et al. (2008) used a DNA chip that could hold 6,000 strands. With 90,000 strands on a chip now possible in more modern technology, one main challenge (from the optimization perspective) is to determine the best way to exploit such massive population sizes. Wedge et al. (2009) have recently explored such questions in silico simulations using contrived search landscapes, as well as real trials on the DNA landscape, revealing, among other findings, that higher-than-standard mutation rates consistently outperformed a range of other setups. This echoes findings in Corne et al. (2003b), which also explored large population sizes

and contrived in silico landscapes, partly to inform the (as then) emerging field of closed-loop protein evolution.

3.1.5 Some Concluding Notes on CL-EMO

For the examples described above, and several more in which CL-EMO has been used, building accurate computational models that could usefully replace real experiments is practically infeasible. The closed-loop alternative offers a more efficient and effective way toward the discovery of innovative solutions, easily making up for the time and expense of tying down the physical kit for the experimental period. One question often worth asking, however, is whether one needs to automate the optimization process at all in such scenarios. There is a processing step in which a computational process (here, e.g., an evolutionary multi-objective optimization algorithm) considers the latest experimental evaluation results, and outputs sample designs for the next sequence of physical evaluations – but this operation could easily be done instead by a domain expert. On the other hand, though, there are several objections to such human involvement: even experts can over-interpret results that are affected by noise or similar factors; similarly, humans are very prone to reason on the basis of simple models, ignoring interactions between parameters. Meanwhile, there is always a very real danger of experts preferring solutions that are (or are close to) known designs.

Problems where accurate computer modeling is infeasible, and for which closed-loop optimization is the efficient solution, are really quite common. For the moment, the main focus of the third author is on problems in modern biology, where there is a growing take-up of multi-objective optimization. Meanwhile, many other substantial areas are able to benefit greatly from CL-EMO; apart from drug discovery and development, large-scale problems such as flood defense design, forest fire control strategies, the location of renewable energy plants, and the task of genetically engineering more pest-resistant food crops and energy crops, can all be seen, to varying degrees, as closed-loop problems.

3.2 Innovation

This section describes a new idea, *innovation*, introduced in Deb and Srinivasan (2005, 2006), which (typically) exploits multi-objective evolutionary computation to find new and innovative design *principles*. Although optimization algorithms are routinely used to find an optimal solution corresponding to an optimization problem, the task of innovation stretches the scope beyond an optimization task and attempts to unveil new and innovative design principles relating to decision variables and objectives, so that a deeper understanding of the problem can be obtained.

Innovation is a common goal for engineers and designers, but there are actually very few (arguably no) systematic procedures for reliably achieving innovations. Goldberg (2002), however, suggests that a “competent” genetic algorithm can be an effective way to achieve an innovative design (and indeed there are numerous examples of innovative designs being discovered by evolutionary computation, including some discussed elsewhere in this chapter). However, the idea of innovation (Deb and Srinivasan 2005, 2006) extends this argument considerably, and gives a systematic procedure that can arrive at a deeper understanding of a given engineering design problem. This systematic procedure may lead to the discovery of new design principles – in particular, principles that are common to the diverse collection of optimal trade-off solutions.

Such common principles may, in many cases, provide a reliable recipe for solving given instances of the problem at hand. In this section, the innovation procedure will be explained and illustrated with two examples in engineering design. The material in this section borrows much from Deb and Srinivasan (2005), which introduces this idea, and contains several more examples. However, before looking at the procedure and examples, it will be helpful to recall some basics about the usually conflicting nature of objectives in the design process.

3.2.1 Multiple Conflicting Objectives in Design

The central idea in innovation involves the presence of at least two conflicting objectives for the design problem at hand. This is far from a limiting constraint – as argued in many places (see in particular Corne et al. 2003a for an introductory account of this argument), almost all realistic problems naturally involve several objectives.

Consider a typical design problem with two or more conflicting goals, such as an engine or generator whose mass needs to be minimized, but whose output needs to be maximized. Such a two-objective optimization task results in a set of Pareto-optimal solutions (see Fig. 7). One of the “extreme” solutions will be the best if the interest is only in mass, while the other extreme solution will be ideal for the output consideration, and there will usually be several solutions in between these extremes, also optimal in a sense, all of which share the property that if they are better than another Pareto-optimal solutions in one objective, they will be worse in the other. The intermediate solutions are invariably good compromises within the extremes, and the solution that may eventually be chosen by the designer will often be among these, and its choice will often be helpfully informed by the knowledge that the designer obtains by viewing the shape and the nature of the trade-offs displayed by the entire set of solutions that form the discovered Pareto front. However, what is of particular interest here is that this set of solutions will typically be very diverse, but all sharing the property of Pareto optimality. The idea of innovation arises from the attempt to see whether this property of Pareto optimality, for any given problem, is manifest in concrete features that the diverse solutions share. Another aspect of this is that the process of obtaining such a wide variety of solutions is itself a significant investment in computation time; *innovation* is a way to exploit this significant investment by performing a posterior analysis of the obtained set of trade-off solutions that may result in a set of “innovized” principles relating to the given design problem.

In designing an electrical motor, for example, this posterior analysis might reveal a feature of the diameter of a certain component and the power output that is shared by all of the Pareto-optimal solutions, but not other solutions. Any such relationships discovered would clearly be of great importance to a designer, and perhaps point toward a recipe for future design tasks in the same domain, as well as spark new theoretical insights into the problem. These are just two of a range of benefits that the so-called “innovized principles” could lead to, as discussed further in Deb and Srinivasan (2005), along with a convincing argument that one can often expect such principles to exist.

3.2.2 How to Innovize

The innovation procedure proposed by Deb and Srinivasan (2005) consists of two phases: in the first phase, the idea is to simply try to obtain the Pareto-optimal solutions of the design problem in question. In the second phase, they then analyze the solutions and extract innovized

principles. The first phase is not as straightforward as it sounds, since, of course, it can usually never be guaranteed that a true optima has been found for a realistic problem, unless an exhaustive search has been performed. However, the idea of the first phase is to do as well as one can in a reasonable time, since it is expected that the chance of obtaining valid principles of Pareto optimality is improved if one has true (or very close to true) Pareto-optimal solutions. In Deb and Srinivasan's procedure, this centrally involves making use of NSGA-II (Deb et al. 2002) (one of the most prominent and effective evolutionary multi-objective optimization algorithms) as the main engine in finding the Pareto front, but initially informed by a single objective method that has been used to find the extreme points on the Pareto front, and followed by various applications of a local search method and the normal constraint method (NCM) (Messac and Mattson 2004) to locally improve the output solutions from NSGA-II, as far as possible.

The second phase of innovization is then the analysis of the assumed Pareto-optimal solutions that emerge from the first phase. There is no fixed recipe for this process, other than to employ the usual common sense and expertise that underpins a data mining and knowledge discovery task in searching for commonality principles among these solutions that may become plausible innovized relationships. Deb and Srinivasan (2005) also pursue "higher level innovations" after this phase, which involve returning to the original problem, but investigating different areas of the design space by looking at neighboring problems (e.g., with different boundaries and constraints on the design task); this then enables new principles to be discovered that are likely to be at a higher level than previously, mapping design constraints to design recipes.

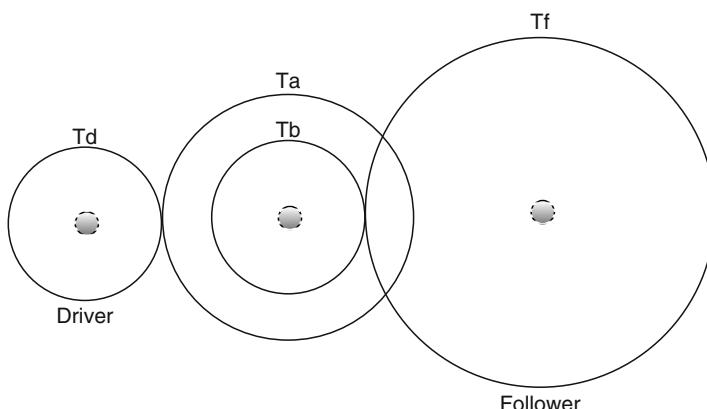
Just two from the increasing collection of results of this above innovization procedure in engineering design applications are now described. These were first described in Deb and Srinivasan (2005), among several other examples.

3.2.3 Example 1: Gear Train Design

Deb and Srinivasan (2005) give the example of the design of a compound gear train, in which a specific gear ratio between the driver and driven shafts is desired. The problem is illustrated in Fig. 9, and is a modification to a problem solved elsewhere (Kannan and Kramer 1994;

Fig. 9

A gear train with four gears (circles). The task is to achieve as close as possible a gear ratio of 6.931:1 between the driver and follower, while minimizing the sizes of each gear.



Deb 1997). The objective is to set the number of teeth in each of the four gears in a way that minimizes the error between the obtained gear ratio and a required gear ratio of 6.931:1, while also minimizing the maximum size of the four gears.

The diameter of a gear is proportional to the (integer) number of teeth, so these objectives can be formalized in terms of four integer decision variables: $\mathbf{x} = (x_1, x_2, x_3, x_4)$ referring, respectively, to the numbers of teeth in gears T_d (driver), T_b , T_a , and T_f (follower). The problem is then to minimize both f_1 and f_2 below:

$$f_1(\mathbf{x}) = \left| \frac{6.931 - \frac{x_3, x_4}{x_1, x_2}}{6.931} \right|$$

$$f_2(\mathbf{x}) = \max(x_1, x_2, x_3, x_4)$$

subject to the following constraints:

$$\frac{f_1(\mathbf{x})}{6.931} \leq 0.5,$$

$$12 \leq x_1, x_2, x_3, x_4 \leq 60$$

The constraints ensure that the difference between the designed gear ratio and the desired gear ratio is no more than 50%. After phase one of the innovation procedure, a collection of assumed Pareto-optimal solutions was obtained.  [Table 4](#) shows the two extreme solutions.

Phase two of the process then revealed several interesting principles relating to the problem, covering the whole set of Pareto-optimal solutions, which is summarized from Deb and Srinivasan's account as follows:

First, in order to minimize the maximal gear size, gears T_d and T_b need to have almost the smallest allowable number of teeth. To get as close as possible to the desired ratio (with error less than 0.1), the T_b and T_d values need to grow somewhat, but still remain close to their lower bounds. Another finding is that the maximum allowed gear size always occurs in a Pareto-optimal solution, either for T_a or for T_f . It is also noted that two distinct types of solutions emerged: (1) gear trains with very low error (very close to the desired gear ratio of 6.931:1), in which there is a great variety of ways in which the numbers on teeth in the four gears combine to almost achieve the 6.931 ratio in the first objective and (2) gear trains with a comparatively large error, with identical first- and second-stage ratios (except the one with the largest error). Although a large error can happen for many different combinations of errors in the two stages, the pressure of the second objective causes both stages of gear-ratios to be identical. Finally, regarding small-error gear trains, half of them have a larger first-stage ratio than second stage, and half have a larger second-stage ratio.

This is a fairly simple and straightforward example, although it brings out several interesting properties of optimal solutions of this type of gear train design problem that are difficult, if not impossible, to infer from the statement of the problem. One implication, for example, concerning recipes for gear train design, is that the process could be guided according to how important it is to closely meet the constraint. If a low error is desired, then it is clearly important

 [Table 4](#)

The extreme solutions obtained for the gear train design problem in Deb and Srinivasan (2005)

Solution	T_d	T_b	T_a	T_f	f_1	f_2
Minimum error	20	13	53	34	~0.00023	53
Minimum maximal gear size	12	12	22	23	3.4171	23

to examine many possible combinations of gear sizes. If a higher error can be allowed, then solutions with minimal size strongly tend to have equal first-stage and second-stage ratios.

3.2.4 Example 2: Welded Beam Design

This is a much-studied problem in the context of single-objective optimization (Reklaitis et al. 1983), in which a beam needs to be welded onto another beam and must carry a certain load F , as illustrated in  Fig. 10.

The problem is to establish the four design parameters (beam thickness and width, respectively, b and t ; length and thickness of weld, respectively, l and h) in a way that minimizes the cost of the beam, and also minimizes the vertical deflection at the end of the beam. The overhanging portion of the beam has a fixed length of 14 in., and is subject to a force F of 6,000 lb. Clearly, an ideal design in terms of cost will be less rigid and hence not ideal in terms of deflection, and vice versa. A formulation of this problem (Deb and Kumar 1995; Deb 2000) gives the objectives as follows, where x indicates the vector of design parameters:

$$f_1(x) = 1.10471 \cdot h^2 \cdot l + 0.04811 \cdot t \cdot b(14.0 + l)$$

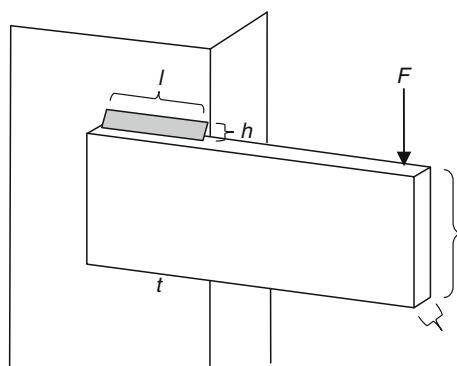
$$f_2(x) = \frac{2.1952}{t^3 \cdot b}$$

subject to these constraints:

$$\begin{aligned} \tau(x) &\leq 13,600 \\ \sigma(x) &\leq 30,000 \\ b &\geq h \\ P_c(x) &\geq 6,000 \\ 0.125 &\leq h, b \leq 5.0 \\ 0.1 &\leq l, t \leq 10.0 \end{aligned}$$

The first constraint specifies that the shear stress at the support location is below the allowable shear stress of the material (13,600 psi); the second ensures that the normal stress at the same location is below the material's allowable yield strength (30,000 psi); the third ensures the

 **Fig. 10**
The welded beam design problem.



obvious practical consideration that the weld is not thicker than the beam, and the fourth ensures that the applied load F is below the allowable buckling load of the beam. The highly nonlinear stress and buckling terms are as follows (Reklaitis et al. 1983):

$$\begin{aligned}\tau(x) &= \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}} \\ \tau' &= \frac{6,000}{\sqrt{2hl}} \\ \tau'' &= \frac{6,000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2(0.707h \cdot l(l^2/12 + 0.25(h+2)^2))} \\ \sigma(x) &= 504,000/t^2 b \\ P_c(x) &= 64,746.022(l - 0.282346t)tb^3\end{aligned}$$

Following phase one of the innovation procedure, a set of Pareto-optimal solutions was obtained, and [Table 5](#) shows the extreme points of this Pareto set, along with an interesting intermediate point which Deb and Srinivasan refer to as T , which comes into the innovized principles discussed next.

Deb and Srinivasan's analysis of the many Pareto solutions obtained (spread liberally between those shown in [Table 5](#)) revealed the following innovized principles:

First, two distinct behaviors were found: from the intermediate transition solution T (shown in [Table 5](#)) toward higher-deflection solutions, the objectives behave differently than in the rest of the trade-off region. For small-deflection solutions, the relationship between the objectives was almost polynomial, with f_1 being roughly proportional to $1/f_2^{0.89}$. Next, it was found that, for all Pareto-optimal solutions, the shear stress constraint is active. In the small-deflection (large-cost) cases, the chosen bending strength (30,000 psi) and allowable buckling load (6,000 lb) are quite large compared to the developed stress and applied load. Any Pareto-optimal solution must achieve the maximum shear stress value (13,600 psi). So, to improve the designs in this region without sacrificing deflection, it would be necessary to use a material with a larger shear strength capacity. A third overall principle found was that the transition point (T) between two trade-off behaviors is related closely to the buckling constraint. Designs with larger deflection (or smaller cost) reduce the buckling load capacity. When buckling load capacity becomes equal to the allowable limit (6,000 lb), no further reduction is allowed. After this point (toward small-deflection solutions), the beam thickness must reduce in inverse proportion to the deflection objective in order to retain optimality.

Table 5

The two extreme solutions and an interesting intermediate solution T obtained by Deb and Srinivasan (2005) for the welded beam design problem. The units of the design parameters are inches

Solution	H	I	T	b	f_1	f_2
Minimum cost	0.2443	6.2151	8.2986	0.2443	2.3815	0.0157
Minimum deflection	1.5574	0.5434	10.000	5.000	36.4403	0.00044
Intermediate solution T	0.2326	5.3305	10.000	0.2356	2.5094	0.0093

Next, for small-deflection solutions, the beam width remains constant. This indicates that for most Pareto-optimal solutions, the width must be set to its upper limit. Although the beam width has opposite effects on cost and deflection, it is involved in the active shear stress constraint, and since shear stress reduces as beam width increases, it can be argued that fixing beam width to its upper limit would make a design optimal. Thus, if in practice the cost objective is not paramount, solutions which have a fixed width (the maximum 10 in. in this case) may be explored, thereby simplifying the inventory. However, along the Pareto trade-off surface, the weld length increases with increasing deflection, and the weld thickness decreases with increasing deflection. Deb and Srinivasan (2005) noted that these phenomena are counter-intuitive and difficult to explain from the problem formulation. However, the innovized principles for arriving at optimal solutions seem to be as follows: for a reduced cost solution, keep beam width t fixed to its upper limit, increase weld length l , and reduce beam and weld thickness (h and b). This “recipe” is valid while the applied load is strictly smaller than the allowable buckling load.

Beyond that point, any reduction in cost must come from reducing beam width below its upper limit, increasing beam thickness, and adjusting the weld parameters so as to make the buckling and shear stress constraints active. Finally, the minimum cost solution occurs when the bending stress equals the allowable strength (30,000 psi), at which point all four constraints become active.

Finally, to achieve very low-cost solutions, the innovized principles are different: for a reduced cost solution, one needs a smaller beam width, but larger beam thickness and weld parameters.

Deb and Srinivasan report a higher level run of the innovation procedure for this case, in which innovation was redone separately for different values of the three allowable limits in the first, second, and fourth constraints above. It was clear that all three cases produced similar dual behavior (different characteristics on either side of a single transition point) to that observed in the original case. All other innovized principles mentioned above (such as the constant nature of beam width, beam thickness being smaller with increasing deflection, and so on) remained valid. And significant further insights were obtained into the overall design problem, detailed in full in Deb and Srinivasan (2005).

3.2.5 Innovization: Concluding Notes

When facing an optimization problem with at least two conflicting objectives, the set of optimal solutions is very diverse. Having found such a set effectively and efficiently using evolutionary multi-objective optimization (judiciously combined with other methods that help locally optimize), the notion of *innovization* is to analyze this set of solutions to see if there are commonalities and patterns that might translate into general design principles for the problem at hand. It turns out that this is true, and interesting new principles (difficult or impossible to have been obtained otherwise) have emerged from several studies to date. The emerging truth seems to be that solutions along a Pareto front often seem to share similarities that seem to be principles of optimality for the problem at hand, irrespective of location on the Pareto front.

In this section, just two case-study results have been borrowed to illustrate the innovization principle. Deb and Srinivasan (2005) show several more examples, including spring design and multiple-disk clutch brake design, while one more recent study (Datta and Deb 2009)

displays an excellent example of the potential impact of innovation (and, hence, indirectly, of evolutionary multi-objective optimization) by finding new innovized principles for the setup parameters of a turning process using a lathe and a cutting tool that are overwhelmingly common in industry workshops. Finally, there is no particular reason to believe that innovation is constrained to engineering design. It will be interesting to see future applications of this idea in other design fields, such as electrical circuits, optical systems, communication networks, and the many other areas in which evolutionary multi-objective optimization is increasingly used.

4 Logistics and Combinatorics Made Easy: Robust Solutions and New Algorithms via Natural Computation

In this section, two areas will be considered that exemplify how natural computing (largely, learning classifier systems and evolutionary computation) has provided highly successful ways to address difficult logistics problems. Logistics usually relates to scheduling and timetabling problems of various kinds, but also included here is the closely related and general field of combinatorial problems in which a discrete collection of items of some kind must be arranged in an optimal way. There are innumerable examples of natural computing applications in this domain, and the first case here is simply a selection of one (of several possibilities) that combines the attributes of “interesting,” “real-world,” and “difficult” (looking here at the case of a real-world truck scheduling problem). Next, moving on to perhaps a more profound area that has emerged from the late 1990s, in which, rather than the use of evolutionary computing to solve “one problem at a time,” the use of natural computing to discover new *algorithms* is considered, which can then in turn be used on entire classes of problems, solving them efficiently and effectively. This is an area within the emerging field of “hyper-heuristics,” but with the particular focus on designing new algorithms which is referred to as “super-heuristics.”

4.1 Safe Streets via Robust Route Optimization

This section describes an application of natural computation in a critical seasonal logistical task, covered more fully in Handa et al. (2006). Local authorities in countries such as the UK, with marginal winter climates, are responsible for the precautionary gritting/salting of the road network in order to allow safer travel in icy conditions. This winter road maintenance task is extremely challenging as well as critically important to the locality, with a potentially major impact on both business and day-to-day life.

As Handa et al. (2006) note, in the case of the UK there are around 3,000 precautionary gritting routes that cover about 120,000 km (30% of the entire UK road network). On nights with forecasted snow or ice, these routes need to be treated so as to ensure the safety of road users. This typically costs between £200 and £800 per km of road (Cornford and Thornes 1996). Accurate road surface temperature prediction is required, in order to decide which roads need to be treated, however this decision can often be uncertain. Optimization of the route to be traveled by the gritting/salting trucks also plays a crucial role here. The consequences of a wrong decision – not treating a road that eventually becomes dangerous – are serious, but if grit or salt is spread when it is not actually required, there are obvious financial and environmental drawbacks. The goal of gritting route optimization is to minimize the

financial and environmental costs, while ensuring that roads that need treatment will be gritted in time. Further, it is essential that gritting routes are planned in advance, to enable effective use of limited resources (e.g., trucks and salt).

Mostly, the design of gritting routes relies heavily on local knowledge and experience. A “static,” often paper-based approach is typically used to optimize gritting routes, staying within constraints imposed by the road network itself, vehicle capacities, the number of vehicles, and the available personnel. This section describes the application of an evolutionary algorithm to this task. Covered in more detail in Handa et al. (2006), discussed here is a Salting Route Optimization (SRO) system that combines evolutionary algorithms with the latest version of the Road Weather Information System (XRWIS) commonly used by local authorities.

4.1.1 The Salting Route Optimization (SRO) System

A very important aspect of the SRO discussed here is its integration with XRWIS, which, recently trialed by the UK Highways Agency, is a high-resolution route-based forecast system that predicts road temperature for a 24-h period. XRWIS models surface temperature and condition at thousands of sites in the road network. Data are collected along each gritting/salting route by conducting a survey of the “sky-view factor” (a measure of the degree of sky obstruction by buildings and trees) (Chapman et al. 2002). This is then combined with other geographic, land use, and updated meteorological data to predict road conditions at typical spatial and temporal resolutions of 20 m and 20 min, respectively. The output is displayed as a color-coded map of forecast road temperatures and conditions that is then disseminated to highway engineers.

In the SRO, XRWIS provides forecast temperature distributions over time that are then input to an evolutionary algorithm module. Each temperature distribution (different distributions for different future timepoints), along with commercially available routing data, is transformed into an instance of a capacitated arc routing problem (CARP) (Lacomme et al. 2004). That is, each temperature distribution suggests a specific set of roads on the networks that need to be treated. The CARP is then defined as the need to find routes that serve this specific set of roads in a reasonable time frame, using no more than the available vehicle numbers and capacities, and ideally minimizing the number of vehicles used. An important point is that each timepoint leads to a different CARP instance, since the set of roads that require treatment may be different. The overall goal of the SRO system is not only to find a suitable series of salting routes, which ensure that the roads that require treatment are treated in time, but also to ensure that the routes do not vary too much, which in turn causes considerable confusion and distraction to the workforce. In this sense, the SRO system finds a *robust* solution, which ensures we can cover the most important sections of the road network.

Given the series of CARP instances, the evolutionary algorithm module finds solutions that are simultaneously good for all or many of these instances. In particular, a specially designed memetic algorithm is used (a combination of evolutionary and local search) as described next. In this approach, the fitness of a solution is calculated according to the entire ensemble of CARP instances. However, at each generation, the operators and local search processes concentrate on a specific instance. The different instances are weighted, and this weighting controls the selection of the instance in each generation, in a way described in the following.

4.1.2 Robust Solutions for Salting Route Optimization

Searching for robust solutions is currently a significant topic in the field of optimization in uncertain environments, since in many problems the decision variables or environmental parameters are subject to noise. In this case, Handa et al. (2006) required that solutions to the different CARP instances be as similar to each other as possible (so that daily changes in the temperature distribution do not lead to significant disturbance in the route to be followed), while at the same time requiring good performance in terms of the costs of the routes. Handa et al. modeled a robust SRO solution as one which optimized the following:

$$F(X) = \int E(X, a)p(a)da$$

in which X and a indicate route design variables (routes and possible temperatures), $E(X, a)$ indicates the distance cost of gritting routes X given temperature a , while $p(a)$ indicates the probability of temperature a . Hence, the idea is to find ideal gritting routes for each temperature distribution, but weighted by the prior probabilities of the forecast temperatures.

Although the distribution in temperature will vary daily across a road network, warmer (colder) sections are usually warmer (colder) than the rest of the network. So, even on cold nights, some warmer sections may not require salting, whereas colder sections may need treatment even in relatively warm conditions. The fitness function, as stated above, is impossible to compute exactly since its components are largely unknown; instead it is approximated by using a number of typical temperature distributions. Considering this and other issues, the fitness function used by Handa et al. (2006) was as follows, given a set of temperature distributions A_e :

$$F(X) = \sum_{a \in A_e} w_i \frac{E(X, a_i) - E^*(a_i)}{E^*(a_i)}$$

in which $E^*(a_i)$ represents the difficulty of finding a good route for temperature distribution a_i , and the w_i are weights, summing to 1, which balance the importance of different temperature distributions during the optimization process. The weights are adapted during evolution in a way that maintains a focus on the routes that are proving more costly, while the $E^*(a_i)$ values are lower bounds on the cost of the routes for each temperature distribution a_i , actually predetermined by prior runs of the memetic algorithm for this purpose described in Handa et al. (2005).

Handa et al. (2006) used a permutation-based encoding as follows. An individual solution comprised a permutation of arc IDs (road sections), interspersed with symbols representing individual trucks. For example, the individual:

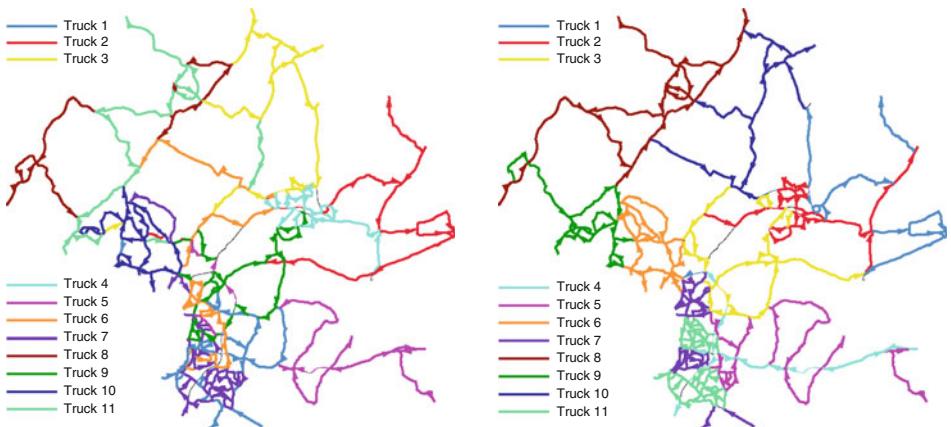
2 6 s1 5 4 7 1 s2 8 3

indicates a gritting route for two trucks; truck 1's route is road sections 5, 4, 7, and 1 (in that order), and truck 2's route is road sections 8, 3, 2, and 6 (note the wraparound involved in the interpretation).

At each generation of the memetic algorithm, crossover (the EAX operator proposed by Nagata and Kobayashi 1997) and local search methods are applied with regard to only one CARP instance (i.e., one temperature distribution) in every generation. That is, for example, the local search is guided by the fitness according to the selected instance only. The choice of instance is made stochastically according to the current weights of the temperature distributions. However, between generations, the fitness of each solution is calculated according to the

Fig. 11

Left: routes optimized by the SRO system for a cold day in South Gloucestershire; Right: existing routes obtained by human experts.



ensemble of instances using the fitness function described, and this then guides the selection of parents for the next generation.

4.1.3 Comparisons and Conclusions

In experiments by Handa et al. (2006) to test and validate this approach, robust solutions were evolved by using ten different temperature distributions, and these were then compared with the routes currently used by South Gloucestershire Council in the UK.

• *Figure 11* shows an example of routes found for a cold day, comparing the SRO system's routes (on the *left*) with the existing routes (on the *right*).

In comparison with the routes that were in use at the time, the robust solutions delivered by the SRO were able to provide more than 10% savings in terms of total distance traveled by the available trucks.

The SRO system was developed for finding optimized robust solutions for salting trucks, and as such it is an excellent example of an important real-world combinatorial problem that can be solved effectively via a system with natural computation at the core. In this case, especially given the integration with the XRWIS, the system can be regarded as proof of concept for similar tasks that need careful planning in relation to weather conditions, such as waste collection and parcel delivery.

4.2 Hyper and Super Heuristics

This section briefly considers a fairly new method in search and optimization, variously called hyper-heuristics or, as an emerging term in the community referring to a specific brand of approach, super-heuristics. In the context of selected applications of natural computation, the special aspect of super-heuristics is that they represent the use of a good global optimization or

learning method – hence, typically evolutionary computation or a learning classifier system – to discover new *algorithms* that solve problems of a given kind. This is as opposed to, and substantially more general than, using optimization or learning to solve a single problem instance.

In very broad terms, the general notion of hyper-heuristics refers to the idea of using an algorithm that manipulates a set of heuristics in order to solve a given problem. This is indeed a very common activity these days, and can be seen in many published applications of evolutionary algorithms, and of meta-heuristics in general (including, e.g., tabu search and simulated annealing). Typically, such an approach is sometimes called hyper-heuristic in the case that the encoding used involves lower-level heuristics in an integrated way. That is, rather than an encoding of a solution being a direct representation of a solution, the encoding is instead an indirect representation (examples will be looked at later). The interesting point is that, in some cases, an encoded solution for a given problem can actually be interpreted as an algorithm that can be applied to a large collection of instances of that problem, not just the instance currently being solved. In many so-called “hyper-heuristic” applications, this reusability of the encoding of a solution is only a side effect. When the term “super-heuristic” is used herein, this is meant to refer to the idea that evolving new general and reusable algorithms (for classes of instances, rather than a single instance of interest), is the specific goal of the process. However, it is noted that “super-heuristics” seems to have been first used in the literature by Lau and Ho (1999), to denote something more akin to standard hyper-heuristics, in which a set of heuristics are engineered by higher level algorithms in order to solve a specific problem instance. This section describes the ideas, amid some examples and historical notes. For alternative and more detailed accounts, a 2003 book chapter is recommended (Burke et al. 2003), or the current Wikipedia article on hyper-heuristics.

4.2.1 Potential Impact of Super-heuristics

The impact and importance of super-heuristics is partly evidenced by a negative point: despite a large collection of case studies, standard applications of meta-heuristics tend to be “one-off” and resource-intensive. For example, a particle-swarm optimization method developed to solve a small company’s daily process scheduling problems may seem successful in its own terms, but its existence does not necessarily accelerate the potential for other companies with similar (but not the same) problems to develop similar solutions. And, typically, the solution itself may be resource intensive, tying up considerable computing resources every morning. Also, the development of this solution will have typically been influenced by a perceived goal to produce solutions as optimal as possible, despite the fact that daily uncertainties and perturbations to the production process underline this optimality – that is, a large number of “reasonably good” solutions will have worked just as well.

In contrast, super-heuristics seem to open up the possibility for producing solutions that, though having an initial cost in development time, are much more flexible. The “solution” in this context would be a fast, constructive algorithm that tends to work well (as well as run much more quickly than a typical meta-heuristic implementation) on the problems typically faced by the company, and may well generalize to similar problems more successfully and easily than the meta-heuristic approach.

4.2.2 Hyper-Heuristics: Further Notions and Examples

Suppose there is an instance of a problem to solve. In particular, it is easier to think in terms of combinatorial and logistics problems, the kind in which a solution might be built step by step by making a series of decisions. For example, if a collection of student examinations have to be timetabled, first a room and a time for the largest exam might be found; then which exam to look at next might be decided, then where to place this next exam might be decided, and so on. For such problem domains, there is usually an available collection of “low-level” heuristics. For example, in timetabling, a common heuristic is to first sort the events that have to be timetabled according to some measure of difficulty. There are several such measures, based on the fact that some events are more difficult to place than others (e.g., can only fit in a small number of rooms, and potentially clash with many of the other events). One way to do timetabling constructively (such algorithms are often called “greedy”) is to repeatedly choose an event to timetable based on a difficulty measure, and then timetable it by finding a place and a time that suits. Each potential difficulty measure can be considered a different heuristic. Similarly, deciding where and when to place the event are also activities that can be based on a range of specialized heuristics. Very similar can be said of other, if not all, logistic or combinatorial problems.

With independent roots in the field of automated planning and scheduling systems (Minton 1988; Gratch et al. 1993; Cross and Walker 1994), an early and influential example of the hyper-heuristic approach being used for solving specific problem instances (i.e., one at a time) is concerned with open-shop scheduling problems in Fang et al. (1994). In Fang et al.’s work, several low-level heuristics were considered, all of which were relevant to the problem of “open-shop” scheduling, in which there are, say, j jobs that need to be scheduled, each consisting of a certain number of tasks. Each such task must use a specific resource (usually called a machine) for a specific amount of time, although the tasks that comprise a given job may be done in any order (when the order of tasks within a job is constrained, it is a job-shop problem). For example, PCs may arrive at a processing center with the operating system installed, and need to have a number of applications installed (for which order of installation is unimportant) by a number of experts, each expert in the installation of a particular application. Each “job” is a PC, which may have its own individual specification and subset of applications that need to be installed; this amounts to an open-shop scheduling problem.

Fang et al. (1994) used an evolutionary algorithm which constructed solutions as follows. A chromosome was a series of pairs of integers $[t_0, h_0, t_1, h_1, \dots]$ interpreted from left to right, meaning: for each i , “consider the t_i -th uncompleted job (always interpretable, when treating the list of uncompleted jobs as circular) and use heuristic h_i to select a task to insert into the growing schedule in the earliest place where it will fit.” Examples of the lower-level heuristics used are:

- Choose the task with the largest processing time
- Choose the task with the shortest processing time
- Find the tasks that can start earliest (there may be more than one) and choose the one with largest processing time
- Find the tasks that can be inserted into a gap in the schedule so far, and pick one that best fills this gap

This approach was called “evolving heuristic choice,” and led to excellent results on benchmark problems, including some new best results at the time of publication, and it marked the beginning of a wave of interest in what were later termed “hyper-heuristic” approaches.

An example following this work was that of Hart and Ross (1998), who looked at job-shop scheduling problems (where the ordering of tasks within a job is predetermined – for example, in the software installation example, it could well be the case that applications need to be installed in a certain order). Their approach relied on the fact that there is always an optimal schedule which is “active,” meaning that to get any task completed sooner, one would need to change the order in which tasks from different jobs get processed on one or more of the machines. Meanwhile, a well-known heuristic algorithm (due to Giffler and Thompson 1960) that generates active schedules was exploited. The explanation by Hart and Ross (1998) and in Burke et al. (2003) is now followed in explaining their approach. Giffler and Thomson’s active-schedule generation algorithm is as follows:

1. Let C = the set of all tasks that can be scheduled next
2. Let t = the minimum completion time of tasks in C , and let m = machine on which it would be achieved
3. Let G = the set of tasks in C that are to run on m whose start time is $< t$
4. Choose a member of G , insert it in the schedule
5. Go to step 1

In step 4, there is a choice to be made, which was exploited in Hart and Ross’ hyper-heuristic approach. Now consider a simplified version of this algorithm, which only generates so-called “non-delay” schedules.

1. Let C = the set of all tasks that can be scheduled next
2. Let G = the subset of C that can start at the earliest possible time
3. Choose a member of G , insert it in the schedule
4. Go to step 1

This time, there is a choice to be made in step 3. Hart and Ross’ approach was to use an encoding of the form $[a_1, h_1, a_2, h_2, \dots]$, again interpreted from left to right, where the a s are 0 or 1, indicating whether to use an iteration of the Giffler and Thompson algorithm or an iteration of the non-delay algorithm, in order to decide on the next task to schedule, and the h s indicate which of the 12 heuristics to use to make the choice involved in the selected algorithm. This method again produced excellent results on benchmark problems.

Finally, before moving on to two examples of what can be called “super-heuristics” (i.e., where general problem solvers are evolved, rather than algorithms for one instance at a time), an early real-world application of the hyper-heuristic approach is briefly mentioned. Described in Hart et al. (1998), the problem that needed to be solved was to schedule the collection of live chickens from farms in Scotland and Northern England, for delivery to one of two processing factories. A given instance of the problem arises from a set of orders from supermarkets and other retailers, which have to be fulfilled within given time windows. The specific resources that needed scheduling were of two types: the collection of live chickens from farms was done by a set of “catching squads” who moved around the country in minibuses; the delivery of chickens to processing factories was done by a set of lorries. In general, catching squads needed to move from farm to farm collecting chickens, and lorries needed to arrive at farms in time to be loaded with chickens caught by the squads, and then either move to another farm if able to hold more, or proceed to unload at a processing plant (and then perhaps back to a farm). The principal aim was to keep the factories supplied with work, while attempting to ensure that live chickens did not wait too long in the factory yard, for veterinary and legal reasons. There were several constraints. For example, different types of

catching squad were distinguished by differences in their contractual arrangements, relating to the amounts of work they would do per day or week (including, e.g., guaranteed minimum amounts of work). Meanwhile, the order in which a given squad could visit farms in 1 day was constrained according to the status of each farm in terms of certain chicken diseases, while lorry schedules also were subject to a range of associated constraints. Overall, the target was to create good schedules satisfying the many constraints, but that were also generally similar to the kinds of work pattern that the staff were already familiar with, and to do so quickly and reliably.

After several approaches which did not work very well, using what were the standard styles of evolutionary algorithm approach at the time (experts in classical scheduling methods had already been consulted by the company, and had tended to retreat in terror once the problem had been described to them), the eventual solution used two evolutionary algorithms in two stages. The first was a hyper-heuristic approach to assign tasks to individual catching squads in a way that was able to cover the current set of customer orders. In detail, a chromosome specified a permutation of customer orders followed by two sequences of heuristic choices. The first sequence of heuristics specified ways to split each order into convenient workloads, and the second sequence of heuristics specified how to assign those workloads to catching squads. The second stage was an evolutionary algorithm that took the set of tasks produced from the first stage, and delivered a schedule of lorry arrivals at each factory. For this real industry problem, a hyper-heuristics approach was central to a solution that worked successfully, whereas no previous approach had met the required standards.

Before moving on to “super-heuristics,” we note that the surface of applications that have found hyper-heuristics to be a highly flexible and successful approach has barely been scratched, albeit at the time of writing the application areas tend to be not very diverse, with most either involving timetabling (e.g., Terashima-Marin et al. 1999; Cowling et al. 2000; Burke et al. 2002; Bilgin et al. 2006) or scheduling (e.g., Hart and Ross 1998; Cowling et al. 2002; Ayob and Kendall 2003). For a much more comprehensive discussion of hyper-heuristics, readers may refer again to Burke et al. (2003), as well as Özcan et al. (2008).

4.2.3 Super-heuristics: Evolving and Learning New and Effective Algorithms

In an increasingly influential thread of research, Ross et al. (2002, 2003) extended the notion of hyper-heuristics to see whether new constructive algorithms, which could deal effectively with large sets of problem instances, rather than one instance at a time, could be evolved. In what is termed here as a “super-heuristic” approach, Ross et al. (2002, 2003) used a learning classifier system called XCS (Wilson 1998), and later an evolutionary algorithm, to try to learn an algorithm for solving hard bin-packing problems. The learning was done in Ross et al. (2003) with an evolutionary algorithm aiming to optimize the parameters for a fast constructive bin-packing algorithm, training on a set of test problems (i.e., a collection of different problem instances was involved in the fitness function). When the learned algorithm was then tested on a different set of test problems, its performance was found to be clearly competitive with state-of-the-art human-designed bin-packing constructive algorithms.

In bin-packing (as with many algorithms, and as discussed with scheduling), a typical constructive algorithm will build a solution one step at a time, each step involving the use of some heuristic to choose the next item to pack into a bin, and maybe another heuristic to

choose which bin to place it in (or a single heuristic covering the combined decision). The overall goal is to pack a given collection of items of different sizes into a set of fixed capacity bins, using as few bins as possible. In detail, the overall idea in Ross et al. (2002) and their later work (2003) is as follows. At each stage during such a constructive algorithm, there is a particular problem “state” that is characterized by the set of items left to pack, and the current partial packing of items into bins. In this state, it is reasonable to infer that some heuristics will be better than others for deciding on the next item/bin placement. So, Ross et al.’s approach was to define a constructive algorithm as a set of rules. Each rule in the set referred to a particular problem state, and specified what heuristic to use when in that state. Clearly, there are far more potential problem states than one can expect to be represented by the left-hand sides of such a rule; the method gets around this by having the rules essentially refer to points in the space of potential problem states, and the rule that “fires” at any particular time is the one that is closest to the current problem state.

The approach was first tested using 890 benchmark bin-packing problems in Ross et al. (2002), of which 667 were used to train the XCS learning classifier system, and 223 for testing. The single resulting learned constructive algorithm was able to achieve optimal results on 78.1% of the problems in the training set, and 74.6% of the problems in the unseen test set. This compared well with the best single heuristic tested, which achieved optimality 73% of the time. A notable finding in that work was that when the training set was confined to some of the harder problems, the learned algorithm was able to solve seven out of ten of those problems to optimality (compared with zero out of ten for the comparison human-designed heuristics). This approach was improved in Ross et al. (2003), with many interesting findings that showed highly competitive results for evolved algorithms on hard unseen problems.

Finally, we have a brief look at a different style of super-heuristic approach applied to a different domain, specifically the work of Fukunaga (2008), which concerns the satisfiability (SAT) problem. A SAT problem instance is a conjunctive normal form (CNF) expression, such as “(A or B or D) and (B or not(C)) and (D or E) . . .,” involving a number of logical variables (A, B, . . .), which may either be true or false, which in turn are the elements of a number of clauses, conjoined into the full statement. The problem is to discover whether or not an assignment of truth values to each of the variables exists, which results in each of the conjuncts, and therefore the entire statement, being true. Fukunaga’s work exploited a well-known general local search framework for SAT, as follows:

1. Generate an assignment A of truth values at random (e.g., $A = T, B = F, C = F, \dots$)
2. For a given maximum number of iterations:
 - 2.1. If A satisfies the formula, return YES
 - 2.2. Choose a variable V with a *Variable Selection Heuristic*
 - 2.3. Change A by flipping the value of variable V
3. Return UNKNOWN

The algorithm uses a “Variable Selection Heuristic” in Step 2.2, and this in turn was the focus of Fukunaga’s investigations. There are several well-known examples of variable selection heuristics, which are human-designed and typically used within the above algorithm framework. One example is GSAT (Selman et al. 1992), which involves choosing the variable that, if flipped, would cause the highest net gain in satisfied clauses, breaking ties randomly. Another, HSAT (Gent and Walsh 1993), works as GSAT, but breaks ties in favor of *age* – so, the variable that was last flipped longest ago in the overarching local search process is the one chosen to break the tie. Yet another, of several more, is the so-called GWSAT(p) (Selman et al. 1994), in

which, with probability p , a random variable from a random unsatisfied clause is selected, else GSAT is used.

Fukunaga (2008) noticed that variable selection heuristics in the SAT literature have certain common building blocks, including

- Scoring variables via a gain metric
- Selecting a variable from a subset of variables
- Ranking variables, and choosing the best (or second best)
- Consideration of a variable's "age"
- Branching (if x do A , else do B)

An insightful comment that Fukunaga makes is that in the history of SAT heuristics, developments typically come from finding new ways to combine these building blocks, rather than entirely novel heuristics. This begs a number of questions, one of which is whether or not automated methods may be able to find better combinations of these building blocks. The latter is in fact exactly what Fukunaga (2008) investigated, by using GP, with a function and terminal set designed in such a way that novel heuristics could be expressed in terms of the above ingredients. As with the previous super-heuristic approach discussed, the GP experiments involved using a large set of different SAT instances in the fitness function, and Fukunaga (2008) evaluates the results by testing the evolved variable selection heuristics on unseen test sets.

On a collection of 1,000 unseen test instances, Fukunaga's evolved variable selection heuristics are very competitive with the state-of-the-art variable selection heuristics, GWSAT, WalkSAT, and Novelty (McAllester et al. 1997). A handful of the new heuristics found in this way dominated the state-of-the-art heuristics in terms of success rate and speed. A further rather interesting finding was that one of the heuristics in a random search of expression trees was almost as good in terms of success rate, but usually faster, than the human-designed state-of-the-art heuristics.

4.2.4 Some Concluding Notes

The super-heuristics concept has the potential to play a major role in optimization over the next few years. One way to view this development is as a thrust toward more "general" optimization systems, which, for a wide variety of application areas, is a significant goal. In just one example of application area, *timetabling*, there has been very extensive research in recent years along the lines of hyper-heuristics and upper-heuristics; this has followed a statement in Ross et al. (1997), which was, "... all this naturally suggests a possibly worthwhile direction for timetabling research involving GAs. It is suggested that a GA might be better employed in searching for a good algorithm rather than searching for a specific solution to a specific problem." In agreement with Burke et al. (2003), we emphasize that this suggestion can be generalized to a much wider range of problem areas than has currently been addressed with hyper- and super-heuristic technologies.

5 Design: Art, Engineering, and Software

This penultimate section considers the theme of "Design," and discusses three quite contrasting examples. Design is an area of especial interest when we consider what natural inspiration

has to offer to practitioners of various sorts. Today, and for some considerable time still to come, the world is, to most intents and purposes, filled with two kinds of artifacts – those designed by nature, and those designed by human designers. The chief difference between these two kinds of artifacts is the specific design method that was employed. The naturally designed artifacts, as most scientists would agree, were designed by an evolutionary process – essentially an iterated process of randomized generation and test, in which new designs, often failures, sometimes improvements, emerge via slight random changes or randomized recombinations of old designs. With a “survival of the fittest” principle built in to this strategy, the successes are more often chosen than the failures when it comes to being the foundation for (or the parents of) new designs. Over time, this process continues to evolve new designs that are successful in their environment, and the examples seen today include everything from archaea to artichokes, baobabs to brains, *Escherichia coli* to elephants, and from wasps to the sophisticated set of processes that lead to the construction of wasp nests. It is overwhelmingly the case, however, that human-designed artifacts have not adopted this process. Humans prefer to design things in a rational way, which prefers the adoption of designs that have worked before for similar problems, and rejects the notion of any randomized exploration. Humans tend to stick to a battery of accepted design rules for the application in hand, and usually opt for a step-by-step constructive approach, rather than generating and later discarding many different designs at once.

Some criticisms of the human way of designing can be summed up in the following statement: the overreliance on established design rules imposes severe constraints on innovation, and probably limits the effectiveness of the resulting designs. Meanwhile, nature’s method for design may well not be perfect – it does indeed seem wasteful – however, it certainly beats the human method for innovation. One cannot yet design, with a rational approach, a biological flying machine as efficient as a mosquito, or an energy transduction system as efficient as photosynthesis. Meanwhile, it is notable that randomization is an integral part of nature’s method – undirected perturbations to designs tend to be anathema to the human approach, but are continually tried and tested in nature. Overall, it seems abundantly clear that nature has a lot to teach us about how to design things.

Perhaps unsurprisingly to most of readers, but nevertheless, it is hoped, inspiringly, the documented experiences so far in the arena of natural computation in design show that novel, effective, and unprecedented designs can be found by applying nature’s method to design the artifacts that need to be created. The next subsections discuss one of the more prominent and exciting examples in recent years, NASA’s use of evolutionary techniques to come up with entirely novel antenna designs that have been deployed on satellite missions. But before that we look at an example of the use of interactive evolutionary computation in artistic design, and this section ends with a brief look at how natural computation is making headway into the design of software.

5.1 Interactive Evolutionary Design of Batik Patterns

Evolutionary Art Systems (EASs) are increasingly popular (Romero and Machado 2008), commonly using evolutionary computation, usually interactively (e.g., Sims 1991; Lutton 2006), to generate aesthetic artworks. In some real-world applications, focusing on particular niches in art and design, EASs have been developed specifically to facilitate a designer’s activity. One recent such case, which is described here, is by Li et al. (2009), in which an EAS

tool is described for helping designers of Batik patterns, a traditional art in Indonesia and southeast Asia. Batik is a form of painting or writing on cotton cloth, applied with the aid of a tool called a cap (Kerlogue and Zanetini 2004). Nowadays, Batik is used in fashion, furnishing fabrics, and household accessories, as well as paintings and ornamentations in rooms and offices. However, fine-quality handmade Batik is very expensive, so it is potentially valuable to consider ways that would decrease Batik designers' effort and increase production of Batik.

Li et al. (2009) investigated the potential for an EAS-based Batik design system with such goals in mind. In doing so, however, they had to consider the difficulties commonly faced by EAS. First, the evolutionary process is often quite limited by the lack of an explicit correlation between genotypes and phenotypes. Essentially, the common ways in which aesthetic works tend to be encoded by manipulable genes (think of fractal patterns encoded in the typical way by mathematical formulae) are far removed from the works themselves, so that, for example, when human designers select what they think are good parents, they may find that none of the promising features they saw in the parents actually appears in the next generation. Another common difficulty is that the process can be tiresome for a human designer, spending hours sitting at a computer rating generated images. Li et al.'s work attempted to develop a Batik design system with innovations that addressed these issues. In particular, they devised a suitable encoding for various Batik styles, and they devised an "out-breeding" mechanism that provided an additional way to generate new patterns, which seemed to be on the aesthetic path being pursued by the designer. These issues are elaborated in the following subsections, but the reader is referred to Li et al. (2009) for a more complete account.

5.1.1 Encoding Batik Patterns

Li et al. (2009) explored the space of geometrical patterns used in Batik, and classified them into categories. They found that the most common features were repetition, and certain geometric transformations such as rotation, translation, and reflection. This led to a way to encode patterns in genotypes, which specify a number of nonredundant primitives along with transformations. The encoding is therefore based directly on features of Batik patterns, most basic elements of which include: triangle, polygon, circle, dot, star, and flower. Each feature is generated from one gene in the genotype.

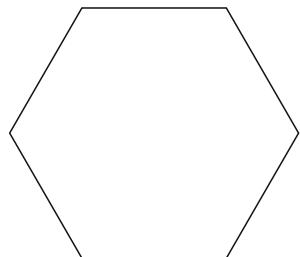
A genotype consists of a variable number of genes, each of which represents one feature in the phenotype. Every gene has two evolvable attributes. The first part is a specific basic pattern (e.g., a simple representation of a flower petal, or a circle, or a triangle); the second part, the transformation, is a vector of matrices, which each represent a transformation of the unit set. A matrix is encoded by six numbers, indicating a 2D linear transformation together with a translation. This representation is straightforward and easy to manipulate. The resulting pattern is made up of the union of the patterns induced by the different genes.  [Figure 12](#) shows some examples of single simple genes in this encoding, with their interpretations above, while  [Fig. 13](#) shows some patterns produced using the system, contrasted with some human-designed similar Batik patterns.

5.1.2 Boosting the Evolutionary Process

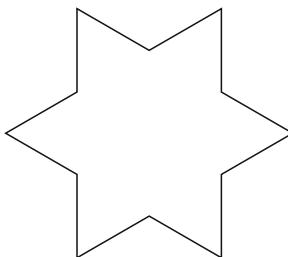
Li et al. (2009) use what they call an "out-breeding" mechanism to invigorate the pool of patterns produced during the interactive evolutionary process. In their EAS, two separate

Fig. 12

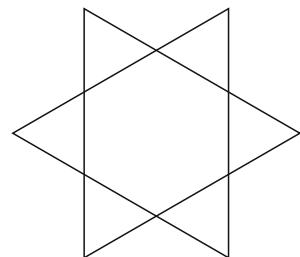
Simple examples of Batik pattern genes, and their interpretations.



(1,1, straight line, rotate, 6)



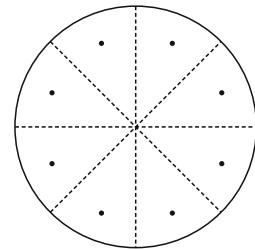
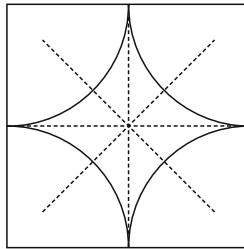
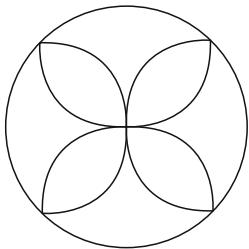
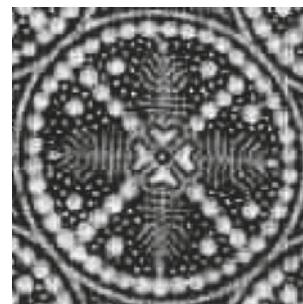
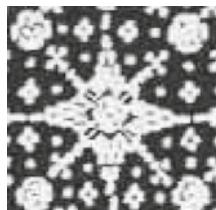
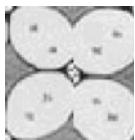
(2,1, straight line, rotate, 6)



(2,2, straight line, rotate, 6)

Fig. 13

Above: some real-world Batik patterns; below: similar individuals generated by the mathematical model, such as appear in the initial population of the Batik interactive evolutionary system.

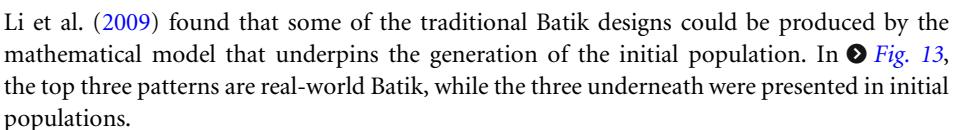


populations of patterns are maintained, displayed to the designer on separate panels. One population evolves in the normal way, based on treating the user's feedback as the fitness function. However, the second population evolves toward individuals that are maximally dissimilar to what seem to be the user's preferences, hence injecting considerable diversity into the displayed patterns. Whenever the first population seems to be stagnating, individuals in the second population will be introduced to the first, contributing diverse input to the gene pool.

The crux of this mechanism is the idea of “dissimilarity,” which requires a way to compare patterns. Li et al. (2009) preferred to investigate a measure that was related to the visual difference between patterns, expecting that a method based only on genotypic difference would not be satisfactory. They use a metric based on singular value decomposition (SVD) (Wang et al. 2000). In their approach, a pattern A is interpreted as a matrix, and they represent each pattern in terms of the singular values arising from the SVD of A , which in turn are likely to capture salient features of the visual perception of A . A similarity metric between two patterns is then defined on the basis of a normalized comparison of their vectors of singular values. The outbreeding process then operates as follows: In each generation, while one population continues to regenerate patterns according to the normal process, guided by the user’s evaluations, the outbreeding population regenerates in a way guided by using dissimilarity as the fitness measure, measured in terms of dissimilarity from the pattern that the user currently perceives as best. Li et al. (2009) report that the outbreeding mechanism is very effective in aiding the search for innovative patterns, and find that the “outbred” populations tend to be more elaborate and attractive than the “main” population!

Meanwhile, concerning the “standard” interactively evolved population, one notes that the generation of the initial population, and the subsequent evolution based on user-supplied fitnesses, relies on a collection of typical genetic operators as follows. The initial population is informed by using a mathematical model of Batik pattern space based on Li et al.’s preliminary characterization. The model is used to generate collections of genes, and then mutation operators are applied to these: either Gaussian mutation (in which each point in the basic pattern element of each gene is perturbed by the same random amount), or style mutation (in which the elements of a gene reflecting line styles are perturbed, for example from straight-line to curve). During the subsequent interactive evolution process, new patterns are produced by crossover and mutation of patterns deemed good by the user. Standard types of crossover and mutation are used for this in Li et al.’s work so far, for example, including linear combination and gene-swap-based crossover. Also, as explained fully in Li et al. (2009), their system has other features that are meant to aid the user’s design process, such as the ability to retrieve patterns that were produced earlier in the evolution.

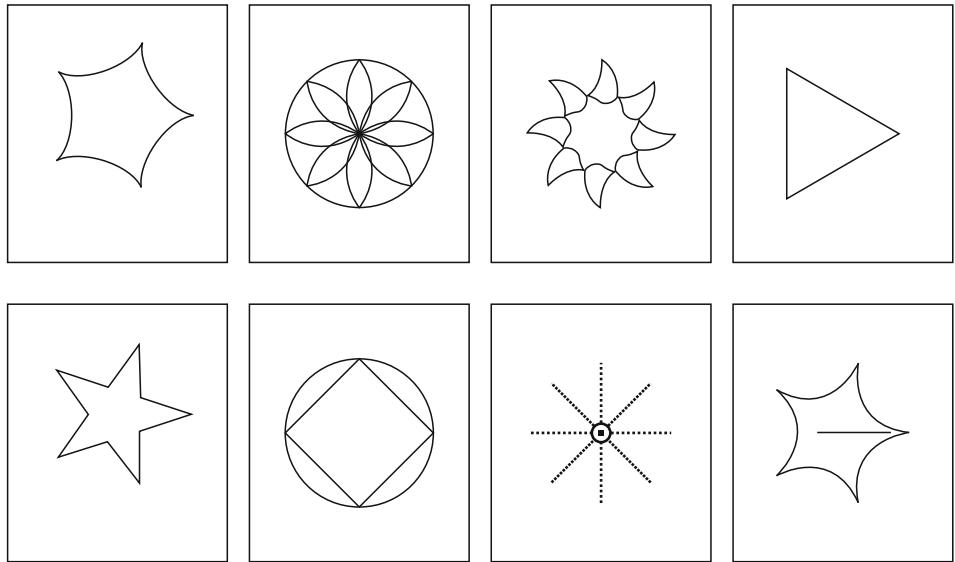
5.1.3 Empirical Notes

Li et al. (2009) found that some of the traditional Batik designs could be produced by the mathematical model that underpins the generation of the initial population. In  Fig. 13, the top three patterns are real-world Batik, while the three underneath were presented in initial populations.

Li et al. (2009) report on five experiments using their system, aimed partly at evaluating the outbreeding technique; each experiment ran the process twice, with and without outbreeding (but starting from the same initial populations). They measured, in particular, the time investment of the user before a satisfactory design was achieved. They found that, with the outbreeding mechanism in place, the design process took on average only 54% of the time taken using the interactive system without outbreeding. Further, the time with outbreeding was roughly 17% of the time it tends to take to design a new Batik pattern by hand. Further experiments confirmed in other ways that the outbreeding mechanism was effective in producing patterns, throughout the process, that tended to be well evaluated by users.  Figure 14 shows the initial population used for all of these experiments, and

Fig. 14

Initial populations used in Li et al.'s experiments.

**Fig. 15**

Tessellations of final-population designs using the Batik pattern interactive evolutionary system (with the outbreeding mechanism).

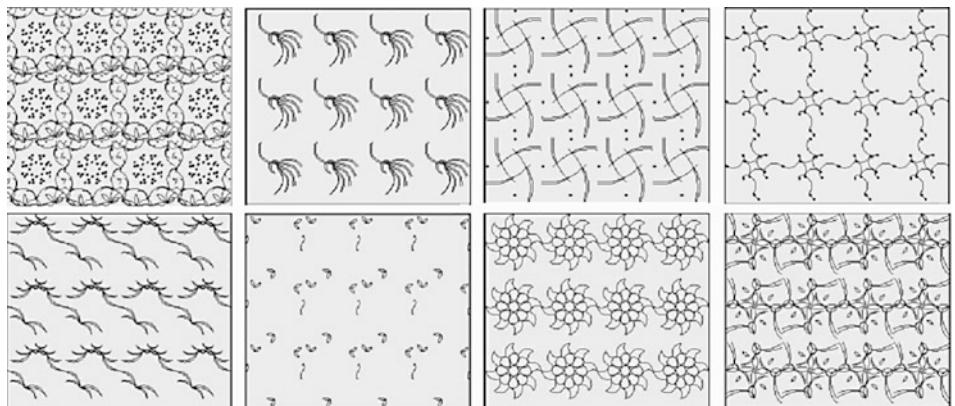


Fig. 15 shows some final-population designs that satisfied the users (produced with the outbreeding mechanism in operation), converted into tessellations.

5.1.4 Final Points and Notes

The interactive Batik design system described and discussed here is a nice example of how interactive evolutionary computation is beginning to be used in an increasing number of

applications that involve creativity. Experience with this system so far shows how it can both speed up and invigorate the process of generating interesting new patterns in the Batik “domain.” One of the keys to success in such enterprises is the wise design of the encoding, and we have seen a good example of that in this case. Li et al. (2009), as already seen, also showed an innovative approach to dealing with some of the ever-present problems (and, hence, research issues) in interactive evolution. The outbreeding mechanism was able to enhance diversity in the process, and at the same time reduce the time (and hence fatigue) of human users.

5.2 Novel Antennas for Satellites: Discarding the Rule Book

As elaborated further in Hornby et al. (2006), current practice in antenna design almost invariably involves designing and optimizing them by hand, and this approach is very limited as a way to develop new and better antenna designs. It requires significant time and expertise from human experts in the domain. An ongoing alternative in antenna design (in common with an increasing variety of such specialist areas) is to investigate evolutionary algorithms for this purpose. This has been happening since the early 1990s, with increasing success and take-up as developments in processing power have been seen, and also improvements in the quality of software simulations of antenna performance. To date, many types of antenna have been investigated using evolutionary design approaches. A particularly interesting and useful aspect of this approach is the opportunity to evolve antenna designs specifically for performance in a particular environment, so that the fitness function takes into account the effects of structures surrounding the antenna’s intended position. This consideration of the immediate environment is extremely difficult for human expert antenna designers to take into account.

This section summarizes the work reported in Hornby et al. (2006) and other publications from that group, which describe the experience and results of using evolutionary algorithms to evolve antennas for spacecraft associated with a number of NASA missions; in particular, two antennas designed for NASA’s Space Technology 5 (ST5) mission, and an antenna for a Tracking and Data Relay Satellite (TDRS) for a mission due to operate after 2010.

5.2.1 Antennas for NASA’s Space Technology 5 Mission

NASA’s Space Technology 5 (ST5) mission had the goal of launching multiple miniature spacecraft to test various innovative concepts for application in future space missions. Three miniaturized satellites were involved in ST5, called micro-sats, designed to measure the effects of solar activity on the Earth’s magnetosphere. These micro-sats were approximately half a meter across and half a meter high, weighing around 25 kg when fully fuelled, and each had two antennas, centered on the top and bottom. They were originally designed to operate in a geosynchronous orbit at approximately 35,000 km above Earth, and had a stringent set of requirements for the communication antennas. Details of the specific requirements are in Hornby et al. (2006), and one need not discuss them here, but (in common with similar antenna design tasks) these requirements were in terms of constraints on the gain patterns, voltage standing wave ratios, and input impedances, at both the transmit and receive frequencies; also the mass of each antenna had to be below 165 g, and the shape had to fit within a cylinder with height and diameter both below 16 cm.

To meet the initial design requirements in this instance, the team decided to constrain their search to a monopole wire antenna with four identical arms, equally spaced around the vertical axis. An evolutionary algorithm was therefore set to work to evolve the shape of a single arm, which in turn defined the entire antenna. Importantly, the encoding used by the team was one that allowed almost arbitrary designs for the arm, with no reference to the limited collection of known standard designs. Essentially it was a GP style approach, in which each node in a tree was an antenna-construction operator. Interpreting the tree top down from the root node, and given an initial “feed-wire” of a given small length and orientation, the operators and leaves of the tree effectively specified three-dimensional movements in the style of “turtle graphics,” adding sections of wire of specific lengths and orientations to the current partial design.

Having decoded a tree into an antenna design, the antenna was simulated by means of a sophisticated simulation platform, which yielded estimated performance characteristics which then had to be automatically evaluated against the design requirements. In common with the design requirements themselves, readers are referred to Hornby et al. (2006) for details of the fitness function, but suffice it to say that the requirements themselves and the simulation results are both curves involving performance characteristics at different spatial locations and frequencies, and the fitness function involved such things as estimates of distances between desired and actual curves, weighted in specific ways according to the importance of different requirements.

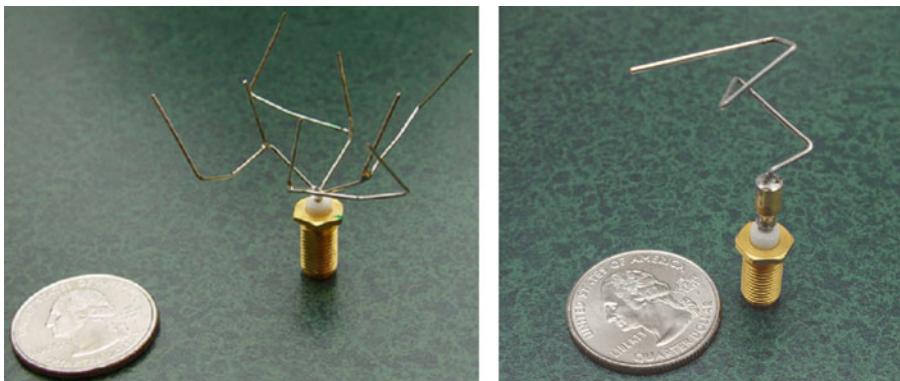
It so happened that the requirements for the ST5 mission changed while these initial antennas were being designed. New mission requirements effectively forced a single-arm antenna design, and this led to the need to redesign the fitness function for the antenna design process. In the operating environment context of Hornby et al.’s work, it is of particular interest and importance to note that an extremely effective antenna design was produced, for the initial set of requirements, in a short time when compared with the human expert design process. Moreover, with mission requirements altered partway through the process, the evolutionary algorithm approach needed only relatively minor modification and was still able to quickly produce an effective antenna for the new requirements.

To meet the initial mission requirements, the best evolved antenna design that emerged, “ST5-3-10,” is shown in  Fig. 16 on the left. This antenna met the initial mission requirements, and was indeed all set to be used on the mission itself, until the mission’s orbit (and hence many other aspects) was revised. The new evolved best antenna following the new requirements was the one shown on the right in  Fig. 16, the so-called “ST5-33-142-7.” The latter antenna design, which was delivered for prototype fabrication less than a month after the changes to the ST5 mission requirements, was found fully compliant with specifications when the prototype was tested, and on 22 March 2006 the ST5 mission was successfully launched into space using evolved antenna ST5-33-142-7. Hornby et al. (2006) report that this was the first computer-evolved antenna to be deployed for any application and the first evolved hardware in space. (One notes that this is clearly valid, if one confines oneself to hardware produced, by whatever means, in the local solar system; but one does not know about elsewhere.)

Hornby et al. (2006) note that the evolved antenna has a number of advantages over human-designed alternatives. These advantages include reduced power consumption, fabrication time and complexity, and improved performance. The ST5 mission managers had actually hired a contractor to produce antenna designs in addition to awaiting the findings of the evolutionary approach. The contractor used conventional design practices, and came up with a variant of one of the many standard designs. When this design was compared in simulation with the evolved design, it was found that if an ST5 craft used two evolved antennas (recall that

Fig. 16

Photographs, reproduced with permission, of prototype-fabricated evolved antennas. Left: the best obtained antenna for the initial ST5 mission requirements, ST5-3-10; right: the best obtained following the revised specifications, ST5-33-142-7.



each craft had two antennas), efficiency would be 93% improved over the situation where the craft instead used two of the contractor-designed antennas. Among other explication of the various benefits in Hornby et al. (2006), it is noted that the evolved antenna required approximately 3 person-months to design and fabricate, versus approximately 5 months for the human-designed one.

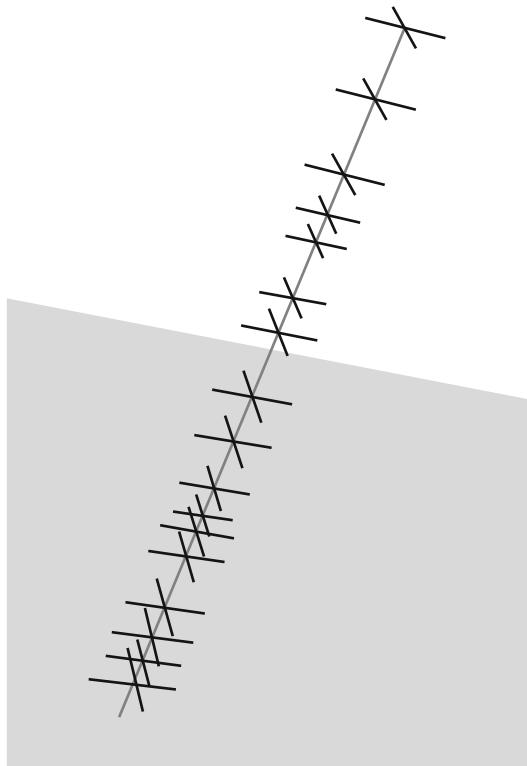
5.2.2 An Antenna for NASA's TDRS-C Communications Satellite

Later in 2006, the same team evolved an “S-band phased array” antenna element design for NASA’s TDRS-C communications satellite, part of a mission that was scheduled for launch sometime between 2010 and 2020. This time the evolutionary algorithm was combined with a hill-climbing algorithm, and the antenna design was somewhat more constrained toward a standard style; nevertheless the resulting design was simpler than the potential competing human designed antennas, consequently reducing testing and integration costs.

As Hornby et al. (2006) reports, the TDRS-C mission will carry several antennas, including among them a 46 element phased array antenna. Readers unfamiliar with the terminology may see [Fig. 17](#), from which it becomes clear what the individual elements are in the phased array. The design and performance specifications for this antenna involved electromagnetic performance issues, as was the case for the ST5 missions, but also certain constraints on the elements and their spacing.

A simpler encoding was used by the team for this case, in which an antenna was represented as a fixed length list of real numbers. Antenna parameters were determined from these simple “genes” in a fairly straightforward way, in which the majority of successive pairs of genes referred to the distance to the next element along the antenna’s axis, followed by the size of the next element.

In a similar process used for the ST5 mission antennas, the team set up around 150 separate experiments that each ran an evolutionary algorithm for a total of 50,000 evaluations (antenna simulations) each – the separate evolutionary algorithms each represented a random

Fig. 17**Best evolved TDRS-C antenna.**

point in parameter space, with different population sizes, mutation rates, and so forth. The best antennas from each of these 150 runs was then subject in a second stage to further improvement via a hill-climbing algorithm for 100,000 evaluations. Finally, the best of these were subject to further hill-climbing.

At the end of this process, most of the evolved antennas were very close to meeting the rather stringent mission specifications, and one of the evolved antennas exceeded the specifications. That one, shown in [Fig. 17](#), was further analyzed by an accurate electromagnetic software (WIPL-D version 5.2), and was subjected to some fine tuning via another evolutionary algorithm, and finally a resulting antenna design was fabricated and tested. The final design, shown in [Fig. 17](#), exceeds the design specifications, and it remains up to the mission leaders whether it is deployed in the TDRS-C mission.

5.2.3 Concluding Points

In this section, the work of Hornby et al. (2006) in evolving antennas for two NASA missions has been described. For both the ST5 mission and the TDRS-C missions it took approximately

3 months to set up the evolutionary algorithms and produce the initial evolved antenna designs. Following the revision in ST5 requirements, it took roughly 1 month for the team to evolve antenna ST5-33-142-7, and the team is indeed very confident (Hornby et al. 2006) that a change in requirements for the TDRS-C mission will result in a similarly fast redesign of an antenna meeting the new requirements.

As well as benefits in relative speed and ease of design, the evolutionary algorithm approach to designing antennas leads to many other advantages over manual design. One such advantage is the potential for performance characteristics that are simply unachievable with conventional design styles. Antenna design is one of several areas in which there is potential for unexplored areas of design space to be examined. These are areas of design space that human experts tend to steer away from, since the current state of theory and understanding is quite limited to the properties of a range of conventional designs. Evolutionary algorithms are far less wary of such ill-understood areas of design space, and, by finding exemplars in such areas that have outstanding performance (such as the ST5 designs discussed in this section), they may lead to more systematic study of such regions of the design space, leading to new design principles and new scientific insights.

5.3 Evolution in Software Design

As noted in Arcuri and Yao (2008), software testing is used to find bugs in computer programs (Myers 1979). Even though successful testing is no guarantee that the software is bug free, testing increases confidence in the software's reliability, and is an integral and extremely important part of modern software engineering. However, testing is very expensive, time consuming, and tedious, amounting to around half the total cost of software development (Beizer 1990). This investment in testing is not begrimed, since releasing bug-ridden software can be immensely more costly in the long run. In fact, it is often argued that far more testing should be done than is usually the case – in the USA, for example, it is estimated that around \$20 billion per year could be saved if better testing was done (Tassey 2002). The need for cheaper and faster testing is clear.

This section looks at recent work by Arcuri and Yao (2008), which is part of an area of research called search-based software engineering. In this particular thread, the idea is to investigate the use of evolutionary computation to improve aspects of the testing process. In particular, Arcuri and Yao (2008) are concerned with unit tests (Ellims et al. 2006). This relates to writing small pieces of software code that test as many parts of the project as possible. For example, the test code might call, with specific inputs, a Java method that adds two integers; the returned value is then checked against the expected value. If there is a difference, one can be sure that there is something wrong with the code. However, since testing all possible inputs of a method is usually infeasible, a suitable subset of tests needs to be chosen. Writing code for such “unit tests” requires some way to decide on a good collection of specific input cases, and is a very resource-hungry exercise.

Automated ways to generate unit tests are clearly of interest to the software design process, and this is the topic of Arcuri and Yao's (2008) work. Various approaches have been studied to automatically generate unit tests (McMinn 2004), but there is no known way to generate an optimal set of unit tests for any given program. Also, comparatively little has been done in this area for object-oriented (OO) software. This section describes Arcuri and Yao's (2008) recent

work that focuses on a particular type of OO software construct: *containers*. These are data structures (like arrays, lists, vectors, trees, etc.) designed to store arbitrary types of data. What usually distinguishes a container class is the computational cost of operations such as insertion, deletion, and retrieval of data objects. They are used in almost all OO software, so their reliability in commercial code is paramount.

Arcuri and Yao (2007) presented a framework for automatically generating unit tests for container classes, in the context of white box testing. They analyzed a number of different search algorithms, and compared them with more traditional techniques. They used a search space reduction that exploits the characteristics of the containers. Without this reduction, the use of search algorithms would have required too much computational time.

5.3.1 About Testing Java Containers

In each of the many kinds of Java containers (arrays, vectors, lists, trees, and so forth), one usually expects to find methods such as insert, remove, and find. The implementations (and hence computational expense) of these methods can vary much between containers; also, the behavior of such methods is often a function of the container's current contents. This situation considerably complicates the design of unit tests (McMinn and Holcombe 2003, 2005) – just because it is found that a method yields the correct result with certain inputs, it does not mean that it will always give the correct result with those inputs, perhaps depending on the current contents of the container.

The approach to testing containers therefore explicitly considers the sequence, S_p , of function calls. During a testing operation on such a container, the container is referred to here as a “Container under Test” (CuT), and a function call (FC) can be seen as a triple:

<object reference; function name; input list>

This simply refers to calling the given function (method) of the given object (container) with the given list of inputs. In Arcuri and Yao’s work, the CuT is subjected to a single sequence S_i of such FCs, rather than a different sequence for each of the functions’ branches. Naturally, in the unit test Java code, each FC is embedded in a different try/catch block, so that the paths that throw up exceptions do not forbid the execution of the subsequent FCs in the sequence.

The goal in testing is to achieve a maximal level of “coverage”; broadly speaking, this refers to the amount of code that is tested. All software is replete, for example, with case statements and “if X then Y else . . .”-style branches, and, without suitable design of test cases, many branches of the code may end up not being followed during the testing process. Given a suitable coverage-related criterion (there are several), it is then important to aim for the short sequence of function calls while achieving excellent coverage. Arcuri and Yao (2008) used *branch coverage* as their coverage criterion, although their approach is easily extensible to other coverage criteria.

In Arcuri and Yao’s formulation, they consider a coverage function $\text{cov}(S_i)$ which, in relation to a given CuT, returns the number of code branches covered when tested with the sequence of functional calls S_i . Where $\text{len}(S_i)$ is simply the number of function calls in the sequence, Arcuri and Yao attempt to optimize both $\text{cov}(S_i)$ and $\text{len}(S_i)$, preferring a shorter sequence in the case that the coverage of two sequences is the same. To some extent, it is clear that this is a multi-objective problem (see Deb 2001, and Sects. 3.1 and 3.2), however, Arcuri and Yao indicate a definite order of preference in this domain (coverage more important than

length), which influences their decision to treat it as a single-objective problem. They therefore attempted to find sequences that optimized $\text{cov}(S_i) + 1/(1 + \text{len}(S_i))$, although with various modifications and adaptations detailed in Arcuri and Yao (2008).

5.3.2 Smoothing the Test Landscape

Arcuri and Yao (2008) detail several complex factors involved in the enterprise of treating unit testing as an application for evolutionary computation, along with their solutions to these issues. Here, only one such issue will be discussed, of particular pertinence to the “engineering” of problems when considering artificial evolution of solutions. This relates to helping the evolutionary process by making the fitness assessments more informative. The problem in this case is that the number of branches covered (i.e., the number returned by $\text{cov}(S_i)$) does not give any indication of how close the sequence S_i is to being able to cover additional branches. Put another way, two sequences S_i and S_j may have the same coverage value, but one may be much “nearer” than the other (i.e., requiring a mutation to just one of its FCs) to a sequence that has higher coverage.

In many branch statements, in which the predicates accessing the branch are quite simple, random sequences of FCs will have little difficulty finding inputs that force coverage of all its branches (this is why a random search, as will be seen later, tends to achieve good coverage). But when the predicate is more complex, it is typically the case that only a very small portion of the space of potential inputs will lead to certain branches being covered. The search is likely to fail.

One approach to this issue is to consider the *Branch Distance* (BD) (Korel 1990). Any particular branch will be entered if a given statement is true (such as $0.2 < x < 0.3$); the BD is a real number that tells how far the relevant predicate is from being true (in the latter case, BD will be low if $x = 0.4$ and high if $x = 10$). Making use of such information in the coverage metric would help the evolutionary search process, by helping to distinguish between pairs of sequences that would otherwise have the same simple coverage value. BD is the topic of much research effort in software testing (e.g., Baresel et al. 2002; Harman et al. 2002; McMinn and Holcombe 2004). This research tends to consider approaches in which different test sequences focus on different branches, without considering the issues involved with the precise sequencing of function calls affecting the results. A difference in Arcuri and Yao’s approach is the attempt to evolve a single test sequence that covers all branches.

The technique they adopt is to modify the $\text{cov}(S_i)$ defined earlier, incorporating within it a simple measure of branch distance for any uncovered branch, which takes into account how many times the predicate associated with a branch is evaluated. For example, if only one FC in the sequence invokes a predicate with two branches, then only one branch will be covered. However, if in another sequence there are two FCs that test this predicate, both invoking the same branch, then it can be said that this second sequence is closer to covering the second branch, since (if coverage of this other branch is possible at all) this requires only mutation of the input list of one of the FCs in the sequence. There are various ways in which the coverage metric could be modified to take branch distance into account, and it turned out important to use different variations in different circumstances, as is mentioned later, and of course discussed more fully in Arcuri and Yao.

5.3.3 Evaluating Natural Computation for This Task

Arcuri and Yao (2008) tested five approaches: random search (RS), hill-climbing (HC), simulated annealing (SA), a genetic algorithm (GA), and a memetic algorithm (MA). RS is a natural baseline used to understand the effectiveness of other presumably more sophisticated algorithms, and often it brings surprisingly good results. In the current context, one can expect RS to give good results in terms of coverage. The RS worked simply by repeatedly generating random sequences, evaluating them, and returning the best at the end of the process; for pragmatic reasons, it was necessary to specify a maximum length for each sequence.

For the other methods, it was necessary to design neighborhood (and genetic) operators that would operate on sequences to produce variants. In all cases, the encoding of a sequence of FCs was entirely straightforward. The “chromosome” is simply an explicit (variable length) sequence of FCs, each a triple as described above. For neighborhood (mutation) operators, the natural choice was made to use operators of the following type:

- removing an FC from a sequence;
- inserting a new FC into the sequence, in a random position;
- modifying the parameters of a randomly chosen FC in a sequence.

In the GA, single point crossover was also used, in which a child sequence was generated by using the first K (randomly chosen) FCs in one parent, and completing the child with the FCs from position $K + 1$ onward in the second parent. The MA was a simple hybrid of the GA and HC, which repeatedly ran HC on each new individual produced by the GA, until a local optimum was reached. Although the RS, HC, GA, and MC were fairly standard in their SA implementation, various modifications and sophistications were included to control the acceptance of new mutants during the search, in an attempt to balance the coverage and length considerations. These details, and of course other parametric details of all of the algorithms, are explained in Arcuri and Yao (2008).

Arcuri and Yao (2008) performed tests on separate Java containers that implemented Vector, Stack, LinkedList, Hashtable, and TreeMap, respectively, from the Java API 1.4, package java.util, and BinTree and BinomialHeap from the examples in Visser et al. (2006). Described here is only a selection of their results, focusing on the four cases which involved the largest number of public functions under test (PuT). These were: Vector (34 PuT), LinkedList (20 PuT), Hashtable (18 PuT) and TreeMap (17 PuT), respectively, with 1,019, 708, 1,060, and 1,636 lines of code, and achievable coverage of 100, 84, 106, and 191 branches. The latter figures for achievable coverage are based on Arcuri and Yao’s experience of around a year’s worth of experimentation, with inspection of the container code confirming that non-covered branches seem unreachable.

Each of the five algorithms was tested, to a limit of 100,000 sequence evaluations per trial, using 100 trials per algorithm and container pair; a selection of Arcuri and Yao’s results are summarized in  Table 6. When one considers the coverage results in the context of the highest achievable coverage mentioned above, it turns out that only TreeMap presents a particularly difficult coverage task. The MA achieves the best mean coverage result on TreeMap, and indeed the MA is reported by Arcuri and Yao (2008) as statistically superior to the other algorithms in all cases except Vector (based on a Mann–Whitney U test). In all container cases except Vector (including the others reported in Arcuri and Yao 2008), the MA shows either the best mean coverage, or it shares first position for coverage while having a better mean length. Not surprisingly, random search tends to have worse performance than the other algorithms,

Table 6

Some results from Arcuri and Yao (2008), showing coverage and lengths obtained when evolving sequences of function calls for five separate containers, using five algorithms: random search (RS), hill-climbing (HC), simulated annealing (SA), genetic algorithm (GA), and memetic algorithm (MA)

Container	Algorithm	Mean coverage	Variance in coverage	Mean length	Variance in length
Vector	RS	85.21	1.52	56.99	7.73
	HC	100.00	0.00	47.67	1.05
	SA	99.99	0.01	45.76	1.11
	GA	99.99	0.01	46.87	1.63
	MA	100.00	0.00	47.89	2.64
LinkedList	RS	69.96	1.82	55.27	14.00
	HC	84.00	0.00	38.48	10.27
	SA	82.47	2.25	33.60	5.29
	GA	83.83	0.26	36.66	3.64
	MA	84.00	0.00	36.43	3.58
Hashtable	RS	92.92	1.17	54.45	25.97
	HC	106.00	0.00	35.25	0.19
	SA	105.84	0.74	34.98	0.77
	GA	101.14	6.50	31.10	6.31
	MA	106.00	0.00	35.01	0.01
TreeMap	RS	151.94	5.85	54.11	26.87
	HC	188.76	0.71	51.23	10.08
	SA	184.19	5.75	40.68	5.88
	GA	185.03	3.46	42.14	8.44
	MA	188.86	0.65	50.55	10.31

although it can often achieve reasonable coverage. When coverage from RS is good, however, the length of the sequence of FCs tends to be poor; this is readily understood given the nature of the “difficult” branches in testing, as also indicated above. Finally, it should be pointed out that Arcuri and Yao’s system was not able to generate inputs that could cover all branches in the CuT; for example, for pragmatic reasons, branches in private methods were not considered, while around 10% of the public methods were not directly callable.

5.3.4 On Related and Similar Work

Arcuri and Yao (2008) point out the difficulties in comparing their approach with traditional systems in software testing, including the fact that there is no common benchmark scenario, and no reasonable way to replicate the ways that other authors instrumented the software to be tested. However, they point out that traditional techniques (e.g., King 1976; Doong and Frankl 1994; Buy et al. 2000; Marinov and Khurshid 2001; Boyapati et al. 2002; Visser et al. 2004;

Xie et al. (2004, 2005) tend to have considerable challenges with scalability, and invariably rely on considerable prior effort, such as the need to generate algebraic specifications or other formal representations of the functions to be tested; this is particularly tricky when predicates are highly nonlinear, involve loops, nonlinear data types, and so forth. Arcuri and Yao's evolutionary computation-based approach, however, needs no such prior specification effort, and is applicable to any container. Meanwhile, although there are many difficulties with direct comparison with results from traditional techniques reported in the literature, the evolutionary computing approach, especially the MA, seems to have significant benefits in terms of speed. However, much further work is warranted in this field, including hybrids of natural computation and traditional approaches.

As noted by Arcuri and Yao, the use of natural computation in software testing has been gaining a research following in recent years. Other examples include Tonella (2004) who used evolutionary algorithms for generating unit tests of Java programs, while Wappler and Wegener (2006) used strongly typed genetic programming (STGP) also for testing Java programs. Seesing (2006) also investigated STGP for a similar purpose, while Liu et al. (2005) used a hybrid approach, involving ant colony optimization (see the chapter [Swarm Intelligence](#) in this volume) to optimize the sequence of function calls, and a multi-agent evolutionary algorithm to optimize the input parameters of those function calls. Meanwhile, Arcuri and Yao's research described in this section began with presenting new encoding and search operators and a dynamic search space reduction method for testing OO containers (Arcuri and Yao 2007), also testing the Estimation of Distribution Algorithms on this problem (Sagarna et al. 2007).

5.3.5 Summary Thoughts

In this section, we have seen an example of how evolutionary computation is beginning to be used in software engineering. The work focused on showed a comparison with a selection of other methods, and also discussed comparisons with standard techniques in the software engineering industry, and found advantages for the evolutionary computation approach in both scenarios. The empirical tests by Arcuri and Yao (2008) showed that their MA usually performs better than the other algorithms tried. However, there remain clear challenges for improvement (e.g., the performance on the TreeMap container was not completely satisfactory).

Arcuri and Yao conclude, based on their results as well as the related literature, that in testing the OO software, nature-inspired algorithms seem to be better than the standard techniques based on symbolic execution and state matching, since they seem able to solve more complex test problems in less time. Arcuri and Yao's work also included the unusual approach of trying to cover every branch at the same time with a single sequence. This has yet to be compared to the traditional approach of testing each branch separately.

6 Concluding Notes

A *selection* of application areas in which natural computation shows its value in real-world enterprises of various sorts has been discussed. This selection has been quite eclectic. Other authors would have chosen a different set; at another time, the same authors would have chosen a different collection. The main message we mean to convey by such statements is that, for the purposes of demonstrating the significant impact and potential of natural computation

in practice, there is certainly no shortage of documented examples that could be selected. We presented here just ten applications, ranging from specific problems to specific domains, and ranging from cases familiar to the authors to highlights known well in the general natural computation community. However, all of them share, one hopes, the property of displaying (each in their own way) a clear indication of the proven promise or great potential for the impact of nature-inspired computation in high-profile and important real-world applications. Similarly, it is hoped that these applications share the property of being inspiring to both students and practitioners; many were selected on the basis of proving particularly popular with students, in the context of getting them interested in the study of natural computation.

When designing a chapter such as this, the first problem one faces is that natural computation is almost too successful in practice. One may ask, for example, why does one not mention more from the thousands of successful real-world applications of neural computation, or fuzzy systems? Well, first of all, they have indeed just been mentioned. But second of all, the positioning of this article in the Broader Perspective part of the Handbook suggests a focus on the novel and unusual; on the less generally known, and on areas whose potential is clear, yet only beginning to be realized in practice.

Naturally, therefore, the center of gravity in this article has turned out to be evolutionary computation. Sandwiched between the more familiar, tried and tested topics of neural and fuzzy systems, and the several emerging areas of natural computation that are currently less “on the map” with application studies, evolutionary computation is a highly flexible child of natural computation that excels in displaying the promise for this field. But, in passing, we saw, in Blondie24, how different areas of nature-inspired computation collaborate to remarkable effect. We have also seen, in the aircraft maneuver study, and in the discussions of super-heuristics, how learning classifier systems – themselves inspired by the adaptive behavior of intelligent organisms – contribute toward natural computing’s expanding gallery of successes. Meanwhile, other chapters in this volume cover some of the successes of swarm intelligence, SA, artificial immune systems, and more.

The real-world value of some of the more established natural computing techniques has been proven to be unquestionably immense. It is worth pointing out that this was never anticipated in the “early days” for each individual technique. In the case of neural computation, for example, Minsky and Papert’s analysis of the capabilities of two-layer networks led (if not deliberately) to much skepticism and delay in the exploration and take-up of neural networks for pattern recognition. In evolutionary computation’s earliest days, the algorithms were usually considered as intellectual curiosities, with the occasional promising application studies considered as one-offs. There seems to be a lesson here for the promise and potential of the several less mature and emerging natural computing ideas – those discussed in this volume, as well as others. In anticipation, one waits and sees.

References

- Allen F, Karjalainen R (1999) Using genetic algorithms to find technical trading rules. *J Financial Econ* 51:245–271
- Angeline PJ (1996) Genetic programming’s continued evolution. In: Angeline PJ, Kinnear K (eds) *Advances in genetic programming*, vol 2. MIT Press, Cambridge, pp 89–110
- Arcuri A, Yao X (2007) A memetic algorithm for test data generation of object-oriented software. In: IEEE congress on evolutionary computation (CEC), Singapore, pp 2048–2055
- Arcuri A, Yao X (2008) Search based software testing of object-oriented containers. *Inf Sci* 178:3075–3095

- Ayob M, Kendall G (2003) A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In: Proceedings of the international conference on intelligent technologies, Chiang Mai, Thailand, pp 132–141
- Banzhaf W, Nordin P, Keller RE, Francone FD (1998) Genetic programming – An introduction: On the automatic evolution of computer programs and its applications. Morgan Kaufmann, San Francisco, CA
- Baresel A, Stahmer H, Schmidt M (2002) Fitness function design to improve evolutionary structural testing. In: Genetic and evolutionary computation conference (GECCO). Morgan Kaufmann, San Francisco, New York, CA, pp 1329–1336
- Becker LA, Seshadri M (2003a) Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Computer Science Technical Report WPI-CS-TR-03-09. Worcester Polytechnic Institute, Worcester, Massachusetts, USA
- Becker LA, Seshadri M (2003b) Cooperative coevolution of technical trading rules. Computer Science Technical Report WPI-CS-TR-03-15. Worcester Polytechnic Institute, Worcester, Massachusetts, USA
- Becker LA, Seshadri M (2003c) GP-evolved technical trading rules can outperform buy and hold. In: Proceedings of sixth international conference on computational intelligence and natural computing, North Carolina, USA, 26–30 September 2003
- Beizer B (1990) Software testing techniques. Van Nostrand Rheinhold, New York
- Bilgin B, Ozcan E, Korkmaz EE (2006) An experimental study on hyper-heuristics and final exam scheduling. In: Proceedings of the 2006 international conference on the practice and theory of automated timetabling, Brno, Czech Republic, pp 123–140
- Box GEP (1957) Evolutionary operation: A method for increasing industrial productivity. *Appl Stat* 6:81–101
- Box G, Hunter W, Hunter J (2005) Statistics for experimenters: design, innovation, and discovery, 2nd edn. Wiley, New York
- Boyapati C, Khurshid S, Marinov D (2002) Korat: Automated testing based on java predicates. In: Proceedings of the international symposium on software testing and analysis (ISSTA). ACM, New York
- Brabazon A, O'Neill M (2005) Biologically inspired algorithms for financial modelling. Natural computing series. Springer, New York
- Branke J, Deb K (2005) Integrating user preferences into evolutionary multi-objective optimization. In: Knowledge incorporation in evolutionary computation. Springer, New York, pp 461–477
- Brockhoff D, Zitzler E (2006) Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: Parallel problem solving from nature – PPSN IX. Lecture notes in computer science, vol 4193. Springer, New York, pp 533–542
- Burke EK, MacCarthy BL, Petrovic S, Qu R (2002) Knowledge discovery in a hyperheuristic for course timetabling using case based reasoning. In: Proceedings of the fourth international conference on the practice and theory of automated timetabling (PATAT'02) Gent, Belgium. Springer, Berlin
- Burke EK, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics an emerging direction in modern search technology. In: Glover F, Kochenberger GA (eds) Handbook of metaheuristics. Springer, New York, pp 457–474
- Buy U, Orso A, Pezze M (2000) Automated testing of classes. In: Proceedings of the international symposium on software testing and analysis (ISSTA), pp 39–48
- Chapman L, Thornes JE, Bradley AV (2002) Sky-view factor approximation using GPS receivers. *Int J Climatol* 22(5):615–621
- Chellapilla K, Fogel DB (1999a) Evolution, neural networks, games, and intelligence. *Proc IEEE* 87(9):1471–1496
- Chellapilla K, Fogel DB (1999b) Evolving neural networks to play checkers without expert knowledge. *IEEE Trans Neural Netw* 10(6):1382–1391
- Chellapilla K, Fogel DB (2001) Evolving an expert checkers playing program without using human expertise. *IEEE Trans Evol Comput* 5(4):422–428
- Chen SH (2002) Genetic algorithms and genetic programming in computational finance. Kluwer, Boston, MA
- Chen SH, Yeh CH (1996) Toward a computable approach to the efficient market hypothesis: an application of genetic programming. *J Econ Dyn Cont* 21:1043–1063
- Cheng SL, Khai YL (2002) GP-based optimisation of technical trading indicators and profitability in FX market. In: Proceeding of the ninth international conference on neural information processing (ICONIP'02), vol 3, Singapore, pp 1159–1163
- Chernoff H (1972) Sequential analysis and optimal design. SIAM monograph. SIAM, Philadelphia, PA
- Coello C (2000) An updated survey of GA-based multi-objective optimization techniques. *ACM Comput Surv (CSUR)* 32(2):109–143
- Coello C (2006) Twenty years of evolutionary multi-objective optimization: A historical view of the field. *IEEE Comput Intell Mag* 1(1):28–36
- Corne D, Jerram N, Knowles J, Oates M (2001) PESA-II: Region-based selection in evolutionary multiobjective optimization. In: Spector L, Goodman ED, Wu A, Langdon WB, Voigt H-M, Gen M, Sen S,

- Dorigo M, Pezeshk S, Garzon MH, Burke E (eds) Proceedings of GECCO-2001: Genetic and evolutionary computation conference. Morgan Kaufmann, San Mateo, San Francisco, CA, pp 283–290
- Corne D, Deb K, Fleming P, Knowles J (2003a) The good of the many outweighs the good of the one: Evolutionary multiobjective optimization. *IEEE Connections Newsletter* 1(1):9–13. ISSN 1543-4281
- Corne D, Oates M, Kell D (2003b) Fitness gains and mutation patterns: Deriving mutation rates by exploiting landscape data. In: De Jong K, Poli R, Rowe J (eds) Foundations of genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 347–364
- Cornford D, Thornes JE (1996) A comparison between spatial winter indices and expenditure on winter road maintenance in Scotland. *Int J Climatol* 16:339–357
- Cowling P, Kendall G, Soubeiga E (2000) A hyperheuristic approach to scheduling a sales summit. In: Burke EK, Erben W (eds) Practice and theory of automated timetabling III: Third international conference, PATAT 2000, Konstanz, Germany, August 2000, selected papers. LNCS, vol 2079. Springer, pp 176–190
- Cowling P, Kendall G, Soubeiga E (2002) Hyperheuristics: A robust optimisation method applied to nurse scheduling. Technical Report NOTTCS-TR-2002-6. School of Computer Science & IT, University of Nottingham, Nottingham, England
- Cross SE, Walker E (1994) Dart: Applying knowledge-based planning and scheduling to crisis action planning. In: Zweben M, Fox MS (eds) Intelligent scheduling. Morgan Kaufmann, San Francisco, CA
- Datta R, Deb K (2009) A classical-cum-evolutionary multi-objective optimization for optimal machining parameters. In: Proceedings of NABIC. IEEE CIS Press
- Davies ZS, Gilbert RJ, Merry RJ, Kell DB, Theodorou MK, Griffith GW (2000) Efficient improvement of silage additives by using genetic algorithms. *Appl Environ Microbiol* April:1435–1443
- Deb K (1997) Mechanical component design using genetic algorithms. In: Dasgupta D, Michalewicz Z (eds) Evolutionary algorithms in engineering applications. Springer, New York, pp 495–512
- Deb K (2000) An efficient constraint handling method for genetic algorithms. *Comput Meth Appl Mech Eng* 1862(4):311–338
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
- Deb K, Kumar A (1995) Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. *Complex Syst* 9(6):431–454
- Deb K, Srinivasan A (2005) Innovization: Innovation of design principles through optimization. KanGAL Report No. 2005007
- Deb K, Srinivasan A (2006) Innovization: Innovating design principles through optimization. In: Proceedings of GECCO. ACM, New York, pp 1629–1636
- Deb K, Agrawal S, Pratap A, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6(2):182–197
- Doong R, Frankl PG (1994) The ASTOOT approach to testing object-oriented programs. *ACM Trans Softw Eng Methodol* 3:101–130
- Ellims M, Bridges J, Ince DC (2006) The economics of unit testing. *Emp Softw Eng* 11(1):5–31
- Evans JRG, Edirisinghe MJ, Eames PVCJ (2001) Combinatorial searches of inorganic materials using the inkjet printer: Science philosophy and technology. *J Eur Ceramic Soc* 21:2291–2299
- Fang H-L, Ross PM, Corne D (1994) A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: Cohn A (ed) Proceedings of ECAI 94: 11th European conference on artificial intelligence. Wiley, Amsterdam, The Netherlands, pp 590–594
- Farnsworth GV, Kelly JA, Othling AS, Pryor RJ (2004) Successful technical trading agents using genetic programming. SANDIA Report SAND2004-4774. SANDIA National Laboratories, California
- Fisher R (1971) The design of experiments, 9th edn. Macmillan, New York
- Fogel D (1998) Evolutionary computation. The fossil record. Selected readings on the history of evolutionary computation. IEEE Press, Piscataway, New Jersey, USA
- Fogel DB (2002) Blondie24: Playing at the edge of AI. Morgan Kaufmann, San Francisco, CA, ISBN 1-55860-783-8
- Fogel DB, Hays TJ, Hahn SL, Quon J (2004) A self-learning evolutionary chess program. *Proc IEEE* 92 (12):1947–1954
- Fogel DB, Hays TJ, Hahn SL, Quon J (2006) The Blondie25 chess program competes against Fritz 8.0 and a human chess master. In: Louis S, Kendall G (eds) Proceedings of 2006 IEEE symposium on computational intelligence & games. IEEE, Reno, pp 230–235
- Fonseca C, Fleming P (1995) An overview of evolutionary algorithms in multiobjective optimization. *Evol Comput* 3(1):1–16
- Fonseca C, Fleming P (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Trans Syst Man Cybernetics – Part A* 28(1):26–37
- Fukunaga A (2008) Automated discovery of local search heuristics for satisfiability testing. *Evol Comput* 16 (1):31–61
- Fyfe C, Marney JP, Tarbert H (1999) Technical trading versus market efficiency: A genetic programming approach. *Appl Finan Econ* 9:183–191

- Gent IP, Walsh T (1993) Towards an understanding of hill-climbing procedures for SAT. In: Proceedings of AAAI'93. AAAI Press/MIT Press, Menlo Park, pp 28–33
- Giffler B, Thompson GL (1960) Algorithms for solving production scheduling problems. *Oper Res* 8(4):487–503
- Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Boston, MA
- Goldberg DE (2002) The design of innovation: Lessons from and for competent genetic algorithms. Kluwer, Boston, MA
- Gratch J, Chein S, de Jong G (1993) Learning search control knowledge for Deep Space network scheduling. In: Proceedings of the tenth international conference on machine learning, Amherst, MA, pp 135–142
- Grefenstette JJ (1988) Credit assignment in rule discovery systems based on genetic algorithms. *Mach Learn* 3:225–246
- Handa H, Chapman L, Yao X (2005) Dynamic salting route optimisation using evolutionary computation. In: Proceedings of the 2005 congress on evolutionary computation, Edinburgh, Scotland, vol 1, pp 158–165
- Handa H, Chapman L, Yao X (2006) Robust route optimization for gritting/salting trucks: A CERCIA experience. *IEEE Comput Intell Mag* February:6–9
- Harman M, Hu L, Hierons R, Baresel A, Stamer H (2002) Improving evolutionary testing by flag removal. In: Genetic and evolutionary computation conference (GECCO). Morgan Kaufmann, San Francisco, New York, CA, pp 1351–1358
- Hart E, Ross PM (1998) A heuristic combination method for solving job-shop scheduling problems. In: Eiben AE, Back T, Schoenauer M, Schwefel H-P (eds) Parallel problem solving from nature V. LNCS, vol 1498. Springer, Berlin, pp 845–854
- Hart E, Ross PM, Nelson J (1998) Solving a real-world problem using an evolving heuristically driven schedule builder. *Evol Comput* 6(1):61–80
- Holland JH (1992) Adaptation in natural and artificial systems. MIT Press, Cambridge, MA, USA
- Holland JH, Holyoak KJ, Nisbett RE, Thagard PR (1986) Induction: Processes of inference, learning, and discovery. MIT Press, Cambridge
- Hornby GS, Globus A, Linden DS, Lohn JD (2006) Automated antenna design with evolutionary algorithms. In: AIAA Space, San Jose, CA
- Hunter WG, Kittrell JR (1966) Evolutionary operation: A review. *Technometrics* 8(3):389–397. Available at: <http://www.jstor.org/stable/1266686>
- Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *ASME J Mech Des* 116(2):405–411
- Kerlogue F, Zanetini F (2004) Batik: Design, style and history. Thames and Hudson, London
- King JC (1976) Symbolic execution and program testing. *Commun ACM* 19:385–394
- Knight CG, Platt M, Rowe W, Wedge DC, Khan F, Day PJ, McShea A, Knowles J, Kell DB (2008) Array-based evolution of DNA aptamers allows modelling of an explicit sequence-fitness landscape. *Nucleic Acids Research November:e6*
- Knowles J (2006) ParEGO: A hybrid algorithm with online landscape approximation for expensive multi-objective optimization problems. *IEEE Trans Evol Comput* 10(1):50–66
- Knowles JD (2009) Closed-loop evolutionary multiobjective optimization. *IEEE Comput Intell Mag August*:77–91
- Korel B (1990) Automated software test data generation. *IEEE Trans Softw Eng* 16:870–879
- Koza JR (1992) Genetic programming: On the programming of computers by means of natural selection. MIT Press, Cambridge
- Lacomme P, Prins C, Ramdane-Cherif W (2004) Competitive memetic algorithms for arc routing problems. *Ann Oper Res* 131:159–185
- Lau TWE, Ho Y-C (1999) Super-heuristics and their application to combinatorial problems. *Asian J Cont* 1(1):1–13
- Li Y, Hu CJ, Yao X (2009) Innovative Batik design with an interactive evolutionary art system. *J Comput Sci Technol* 24(6):1035–1047
- Liu X, Wang B, Liu H (2005) Evolutionary search in the context of object oriented programs. In: MIC2005: The sixth metaheuristics international conference, Vienna, Austria
- Lo AW, Mamaysky H, Wang J (2000) Foundations of technical analysis: Computational algorithms, statistical-inference, and empirical implementation. *J Finance* 55:1705–1770
- Lohpetch D, Corne D (2009) Discovering effective technical trading rules with genetic programming: Towards robustly outperforming buy-and-hold. In: World congress on nature and biologically inspired computing (NABIC). IEEE Press
- Lohpetch D, Corne D (2010) Outperforming buy-and-hold with evolved technical trading rules: Daily, weekly and monthly trading. In: EvoApplications. Proceedings of EvoStar 2010. LNCS, vol 6025. Springer pp
- Lutton E (2006) Evolution of fractal shapes for artists and designers. *Int J Artif Intell Tools* 15(4):651–672
- Marinov D, Khurshid S (2001) TestEra: A novel framework for testing java programs. In: IEEE international conference on automated software engineering (ASE), San Diego, California, USA. Kluwer, The Netherlands.
- Marney JP, Miller D, Fyfe C, Tarbert H (2000) Technical analysis versus market efficiency: A genetic

- programming approach. *Computing in Economics and Finance*, Society for Computational Economics, Barcelona, Spain (paper #169)
- Marney JP, Fyfe C, Tarbert H, Miller D (2001) Risk adjusted returns to technical trading rules: A genetic programming approach. *Computing in Economics and Finance*, Society for Computational Economics, Yale University, USA
- Marney JP, Tarbert H, Fyfe C (2005) Risk adjusted returns from technical trading: A genetic programming approach. *Appl Financial Econ* 15: 1073–1077
- McAllester D, Selman B, Kautz H (1997) Evidence for invariants in local search. In: *Proceedings of the 14th national conference on artificial intelligence*. AAAI Press/MIT Press, Menlo Park, Providence, Rhode Island, USA, pp 321–326
- McMinn P (2004) Search-based software test data generation: A survey. *Softw Test Verif Reliab* 14 (2):105–156
- McMinn P, Holcombe M (2003) The state problem for evolutionary testing. In: *Genetic and evolutionary computation conference (GECCO)*, Chicago, Illinois, USA, pp 2488–2500
- McMinn P, Holcombe M (2004) Hybridizing evolutionary testing with the chaining approach. In: *Genetic and evolutionary computation conference (GECCO)*, Seattle, Washington, USA, pp 1363–1374
- McMinn P, Holcombe M (2005) Evolutionary testing of state-based programs. In: *Genetic and evolutionary computation conference (GECCO)*, Washington, DC, USA, pp 1013–1020
- Messac A, Mattson CA (2004) Normal constraint method with guarantee of even representation of complete Pareto frontier. *AIAA J* 42(10):2101–2111
- Miettinen K (1999) Nonlinear multiobjective optimization. Springer, New York
- Minton S (1988) Learning search control knowledge: An explanation-based approach. Kluwer Academic Publishers Norwell, MA, USA
- Murphy JJ (1999) Technical analysis of the financial markets. New York Institute of Finance, New York
- Myers G (1979) The art of software testing. Wiley, New York
- Myers R, Montgomery D (1995) Response surface methodology: Process and product optimization using designed experiments. Wiley, New York
- Nagata Y, Kobayashi S (1997) Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In: *Proceedings of the seventh international conference on genetic algorithms*, East Lansing, Michigan, USA, pp 450–457
- Neely C (2001) Risk-adjusted, ex ante, optimal technical trading rules in equity markets. Working Papers 99-015D, Revised August 2001, Federal Reserve Bank of St. Louis
- Nolfi S, Floreano D (2004) Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. Bradford Book. MIT Press, Cambridge
- O'Hagan S, Dunn WB, Brown M, Knowles JD, Kell DB (2005) Closed-loop, multiobjective optimization of analytical instrumentation: Gas chromatography/time-off light mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Anal Chem* 77(1):290–303
- O'Hagan S, Dunn W, Knowles J, Broadhurst D, Williams R, Ashworth J, Cameron M, Kell D (2007) Closed-loop, multiobjective optimization of two-dimensional gas chromatography/mass spectrometry for serum metabolomics. *Anal Chem* 79(2):464–476
- Özcan E, Bilgin B, Korkmaz EE (2008) A comprehensive analysis of hyper-heuristics. *Intell Data Anal* 12 (1):3–23
- Potvin JY, Soriano P, Vallée M (June 2004) Generating trading rules on the stock markets with genetic programming. *Comput Oper Res* 31(7):1033–1047
- Pring MJ (1980) Technical analysis explained. McGraw-Hill, New York
- Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, Farnborough, Hampshire, UK
- Rechenberg I (2000) Case studies in evolutionary experimentation and computation. *Comput Meth Appl Mech Eng* 186(2–4):125–140
- Reklaitis GV, Ravindran A, Ragsdell KM (1983) Engineering optimization methods and applications. Wiley, New York
- Romero J, Machado P (2008) The art of artificial evolution: a handbook on evolutionary art and music. Springer, Heidelberg
- Ross P, Hart E, Corne D (1997) Some observations about GA-based exam timetabling. In: Burke EK, Carter M (eds) *Practice and theory of automated timetabling II: Second international conference, PATAT 1997*, Toronto, Canada, August 1997, selected papers. LNCS, vol 1408. Springer, pp 115–129
- Ross P, Schulenburg S, Marín-Blázquez JG, Hart E (2002) Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In: *Genetic and evolutionary computation conference (GECCO 2002)*, New York
- Ross P, Marín-Blázquez JG, Schulenburg S, Hart E (2003) Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. In: *Proceedings of the genetic and evolutionary computation conference (GECCO 2003)*. Lecture notes in computer science, vol 2723. Springer, Chicago, Illinois, USA, pp 1295–1306
- Ruggiero MA (1997) Cybernetic trading strategies. Wiley, New York

- Russell SJ, Norvig P (2003) Artificial intelligence: A modern approach, 2nd edn. Prentice Hall, Upper Saddle River, NJ, pp 163–171
- Sagarna R, Arcuri A, Yao X (2007) Estimation of distribution algorithms for testing object oriented software. In: IEEE congress on evolutionary computation (CEC), Singapore, pp 438–444
- Samuel AL (1959) Some studies in machine learning using the game of checkers. *IBM J Res Develop* 3:210–219
- Schaeffer J (1996) One jump ahead: Challenging human supremacy in checkers. Springer, New York, p 97, 447
- Schaeffer J, Lake R, Lu P, Bryant M (1996) Chinook: The world man–machine checkers champion. *AI Mag* 17:21–29
- Seising A (2006) Evotest: Test case generation using genetic programming and software analysis. Master's thesis, Delft University of Technology
- Selman B, Levesque HJ, Mitchell DG (1992) A new method for solving hard satisfiability problems. In: Tenth AAAI, San Jose, pp 440–446
- Selman B, Kautz HA, Cohen B (1994) Noise strategies for improving local search. In: Proceedings of the 12th national conference on artificial intelligence. AAAI Press/MIT Press, Menlo Park, Seattle, Washington, USA, pp 337–343
- Sharpe WF (1966) Mutual fund performance. *J Business* 39(S1):119–138. doi:10.1086/294846
- Shaw RL (1998) Fighter combat: Tactics and maneuvering. United States Naval Institute Press, Annapolis, Maryland, USA
- Sims K (1991) Artificial evolution for computer graphics. In: Proceedings of the 18th annual conference on computer graphics and interactive techniques (SIGGRAPH 1991). ACM, New York, Las Vegas, Nevada, USA, pp 319–328
- Smith RE, Dike BA (1995) Learning novel fighter combat maneuver rules via genetic algorithms. *Int J Expert Syst* 8(3):247–276
- Smith RE, Dike BA, Mehra RK, Ravichandran B, El-Fallah A (2000) Classifier systems in combat: Two-sided learning of maneuvers for advanced fighter aircraft. *Comput Meth Appl Mech Eng* 186:431–437
- Smith RE, Dike BA, Ravichandran B, El-Fallah A, Mehra RK (2002) Discovering novel fighter combat maneuvers: Simulating test pilot creativity. In: Bentley P, Corne D (eds) Creative evolutionary systems. Morgan Kaufmann, San Francisco, CA, pp 467–486
- Tassey G (2002) The economic impacts of inadequate infrastructure for software testing. Final Report. National Institute of Standards and Technology
- Terashima-Marín H, Ross PM, Valenzuela-Rendón M (1999) Evolution of constraint satisfaction strategies in examination timetabling. In: Banzhaf W et al. (eds) Proceedings of the GECCO-99 genetic and evolutionary computation conference, Orlando, Florida. Morgan Kaufmann, San Francisco, pp 635–642
- Thompson A, Layzell P (1999) Analysis of unconventional evolved electronics. *Commun ACM* 42(4):71–79
- Tonella P (2004) Evolutionary testing of classes. In: Proceedings of the international symposium on software testing and analysis (ISSTA), pp 119–128
- Trianni V, Nolfi S, Dorigo M (2006) Cooperative hole avoidance in a swarm-bot. *Robot Autonomous Syst* 54(2):97–103
- Tuerk C, Gold L (1990) Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T₄ DNA polymerase. *Science* 249(4968):505
- Visser W, Pasareanu CS, Khurshid S (2004) Test input generation with java pathfinder. In: Proceedings of the international symposium on software testing and analysis (ISSTA), Boston, Massachusetts, USA
- Visser W, Pasareanu CS, Pelánek R (2006) Test input generation for java containers using state matching. In: Proceedings of the international symposium on software testing and analysis (ISSTA), Portland, Maine, USA, pp 37–48
- Wang Y, Tan T, Zhu Y (2000) Face verification based on singular value decomposition and radial basis function neural network. In: Proceedings of fourth Asian conference on computer vision, Taiwan, pp 432–436
- Wang SF, Wang S, Takagi H (2006) User fatigue reduction by an absolute rating data-trained predictor in IEC. In: Proceedings of 2006 congress on evolutionary computation, pp 2195–2200
- Wappler S, Wegener J (2006) Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. In: Genetic and evolutionary computation conference (GECCO), Seattle, Washington, USA, pp 1925–1932
- Wedge D, Rowe W, Kell D, Knowles J (2009) In silico modelling of directed evolution: Implications for experimental design and stepwise evolution. *J Theor Biol* 257:131–141
- Wilson S (1998) Generalisation in the XCS classifier system. In: Koza J (ed) Proceedings of the third genetic programming conference, Madison, Wisconsin. Morgan Kaufmann, San Francisco, CA, USA, pp 665–674
- Xie T, Marinov D, Notkin D (2004) Rostra: A framework for detecting redundant object-oriented unit tests. In: IEEE international conference on automated software engineering (ASE), Linz, Austria. IEEE Computer Society, Washington, DC, pp 196–205
- Xie T, Marinov D, Schulte W, Notkin D (2005) Symstra: A framework for generating object-oriented unit tests using symbolic execution. In: Proceedings of the 11th international conference on tools and algorithms for the construction and analysis of systems, Edinburgh, UK, pp 365–381

Section VII

Broader Perspective – Alternative Models of Computation

David W. Corne

53 Artificial Life

Wolfgang Banzhaf¹ · Barry McMullin²

¹Department of Computer Science, Memorial University
of Newfoundland, St. John's, NL, Canada

banzhaf@cs.mun.ca

²Artificial Life Lab, School of Electronic Engineering, Dublin City
University, Ireland
barry.mcmullin@dcu.ie

1	<i>Introduction and Historical Overview</i>	1806
2	<i>Fundamental Questions</i>	1808
3	<i>Theory of and Formalisms in Artificial Life</i>	1817
4	<i>Applications</i>	1822
5	<i>Conclusion</i>	1826

Abstract

Artificial life has now become a mature inter-discipline. In this contribution, its roots are traced, its key questions are raised, its main methodological tools are discussed, and finally its applications are reviewed. As part of the growing body of knowledge at the intersection between the life sciences and computing, artificial life will continue to thrive and benefit from further scientific and technical progress on both sides, the biological and the computational. It is expected to take center stage in natural computing.

1 Introduction and Historical Overview

The phrase *artificial life*, referring to a specific coherent research programme, is normally attributed to Chris Langton. He specifically applied the term as a title for the “interdisciplinary workshop on the synthesis and simulation of living systems” that he organized in September 1987, in Los Alamos, New Mexico (Langton 1989). It was then adopted for a biannual international conference series (now up to *Artificial Life XI*, held in Winchester UK, in August 2008), and alternating with the biannual *European Conference on Artificial Life* (ECAL), first held in Paris in 1991. The *Artificial Life journal* (with Langton as first Editor-in-Chief) was founded in 1993. The journal and the two conference series are now formally coordinated through the International Society for Artificial Life (ISAL) (<http://www.alife.org>), which was established in 2001.

Langton’s original announcement of the 1987 workshop defined the new field as follows:

- ▶ Artificial life is the study of artificial systems that exhibit behavior characteristic of natural living systems. This includes computer simulations, biological and chemical experiments, and purely theoretical endeavors. Processes occurring on molecular, cellular, neural, social, and evolutionary scales are subject to investigation. The ultimate goal is to extract the logical form of living systems. (Langton 1987)

It is clear that, from the outset, Langton’s vision was for a very broad, thoroughly interdisciplinary, endeavor, and indeed that is how the field has largely developed. There has been a consistent focus on using tools of computer science to model and simulate biological systems; but also a specific goal of using *synthesis* to start an exploration of the space of *possible* life. Indeed, Langton’s own first use of the phrase “artificial life” was in his paper “*Studying artificial life with cellular automata*” (Langton 1986). This was presented two years before the first artificial life workshop, at the conference *Evolution, Games, & Learning* organized by the Los Alamos Center for Non-Linear Studies (CNLS) in May 1985. In this paper, Langton was already introducing “artificial life” as a direct lineal descendant and inheritor of von Neumann’s seminal research programme, begun in the late 1940s, combining “automata theory” with problems of biological organization, self-reproduction, and the evolution of complexity (von Neumann 1949, 1951, 1966).

But while von Neumann’s influence was certainly strong, there were many other contributions and precursors to the field. The phrase “artificial life” is, in part, a deliberate play on “Artificial Intelligence” or AI. The original premise of AI, dating from Turing’s 1950 paper on “*Computing machinery and intelligence*” (Turing 1950), was that “intelligence” was an intrinsically computational or software phenomenon. It could therefore be divorced from

any particular underlying hardware implementation, including all the particular biological details of terrestrial life. By the 1980s however, after 30 years of intensive and well-resourced efforts, AI was still showing little clear progress toward its central goal of realizing human-level intelligence by means of computer programs. Indeed, as early as 1977, Popper was already declaring:

- ▶ “... computers are totally different from brains, whose function is not primarily to compute but to guide and balance an organism and help it to stay alive. It is for this reason that the first step of nature toward an intelligent mind was the creation of life, and I think that should we artificially create an intelligent mind, we would have to follow the same path. (Popper and Eccles 1977)

In 1988, the mounting challenges and criticisms of GOFAI (“good old fashioned AI” (Haugeland 1989)), were summarized in a special edition of *Daedalus*, the Journal of the American Academy of Arts and Sciences (Graubard 1988). This documented a renaissance of interest in the biological underpinnings of intelligence. Artificial life can therefore be seen as just one of the particular results of this splintering of AI, along with subsumption-architecture robotics (Brooks 1985), parallel distributed processing (Rumelhart et al. 1986) (the “new” connectionism), Neural Darwinism (Reeke and Edelman 1988; Edelman 1987), embodied intelligence (Varela et al. 1992), etc. And indeed, in some ways, artificial life continues to function as an integrative term, bridging between these more specialized and disparate research programmes.

A specific debate that artificial life has inherited from its roots in AI, is that between “weak” and “strong” forms. In the AI case, a distinction is made between the claim that an AI system provides a more or less effective *simulation* of intelligence, and the claim that it is “really” intelligent. The distinction is particularly associated with the philosopher John Searle, whose famous “Chinese Room Argument” claimed to prove that however well digital computers might simulate intelligence, computation could never be sufficient to genuinely *realize* intelligence (Searle 1980). Clearly an analogous distinction can be drawn within artificial life, between merely simulating and actually realizing “real” life (Levy 1993). In that case, it is generally focused on whether “life-like” organization, which exists only in a purely computational/virtual world, could ever properly be described as genuinely living. However, Searle’s abstract philosophical argument cannot be transferred directly into the artificial life domain as it relies on conscious introspection. Perhaps because of this, the issue has received comparatively less intense attention in this field. In any case, while much artificial life research is carried out in computational/virtual worlds, and thus might be subject to this kind of debate, it is generally considered that work in so-called wet artificial life, working *in vitro* with real chemical systems (e.g., Luisi and Varela 1989; Rasmussen et al. 2008), is certainly immune to this kind of critique.

Other significant influences and inspirations for artificial life include morphogenesis (Turing 1952), cybernetics (Wiener 1965), general systems theory (Bertalanffy 1976), mathematical/relational biology (Rashevsky 1940), theoretical biology (Waddington 1969), self-organization (Yovits and Cameron 1960), hierarchy theory (Pattee 1973), autopoiesis (Maturana and Varela 1973), and genetic and evolutionary algorithms (Fogel et al. 1966; Holland 1975; Rechenberg 1975).

A specific precursor to the modern artificial life conference series was the conference on *Automata, Languages, Development* (Lindenmayer and Rozenberg 1976) organized by Rozenberg and Lindenmayer, March–April 1975 in Noordwijkerhout, Holland. One of the

participants, Alvy Ray Smith, has subsequently commented that (<http://alvyray.com/Papers/PapersCA.htm>):

- ▶ I like to call this conference ALife0 — for Artificial Life 0 conference — since it was the first attempt I know to attempt cross-fertilization between biologists and computer scientists. Many of the players at this conference were present for ALife1, ALife2, etc [...] Other participants at ALife0 were Karel Culik, Pauline Hogeweg, John Holland, Aristid Lindenmayer, and Stanislaw Ulam.

In summary, artificial life is now a mature and well-established discipline (or “inter-discipline”) in its own right. The following sections will present the fundamental questions it is concerned with, a selection of specific theories and formalisms it relies on, and we review some applications and conclude with a discussion of the key open problems for future research.

2 Fundamental Questions

2.1 What Is (Artificial) Life?

- ▶ By extending the horizons of empirical research in biology beyond the territory currently circumscribed by life-as-we-know-it, the study of Artificial Life gives us access to the domain of life-as-it-could-be, and it is within this vastly larger domain that we must ground general theories of biology and in which we will discover practical and useful applications of biology in our engineering endeavors. (Langton 1995)

The question of “what is life?” is clearly fundamental to a discussion of the field of artificial life, as the answer (or answers) would be expected to underpin the scope, problems, methods, and results of its investigations.

The starting point is, of course, natural life, the domain of conventional biology. Historically, it was conjectured that the very distinctive properties and characteristics of natural living systems reflected a fundamental *constitutional* distinction: that living systems incorporated (or were literally “animated by”) a distinct and unique “substance” – the so-called *elan vital* that identified this class of theories as forms of *vitalism*. However, from the pioneering inorganic synthesis of urea by Wöhler in 1828 onward, vitalism progressively lost credibility. With the explosive growth of molecular understanding of basic biological mechanisms in the second half of the twentieth century (traditionally dated from the elucidation of the chemical structure of DNA in 1953), vitalism in this sense has now all but disappeared from scientific consideration. There is now a clear scientific consensus that there is no *material* distinction between the domains of living and nonliving systems; rather the differences (which are, of course, still objectively real and substantial) arise from differences in material organization.

However, it is still very much an open question as to what exactly these differences in organization *are*. Conventional molecular biology continues to make rapid progress in teasing out the detailed molecular basis for particular low-level mechanisms of operation in natural organisms; but by focusing on dismantling organisms into molecular components, these investigations, though still valuable in themselves, risk eliminating precisely the higher order organization that is distinctive of natural life. More recently, and partly in reaction to this, there has been growing interest in what is termed *systems biology*: The explicit study of how low-level molecular mechanisms are successfully integrated into higher level “systems” (Wolkenhauer 2007); and also, in *synthetic biology* that investigates the synthesis of entirely novel classes of living systems, either by manipulation of natural organisms and their components

(Smith et al. 2003; Regis 2008) or even by attempting de novo creation of life from entirely nonbiological substrates (Rasmussen 2008).

All these strands of investigation are still essentially premised on the study of existing natural life on Earth; or, as Langton puts it, “life-as-it-is” rather than “life-as-it-could-be” (Langton 1992b) (even synthetic biology, in its current state, largely relies on preexisting biological components, or close analogues of them).

By contrast, the field of artificial life, from its inception, has had a specific goal of investigating the most general possible formulations and instantiations of “living phenomenology.” In particular, it has explicitly brought to bear the possibility of creating computer-synthesized universes, or *virtual worlds*, in which even the most basic laws of “physics” or “chemistry” can be arbitrarily manipulated, in whatever ways may prove most conducive to any particular study.

Of course, this lack of hard boundaries makes “artificial life,” as a field of study, significantly ill-defined. Unlike the case for natural life, there are, as yet, no clear criteria for what virtual world phenomena should qualify as “living” or sufficiently “life-like” to legitimately count as lying within this field. In large measure, this simply reflects the continuing debate and investigation within conventional biology, of what specific organizational (as opposed to material) system characteristics are critical to properly living systems. The key advantage and innovation in artificial life is precisely that it has this freedom to vary and explore possibilities that are difficult or impossible to investigate in natural living systems. In this context, a precise definition of “life” (natural or artificial) is not a necessary, or even especially desirable condition for progress.

2.2 Hierarchy and Emergence

A specific and pervasive feature of natural life is hierarchical organization – from molecules to cells to tissues to organs to multicellular organisms to societies of organisms. Hierarchical design is also common in artificial engineered systems – for example, from semiconductor materials to logic gates to processors to computers to the Internet. At first sight, the hierarchical structure of natural living systems might be viewed as simply the natural outcome of evolutionary optimization for certain engineering benefits: modularity, standardization, ease of repair, robustness, etc. However, there are significant difficulties in such a simplistic account. In particular, evolution by natural selection must operate without foresight: So the opening up of evolutionary “potential” per se, via a new hierarchical level of organization, cannot be a direct locus for immediate selection to operate. The fundamental question here relates to the emergence and establishment of new levels of Darwinian “actors.” This is a process that has evidently occurred repeatedly in natural evolution (the origin of the first protokaryotes; the origin of the eukarotes; multicellularity, sociality, etc.); however, without circular reasoning, this cannot itself simply be assigned to a process of Darwinian selection (not, at least, at the same hierarchical level). In natural biology, this is the focus of investigation into so-called *major transitions in evolution* (Maynard Smith and Szathmáry 1997).

In the case of artificial life, while there have been some examples of model systems demonstrating some degree of emergence of hierarchical organization (e.g., Rasmussen et al. 2001b), the achievement to date has been limited, both in terms of evolutionary potential and in number of hierarchical levels, and is the subject of significant ongoing debate within the

community (Gross and McMullin 2001; Rasmussen et al. 2001a). This can be expected to be a particularly active area of continuing research (Baas 1994).

2.3 Constructive and Autopoietic Systems

A problem closely associated with emergence and hierarchy, yet somewhat distinct, is that of “self-construction” systems. Natural life, at all hierarchical levels, exhibits a characteristic ability to maintain its system organization while simultaneously turning over components at the lower hierarchical levels. Thus, a cell can stably maintain itself (and even grow and divide) while continuously regenerating all its significant molecular components (metabolizing environmental substrates as necessary); similarly, a multicellular organism maintains its organization while constantly replacing its component cells; and a social insect colony may long outlive most, if not all, of its constituent individual organisms. While this is a typical phenomenon of natural life, it is very different from the behavior of conventional engineered systems. The latter may well have complex hierarchical organization, but this organization is typically static: The system is assembled once, and then retains its fixed compositional structure for its functional lifetime. Indeed, it is typical of engineered systems that failure of any component, at any hierarchical level, will at least significantly impair system level function, and will commonly result in full system failure. Although so-called “fault-tolerant” systems may be engineered to incorporate significant redundancy, this often involves static assembly of additional components at manufacture time. Functional lifetime is extended as failing components are automatically removed from service; but once this pre-built redundancy is exhausted, system failure follows from any further component failure. Further, even such redundant systems are generally vulnerable to failures in the “failure-detecting” components that cannot also simply be duplicated without potentially entering into infinite regress.

This is a complex problem to address, but natural life provides a “proof of principle” that such self-constructing organizations are possible. Artificial life certainly offers a potentially very fruitful avenue for further investigation. In particular, artificial life allows “virtual worlds” to be formulated in which problems of self-construction can be simplified, and the core organizational mechanisms can be exposed and subjected both to mathematical analysis and experimental exploration. While a variety of work in artificial life bears on this, some of the most direct contributions might be summarized as follows:

- The Chilean biologists Maturana and Varela pioneered the abstract concept of *autopoiesis* (literally “self-production”) as a description of the core mechanism of organizational self-maintenance in biological cells (Varela et al. 1974). Almost 15 years before the modern computational study of artificial life was even named (Langton 1989), they were already using a molecular level, agent-based, abstract chemistry to give a concrete demonstration of this abstract theory of cellular organization. In essence, the proposal is that biological cells are dynamic, self-sustaining, chemical networks (an abstraction of cell metabolism), which also create and maintain a spatial boundary (an abstraction of a cell membrane), where the contained network and the boundary reciprocally rely on each other for stability. While some technical deficiencies were identified in the original presentation of this work (McMullin and Varela 1997), it has given rise to a sustained and continuing programme of active research (McMullin 2004).

- Also in the 1970s, Holland proposed the so-called α -universes (Holland 1976) and, independently, Hofstadter described the *typogenetics* system (Hofstadter 1979). Both involved one-dimensional fragments of computer code, which could interact with each other. In both cases, they were explicitly inspired by aspects of the molecular replication and translation machinery of biology, and were concerned with understanding the reflexive interactions that arise when the “same” class of entities (molecules) can sometimes function as executable “code” (enzyme/protein) and sometimes as literal “data” (nucleic acid), and the possibility of collective self-reproduction and maintenance. These were, however, restricted to theoretical/analytic treatments, and were not subject to empirical investigation until much later (McMullin 1992; Snare 1999; Kvasnicka et al. 2001).
- This same tension between syntax and semantics in materially instantiated dynamic systems is also at the core of Pattee’s analysis of what he terms “semantic closure” (Rocha 2000; Cariani 1992).
- A related, but independent line of investigation was pioneered by Kauffman and others, in the form of “collectively autocatalytic sets” (Farmer et al. 1986; Kauffman 1993). Such a set is formally similar (though more strictly defined) to the contained chemical network of an autopoietic system, but without the requirement for a self-generated spatial boundary. The key result here was the demonstration, in completely abstract virtual “chemistries” that such collectively autocatalytic sets can spontaneously arise, quickly and under relatively weak constraints on the underlying chemistry.
- Fontana and Buss self-consciously launched a mathematical and computational investigation of “constructive dynamical systems,” based around the so-called *alchemy* system (Fontana and Buss 1994). These systems deliberately diverge from the classical concept of “dynamical systems” by relaxing the normally strict demarcation between “state variable” and “dynamic law.” The authors demonstrated and analyzed a number of organizational phenomena, including forms of “self-sustaining closure” where more-or-less complex dynamic aggregates of components successfully sustain themselves, even as all the individual components are continuously diluted or degraded.
- As a comprehensive overview treatment, Dittrich et al. (2001) provide a summary of the general approach of building computational models of arbitrary “artificial chemistries,” as a platform for investigating these and other problems. More recently, Dittrich and Speroni have developed a primarily algebraic mathematical analysis in a comprehensive way, in the form of “*Chemical organisation theory*” (Dittrich and di Fenizio 2007). However, while an important advance, the difficulties of incorporating dynamics (chemical kinetics) and the interaction of chemical self-production and spatial demarcation (the self-constructed “boundary” of autopoietic theory) remain substantial.
- A quite different and more radical approach to the problem of biological self-maintenance was proposed by the theoretical biologist Robert Rosen over an extended series of works (Rosen 1959, 1972, 1985, 1991). In brief, Rosen argues that the self-constructing organization characteristic of natural living systems, which he termed “closure under efficient causation” transcends the possibilities of conventional dynamical systems in a fundamental way; with the consequence that it cannot be realized in any “computational” universe. This is a complex and contentious theoretical claim which, not surprisingly, continues to be the focus of considerable critique and criticism (e.g., Chu and Ho 2006). Rosen’s work was also a strong influence in Kampis’ development of a comprehensive mathematical treatment of what he terms “self-modifying” systems (Kampis 1991).

2.4 Complexity (and Its Growth)

- ▶ There is a concept which will be quite useful here, of which we have a certain intuitive idea, but which is vague, unscientific, and imperfect . . . I know no adequate name for it, but it is best described by calling it “complication.” It is effectivity in complication, or the potentiality to do things. I am not thinking about how involved the object is, but how involved its *purposive operations* are. In this sense, an object is of the highest degree of complexity if it can do very difficult and involved things. (von Neumann 1949, p. 78, emphasis added)

Arguably the most fundamental problem in the theory of biology is that of the growth of “complexity”; with “complexity” understood in the sense explained by von Neumann above, of the ability to do complicated things. While this is certainly an informal definition, it suffices to clearly demarcate this notion from purely syntactic, combinatorial, or computational concepts of complexity.

Von Neumann is a key figure in the early development of abstract computational modeling approaches to understanding biological phenomena; indeed, in this respect he can be considered as having instigated the earliest investigations in artificial life, as the term is now understood (McMullin 2000a). In particular, in the late 1940s, (von Neumann 1949) he started to draw attention to an apparent paradox arising from the contrast between any mechanistic (i.e., not vitalistic) understanding of living organisms and common experience of engineering artificial mechanisms or automata. While it is quite generally possible to design machines that construct other machines, this process is normally *degenerative in complexity*: a machine of a given complexity (such as an automated factory) can only construct machines of comparatively lower complexity (such as cars or phones or televisions etc.). Whereas, if the theory of Darwinian evolution is accepted, then biological “machines” (organisms) must be capable of constructing descendant machines of greater complexity. Granted, these increases in complexity may occur only in very small increments, and even then only in some lineages and accumulating over geological periods of time; but nonetheless it must be possible, in principle, for biological organisms to give rise to offspring more complex than themselves.

Von Neumann quickly developed an outline resolution of this paradox, in the abstract form of what he termed a “general constructive automaton.” This was inspired by Turing’s earlier formulation of a universal computing automaton (Turing 1936), and also by von Neumann’s own contemporaneous involvement in designing and building the earliest electronic stored program digital computers (von Neumann 1945). In essence, a general constructive automaton is a *programmable* constructor or assembler, capable of constructing an indefinitely large set of target automata – loosely, a “universal” set comprised of all automata that could be assembled from a given, finite, set of primitive components. Significantly, von Neumann hypothesized that a general constructive automaton, relative to a specific set of components, might itself be realizable as an assembly of these same components. With this conceptual architecture and some relatively minor technical elaboration, von Neumann showed that this would then give rise to an indefinitely large set of automata, spanning an indefinite large range of complexity, all of which would be fully connected by a network of heritable “mutations” (essentially, chance variations in the construction programs). That is, there would exist mutational pathways leading from the simplest to arbitrarily complex members of this set; with all these machines also, incidentally, being capable of self-reproduction. (While terminology in the field is not completely consistent, in this

chapter “(self-)reproduction” is reserved to mean this von Neumann style process, involving separate actions of copying and decoding; whereas “(self-)replication” is used to denote a process of copying only.)

In retrospect, this was already an astonishing achievement. Von Neumann effectively described the abstract architecture of biological self-reproduction, based on separate processes of syntactic copying (replication) and semantic “decoding” (translation) of a passive information carrier; he showed how this architecture supported self-reproduction, heritable mutation, and thus evolutionary growth of complexity; and he presented all these, at least in outline form, in 1948, five years before the chemical structure of DNA was even identified (von Neumann 1949).

However, as yet this was still only a sketch of a solution. To make it properly convincing, he needed to present a concrete set of “primitive parts” and show that with these, it would actually be possible to realize an example of a general constructive automaton. This proved to be a complicated and potentially intractable problem: any plausible “real world” set of components would introduce many ancillary complications – mechanics, thermodynamics, etc. Following a suggestion from Stanislaw Ulam, von Neumann instead formulated an artificial, virtual, world that would simplify away these ancillary complications. He proposed a two-dimensional “tessellation automaton,” or, as it is now called, a *cellular automaton* (see [Sect. 3.1.1](#) for a general introduction to CA). Within this simplified virtual world, von Neumann then successfully developed a fully detailed design for an example of a general constructive automaton. He had planned this as a first step in a much more general and comprehensive “theory of automata.” However, he put his unfinished manuscript aside in 1953; and due to his untimely death in 1957, was never to return to this project. The manuscript was subsequently edited by Burks and formally published in 1966 (Burks 1966).

Von Neumann’s achievement, both in theoretical insight and technical implementation, was considerable, and it gave rise to an extended program of further research. Many different cellular automata worlds have been formulated and investigated, and many variants on von Neumann’s architecture for evolvable self-reproducing automata have been demonstrated (Thatcher 1970; Codd 1968; Langton 1984b). Conway’s *Game of Life* has particularly popularized the study of life-like phenomena in cellular automata worlds (Berlekamp et al. 1982). More recently, it has become feasible to actually implement von Neumann’s original designs on real computers (Pesavento 1995).

Nonetheless, seminal as this work was, it also explicitly left certain questions still very much unresolved. In particular, as von Neumann himself was clearly aware, the mere existence of mutational pathways from simple to complex automata does not mean that evolution will actually follow such pathways. Indeed, in von Neumann’s original cellular automaton world, the embedded self-reproducing machine was extremely fragile. It could only operate successfully at all on the condition that the rest of the world was completely quiescent; and, of course, once it completes one round of reproduction, that could no longer be the case. Parent and offspring would then interfere with each other’s structure and operation, and both would rapidly disintegrate. While this problem could be superficially avoided by constraining each complete, successively constructed automaton to restrict its further operation to separate regions of the space, it rules out any possibility of natural selection; and, in any case, cannot be sustained in the long term if there is any possibility of stochastic, component level, malfunction, or failure (which itself is actually *desirable*, as a source of the random variation that is the grist to the mill of Darwinian evolution).

2.5 Coreworlds: Spontaneous Evolution of Computer Programs

- ▶ Nothing in Biology makes sense except in the light of evolution. (Dobzhansky 1973)
- ▶ Discovering how to make such self-reproducing patterns more robust so that they can evolve to increasingly more complex states is probably the central problem in the study of artificial life. (Farmer and d'A. Belin 1992)

Although the cellular automata models of von Neumann and his successors have not, to date, demonstrated extended evolutionary dynamics, various other artificial life models have addressed evolution more directly, through attempts to demonstrate evolution among populations of virtual (software) agents in virtual worlds. The most systematically investigated framework is to envisage these as small machine code programs, each executed by a separate, parallel processor, but all sharing a single main memory (“core”). Generically, these approaches can be referred to as *coreworlds*. In principle, the parallelism could be implemented directly with a sufficiently large pool of hardware processing elements, dynamically allocated as agents are created or destroyed; but it is typically just realized with conventional timeslice-based multithreading of a much smaller pool of hardware CPUs (often just one).

The earliest work adopting this methodological approach can be traced to the seminal investigations of Nils Barricelli, carried out between 1953 and 1963 (Barricelli 1957, 1963). In fact, this work was carried out on the original computer designed and built at the Princeton Institute for Advanced Studies (IAS) between 1946 and 1952 under the direction of von Neumann. Barricelli was a visitor to the institute at von Neumann’s invitation; and took the opportunity to conduct an investigation of the role of “symbiogenesis” in the origin of life. He programmed the IAS computer to directly model patterns of numbers that could interact with each other within a circular array (effectively the main memory system of the machine), according to some fixed local rules. This is conceptually similar to a one-dimensional cellular automaton. Although these patterns of numbers did not have the general computing ability of Turing machines, Barricelli demonstrated the existence of patterns that could self-replicate, as well as various forms of competitive and cooperative (symbiotic) interaction between patterns. Somewhat strangely, in his published description of this work, Barricelli only obliquely related it to von Neumann’s own contemporary work on self-reproducing and evolving automata. In any case, the insight and innovation of this work was not widely recognized at the time, and has only recently been properly reinstated through George Dyson’s investigations of the early history of digital computing at the IAS (Dyson 1997, 1998).

The first attempt to study the dynamics of competition among co-resident *general purpose programs* in a shared memory was pioneered by Vysotsky et al. at Bell Labs in the early 1960s (though not published until 1972), and appears to have been independent of Barricelli’s study (Vysotsky 1972). This was effectively created as a form of computer programming game in which different programmers provided hand-coded programs, which were then instantiated in a common core, and executed to see which would survive longest. Again, self-replication was a key feature. An interesting aspect was that, unlike almost all later work, these programs were implemented in the native (IBM 7090) machine code, rather than in a virtual machine code to be executed by an interpreter. This allowed comparatively much higher execution speed, but with some associated limitations. For example, as programs could not be written to be position independent, successful self-replication required active relocation processing, as opposed to simple self-copying. In this early implementation, a “lethal” program was relatively quickly developed by R.H. Morris, and the original game was put aside.

The α -universes described by Holland in the mid-1970s (Holland 1976) may again be mentioned here. Although formulated independently, and with quite different motivations, this proposal was somewhat similar to the Vyssotsky system. In particular, it again envisaged an essentially linear core memory inhabited by concurrently executing fragments of computer code. However, in other ways this was actually much closer to Barricelli's work in that the specific intention was to investigate the spontaneous emergence of crude collective self-reproduction activity, rather than simple competition between programs pre-coded by human programmers; but, in any case, as already mentioned in [Sect. 2.3](#), this system was not experimentally investigated until much later (McMullin 1992).

The more direct descendent of the Vyssotsky system was the *Core War* game, developed by Dewdney and others in the early 1980s. This now relied on an interpreter and offered much more varied gameplay opportunities (Dewdney 1984). Following the establishment of an international tournament (Dewdney 1987), *Core War* has had a sustained following, and remains active to this day (<http://corewars.org/>).

Although the *Core War* framework is predicated on the idea of human programmers coding the competing programs, it does also naturally lead to the question of whether, under conditions of potential mutation during self-replication, there could be an autonomous and sustained evolutionary process, not relying on programmer intervention. This was explicitly studied by Rasmussen et al. in a system they called the *coreworld* (Rasmussen et al. 1990). In this instantiation, however, the system suffered from a similar limitation to that of the von Neumann style cellular automata systems. While the world could be seeded with an initial self-replicating program, as its offspring filled up the core they quickly began overwriting each other and the self-replicating functionality and thus the potential for sustained Darwinian evolution was lost. This issue was addressed by Ray, in the *Tierra* system, developed and investigated in the early 1990s (Ray 1992). *Tierra* was based on the idea of competing and interacting self-replicating programs in a shared core memory. Programs were now given an ability to employ "memory protection" to limit overwriting by other programs. On its own, this would simply lead to the core filling with programs once, and then no further self-replication (much less evolution) would be possible. To overcome this, Ray added a mechanism for random removal of programs (the so-called "reaper"). This addition allowed for continuing self-replication and the possibility for longer term evolution. He also adopted a significantly different instruction set, incorporating a biologically inspired "template" addressing mode. With these innovations, Ray was able to demonstrate an extended process of evolution, with multiple episodes of Darwinian natural selection. The system produced a surprising array of evolutionary phenomena, including optimization of program length, obligate parasitism, and facultative "hyper-parasitism" (as a defense against obligate parasites).

Tierra has led to a wide variety of related work, such as:

- The *Avida* system of Adami and Brown that introduced a more conventional 2-D spatial world in which programs would colonize nodes, and also be evaluated for their ability to complete extrinsically defined computing tasks (Adami and Brown 1994).
- A proposed extension to *Tierra* into a multiprocessor networked environment ultimately to be distributed across the wider Internet, under suitable "sandbox" execution controls (Ray 1995).
- *Nanopond* (<http://adam.ierymenko.name/nanopond.shtml>), a highly simplified implementation of a 2-D spatial "program evolution" system, drawing on elements of both *Tierra* and *Avida*, but implemented in less than 1,000 lines of C source code.

- The *Amoeba* system, developed by Pargellis, which specifically demonstrated the possibility of spontaneous emergence of a self-replicating program in coreworld-like systems (Pargellis 2001).

Evolution in coreworlds can also usefully be compared and contrasted with several other related but divergent research fields and investigations:

- There has been a separate line of research into the logic of so-called *Quine* programs. These are defined as computer programs that produce their own source code as output. The original discussion is generally credited to Bratley and Millo (1972), though the term “Quine program” is usually attributed to Hofstadter (1979). The existence of Quine programs in general-purpose programming languages essentially follows from Kleene’s recursion theorem. But while the formulation of specific Quine programs in different languages has been a hobby among programmers, these programs have not generally been embedded within an execution environment in which exponential growth, Darwinian selection, and evolution could actually take place. It has not therefore, particularly influenced research approaches in artificial life. That said, through its definitional restriction to programs that must self-reproduce without having direct access to self-inspection, the study of Quine programs draws attention to the logical distinction between proper von Neumann style genetic “self-reproduction” and the “self-replication” by copying normally adopted in coreworlds. It is, in fact, a very open question as to what the effect of this architectural choice is on evolutionary potential (McMullin et al. 2001).
- The replication and propagation of programs in controlled coreworld environments obviously also lead to the idea of self-replicating programs propagating in open, networked, computer environments. This is the realm of computer malware – computer viruses, worms, etc. And indeed, this connection was explicitly made by Dewdney already in the immediate aftermath of the Morris Internet worm incident in 1988 (Dewdney 1989). However, while malware development certainly involves an arms race between the human developers on both sides, there is to date no evidence of effective *autonomous* evolution of “free-living” malware.
- There is significant overlap in inspiration between evolution in coreworld-like systems, and work in evolutionary algorithms, and, especially, *genetic programming* (GP) (Koza 1992; Willis et al. 1997; Banzhaf et al. 1998). This, in turn, has a long history extending back at least to Samuel’s late 1950s investigations in machine learning (Samuel 1959). However, the major distinction is that GP is generally driven, directly or indirectly, by an externally provided evaluation function (used as an extrinsic “fitness” to drive an imposed evolutionary algorithm) rather than the coreworld approach, which is to investigate spontaneous autonomous evolutionary dynamics, in which the software agents are responsible for their own replication and relative fitness emerges from their bottom-up ecological interactions.
- Investigations into the evolution of cooperation by Axelrod and others should also be mentioned here (Axelrod 1984, 1987). Although not generally using a coreworld-like framework, and focused on a relatively narrowly defined pattern of interaction (the iterated prisoner’s dilemma), using an extrinsically applied genetic algorithm, this examination of the problem of cooperation, and coevolution between “cooperating” and “defecting” strategies has been very influential in the wider fields of political science, ecology, and complexity theory.

To summarize, the general experience of the investigation to date of evolution in core-world-like systems is that the evolutionary potential of these systems is interesting but still strictly limited. Indeed, they can be viewed as formally similar to pure artificial “replicator chemistries,” comparable to the systems already mentioned in [Sect. 2.3](#) above, and exhibiting similar phenomena of collective autocatalytic closure and self-maintenance (McMullin [2000b](#)). Thus, both von Neumann-style genetic self-reproduction in cellular automaton worlds, and replication by simple copying or self-inspection in coreworlds, naturally lead directly into the problems of biological robustness, self-maintenance, and hierarchical organization already discussed above. The integration of self-reproducing or self-replicating programs with self-maintenance and individuation (autopoiesis), and the demonstration of sustained evolutionary growth of complexity in a purely virtual world remains perhaps the key “grand challenge” problem in the field of artificial life.

3 Theory of and Formalisms in Artificial Life

A number of formalisms have been effective and widely used in artificial life systems. Among them are cellular automata, rewriting systems, and complex dynamical systems. Other formalisms have been used to analyze artificial life systems, like network analysis. In this section, the most important synthetic and analytic tools will be described.

3.1 Automata

The theory of abstract automata defines finite state automata as behavioral models of machines that possess a finite number of states, a set of transitions between those states, and machine actions (such as “read input” or “write output”). Based on this notion, an entire field in computer science (automata theory) has been formulated, which addresses their mathematical features and the use of these entities. One variant of automata of particular relevance to artificial life are cellular automata.

3.1.1 Cellular Automata

A cellular automaton (CA) is composed of a large interconnected collection of component finite state automata. These possess, besides states and transitions, an additional feature, a location within a neighborhood, often constituted as a grid. The neighboring “cells” (although the word “cell” here gives rise to the term “cellular automaton,” note that it is *not* being used with its specifically biological sense; it rather just means a discrete, spatially demarcated, component of a larger, static, locally interconnected, array) or “nodes”, which are themselves finite state automata with a different location, receive the state of the particular automaton as input, and also, reciprocally, provide their own states as inputs that inform a particular automaton which state transition to realize. CA are typically defined to be structurally homogenous, that is all cells contain the same finite state automaton and all share the same local neighborhood pattern. Of course, cell *states* vary dynamically in time and across the cells of the automaton; and it is such spatiotemporal patterns of states that are used to represent so-called *embedded* machines or automata. Such a model can thus be used to model spatially extended systems and study their dynamical behavior. CA have also been used as a modeling

tool to capture behavior otherwise modeled by partial differential equations (PDEs); and indeed can be regarded as a discrete time, discrete space, analogue of PDEs.

Cellular automata play a particularly useful role, in that they possess the twin features of being applicable for synthesis of system behavior, and for analysis of system behavior. They have been thoroughly examined in one-dimensional and two-dimensional contexts (Wolfram 1986, 1994), and have been considered in the fundamental contexts of computability (Wolfram 1984) and parallel computing (Toffoli and Margolus 1987; Hillis 1989).

As mentioned previously (❸ Sect. 2.4), von Neumann, following a suggestion of Ulam, originally introduced CAs as a basis to formulate his system of self-reproducing and evolvable machines. The von Neumann CA was two dimensional with 29 states per cell. According to Kemeny, von Neumann's core design for a genetically self-reproducing automaton would have occupied a configuration of about 200,000 cells (a size dominated by the "genome" component, which would stretch for a linear distance of about 150,000 cells) (Kemeny 1955, p. 66).

In fact, von Neumann's contributions stand at the cradle of the field of artificial life by showing, for the first time, a mathematical proof that an abstract "machine-like" entity can be conceptualized, which can construct itself, with that copy being able to achieve the same feat in turn, while also having the possibility of undergoing heritable mutations that can support incremental growth in complexity. This was a breakthrough showing that above a certain threshold complexity of an entity, this entity is constructionally powerful enough not only to reproduce itself (and thus maintain that level of complexity) but to seed a process of indefinite complexity growth.

While von Neumann achieved this result in principle, a large number of scientists have since succeeded in pushing down the complexity threshold. In the meantime, much simpler CAs (with smaller number of states per cell) have been shown to support self-reproducing configurations, many of those as 2D loops (Langton 1984; Sipper 1998). In fact, the smallest currently known example in this form is a cellular automaton with eight states per cell where the self-reproducing configuration is of size five cells (Reggia et al. 1993). However, in many of these cases, the core von Neumann requirement to embody a "general constructive automaton" in the self-reproducing configuration has been relaxed: So the potential for evolutionary growth of complexity is correspondingly impoverished.

Cellular automata also have been used in the design of the Game of Life (Gardner 1970, 1971), which draws inspiration from real life in that replication and transformation of moving entities, again in a 2D CA, were observed (for a comprehensive introduction, see Berlekamp et al. (1982) and Conway's Game of Life, <http://en.wikipedia.org/wiki/conway> and online resources therein).

The discrete nature of states, space, and time available in cellular automata has led to a number of applications of these tools, many of which could be considered artificial life applications in the widest sense. For example, the patterning of seashells has been modeled using discrete embodiments of nonlinear reaction-diffusion systems (Meinhardt 2003). Scientists in Urban Planning have started to use the tool of cellular automata to model urban spread (White and Engelen 1993).

3.2 Rewriting Systems

Rewriting systems are a very general class of formal systems in mathematics, computer science, and logic, in which rules for replacing formal (sub)structures with others are

repeatedly applied. Rewriting systems come in many forms, such as term rewriting, string rewriting, equation rewriting, graph rewriting, among others, and have been at the foundation of programming languages in computer science and automated deduction in mathematics. String rewriting systems, first considered by Axel Thue in the early part of the twentieth century (Dershowitz and Jouannaud 1990; Book and Otto 1993), have a close relationship with Chomsky's formal grammars (Salomaa 1973). Many rewriting systems have the goal of transforming particular expressions into a normalized form, deemed to be stable in some sense. Prior to reaching this stable form from a starting expression, the rewriting process goes through a series of transient expressions. For example, mathematical equations can be formally rewritten until they reach a form in which it is possible to read off solutions.

It can be seen, therefore, that rewriting allows a sort of dynamics of symbolic systems. It is worth noting that rewriting systems provide the basis for the λ -calculus (Barendegt 1984), which has been used to implement artificial chemistries.

In a more general artificial life context, transformations based on rewriting systems can be used to describe growth and developmental processes. For example, besides transforming complex expressions into normal form, another approach uses rewriting to generate complex objects from simple start objects. This has provided the formal basis to studies of biological development through what were later called L-systems (Lindenmayer 1968). The difference between L-systems simulating growth and development and formal grammars lies in the order of rewriting. While formal grammars consider rewriting in a sequential fashion, L-systems consider parallel (simultaneous) rewriting, much closer to the asynchrony of the growth of real cells. But central to all these models is the recursive nature of rewriting systems (Herman et al. 1975). Lindenmayer and, later, Prusinkiewicz and Lindenmayer (1990) have pioneered the use of turtle commands of computer languages like LOGO to visualize plant patterning and growth.

P-systems (Paun 1998; Paun and Rozenberg 2002) are a more complex class of rewriting systems used in artificial life. They are similar to artificial chemistries in that production of symbolic chemicals is prescribed by reaction rules that could also be considered as rewriting rules. In addition to reactions, however, P-systems contain membranes, which limit the availability and direct the flow of material. This provides a higher-level means of structuring environments, something artificial chemistries with well-stirred reactions are unable to do (Dittrich et al. 2001).

In the meantime, a large number of applications of rewriting systems have appeared. For example, such classical artificial life systems as *Tierra* have been reformulated as rewriting systems (Sugiura et al. 2003); Giavitto et al. (2004) describe applications of rewriting systems in modeling biological systems, and virtual cities have been modeled with the help of L-systems (Kato et al. 1998), just to name three examples.

3.3 Dynamical Systems Modeling with ODE

Artificial life systems are often couched in terms of the mathematical language of dynamical systems, captured by ordinary differential equations. This formalism is used for describing the continuous change of state of a system via the change in time of n quantified state variables $\mathbf{q}(t)=\{q_1(t), q_2(t), \dots, q_n(t)\}$. A very simple and linear example of a dynamical system is a system

consisting of two-state variables q_1, q_2 , which develop according to the following differential equation

$$\frac{dq_1}{dt} = a_{11}q_1 + a_{12}q_2 \quad (1)$$

$$\frac{dq_2}{dt} = a_{21}q_1 + a_{22}q_2 \quad (2)$$

Coefficients a_{11}, \dots can be lumped together into a matrix A , and the stability of the dynamical system can be examined based on a solution to the equation $d\mathbf{q}/dt = A \times \mathbf{q}(t) = \mathbf{0}$.

3.3.1 Nonlinear Dynamical Systems

More interesting than this basic dynamical system are those which provide for nonlinear interactions between components and the environment. This can be achieved by higher order terms in the coupling of state variables. For instance

$$\frac{dq_1}{dt} = a_{11}q_1 + a_{12}q_2^2 \quad (3)$$

$$\frac{dq_2}{dt} = a_{21}q_1^2 + a_{22}q_2 \quad (4)$$

would already provide such an example.

Chaotic Systems

Nonlinear couplings are the hallmark of chaotic systems, as nonlinearities in the coupling of state variables tend to amplify small fluctuations (e.g., noise in state variables) into large global state changes. Of course, all these depend on the size of the fluctuations. It is well known that as long as fluctuations are small relative to the strength of nonlinear interactions, systems act in quasi-linear mode and can be formally linearized. However, once either the fluctuation signal is big enough, or the nonlinear interactions become stronger, systems tend to behave in less predictable ways, for example, start oscillating, and might even transit into a chaotic regime characterized by short-term predictability and long-term unpredictability (Argyris et al. 1994). In the artificial life community, the notion of chaos has played an important role as the extreme case of unpredictability beyond life, much as the predictability of dead matter lies at the other end of the spectrum. This notion was first developed in the context of cellular automata (Langton 1992a), but later extended to other systems.

Constructive Systems

Dynamical systems can be extended by considering a notion of change in the number of state variables. For instance, if each of the variables characterizes the concentration of a chemical substance, and some of the substances are consumed or produced for the first time, this can be captured by changing the number and/or assignment of state variables. Simulation might proceed by alternating between a low-level simulation of, for example, individuals of species, and high-level events that signal the birth or extinction of a particular species (Ratze et al. 2007). While couched here in ecological terms, the same approaches apply to the substances of artificial chemistries. Constructive dynamical systems are an active area of research whose exploration has barely begun.

3.4 Complex Systems

The relation between complex systems and artificial life is complicated. Some might argue that artificial life is a subfield of complex systems. This opinion draws from the argument, valid in both complex systems research and artificial life, that these kinds of systems work by producing emergent effects through the interaction of many entities. As such, living systems (and their artificial counterparts) are complex systems, but not the only ones, since, for instance, societies, economies, or legal and norm systems might legitimately be called complex systems too. On the other hand, one might argue that if systems like societies or economies are behaving as complex systems, making use of emergent phenomena and top-down causation, they really can be considered to be alive. If that argument is followed it could be stated that complex systems are part of the research in artificial life.

In this section, it is preferred not to resolve this tension, but rather to focus on one area of research that one legitimately can consider as at the intersection of the two: networks. Networks and their associated formalisms are peculiar since they provide evidence for a phenomenon frequently seen in artificial life systems, as well as in complex systems: There is no easy way to dissect these systems into cause and effect relationships. Indeed, networks allow us to formalize the concept that causes and associated effects have a many-to-many relationship.

If a simple cause–effect relationship is visualized as an edge between two nodes or vertices in a graph, while the nodes themselves symbolize the entities that might or might not be in these relationships, one can easily see that networks, with nodes of in-degree and out-degree larger than one will be able to capture multiple cause–multiple effect relationships. This is a key characteristic of complex systems, as opposed to simple systems, where simple pathways exist between causes and their associated effects.

3.5 Networks

The recent surge in network research (Albert and Barabasi 2002) can be attributed to the realization that networks are the quintessential structure to capture emergent phenomena. Without network infrastructure, emergent behavior of a system would be transient and very quickly disappear, like fluctuations. It is when a network infrastructure is in place that originally weak and transient signals of emergence can be captured and stabilized into identifiable phenomena.

Network studies have revealed widespread characteristics in complex systems, such as small-world features or scale-free connectivity (Watts 1999; Strogatz 2001) that have been ascribed to the generation or evolution of these networks. The dynamics of network evolution is an active area of research (Dorogovtsev and Mendes 2003).

But even the much simpler formalisms of static networks provide useful insights into complex systems and artificial life systems, because there is a deep relationship between autocatalytic sets (Farmer et al. 1986) and closed paths on networks, which in mathematical terms is the result of a connection between graph theory and algebra. Networks can be formally represented by directed graphs $G = G(N, E)$, defined by the set of N nodes and the set of E edges or connections between those nodes. Edges or links between nodes are often represented by an ordered pair of nodes, called the connectivity matrix. If connections are only present or absent, symbolized by a “0” or “1” entry in the matrix, this is called the adjacency matrix A of a graph G . The adjacency matrix has a set of eigenvalues and eigenvectors, which can be easily computed by solving the characteristic equation of A : $|A - \lambda I| = 0$, where λ are

eigenvalues of A and I is the unit matrix. Given that A is a nonnegative matrix, the Perron–Frobenius theorem states that the largest eigenvalue λ_1 of A is real. If this eigenvalue is $\lambda_1 \geq 1$, then there exists a closed path through the graph (Rothblum 1975). Closed paths are deeply related to autocatalytic sets, which were first introduced in chemical systems where each sort of molecules present is catalyzed by at least one other sort of molecules (Eigen 1971). This relation would be considered equivalent to an edge in the graph of all molecular sorts (nodes). Autocatalytic sets correspond therefore to closed paths in that graph.

In the wider context of a theory of organizations (Fontana and Buss 1994; Speroni di Fenizio et al. 2000), autocatalytic sets or closed and self-maintaining sets (organizations) are the key to life-like processes, in that they maintain whatever structure or organization has come into existence. This is the core of the previous statement that emergent phenomena need the stabilizing infrastructure of a network (and its closed paths) to move from the status of a fluctuation in a system to that of an identifiable phenomenon.

The deep connections between algebra and graph theory indicated here would merit a much more detailed discussion, yet space is a constraint here. Suffice it to say that networks are the stage for a dynamics of the chemical (or otherwise) sorts of agents symbolized by the nodes. Thus, it can be said that a dynamical system can be imposed on these networks, and once this dynamics starts to interact with the structure of the network, the most interesting phenomena will be observed (Jain and Krishna 2001).

3.6 Other Formalism

The list of formalisms discussed here is not exhaustive. For instance, the area of statistical mechanics has also contributed valuable insights into artificial life (see, e.g., Adami 1998). Category theory, chemical organisation theory, and other approaches can also be legitimately mentioned as contributing to our understanding of artificial life phenomena.

4 Applications

A huge number of applications can be considered to make use of artificial life techniques in the wider sense (Kim and Cho 2006). It is difficult to even list these applications comprehensively. Instead, a few applications have been selected here to demonstrate the width of the applicability of its concepts.

4.1 Biology

The most obvious application of artificial life is in biology. It was already noted that there is a close relation between artificial life models and synthetic biology (Ray 1994) and that artificial life models in fact are a useful tool for the exploration of questions in biology (Taylor and Jefferson 1994). For a more recent example, see Strand et al. (2002).

4.1.1 Synthetic Biology

The current state of synthetic biology does not acknowledge this connection extensively, despite, historically speaking, emerging from artificial life research. Key questions in this

area today are, how to generate biological entities (cells, proteins, genes, organisms) that have some purposefully designed function (Ferber 2004). The question how systems function that can be ascribed to be “living” has been separated off into the field of “systems biology,” another newcomer to biology with inheritance from artificial life (Kitano 2002). Artificial life was always concerned with “Life as it could be,” that is, alternative designs that can be termed living, designed for a purpose (as engineers would do), or for scientific inquiry, and it was following systems thinking long before systems biology was established (Langton 1997).

4.1.2 Health/Medicine

The modern health-care system is so complex now, that complex systems thinking and ideas from artificial life are ripe for application (Plsek and Greenhalgh 2001). This will entail the understanding of system behavior and response to attempted changes, financial issues such as exploding health-care costs and redesign of the system to become more adaptive.

Medicine has long benefited from its relation to biology, which will result in new applications for artificial life methods as well (Coffey 1998; Hamarneh et al. 2009). Notably, regenerative medicine will be the beneficiary of model systems provided by artificial life (Semple et al. 2005).

4.1.3 Environmental Science and Ecology

In environmental science, the *Gaia* theory (Lovelock 1989) has benefited from simulations provided by artificial life models (Gracias et al. 1997; Downing 2000; Lenton 2002). Artificial life approaches have been used to design sustainable architecture, at least on a local level (Magnoli et al. 2002). In the design of other systems, for example, for recycling, artificial life models have been successfully applied (Okuhara et al. 2003).

Ecosystem research in general has made use of artificial life models, from the setting up of “artificial ecosystems” whose behavior can then be explored in the computer (Lindgren and Nordahl 1994) to the examination of characteristics necessary to provide opportunity for ecosystem formation (Wilkinson 2003). Further, artificial life models have been applied to questions of astrobiology (Centler et al. 2003), another offshoot with some roots in artificial life research.

4.2 Engineering

A number of applications in the engineering subdisciplines have been examined, with autonomous robotics being a very prominent one.

4.2.1 Autonomous Robotics

While it is natural to expect artificial life to contribute to biology, its contribution to the field of engineering is less to be expected. But given the tight connection between behavior and autonomous robotics, it is perhaps less a surprise. In fact, an entire conference series has been

termed “from animals to animats” (Meyer and Wilson 1991; Asada et al. 1998), carrying with it the notion of machines that behave adaptively. Recent reviews and summaries of autonomous robotics as the field pertains to artificial life can be found in Eaton and Collins (2009), Harvey et al. (2005), and Pfeifer et al. (2005).

Often, emergence of behavior or functionality is studied in behaving artifacts like robots, using the bottom-up principles of artificial life. Notably, the interaction of software or hardware parts of robots can be assumed to be closely influenced by artificial life thinking. For instance, the subsumption architecture of Brooks (1990, 1991) has been a closely studied subject in the interaction of behavioral modules. Different sorts of swarm-like behavior (Reynolds 1987; Bonabeau et al. 1999) and, more recently, specifically constructed swarms-bots on the hardware side (Mondada et al. 2004; Groß et al. 2006) have been studied extensively.

Finally, groups of robots cooperating as social robots, and interacting and cooperating with each other, have been studied (Fong et al. 2003). All these applications emphasize the real-world character of robotics problems, and the embodiment of agents (Steels and Brooks 1995).

4.2.2 Transport

Traffic and our societies’ transportation systems are collective systems of an enormous complexity. Their modeling, simulation, and design stand to win substantially from new biological principles (Lucic and Teodorovic 2002; Wang and Tang 2004). Notably, principles gleaned from the organization of insect societies (Deneubourg et al. 1994), and from insect behavior (Theraulaz and Bonabeau 1999), that even give rise to new optimization algorithms (Dorigo and Stützle 2004), are relevant here.

Artificial life applications are not restricted to the movement of physical objects on roads, but are also widespread in other infrastructure networks, such as communication networks (Tanner et al. 2005), notably the Internet (Prokopenko et al. 2005). Early on, it was already proposed to consider computer viruses as a form of artificial life (Dewdney 1989; Spafford 1994) spreading in these communication networks (Bedau 2003).

Already there is a large number of engineering applications for artificial life, but this number is only bound to grow over the coming years (Ronald and Sipper 2000). The reader is referred to (Kim and Cho 2006) for a recent overview of artificial life in engineering.

4.3 Computer Graphics, Virtual Worlds, and Games

Terzopoulos (1999) has summarized a number of applications of artificial life in computer graphics.

4.3.1 Computer Graphics

In particular, Sims’ work is worth mentioning (Sims 1991, 1994). He used virtual organisms that coevolved in competition with each other to demonstrate the power of artificial evolutionary systems for developing naturally looking moving behavior. Terzopoulos built animated systems resembling simulated (“artificial”) fishes, again studying locomotion (Terzopoulos et al. 1994).

The breve visualization environment (Klein 2002) offers a very natural way to interact with artificial life simulations (Spector et al. 2005).

In general, the aim of these tools is animation of agents equipped with virtual sensors and actors, in a life-like fashion (Miranda et al. 2001; Conde and Thalmann 2004). This development was able to close the gap to games (Maes 1995; Reynolds 1999; Rabin 2002).

4.3.2 Virtual Worlds

Magnenat-Thalmann and Thalmann (1994) provide an early overview of how artificial life ideas could be put to use in creating virtual worlds. One key idea, also found in other application areas, is to make use of unpredictability in the form of random events, as they are also present in the real world of life through mutation and chance encounters. This line of inquiry is continued with further work on multi-sensor integration in virtual environments, see, for example, Conde and Thalmann (2004). The more complex the inner workings of agents are, and the more complex the environment is providing stimuli, the more intelligence is required to allow for realistic simulations. Aylett et al. discuss the connection between an adaptive environment and the intelligence of the agents/avatars populating it (Aylett and Luck 2000).

More recently, artificial life virtual worlds have been put to use in social scenarios, such as urban environments. For instance, Shao and Terzopoulos (2007) integrate motor, perceptual, behavioral, and cognitive components in a model of pedestrian behavior. Building on Reynolds (1987, 1999), this allows much more realistic simulation of multi-individual interactions in complex urban settings than was possible previously. An entire area, crowd simulation has now emerged using these techniques (Lee et al. 2007).

Virtual worlds are bound to expand enormously in the future, and artificial life applications will continue to proliferate.

4.3.3 Games

Artificial life helped to grow the area of (evolutionary) game theory (Alexander 2003; Gintis 2009) and provided valuable insights into more traditional board games (Pollack et al. 1997). But its most notable and unique success is in areas where traditional games have not been able to contribute much: multiuser games, distributed over the Internet.

One now classical example, and one of the earliest games that made headlines and really was a commercial success for some time, was Steve Grand's game "Creatures" (Grand et al. 1996). The idea behind this game was very simple: Evolution through breeding of creatures using genomes of characteristics that could be combined in multiple ways. By exchanging creatures over the Internet, a multiuser sphere was created setting in motion something akin to real evolution, with implicit fitness, not controlled by any central agency (Grand 2001). Prior to this commercial application, Ray's Tierra was distributed over the net in a large experiment (Ray 1998).

4.4 Art

The connection between artificial life and art can be easily seen when considering applications in computer graphics. However, there is a much deeper connection, having to do with

creativity and surprise, which is one of the hallmarks of evolutionary systems that, like artificial life, concern themselves with emergent phenomena. A discussion of these connections is provided in Whitelaw (2004). In a 1998 special section of the journal *Leonardo* (Rinaldo 1998), the connection was examined from various viewpoints. Bentley and Corne (2002) and references therein look at creative processes from an evolutionary perspective, which might be argued to be a superset of artificial life approaches. Arts applications range from painting (Todd and Latham 1992, 1994) (even painting by robots (Moura 2004)) via video installations to sculpture (Baljko and Tenhaaf 2006). Creativity is also considered in the area of sound generation and music (Bilotta and Pantano 2002; Berry and Dahlstedt 2003). Again, artificial life work is often put together with evolutionary work in composition (Romero and Machado 2007).

4.5 Artificial Societies and Artificial Economies

Agent-based modeling using artificial life ideas has also been applied to formulate systems best described as artificial societies and artificial economies.

The area of artificial societies was spawned by the seminal work of Axtell and Epstein (Epstein and Axtell 1996), which used the idea of bottom-up effects in social systems to generate models of social systems that could be simulated in a computer. These ideas have been further developed by Epstein (2007). Social simulation in general (Gilbert and Troitzsch 2005) has now moved to center stage in the social sciences, and a new journal has been established to examine artificial societies (<http://jasss.soc.surrey.ac.uk/JASSS.html>). In the context of social science, emergence (Goldspink and Kay 2007) and learning (Gilbert et al. 2006) have become important topics of discussion.

These very same techniques can also be applied to markets, at which point an artificial society mutates into an artificial economy (Zenobia et al. 2008). The formulation of economies as complex adaptive systems is already somewhat older, and Kaufmann (1993) points out models of economies based on ideas of evolution of complex systems. The power of these ideas lies in their natural ability to treat nonequilibrium phenomena, a topic for the most part carefully avoided by many traditional economists, since it would have to be based on nonlinear systems (Arthur 2006). Recently, another subfield of artificial life, artificial chemistries, have been applied to model economies (Straatman et al. 2008).

5 Conclusion

As we have seen in this chapter, artificial life is a vivid research area, spawning interesting research directions, and providing inspiration for a large number of applications. While the boundaries of the area are fuzzy, there is no doubt that over its 20 years of existence as a named field, artificial life has exerted substantial influence on a whole number of other research areas.

To judge the perspectives of a field, it is perhaps best to focus on its main objectives. The proposal of Bedau (Bedau et al. 2001; Bedau 2003) is followed and slightly changed in dividing the central goals of artificial life into three broad thrusts, all posited here as questions:

- (A) What is the origin of life, or how does life arise from the nonliving?
- (B) What are the potentials and limits of living systems?
- (C) How does life relate to intelligence, culture, society, and human artifacts?

These questions belong to the most challenging and most fascinating in all of science today. From each of these questions derives an entire research programme within artificial life that can be formulated with a non-comprehensive number of tasks:

- (A1) Generate a molecular proto-organism *in vitro*.
- (A2) Achieve the transition to life in an artificial chemistry *in silico*.
- (A3) Determine the threshold of “minimal” life in the organic world *in vivo*.
- (A4) Determine whether fundamentally novel living organizations can arise from inanimate matter.
- (A5) Simulate a unicellular organism over its entire life-cycle.
- (B1) Determine what is inevitable in the open-ended evolution of life.
- (B2) Establish a set of mechanisms for the generation of novelty and innovation.
- (B3) Determine minimal conditions for evolutionary transitions from specific to generic response systems.
- (B4) Create a formal framework for synthesizing dynamical hierarchies at all scales.
- (B5) Determine the predictability of evolutionary manipulations of organisms, ecosystems, and other living systems.
- (B6) Develop a theory of information processing, information flow, and information generation for evolving systems.
- (C1) Explain how rules and symbols are generated from physical dynamics in living systems.
- (C2) Demonstrate the emergence of intelligence and mind in an artificial living system.
- (C3) Evaluate the influence of machines on the next major evolutionary transition of life.
- (C4) Provide predictive models of artificial economies and artificial societies.
- (C5) Provide a quantitative model of the interplay between cultural and biological evolution.
- (C6) Establish ethical principles for artificial life.

Inasmuch as life constitutes a large part of the natural world, artificial life will constitute a large part and exert heavy influence on the area of natural computing discussed in this handbook.

Acknowledgments

WB would like to thank the Canadian Natural Science and Engineering Research Council (NSERC) for providing continuous operating funding since 2003 under current contract RGPIN 283304-07. BMcM acknowledges the support of Dublin City University, and of the European Union FP6 funded project ESIGNET, project number 12789.

References

- Adami C (1998) Artificial life, an introduction. Springer, Berlin
- Adami C, Brown CT (1994) Evolutionary learning in the 2D artificial life system “Avida.” In: Brooks RA, Maes P (eds) Artificial life IV: Proceedings of fourth international workshop on the synthesis and simulation of living systems. MIT Press, Cambridge, MA, pp 377–381, <http://citeseer.ist.psu.edu/101182.html>
- Albert R, Barabasi AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74:47–97
- Alexander J (2003) Evolutionary game theory. In: Stanford encyclopedia of philosophy. <http://plato.stanford.edu/>

- Argyris J, Faust G, Haase M (1994) An exploration of chaos. North-Holland, New York
- Arthur W (2006) Out-of-equilibrium economics and agent-based modeling. In: *Handbook of computational economics*, vol 2. North-Holland, Amsterdam, The Netherlands pp 1551–1564
- Asada M, Hallam J, Meyer J, Tani J (eds) (2008) From animals to animats: Proceedings of the tenth international conference on simulation of adaptive behavior, Osaka, Japan, July 2008. Springer, Berlin
- Axelrod R (1984) *The evolution of cooperation*. Basic Books, New York
- Axelrod R (1987) The evolution of strategies in the iterated prisoner's dilemma. In: Davis L (ed) *Genetic algorithms and simulating annealing*. Pitman, London
- Aylett R, Luck M (2000) Applying artificial intelligence to virtual reality: intelligent virtual environments. *Appl Artif Intell* 14(1):3–32
- Baas N (1994) Hyperstructures. In: *Artificial life III*, proceedings of the second workshop on artificial life. Addison-Wesley, Redwood City, CA, Santa Fe, NM, June, 1992
- Baljko M, Tenhaaf N (2006) Different experiences, different types of emergence: A-life sculpture designer, interactant, observer. In: *Proceedings of AAAI fall 2006 symposium on interaction and emergent phenomenon in societies of agents*. Arlington, VA, October 2006, pp 104–110
- Banzhaf W, Nordin P, Keller R, Francone F (1998) *Genetic programming – An introduction*. Morgan Kaufmann, San Francisco, CA
- Barendregt H (1984) *The lambda calculus, its syntax and semantics*. Elsevier-North Holland, Amsterdam, The Netherlands
- Barricelli N (1957) Symbiogenetic evolution processes realized by artificial methods. *Methodos* 9(35–36): 143–182
- Barricelli N (1963) Numerical testing of evolution theories. *Acta Biotheor* 16(3):99–126
- Bedau M (2003) Artificial life: organization, adaptation and complexity from the bottom up. *Trends Cogn Sci* 7(11):505–512
- Bedau MA et al. (February 2001) Open problems in artificial life. *Artif Life* 6(4):363–376
- Bentley P, Corne D (eds) (2002) *Creative evolutionary systems*. Academic Press, San Diego, CA
- Berlekamp ER, Conway JH, Guy RK (1982) What is life? In: *Winning ways for your mathematical plays*, vol 2. Academic Press, London, chap 25, pp 817–850
- Berry R, Dahlstedt P (2003) Artificial life: Why should musicians bother? *Contemp Music Rev* 22(3):57–67
- Bertalanffy LV (1976) *General system theory: foundations, development, applications*. George Braziller, New York
- Bilotta E, Pantano P (2002) Synthetic harmonies: Recent results. *Leonardo* 35:35–42
- Bonabeau E, Dorigo M, Theraulaz G (1999) *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York
- Book R, Otto F (1993) *String-rewriting systems*. Springer Verlag, London
- Bratley P, Millo J (1972) Computer recreations; self-reproducing automata. *Software Pract Experience* 2:397–400
- Brooks R (1990) Elephants don't play chess. *Robot Autonomous Syst* 6:3–15
- Brooks R (1991) Intelligence without representation. *Artif Intell* 47:139–159
- Brooks RA (1985) A robust layered control system for a mobile robot. Technical Report A. I. Memo 864, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, <http://people.csail.mit.edu/brooks/papers/AIM-864.pdf>
- Burks AW (ed) (1966) *Theory of self-reproducing automata [by] John von Neumann*. University of Illinois Press, Urbana, IL
- Cariani P (1992) Some epistemological implications of devices which construct their own sensors and effectors. In: *Toward a practice of autonomous systems-proceedings of the first European conference on artificial life* Paris, France, December 1991. MIT Press, Cambridge, MA, pp 484–493
- Centler F, Dittrich P, Ku L, Matsumaru N, Pfaffmann J, Zauner K (2003) Artificial life as an aid to astrobiology: testing life seeking techniques. In: Banzhaf W, Christaller T, Dittrich P, Kim J, Ziegler J (eds) *Advances in artificial life. Proceedings of ECAL 2003*, Dortmund, Germany, September 2003. Lecture notes in computer science, vol 2801. Springer, Berlin, pp 31–40
- Chu D, Ho WK (2006) A category theoretical argument against the possibility of artificial life: Robert Rosen's central proof revisited. *Artif Life* 12(1):117–134. doi: 10.1162/106454606775186392, <http://www.mitpressjournals.org/doi/abs/10.1162/106454606775186392>, <http://www.mitpressjournals.org/doi/pdf/10.1162/106454606775186392>
- Codd EF (1968) *Cellular automata*. ACM Monograph Series. Academic Press, New York
- Coffey D (1998) Self-organization, complexity and chaos: the new biology for medicine. *Nat Med* 4(8):882–885
- Conde T, Thalmann D (2004) An artificial life environment for autonomous virtual agents with multi-sensorial and multi-perceptive features. *Comput Anim Virtual Worlds* 15:311–318
- Deneubourg J, Clip P, Camazine S (1994) Ants, buses and robots-self-organization of transportation systems. In: *Proceedings of the From Perception to Action Conference*, PerAc-94, IEEE Press, New York, pp 12–23
- Dershowitz N, Jouannaud J (1990) Rewrite systems. In: *Handbook of theoretical computer science*.

- Volume B, Formal Methods and Semantics. North-Holland, Amsterdam, the Netherlands, pp 243–320
- Dewdney AK (1984) Computer recreations: in a game called Core War hostile programs engage in a battle of bits. *Sci Am* 250:14–22
- Dewdney AK (1987) Computer recreations: a program called mice nibbles its way to victory at the first Core Wars tournament. *Sci Am* 256(1):8–11
- Dewdney AK (1989) Computer recreations: of worms, viruses and Core War. *Sci Am* 260(3):90–93
- Dittrich P, di Fenizio PS (2007) Chemical organization theory. *Bull Math Biol* 69(4):1199–1231
- Dittrich P, Ziegler J, Banzhaf W (2001) Artificial chemistries – a review. *Artif Life* 7(3):225–275
- Dobzhansky T (1973) Nothing in biology makes sense except in the light of evolution. *Am Biol Teach* 35: 125–129. http://www.pbs.org/wgbh/evolution/library/10/2/text_pop/l_102_591.html
- Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge, MA
- Dorogovtsev S, Mendes J (2003) Evolution of networks: from biological nets to the Internet and WWW. Oxford University Press, Oxford
- Downing K (2000) Exploring Gaia theory: artificial life on a planetary scale. In: Artificial life VII: proceedings of the seventh international conference on artificial life. MIT Press, Cambridge, MA, p 90
- Dyson G (1997) Darwin among the machines; or, the origins of [artificial] life. Presentation hosted by the Edge Foundation, Inc., 8 July 1997, http://edge.org/3rd_culture/dyson/dyson_p1.html
- Dyson GB (1998) Darwin among the machines: the evolution of global intelligence. Basic Books, New York
- Eaton M, Collins J (2009) Artificial life and embodied robotics: current issues and future challenges. *Artif Life Robot* 13(2):406–409
- Edelman G (1987) Neural Darwinism: the theory of neuronal group selection. Basic Books, New York
- Eigen M (1971) Self-organization of matter and the evolution of biological macro-molecules. *Naturwissenschaften* 58:465–523
- Epstein J (2007) Generative social science: studies in agent-based computational modeling. Princeton University Press, Princeton, NJ
- Epstein J, Axtell R (1996) Growing artificial societies: social science from the bottom up. MIT Press, Cambridge, MA
- Farmer J, Kauffman S, Packard N (1986) Autocatalytic replication of polymers. *Physica D* 22:50–67
- Farmer JD, d'A Belin A (1992) Artificial life: the coming evolution. In: Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) *Artificial Life II*. Santa Fe Institute studies in the sciences of complexity, vol X. Addison-Wesley, Redwood City, CA, pp 815–838. Proceedings of the workshop on artificial life, Santa Fe, NM, February 1990
- Ferber D (2004) Synthetic biology: microbes made to order. *Science* 303:158–161
- Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, New York
- Fong T, Nourbakhsh I, Dautenhahn K (2003) A survey of socially interactive robots. *Robot Autonomous Syst* 42(3–4):143–166
- Fontana W, Buss L (1994) The arrival of the fittest: toward a theory of biological organization. *Bull Math Biol* 56:1–64
- Gardner M (1970) Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life." *Sci Am* 223(4):120–123
- Gardner M (1971) Mathematical games: on cellular automata, self-reproduction, the garden of Eden and the game "Life." *Sci Am* 224(2):112–117
- Giavitto J, Malcolm G, Michel O (2004) Rewriting systems and the modeling of biological systems. *Comparative Funct Genomic* 5:95–99
- Gilbert G, Troitzsch K (2005) Simulation for the social scientist. Open University Press, London
- Gilbert N, den Besten M, Bontovics A, Craenen B, Divina F et al. (2006) Emerging artificial societies through learning. *J Artif Soc Soc Simulation* 9(2)
- Gintis H (2009) Game theory evolving: a problem-centered introduction to modeling strategic interaction, 2nd edn. Princeton University Press, Princeton, NJ
- Goldspink C, Kay R (2007) Social emergence: distinguishing reflexive and non-reflexive modes. In: AAAI fall symposium: emergent agents and socialities: social and organizational aspects of intelligence. Washington, DC, November 2007
- Gracias N, Pereira H, Lima J, Rosa A (1997) Gaia: An artificial life environment for ecological systems simulation. In: Artificial life V: proceedings of the fifth international workshop on the synthesis and simulation of living systems. MIT Press, Cambridge, MA, pp 124–134
- Grand S (2001) Creation: life and how to make it. Harvard University Press, Cambridge, MA
- Grand S, Cliff D, Malhotra A (1996) Creatures: artificial life autonomous software agents for home entertainment. In: Proceedings of the first international conference on autonomous agents, Marina del Rey, CA, February 1997. ACM Press, New York
- Graubard SR (ed) (1988) *Daedalus, Winter 1988: Artificial intelligence*. American Academy of Arts and Sciences, issued as Proc Am Acad Arts Sci 117(1)
- Gross D, McMullin B (2001) Is it the right ansatz? *Artif Life* 7(4):355–365
- Groß R, Bonani M, Mondada F, Dorigo M (2006) Autonomous self-assembly in swarm-bots. *IEEE Trans Robot* 22(6):1115–1130
- Hamarneh G, McIntosh C, McInerney T, Terzopoulos D (2009) Deformable organisms: an artificial life

- framework for automated medical image analysis. Chapman & Hall/CRC, Boca Raton, FL, p 433
- Harvey I, Paolo E, Wood R, Quinn M, Tuci E (2005) Evolutionary robotics: a new scientific tool for studying cognition. *Artif Life* 11(1–2):79–98
- Haugeland J (1989) Artificial intelligence: the very idea. MIT Press, Cambridge, MA
- Herman G, Lindenmayer A, Rozenberg G (1975) Description of developmental languages using recurrence systems. *Math Syst Theory* 8:316–341
- Hillis W (1989) The connection machine. MIT Press, Cambridge, MA
- Hofstadter DR (1979) Gödel, Escher, Bach: An eternal golden braid. Penguin Books, London. First published in Great Britain by The Harvester Press Ltd 1979. Published in Penguin Books 1980
- Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor MI
- Holland JH (1976) Studies of the spontaneous emergence of self-replicating systems using cellular automata and formal grammars. In: Lindenmayer A, Rozenberg G (eds) Automata, languages, development. North-Holland, New York, pp 385–404. Proceedings of a conference held in Noordwijkerhout, Holland, 31 March–6 April 1975
- Jain S, Krishna S (2001) A model for the emergence of cooperation, interdependence and structure in evolving networks. *PNAS* 98:543–547
- Kampis G (1991) Self-modifying systems in biology and cognitive science. IFSR international series on systems science and engineering, vol 6. Pergamon Press, Oxford, NY. Editor-in-Chief: George J. Klir
- Kato N, Okuno T, Okano A, Kanoh H, Nishihara S (1998) An ALife approach to modelling virtual cities. In: IEEE Conference on Systems, Man and Cybernetics, vol 2, San Diego, CA, October 1998. IEEE Press, New York, vol 2, pp 1168–1173
- Kauffman SA (1993) The origins of order: self-organization and selection in evolution. Oxford University Press, Oxford
- Kemeny JG (1955) Man viewed as a machine. *Sci Am* 192(4):58–67
- Kim K, Cho S (2006) A comprehensive overview of the applications of artificial life. *Artif Life* 12(1): 153–182
- Kitano H (2002) Systems biology: a brief overview. *Science* 295:1662–1664
- Klein J (2002) Breve: a 3D environment for the simulation of decentralized systems and artificial life. In: Proceedings of artificial life VIII, the eighth international conference on the simulation and synthesis of living systems, Sydney, Australia, December 2002. MIT Press, Cambridge, MA, pp 329–334
- Koza J (1992) Genetic programming. MIT Press, Cambridge, MA
- Kvasnicka V, Pospíchal J, Kaláb T (2001) A study of replicators and hypercycles by typogenetics. In: ECAL 2001: Proceedings of the sixth European conference on advances in artificial life, Prague, Czech Republic, September 2001. Springer-Verlag, Berlin, pp 37–54
- Langton CG (1984a) Self-reproduction in cellular automata. *Physica D* 10:135–144
- Langton CG (1992a) Life at the edge of chaos. In: Langton C, Taylor C, Farmer J, Rasmussen S (eds) Artificial life II: Proceedings of the second workshop on artificial life. Addison-Wesley, Redwood, CA
- Langton CG (1995) What is artificial life? In: Brown T (ed) comp.ai.alife Frequently Asked Questions (FAQ), November 18th 1995. <http://www.faqs.org/faqs/ai-faq/alife/>
- Langton CG (1997) Artificial life: an overview. MIT Press, Cambridge, MA
- Langton CG (1984b) Self-reproduction in cellular automata. *Physica* 10D:135–144
- Langton CG (1986) Studying artificial life with cellular automata. *Physica* 22D:120–149
- Langton CG (1987) Announcement of artificial life workshop. http://groups.google.com/group/news.announce._conferences/browse_thread/thread/776a2af47daf7a73/7ab70ed772c50437, posted to news.announce.conferences, April 20 1987, 7:22 pm
- Langton CG (ed) (1989) Artificial Life. Santa Fe Institute studies in the sciences of complexity, vol VI. Addison-Wesley, Redwood City, CA. Proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems, Los Alamos, NM, September 1987
- Langton CG (1992b) Preface. In: Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) Artificial life II. Santa Fe Institute Studies in the Sciences of Complexity, vol X. Addison-Wesley, Redwood City, CA. Proceedings of the workshop on artificial life, Santa Fe, NM, February 1990
- Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) (1992) Artificial life II. Santa Fe Institute Studies in the sciences of complexity, vol X. Addison-Wesley, Redwood City, CA. Proceedings of the workshop on artificial life Santa Fe, NM, February 1990
- Lee K, Choi M, Hong Q, Lee J (2007) Group behavior from video: a data-driven approach to crowd simulation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation. San Diego, CA, August 2007, pp 109–118
- Lenton T (2002) Gaia as a complex adaptive system. *Philos Trans R Soc B Biol Sci* 357(1421):683–695
- Levy S (1993) Artificial life: a report from the frontier where computers meet biology. Vintage, New York
- Lindenmayer A (1968) Mathematical models for cellular interaction in development, parts I and II. *J Theor Biol* 18:280–315

- Lindenmayer A, Rozenberg G (eds) (1976) Automata, languages, development. Elsevier, Amsterdam, The Netherlands
- Lindgren K, Nordahl M (1994) Cooperation and community structure in artificial ecosystems. *Artif Life* 1:15–37
- Lovelock J (1989) Geophysiology, the science of Gaia. *Rev Geophys* 27(2):215–222
- Lucic P, Teodorovic D (2002) Transportation modeling: an artificial life approach. In: ICTAI 2002: Proceedings of the 14th IEEE international conference on tools with artificial intelligence. Washington, DC, November 2002, pp 216–223
- Luisi PL, Varela F (1989) Self replicating micelles: a minimal version of a chemical autopoietic system. *Orig Life Evol Biosph* 19:633–643
- Maes P (1995) Artificial life meets entertainment: lifelike autonomous agents. *Commun ACM* 38(11):108–114
- Magnenat-Thalmann N, Thalmann D (eds) (1994) Artificial life and virtual reality. Wiley, Chichester, UK
- Magnoli G, Bonanni L, Khalaf R (2002) Designing a DNA for adaptive architecture: a new built environment for social sustainability. In: Brebbia C, Sucharov L, Pascolo P (eds) Design and nature: comparing design in nature with science and engineering. WIT, Southampton, UK, pp 203–213
- Maturana HR, Varela FJ (1973) Autopoiesis: the organization of the living. In: Autopoiesis and cognition: the realization of the living. Boston studies in the philosophy of science, vol 42. D. Reidel Publishing Company, Dordrecht, Holland, pp 59–138. First published 1972 in Chile under the title De Maquinas y Seres Vivos, Editorial Universitaria S.A.
- Maynard Smith J, Szathmáry E (1997) The major transitions in evolution. Oxford University Press, New York
- McMullin B (1992) The Holland α -universes revisited. In: Varela FJ, Bourgine P (eds) Toward a practice of autonomous systems: proceedings of the first European conference on artificial life. Complex Adaptive Systems. MIT Press, Cambridge, MA, pp 317–326. series Advisors: John H. Holland, Christopher Langton and Stewart W. Wilson
- McMullin B (2000a) John von Neumann and the evolutionary growth of complexity: looking backwards, looking forwards. *Artif Life* 6(4):347–361. <http://www.eeng.dcu.ie/~alife/bmcm-alj-2000/>
- McMullin B (2000b) Some remarks on autocatalysis and autopoiesis. *Ann N Y Acad Sci* 901:163–174. <http://www.eeng.dcu.ie/~alife/bmcm9901/>
- McMullin B (2004) 30 years of computational autopoiesis: a review. *Artif Life* 10(3):277–296. <http://www.eeng.dcu.ie/~alife/bmcm-alj-2004/>
- McMullin B, Varela FJ (1997) Rediscovering computational autopoiesis. In: Husbands P, Harvey I (eds) ECAL-97: Proceedings of the fourth European conference on artificial life, Brighton, UK, July 1997. Complex adaptive systems. MIT Press, Cambridge, MA. <http://www.eeng.dcu.ie/~alife/bmcm-ecal97/>
- McMullin B, Taylor T, von Kamp A (2001) Who needs genomes? In: Atlantic symposium on computational biology and genome information systems & technology, Regal University Center, Durham, NC, March 2001. <http://www.eeng.dcu.ie/~alife/bmcm-cbgi-2001/>
- Meinhardt H (2003) The algorithmic beauty of sea shells. Springer, Heidelberg
- Meyer J, Wilson S (eds) (1991) From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior, London, 1990. MIT Press Cambridge, MA
- Miranda F, Köglér J, Del Moral Hernandez E, Lobo Netto M (2001) An artificial life approach for the animation of cognitive characters. *Comput Graph* 25(6): 955–964
- Mondada F, Pettinari G, Guignard A, Kwee I, Floreano D, Deneubourg J, Nolfi S, Gambardella L, Dorigo M (2004) SWARM-BOT: a new distributed robotic concept. *Autonomous Robot* 17(2):193–221
- Moura L (2004) Man + robots: symbiotic art. Institut d'Art Contemporain, Villeurbanne, France
- Okuhara K, Domoto E, Ueno N, Fujita H (2003) Recycling design using the artificial life technology to optimize evaluation function. In: ECO Design'03: 2003 third international symposium on environmentally conscious design and inverse manufacturing. Tokyo, Japan, December 2003, pp 662–665
- Pargellis AN (2001) Digital life behavior in the amoeba world. *Artif Life* 7(1):63–75. doi: 10.1162/106454601300328025, <http://www.mitpressjournals.org/doi/abs/10.1162/106454601300328025>, <http://www.mitpressjournals.org/doi/pdf/10.1162/106454601300328025>
- Pattee H (ed) (1973) Hierarchy theory. The challenge of complex systems. Georges Braziller, New York
- Paun G (1998) Computing with membranes. Technical Report, Turku Center for Computer Science, TUCS Report 208
- Paun G, Rozenberg G (2002) A guide to membrane computing. *Theor Comput Sci* 287:73–100
- Pesavento U (1995) An implementation of von Neumann's self-reproducing machine. *Artif Life* 2(4):337–354
- Pfeifer R, Iida F, Bongard J (2005) New robotics: design principles for intelligent systems. *Artif Life* 11 (1–2):99–120
- Plsek P, Greenhalgh T (2001) The challenge of complexity in health care. *Br Med J* 323(7313): 625–628
- Pollack J, Blair A, Land M (1997) Coevolution of a backgammon player. In: Artificial life V: Proceedings of the fifth international workshop on the synthesis and simulation of living systems. Nara, Japan, May 1996, pp 92–98

- Popper KR, Eccles JC (1977) *The self and its brain: an argument for interactionism*. Routledge & Kegan Paul plc, London. First published 1977, Springer-Verlag, Berlin. This edition first published 1983
- Prokopenko M, Wang P, Valencia P, Price D, Foreman M, Farmer A (2005) Self-organizing hierarchies in sensor and communication networks. *Artif Life* 11(4):407–426
- Prusinkiewicz P, Lindenmayer A (1990) *The algorithmic beauty of plants*. Springer, New York
- Rabin S (2002) *AI game programming wisdom*. Charles River Media, Hingham, MA
- Rashevsky N (1940) *Advances and applications of mathematical biology*. University of Chicago Press, Chicago, IL
- Rasmussen S (2008) *Protocells*. MIT Press, Cambridge, MA
- Rasmussen S, Knudsen C, Feldberg R, Hindsholm M (1990) The Coreworld: emergence and evolution of cooperative structures in a computational chemistry. *Physica* 42D:111–134
- Rasmussen S, Baas NA, Mayer B, Nilsson M (2001a) Defense of the ansatz for dynamical hierarchies. *Artif Life* 7(4):367–373
- Rasmussen S, Baas NA, Mayer B, Nilsson M, Olesen MW (2001b) Ansatz for dynamical hierarchies. *Artif Life* 7(4):329–353
- Rasmussen S, Bedau MA, Chen L, Deamer D, Krakauer DC, Packard NH, Stadler PF (eds) (2008) *Protocells: bridging nonliving and living matter*. MIT Press, Cambridge, MA
- Ratze C, Gillet F, Mueller J, Stoffel K (2007) Simulation modelling of ecological hierarchies in constructive dynamical systems. *Ecol Complexity* 4:13–25
- Ray T (1994) An evolutionary approach to synthetic biology. *Artif Life* 1(1/2):179–209
- Ray T (1998) Selecting naturally for differentiation: preliminary evolutionary results. *Complexity* 3(5):25–33
- Ray TS (1992) An approach to the synthesis of life. In: Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) *Artificial life II*. Santa Fe Institute studies in the sciences of complexity, vol X. Addison-Wesley, Redwood City, CA, pp 371–408
- Ray TS (1995) A proposal to create two biodiversity reserves: one digital and one organic. <http://life.ou.edu/pubs/reserves/>, Project proposal for Network Tierra
- Rechenberg I (1975) Evolution strategies (in German). Holzmann-Froboeg, Stuttgart, Germany
- Reeke GN Jr, Edelman GM (1988) Daedalus, Winter 1998: Real brains and artificial intelligence. *Proc Am Acad Arts Sci* 117(1):143–173
- Reggia J, Armentrout S, Chou H, Peng Y (1993) Simple systems that exhibit self-directed replication. *Science* 259:1282–1287
- Regis E (2008) What is life? Investigating the nature of life in the age of synthetic biology. Farrar, Straus & Giroux, New York
- Reynolds C (1987) Flocks, herds and schools: a distributed behavioral model. In: *Proceedings of the 14th annual conference on computer graphics and interactive techniques*, July 1987. ACM, New York, Anaheim, CA, July 1987, pp 25–34
- Reynolds C (1999) Steering behaviors for autonomous characters. In: Game developers conference, vol 1999, San Jose, CA, 1999, pp 763–782
- Rinaldo K (1998) Artificial life art, introduction to the special section of Leonardo. *Leonardo* 31:370
- Rocha LM (ed) (2000) *The physics and evolution of symbols and codes: reflections on the work of Howard Pattee*. BioSystems (special issue) 60(1–3): 149–157. Elsevier
- Romero J, Machado P (2007) *The art of artificial evolution: a handbook of evolutionary art and music*. Springer, Berlin
- Ronald E, Sipper M (2000) Engineering, emergent engineering, and artificial life: unsurprise, unsurprising surprise, and surprising surprise. In: *Artificial life VII: Proceedings of the seventh international conference on artificial life*. The MIT Press, Cambridge, MA, pp 523–528
- Rosen R (1959) On a logical paradox implicit in the notion of a self-reproducing automaton. *Bull Math Biophys* 21:387–394
- Rosen R (1972) Some relational cell models: the metabolism-repair systems. In: Rosen R (ed) *Foundations of mathematical biology*, vol II. Academic, New York
- Rosen R (1985) Organisms as causal systems which are not mechanisms: an essay into the nature of complexity. In: Rosen R (ed) *Theoretical biology and complexity*. Academic, Orlando, FL, Chap 3, pp 165–203
- Rosen R (1991) *Life itself*. Columbia University Press, New York
- Rothblum U (1975) Algebraic eigenspace of nonnegative matrices. *Linear Algebra Appl* 12:281–292
- Rumelhart DE, McClelland JL, the PDP Research Group (1986) *Parallel distributed processing*, vol 1: Foundations. MIT Press, Cambridge, MA
- Salomaa A (1973) *Formal languages*. Academic Press, New York
- Samuel A (July 1959) Some studies in machine learning using the game of checkers. *IBM J* 3(3): 210–229, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5392560
- Searle JR (1980) Minds, brains, and programs. *Behav Brain Sci* 3:417–457, includes peer commentaries. Also reprinted (without commentaries) as Haugeland J (ed) (1981) *Mild design*. MIT Press, Cambridge, MA, Chap 10, pp 282–305 and Boden MA (ed) (1990) *The philosophy of artificial intelligence*.

- Oxford readings in philosophy. Oxford University Press, Oxford, Chap 3, pp 67–88
- Semple J, Woolridge N, Lumsden C (2005) Review: *in vitro, in vivo, in silico: computational systems in tissue engineering and regenerative medicine*. *Tissue Eng* 11(3–4):341–356
- Shao W, Terzopoulos D (2007) Autonomous pedestrians. *Graph Model* 69(5–6):246–274
- Sims K (1991) Artificial evolution for computer graphics. *Comput Graph* 25(4):319–328
- Sims K (1994) Evolving 3D morphology and behavior by competition. In: *Artificial life IV: Proceedings of the fourth international workshop on the synthesis and simulation of living systems*. MIT Press, Cambridge, MA, pp 28–39
- Sipper M (1998) Fifty years of research on self-replication: an overview. *Artif Life* 4(3):237–257
- Smith HO, Hutchison CA, Pfannkoch C, Venter JC (2003) Generating a synthetic genome by whole genome assembly: ϕ x174 bacteriophage from synthetic oligonucleotides. *Proc Natl Acad Sci* 100(26):15440–15445
- Snare A (1999) Typogenetics. <http://www.csse.monash.edu.au/hons/projects/1999/Andrew.Snare/thesis.pdf>, Thesis for Bachelor of Science (Computer Science) Honours, School of Computer Science and Software Engineering, Monash University
- Spafford E (1994) Computer viruses as artif life. *Artif Life* 1(3):249–265
- Spector L, Klein J, Perry C, Feinstein M (2005) Emergence of collective behavior in evolving populations of flying agents. *Genet Programming Evolvable Mach* 6(1):111–125
- Speroni di Fenizio P, Dittrich P, Zeigler J, Banzhaf W (2000) Towards a theory of organizations. In: German workshop on artificial life, vol 5, Bayreuth, Germany, April 2000. <http://www.cs.mun.ca/~banzhaf>
- Steels L, Brooks R (1995) The artificial life route to artificial intelligence. Lawrence Erlbaum Associates, New Haven, CT
- Straatman B, White R, Banzhaf W (2008) An artificial chemistry-based model of economies. In: *Artificial Life XI: Proceedings of the eleventh international conference on simulation and synthesis of living systems*. MIT Press, Cambridge, MA, pp 592–602
- Strand E, Huse G, Giske J (2002) Artificial evolution of life history and behaviour. *Am Nat* 159:624–644
- Strogatz S (2001) Exploring complex networks. *Nature* 410:268–276
- Sugiyama K, Suzuki H, Shiose T, Kawakami H, Katai O (2003) Evolution of rewriting rule sets using string-based tierra. In: Banzhaf W et al. (eds) *Advances in artificial life - Proceedings ECAL 2003*, Budapest, Hungary, April 2003. Lecture notes in computer science, vol 2801. Springer, Berlin, pp 69–77
- Tanner H, Jadabaia A, Pappas G (2005) Flocking in fixed and switching networks. In: IEEE conference on decision and control, vol 1, Seville, Spain, December 2005, p 2
- Taylor C, Jefferson D (1994) Artificial life as a tool for biological inquiry. *Artif Life* 1(1–2):1–13
- Terzopoulos D (1999) Artificial life for computer graphics. *Commu of the ACM* 42(8):32–42
- Terzopoulos D, Tu X, Grzeszczuk R (1994) Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In: *Artificial life IV: Proceedings of the fourth international workshop on the synthesis and simulation of living systems*. MIT Press, Cambridge, MA, pp 17–27
- Thatcher JW (1970) Universality in the von Neumann cellular model. In: Burks AW (ed) *Essays on cellular automata*. University of Illinois Press, Urbana, IL, pp 132–186 (Essay Five)
- Theraulaz G, Bonabeau E (1999) A brief history of stigmergy. *Artif Life* 5(2):97–116
- Todd S, Latham W (1992) Artificial life or surreal art. In: *Toward a practice of autonomous systems: Proceedings of the first European conference on artificial life*. MIT Press, Cambridge, MA, p 504
- Todd S, Latham W (1994) Evolutionary art and computers. Academic London
- Toffoli T, Margolus N (1987) *Cellular automata machines*. MIT Press, Cambridge, MA
- Turing A (1936) On computable numbers, with an application to the Entscheidungsproblem. *Proc London Math Soc Ser 2* 42:230–265
- Turing AM (1950) Computing machinery and intelligence. *Mind* LIX(236):433–460, also reprinted as, Boden MA (ed) (1990) *The philosophy of artificial intelligence*. Oxford readings in philosophy. Oxford University Press, Oxford, Chap 2, pp 40–66
- Turing AM (1952) The chemical basis of morphogenesis. *Philos Trans R Soc B* 237:37–72
- Varela FJ, Maturana HR, Uribe R (1974) Autopoiesis: the organization of living systems, its characterization and a model. *BioSystems* 5:187–196
- Varela FJ, Thompson ET, Rosch E (1992) The embodied mind: cognitive science and human experience. MIT Press, Cambridge, MA
- von Neumann J (1945) First draft of a report on the EDVAC. (A corrected version was formally published in the IEEE Ann Hist Comput 15(4):27–75 1993.)
- von Neumann J (1949) Theory and organization of complicated automata. In: Burks AW (ed) (1966) *Theory of self-reproducing automata* [by] John von Neumann. University of Illinois Press, Urbana, IL, pp 27– (Part One), based on transcripts of lectures delivered at the University of Illinois, in December 1949. Edited for publication by A.W. Burks

- von Neumann J (1951) The general and logical theory of automata. In: Taub AH (ed) John von Neumann: collected works. Volume V: Design of computers, theory of automata and numerical analysis. Pergamon Press, Oxford, chap 9, pp 288–328. First published Jeffress LAA (ed) (1951) Cerebral mechanisms in behavior—the Hixon symposium. Wiley, New York, pp 1–41
- von Neumann J (1966) The theory of automata: Construction, reproduction, homogeneity. In: Burks AW (ed) (1966) Theory of self reproducing automata [by] John von Neumann. University of Illinois Press, Urbana, IL, pp 89–250. Based on an unfinished manuscript by von Neumann. Edited for publication by A.W. Burks
- Vyssotsky VA (1972) Darwin: a game of survival and (hopefully) evolution. Software Pract Experience 2: 91–96. <http://www.cs.dartmouth.edu/doug/darwin.pdf>, Attachment to letter appearing in Computer Recreations column, signed by M. D. McIlroy, R. Morris, and V. A. Vyssotsky
- Waddington CH (ed) (1969) Towards a theoretical biology. 2: Sketches. Edinburgh University Press, Edinburgh, UK
- Wang F, Tang S (2004) Artificial societies for integrated and sustainable development of metropolitan systems. IEEE Intell Syst 19(4):82–87
- Watts D (1999) Small worlds: the dynamics of networks between order and randomness. Princeton University Press, Princeton, NJ
- White R, Engelen G (1993) Cellular automata and fractal urban form: a cellular modelling approach to the evolution of urban land-use patterns. Environ Plan A 25:1175–1199
- Whitelaw M (2004) Metacreation: art and artificial life. MIT Press, Cambridge, MA
- Wiener N (1965) Cybernetics, 2nd edn: or the control and communication in the animal and the machine. MIT Press, Cambridge, MA
- Wilkinson D (2003) The fundamental processes in ecology: a thought experiment on extraterrestrial biospheres. Biol Rev 78:171–179
- Willis M, Hiden H, Marenbach P, McKay B, Montague GA (1997) Genetic programming: an introduction and survey of applications. In: Zalzala A (ed) GALE-SIA'97: Second international conference on genetic algorithms in engineering systems: innovations and applications, Glasgow, UK, September 1997. Institution of Electrical Engineers (IEE), Savoy Place, London, UK, pp 314–319, <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00681044>
- Wolfram S (1984) Computation theory of cellular automata. Commun Math Phys 96(1):15–57
- Wolfram S (1986) Theory and applications of cellular automata. Advanced Series on Complex Systems. World Scientific Publication, Singapore
- Wolfram S (1994) Cellular automata and complexity: collected papers. Perseus Books Group, Reading, MA
- Wolkenhauer O (2007) Why systems biology is (not) systems biology. BIOforum Europe 2007, vol 4, pp 2–3
- Yovits MC, Cameron S (eds) (1960) Self-organizing systems. Pergamon Press, Oxford. Proceedings of an interdisciplinary conference, 5 and 6 May, 1959
- Zenobia B, Weber C, Daim T (2008) Artificial markets: a review and assessment of a new venue for innovation research. Technovation. doi:10.1016/j.technovation.2008.09.002

54 Algorithmic Systems Biology — Computer Science Propels Systems Biology

Corrado Priami

Microsoft Research, University of Trento Centre for Computational and
Systems Biology (CoSBi), Trento, Italy
DISI, University of Trento, Trento, Italy
priami@cosbi.eu

1	<i>Introduction</i>	1836
2	<i>Model Specification</i>	1841
3	<i>Knowledge Inference</i>	1853
4	<i>CoSBi Lab: An Algorithmic Systems Biology Platform</i>	1855
5	<i>Conclusions</i>	1859

Abstract

The convergence between computer science and biology occurred in successive waves involving deeper and deeper concepts of computing. The current situation makes computer science a suitable candidate to become a philosophical foundation for systems biology with the same importance as mathematics, chemistry, and physics. Systems biology is a complex and expanding applicative domain that can open completely new avenues of research in computing and eventually help it become a natural, quantitative science. This chapter highlights the benefits of relying on an algorithmic approach to model, simulate, and analyze biological systems. The key techniques surveyed are related to programming languages and concurrency theory as they are the main tools to express in an executable form the key feature of computing: algorithms and the coupling executor/execution of descriptions. The concentration here is on conceptual tools that are also supported by computational platforms, thus making them suitable candidates to tackle real problems.

1 Introduction

Computing and biology have been converging for the past two decades, but with a vision of computing as a service to biology that has propelled bioinformatics. This field addresses structural and static aspects of biology and produced databases, patterns of manipulation and comparison, searching tools and data mining techniques (Spengler 2000; Roos 2001). The most relevant success has been the Human Genome Project that was made possible by the selection of the right language abstraction for representing DNA (a language over a 4-character alphabet) (Searls 2002).

In biology, there is now a heightened interest in system dynamics for interpreting living organisms as information manipulators (Hood and Galas 2003) and moving toward systems biology (Kitano 2002). There is no general agreement on a definition of systems biology, but whatever definition one selects, it must embrace at least four characterizing concepts, as systems biology transitions in the following ways:

1. From qualitative biology toward a quantitative science. Often, biological phenomena are explained through textual descriptions and sometimes cartoons that highlight the interaction of components without any quantitative information. Furthermore, the cartoons are superpositions of different snapshots of the interconnection network of biological elements that vary over time. As a consequence, temporal ordering of events is also not easily retrievable.
2. From reductionism to system-level understanding of biological phenomena. Biology has mainly studied single components of systems following the idea that by putting together the knowledge on the single components the system behavior would also have been elucidated. The emergent behavior due to interaction of subsystems makes the unravelling of the system dynamics impossible.
3. From structural, static descriptions to functional, dynamic properties. The list of components and their three-dimensional structure coupled with the description of their chemical and physical properties does not allow one to infer the way in which those components operate in context. A focus on the interactions (dynamics) is needed from the very beginning of the studies.

4. From descriptive biology to mechanistic/causal biology. Although some attempts to model the dynamics of systems exist (and hence partially move in the right direction with respect to item 3 above), most of the approaches hide within mathematical variables the elementary, mechanistic steps that a system performs to accomplish a goal. As a consequence, it is very difficult to disentangle the causality of events that govern the evolution of a system.

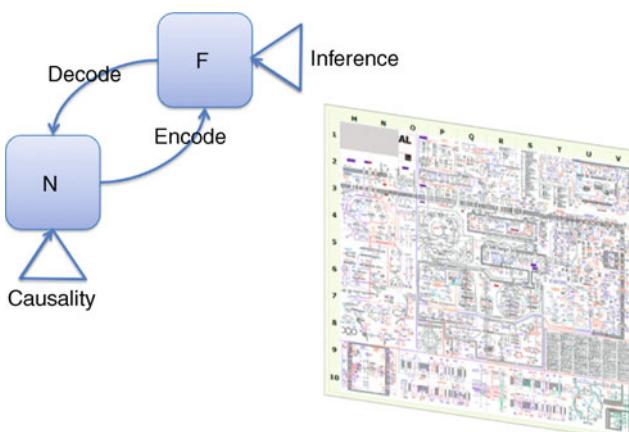
The above features highlight that causality between events, temporal ordering of interactions, and spatial distribution of components, within the reference volume of the system at hand, are becoming essential. This poses new challenges to describe the step-by-step mechanistic behavior that enables phenotypical phenomena, a behavior that both bioinformatics and classical mathematical modeling do not address (Cassman et al. 2005).

The complexity of studying the behavior of whole systems, rather than small isolated parts, calls for a stronger formalization of biological concepts that allows one to automatically manipulate biological knowledge. Therefore, the role of modeling becomes an essential and integral part of the new discipline of systems biology. The metabolic network represented in the right part of Fig. 1 is an illuminating example that informal reasoning cannot provide reasonable outcomes. Models can no longer be simple cartoons like the ones found in biology textbooks and cannot be equations, because they hide both the mechanistic behavior of systems and the inherent concurrency of biology in representations that are too abstract.

Rosen's description of modeling (left part of Fig. 1) perfectly fits the situation (Rosen 1998) as well as his statement "I have been, and remain, entirely committed to the idea that modeling is the essence of science..." Nature is governed by causal laws that make systems vary over time and space. The complexity of real systems makes it impossible to study and analyze them directly. Abstractions are mandatory to encode into a formal system, the relevant characteristics needed to study the properties of the real system in which one is interested. One then uses the formal system corresponding to the abstract representation to perform inferences and proofs of properties. In this way one derives some knowledge that is only true in

Fig. 1

Left Rosen's representation of the relationships between nature and formal models. Right: a graphical representation of a metabolic network.



the abstract world; one then needs to *decode* the findings and reinterpret them in the natural world by establishing a precise coupling between inference and causality. However, this is exactly what computer science has been doing since its early days, by using algorithms and semantics of programming languages, to ensure strong and coherent coupling of phenomena at different levels of abstraction. Therefore, algorithms and formal specification languages are suitable candidates to address the modeling challenges of systems biology.

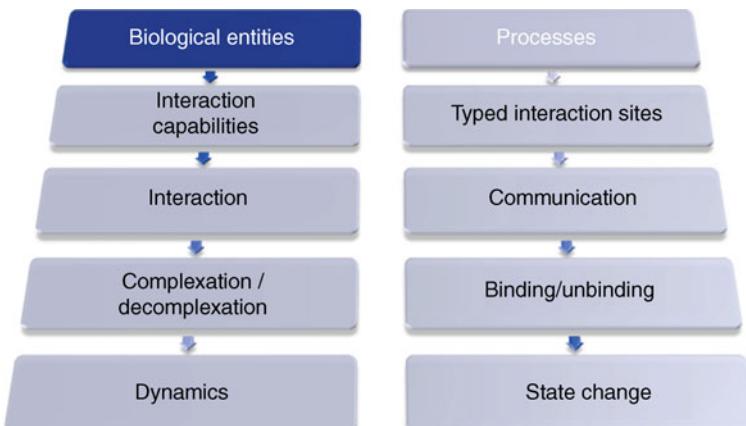
Algorithms expressed by computing languages are a conceptual tool (Wing 2006) that helps elucidate fundamental biological principles by forcing modelers/biologists to think about the mechanisms governing the behavior of the system under question at the right level of abstraction. Since the dynamics of biological systems are mainly driven by quantities such as concentrations, temperatures, gradients, etc., one must clearly focus on quantitative algorithms and languages, that is, the selection mechanism of the next step is determined according to probabilistic/temporal distributions. These distributions allow one to model the kinetics (speed) of interactions that is determined by the chemical and physical quantities listed above. Furthermore, algorithms can help in coherently extracting and organizing general biological principles that underlie the huge amount of data produced by high-throughput technologies, thus producing knowledge from information (data).

Algorithms need a syntax to be described and a semantics to be associated with the descriptions of their intended meaning so that an executor can precisely perform the steps needed to implement the algorithms with no ambiguity. In this way, one is entering the realm of programming languages from both a theoretical and practical perspective.

The use of programming languages to model biological systems is an emerging field that enhances current modeling capabilities (richness of aspects that can be described as well as user-friendliness, compositability and reusability of models) (Priami and Quaglia 2004). The metaphor that inspires this (see ▶ Fig. 2) is one where biological entities are represented as programs being executed simultaneously and the interaction of two entities is represented by the exchange of a message between the programs representing those entities in the

Fig. 2

The metaphor underlying algorithmic systems biology. The left column represents biological objects and their properties and capabilities, while the right column represents the corresponding computer science concepts.



model (Regev and Shapiro 2003). Note that interaction capabilities could also be represented with untyped interfaces or simply through channel names as in most approaches. However, the typing information separates the names used to communicate/interact between entities and the information on the shapes/types of the active domains/receptors that de facto determine the sensitivity or affinity between entities. Therefore, the modeling process is much closer to the natural behavior of systems and helps the modeler think in terms of the fundamental blocks of biological systems. The biological entities involved in the biological process and the corresponding programs in the abstract model are in a 1:1 correspondence, thus coping by construction with the combinatorial explosion of variables needed in the mathematical approach to describe the entire set of states through which a single component can pass.

The simultaneity of the execution of programs requires one to take concurrency into account. Concurrency must not be considered a tool to improve the performance of sequential programming languages and architectures, which is the standard practice in most actual cases. Concurrency is the key concept that permits the execution of algorithms to exhibit the emergent behavior produced at system level by the set of local interactions between components without the need to specify it from the beginning. This aspect is crucial for the predictive power of this approach.

Some programming languages that address concurrency as a core primitive issue and aim at modeling biological systems are emerging, for example, Welch and Barnes (2005) and Dematté et al. (2008b), from the field of process calculi (Bergstra et al. 2001). These concurrent programming languages are suitable candidates to easily and efficiently express the mechanistic rules that propel algorithmic systems biology (Priami 2009). Therefore, it seems natural to check whether the programming and analysis techniques, developed for computer networks and their formal theories, could bring new light to biology when suitably adapted. Note that process calculi are not the only theoretical ground for algorithmic systems biology. Petri nets, logic, rewriting systems, and membrane computing are other relevant examples of formal methods applied to systems biology (for a collection of tutorials see Bernardo et al. (2008) as well as the other chapters of this book). Other approaches that are more closely related to software design principles are the adaptation of UML to biological issues (see www.biouml.org) and statecharts (Harel 2007). Finally, cellular automata (Gutowitz 1990) need to be considered as well with their game of life.

Scalability is an issue for systems biology. Design principles of large software systems can help in developing an algorithmic discipline for systems biology in order to move from toy examples to real case studies. A library of biological components can be easily built and used to derive models of large systems that are ready to be simulated and analyzed just by composing the available modules (Dematté et al. 2008a). Note that here we refer to shallow compositionality (de Alfaro et al. 2001) (i.e., syntactic composition of modules) and hence the well-known problem of composing stochastic semantics does not affect the definition of libraries.

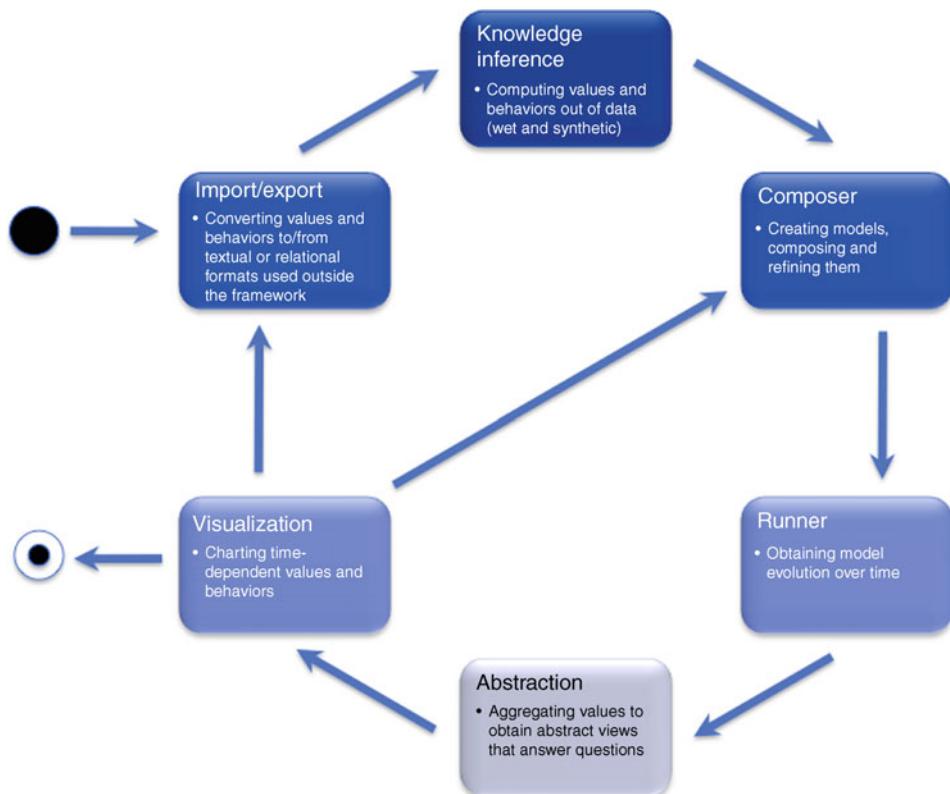
Algorithmic systems biology raises novel issues in computing by stepping away from qualitative descriptions, typical of programming languages, toward a new quantitative computing so that it can fully become an experimental science, as advocated in Denning (2007), that is suitable to support systems biology. This would also foster the move toward a simulation-based science that is needed to address the larger and larger dimensions and complexity of scientific questions and available data. Algorithmic systems biology fully adopts the main assets of computer science: hierarchical, systematic, and algorithmic (computational) thinking and creativity in modeling, programming, and innovating.

1.1 Organization of the Chapter

The organization of the chapter follows a high-level schematic workflow of how models and experiments can be tightly integrated in the algorithmic systems biology vision (see [Fig. 3](#)). The next section deals with modeling languages derived from process calculi, which are described in the chapter [Process Calculi, Systems Biology and Artificial Chemistry](#) of this handbook. The modeling language is the core component of the composer in [Fig. 3](#). In order to feed models with quantitative information needed to drive simulations, [Sect. 3](#) discusses knowledge inference. Then, in [Sect. 4](#) the simulation of models is addressed and the features of a computational platform are described. These aspects are part of the runner box in the figure that mainly relates to the quantitative implementation of the modeling language. The same section also discusses the post-processing of data obtained from the execution of biological models and shows how to infer new hypotheses to restart the workflow. The key activities of post-processing are collected in the boxes *Abstraction* and *Visualization* in the figure.

Fig. 3

A typical workflow for algorithmic systems biology. All boxes contain examples of the activities that can be performed. (Figure prepared by Ivan Mura.)



2 Model Specification

The main challenges for building algorithmic models for the system-level understanding of biological processes include the relationship between low-level local interactions and emergent high-level global behavior; the partial knowledge of the systems under investigation; the multilevel and multi-scale representations in time, space, and size; the causal relations between interactions and the context-awareness of the inner components. Therefore, the modeling formalisms that are candidates to propel algorithmic systems biology should be complementary to and interoperable with mathematical modeling; be algorithmic and quantitative; be interaction-driven, compositional, scalable, and modular; address parallelism and complexity; and express causality.

Most of the issues above are naturally addressed by process calculi (see the chapter ➤ [Process Calculi, Systems Biology and Artificial Chemistry](#)), and therefore the concentration here is on the conceptual and computational platforms developed from that theoretical foundation. The brief history of this field is recalled by grouping the modeling languages into three different generations. The first two are described in the chapter ➤ [Process Calculi, Systems Biology and Artificial Chemistry](#) and the details are not entered into, but their limitations highlighted to model real biological systems in a composable and scalable manner. Guerriero et al. (2008) is referred to for a survey of process calculi applied to biology and here stochastic π -calculus (Priami 1995; Priami et al. 2001) and beta-binders (Priami and Quaglia 2004) are considered as representatives of the two generations of calculi. Then BlenX is examined as an example of the last generation of modeling languages.

2.1 First Generation of Calculi for Biology

For a long time, the only calculus equipped with a quantitative implementation to model and simulate biological systems has been the stochastic π -calculus. This calculus attracted the interest of many researchers and has been used in a variety of case studies that show the feasibility of the programming language approach to model biological systems. Although successful in proving the potential of process calculi, the stochastic π -calculus suffers from some limitations in modeling biology. In fact, it was designed to model and analyze the performance of computer networks and hence it is strictly tuned to computer science. As a matter of fact, many tricks are needed to encode biology, and some natural primitives to describe common biological phenomena are missing.

According to the metaphor, biological entities are represented by π -calculus processes (hereafter for simplicity modeling proteins and their interactions are assumed). Proteins are characterized by functional units called domains. For instance, a protein P with three domains (D_1 , D_2 , and D_3) can be represented by three processes with the same names as the corresponding domains that are glued together to form a process P representing the protein. The π -calculus has no primitive to model physical attachment of sub-processes into larger processes, and therefore it uses the scope of names to identify boundaries. Intuitively it is assumed that the three domains share a secret (a private name) that states that they are bound together to form a protein. Thus, the protein can be modeled as $P = (\text{vx})(D_1|D_2|D_3)$, where (vx) is the declaration of the new name x with scope of the process P . The behavior of P is determined by the parallel composition of its domains and the shape or physical boundaries of P are determined by the backbone channel x that is shared only by the domains.

The scope of the channel determines the physical boundaries of the biological entities. Real biological systems contain multiple copies of the same protein; since concentrations of species affect the selection mechanism of the next action to be performed in a stochastic simulation, multiplicity is modeled through parallel composition yielding $MultiP = P|..|P$. Finally, a system contains multiple species so that it ends up in $Sys = MultiP|MultiP_1|..|MultiP_n$.

The processes encoding the domains implicitly describe the interaction capabilities of the protein through the names used in their syntactic definition. Recall that the π -calculus communications are determined by complementary actions (send “!” and receive “?”) performed on the very same channel name that must be shared between the partners of the interaction. For instance, if $D_3 = a!y.D'_3$, the protein P can interact with all the other entities that are willing to perform an input on the shared channel a , for example, the protein Q made up of the single domain $D_4 = a?w.Q'_4$.

If the interaction of two proteins has to form a complex, the attachment must be programmed through communication of private names so that the scope of the communicated name is changed to include the two proteins. According to the interpretation of the scope of private names as boundaries of physical objects, one has to dynamically manipulate the scope to create and destroy different complexes during the execution. For instance, if one considers the system $((vx)(vy)P)|Q$ after the interaction of D_3 and D_4 , the new system will be $(vy)((vx)(D_1|D_2|D'_3)|D'_4)$. The residual of protein P and the residual of protein Q after the interaction share the private name y that P sent and made available to Q ; therefore, they are interpreted as being physically attached. The detachment of the proteins must be programmed as well by opening the scope of the name y . For instance, if $D'_4 = z!y.D'_4$ and it performs the output of the private name with no other process willing to perform the input on the channel z , the resulting system no longer has a restriction on the name y , which becomes available to the whole system and hence the two proteins detach.

Note that both interaction and complexation are based on a key-lock mechanism of affinity. In fact, the partners of a reaction must share the same channel and must perform synchronous complementary operations on it. Real biology does not work in the same manner. Interaction can happen even if the shape of the active domains that are going to react are not exactly complementary to one another. Hence, a given domain can interact with different strengths and probabilities with whole families of partially complementary domains. In order to model this in π -calculus-like formalisms, one must explicitly write the whole set of alternatives relying on the choice operator in the processes. For instance, if the domain D_3 can interact with n different domains, one should specify it within the protein as $P = (vx)(D_1|D_2|a_1!y.D'_3 + \dots + a_n!y.D'_3)$. Furthermore, any complementary domain must contain a receive operation on the very same channel a_i representing the compatibility between the two entities.

The main limitations (listed below) of the π -calculus family formalisms stem from the fact that these calculi have been designed for artificial, known computer systems that can interact only when the port and addresses of the partners are perfectly and exactly known. Furthermore, the identity of the programs is not an issue in computer science, in which it is enough to ensure a global behavior as described by the algorithms to be implemented.

Processes. The same syntactical concept is used to model many different biological entities at different levels of abstraction, such as domains, proteins, and whole systems.

Restriction. A primitive intended to declare names and identify their scope is used to represent compartments, membranes, and complexes.

Complex/Decomplex. Creation and destruction of complexes of biological entities is not a primitive operation, but shows up from the interplay of private names and the variation of their scopes through open and close operations. It must be programmed through particular classes of communications. A further confusion that arises in this class of calculi is that channel names are used both to define the boundaries of complexes as well as to manipulate the structure of programs. A characterization of the minimal set of primitives needed to model this phenomena is discussed in Cardelli and Zavattaro (2008).

Low-level programming. The reversibility of reactions as well as the complementarity of complexation and decomplexation must be programmed through communication and name passing. The very same structure of communications determines the network of interaction of the biological entities. In other words, one needs to specify all the arcs of the network through a complementary pair of send and receive on the same channel rather than inferring it from the sensitivity or affinity of interaction of entities as it happens in biology.

Interaction. The interaction of biological entities can occur only if the entities share the very same channel name. This implies a perfect key-lock mechanism of interaction that is not realistic in the biological domain. In fact, the interaction can happen with different strengths or probabilities depending on the complementarity of the active domains and on the concentration of the entities in the reaction volume. The non-key-lock mechanism must be coded; in case of complexation/decomplexation, one also needs to program all the reversible interactions for all the different channels representing the different shapes that can create interactions.

Incremental model building. Most of the information related to the interconnection structure of the entities and on the sensitivity/affinity of interaction is hard-coded into the syntax of the processes through the send/receive pairs. This means that whenever new knowledge, for example, on the capability of interaction of entities, is discovered, a new piece of code must be produced and distributed in various parts of the specification that are not well identified. As a consequence, incremental model building is strongly affected with a negative impact on the scalability of the approach. In fact, the compositional modeling style of all the calculi used so far in the biological context is simply a way of structuring a description of a system by identifying sub-systems that run in parallel rather than a true scalable and compositional methodology.

Identity of entities. Biological entities—for example, proteins—can take on different states during their lifetime depending on the interactions in which they participate. One of the main criticisms of the mathematical modeling of these phenomena is that a new variable is needed for any different state of the same biological entity, thus causing to the well-known combinatorial explosion of the variables and consequently equations. The very reason here is that no identity of the considered entities is maintained in the model. In the π -calculus-like approach, the identity of entities in the initial description can coincide with the processes to which names can be assigned. However, during the execution of the programs, prefixes are consumed and the identity is lost as well.

Implementation. Stochastic simulation selects the action to be performed considering the number of equal entities in the system at a given time (e.g., the number of proteins P). If one assumes that entities are represented by processes, one has to count all the pieces of programs that represent the same object. Due to name passing and the naming of channels in general, it could be that two syntactically different programs represent the very same semantic object, and, hence, the same entity. The technical tool used to address this

equivalence is commonly known in the process algebra field as structural congruence. Efficiency in computing is therefore mandatory because it must be done at any step of computation and it is not so immediate for π -calculus. Indeed the entities change their structure after any execution step and their number is then continuously changing. This is due to the fact that processes can be created and updated dynamically. Efficiency in the execution of models is also fundamental due to the large size of the real biological systems. Parallel implementation of algorithms for running models written with process calculi are not available at this stage.

Qualitative versus quantitative descriptions. Biology is mainly driven by quantities, whereas process calculi are usually qualitative descriptions of behavior. Stochastic process calculi handle the kinetics of systems by associating channel names with rates identifying exponential distributions. Most of the time, the kinetic parameters for real biological interactions are not known and must be estimated to fit some phenotypical behavior that is measurable experimentally. Therefore, mixing quantitative and qualitative information imposes the modeler to rewrite the model or to heavily work on it in order to carry out parameter estimation and sensitivity analysis. Furthermore, limiting the description of kinetics to exponential distribution may cause all the high-level abstractions like (such as Hill-functions or even more complex dynamics that are used in mathematical modeling) to explode into elementary steps in order to hide unknown details within models. Unfortunately, most of the time the details to explode these macro-processes are not available and makes it difficult to apply the approach.

Space. Many biological phenomena are highly sensitive to the localization within the reference system volume of the reactants. There is an extremely limited and difficult to manage notion of space, and it must be hard-coded in the language.

Connection with standards. A very limited and preliminary activity is emerging to connect process calculi with SBML (see e.g., Eccher and Priami (2006)). No interoperability between computational tools is available.

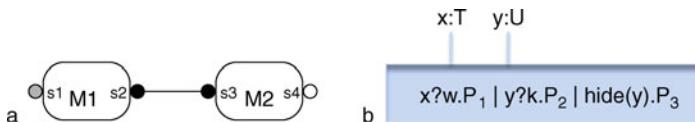
Predictive models. Most of the previous items and the introductory description show how any possible dynamic of the modeled system must be coded in the specification. This means that execution can provide the user with different paths due to different selection of alternatives or different temporal ordering of concurrent events, but all the observed action has been created by the modeler and explicitly programmed. Consequently, the predictive power of the models is limited, although it can still provide much more information on the dynamics of systems and their mechanistic behavior than classical ODE-based mathematical modeling.

2.2 Second Generation of Calculi for Biology

The second generation of process calculi directly defined to model biological systems improved the easiness of expressing basic biological principles considerably. The most representative languages are the κ -calculus (Danos and Laneve 2004) and beta-binders (Priami and Quaglia 2004, 2005) because they are equipped with computational platforms to execute specifications and are innovative with respect to many other extensions of process calculi. The main step ahead is the clear 1:1 correspondence between biological entities and objects of these calculi (see  Fig. 4). Although an entity can occur in several different states, this

Fig. 4

κ -calculus (a) and beta-binders (b) maintain the identity of biological entities by relying on boxes with interfaces that describe the interaction capabilities of the modeled entities. Besides the 1:1 correspondence between biological components and boxes specified in the model, there is also a 1:1 correspondence between functional activities of domains and interfaces (binders). The interfaces of the boxes in the κ -calculus are identified by unique names written within the boxes. Interfaces can be either bound (s_2, s_3), visible (s_4) or hidden (s_1). Protein complexes are formed by joining interfaces through arcs. The rule-based manipulation of systems simply creates, deletes, and changes links, thus manipulating complexes. The interfaces of beta-binders boxes have unique names that act as binders for their occurrences within the box (x, y). The names of the interfaces are equipped with types (T, U) over which a compatibility of interaction is defined, thus relaxing the exact complementarity of actions over the same channel name to allow interactions. The boxes of beta-binders contain a π -calculus like description of their internal behavior. Primitives to manipulate the state of the interfaces that can be active/visible or hidden are provided (see the *hide* primitive in the figure that makes a visible interface hidden).



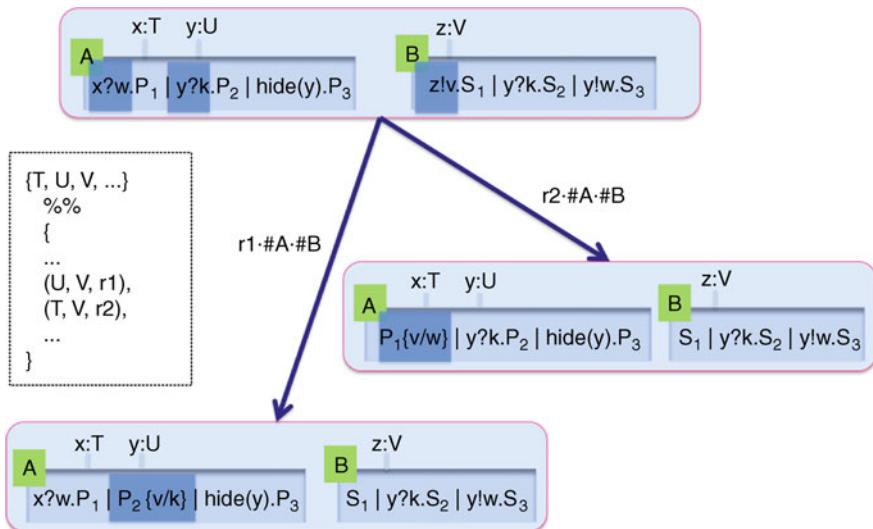
generation of calculi introduces well-defined syntactic boundaries that allow one to group all the states of an entity into single objects. This makes it easier to count and trace the behavior of the modeled entities. Furthermore, the whole set of states through which an entity can pass are implicitly encoded in the program describing the internal behavior of objects, rather than being all listed explicitly and represented with different placeholders, for example, variables in ODEs. Finally, note that other calculi have been defined as well (Cardelli 2005; Regev et al. 2004).

κ -calculus is a formal, rule-based calculus of protein interactions. It was conceived to represent complexation and decomplexation of proteins, using the concept of shared names to represent bonds. The units of κ -calculus are proteins, and operators are meant to represent creation and division of protein complexes. Once the initial system has been specified and the basic reductions have been fixed, the behavior of the system is obtained by rewriting it. This kind of reduction resembles pathway activation. Although this calculus has the merit of being directly inspired by biology, it does not offer a natural support for managing the evolution of compartments. Furthermore, the whole set of possible reactions must be specified since the beginning, so that the predictive power of the models is limited as in the previous generation of calculi.

Beta-binders are strongly inspired by both π -calculus and biological examples. They associate biological entities with boxes. Interaction is still communication-based and can occur either within a box if the same channel name is used or between different boxes if they have compatible interfaces (even if they have different names and hence the interaction is over different channel names). In fact, typed interfaces of boxes enable promiscuity of interaction (sensitivity/affinity versus complementarity). For an example, please see Fig. 5. Note that types here do not play the same role played by names in π -calculus. In fact, the same type can be compatible with many different types, thus allowing interaction over different channels and between different boxes.

Fig. 5

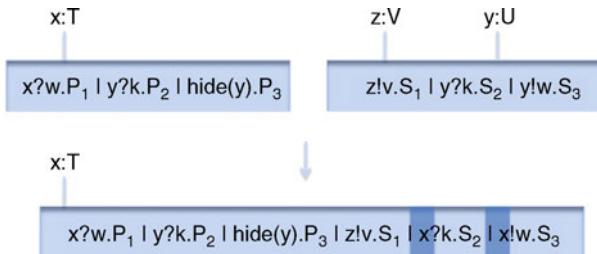
The communication between boxes in beta-binders occurs when there are interfaces with compatible types over which one box is willing to output a value and another box is willing to receive a value. For instance, the box A is willing to receive both on its interfaces $x : T$ and $y : U$, while the box B is willing to send a value v over its interface $z : V$. In a separate file, the compatibility between types is specified by inserting a tuple for any pair of types one wants to be compatible. For instance, the tuple $(U, V, r1)$ means that interfaces with types U and V can interact, and the kinetic of the interaction is specified by a rate $r1$. According to the specified compatibilities and the actions enabled over the corresponding interfaces, the two transitions depicted in the figure can occur independently of the channel names used to specify the send and receive operations. Since the compatibility also specifies the rate of the interaction, the actual rate of the transitions is computed taking into account the number of molecules available in the system ($\#A$ and $\#B$).



The enclosing surfaces (boxes) of entities cannot be nested and maintain a strict correspondence between processes and biological entities. Primitives to manipulate interfaces can be included in the description of the internal behavior of boxes. Finally, the physical rearrangement of boxes is implemented by defining completely general special split and join functions that operate on the structure of boxes and that are not inserted in the flow of control. This choice enables a higher level of nondeterminism with respect to any calculus defined so far, and better accommodates the specification of systems for which there is a considerable lack of knowledge. In fact, one does not need to completely specify, in the flow of control, the merging or division of boxes as it happens in other approaches, but one only needs to specify general conditions on the structure and on the status of the binders of boxes that enables the corresponding joining and splitting. Furthermore, the higher level of nondeterminism enhances the predictive power of models because the modeler does not have to specify the whole set of reactions from the beginning; they are instead computed by the run-time of the language. For an example, please see [Fig. 6](#).

Fig. 6

The figure shows an example of join of two boxes. The join function returns the list of binders of the new box as well as two renaming functions to change the names of the parallel composition of the original boxes to adapt them to the new binders. The function *comp* defines the affinity between the types *T* and *U* and the notation $\beta(x : T)$ denotes an interface (binder) named *x* with type *T*.



$$f_{\text{join}} = \lambda B_1 B_2 Q_1 Q_2. \text{if} [B_1 = \beta(x:T)B_1^* \text{ and } B_2 = \beta(y:U)B_2^* \text{ and } \text{comp}(T,U)] \\ \text{then } (B_1, \sigma_{\text{id}}, \{x/y\}) \text{ else nothing}$$

Although κ -calculus and beta-binders solve many of the problems identified at the end of the previous section for the first generation of calculi, they still suffer from some drawbacks listed below. (Here one uses the same categories used in the previous section, and those not listed are no longer considered an issue; the ones introduced here refer to issues raised by the second generation of calculi that were not present in the previous generation.)

Complex/Decomplex. κ -calculus provides the best support for manipulating complexes. The main limitation is that the rules for creating complexes and manipulating them must all be specified by the modeler on the basis of the structure of the boxes. No support is provided by the implementation of the language to automatically compute the steps to be performed for this fundamental mechanism of biological interaction.

Low-level programming. Almost nothing is changed on this item with respect to the previous generation of calculi. The only improvement is that the interaction network can be implicitly modeled in beta-binders by adequately defining the compatibility function between types.

Interaction. The key-lock mechanism of interaction is relaxed in beta-binders, while it is hidden in the complexation and decomplexation rules of κ -calculus.

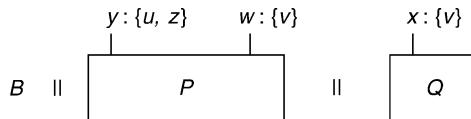
Incremental model building. The identification of entities with boxes in both calculi makes the extension of models easier because the parts of the specifications to be changed are clearly identified.

Implementation. The association between boxes and biological entities simplifies the implementation of the stochastic simulation algorithms. However, the efficiency is still not acceptable and no parallel implementation is available. Furthermore, the generality of split and join functions of beta-binders poses further difficulties to efficiently implement the calculus.

Qualitative versus quantitative descriptions. The most enhanced calculus of the second generation are beta-binders that decouple qualitative descriptions from quantitative ones. In fact, rates are no more merged with the qualitative descriptions, but are externally specified through the compatibility function between the types of the interfaces (see also Fig. 5). However, complex kinetics function cannot yet be incorporated within the models.

Fig. 7

The box enclosing Q acts as a sub-compartment of the other box in the figure. In fact, the binder x can only interact with the binder w assuming that the type v does not occur in any other place in the system. Therefore, the box enclosing Q can only interact internally or with its father compartment through x .



Space. Some preliminary attempt to code compartments and hence implicitly handle localization of molecules is emerging, but very limited computational support is available. For an example, see [Fig. 7](#).

Connection with standards. The situation is still the same as the previous generation of calculi.

Only limited activities have been performed (Ciocchetta et al. 2008b).

Predictive models. Something in this direction is obtained by beta-binders as a result of the higher level of nondeterminism within this calculus. However, the predictive capabilities of models specified through calculi are not yet satisfactory. In other words, one is only slightly ahead of ODEs.

Although the second generation of calculi improve the usage and scalability of the models, some relevant issues remain unresolved and no interoperability with mathematical modeling, (for example, ODEs) is available.

2.3 From Theoretical Calculi to Practical Modeling Languages

The very last step in the design of modeling formalisms is moving from theoretical calculi to real programming languages designed for biology. The most relevant platforms emerged from the two calculi discussed in the previous section. The BlenX (Dematté et al. 2008b) language inspired by beta-binders is considered here as an example, because it also incorporates the complex/decomplex features of κ -calculus. BlenX collects all features now available within modeling environments based on process calculi (see [Fig. 8](#)). Alternative efforts of BioPEPA (Ciocchetta and Hillston 2008) and BIOCHAM (see <http://contraintes.inria.fr/BIOCHAM/>) are also mentioned.

The driving principle to move from theoretical descriptions to practical modeling languages is sure to enrich the syntax of the calculi to facilitate the modeling process. This step must be performed by separating concerns as much as possible—that is, by keeping qualitative and quantitative descriptions distinct as well as using different syntactic categories and semantic actions to represent different fundamental biological principles. Finally, the run-time of the language should take into account, as much as possible, the basic dynamic principles of biological systems, like complexation and decomplexation. It would be useful to specify the minimum information that is needed to infer the dynamic interactions between entities.

The dynamic behavior of BlenX models is specified through three classes of actions:

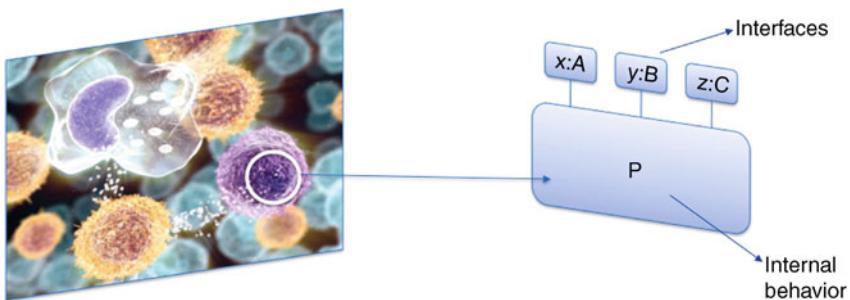
Monomolecular: actions that describe the evolution of single entities (see [Fig. 9](#))

Bimolecular: actions that involve two or more entities (see [Fig. 5](#))

Events: global rewriting rules of the environment that manipulate a set of entities as well as their structure (see [Fig. 10](#))

Fig. 8

BlenX is still based on boxes that are associated with biological entities. Boxes have typed and uniquely named interfaces and their internal behavior is specified similarly to beta-binders, although a richer set of primitives to manipulate interfaces is available and the restriction operator is no longer present, thus simplifying the theoretical development. The dynamics of systems are specified through three classes of actions: monomolecular, bimolecular, and events. Monomolecular actions affect a single box and can be either internal communications or internal primitives like *die* to destroy a box or interface manipulations. Bimolecular actions affect two or more boxes and can be either communications between boxes or complex/decomplex operations. Events are actions that are enabled when global conditions on the structure and cardinality of boxes are satisfied.

**Fig. 9**

BlenX provides many primitives to manipulate interfaces. Represented in the figure is the *expose* that creates a new interface on a box by ensuring that the name *x* is new (note the renaming to avoid clashes), the complementary actions of hiding and unhiding interfaces. There is also a primitive to let time pass without performing anything significant (*delay*) even if the time causes the choice between the alternatives to be performed. There are other two primitives not represented in the figure: *change(x:W)*, which alters the type of an interface named *x* to *W*, and *die*, which kills a box.

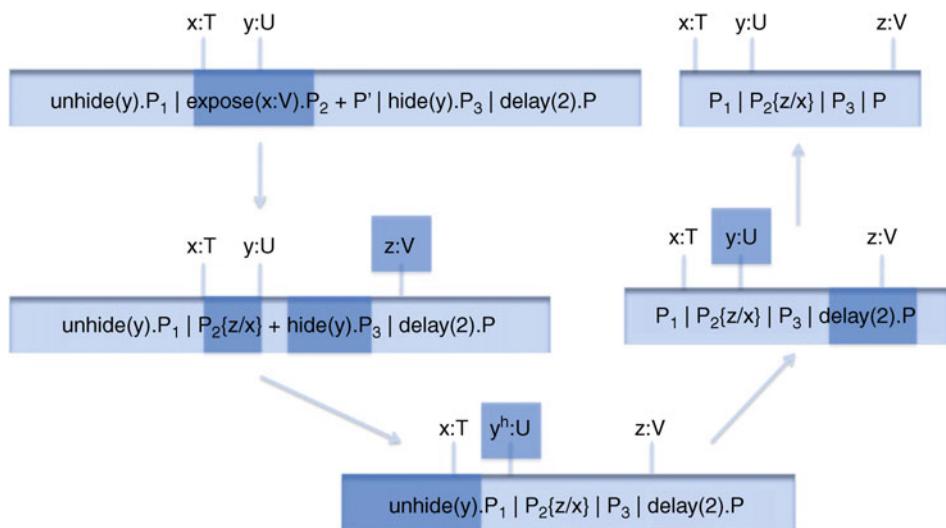
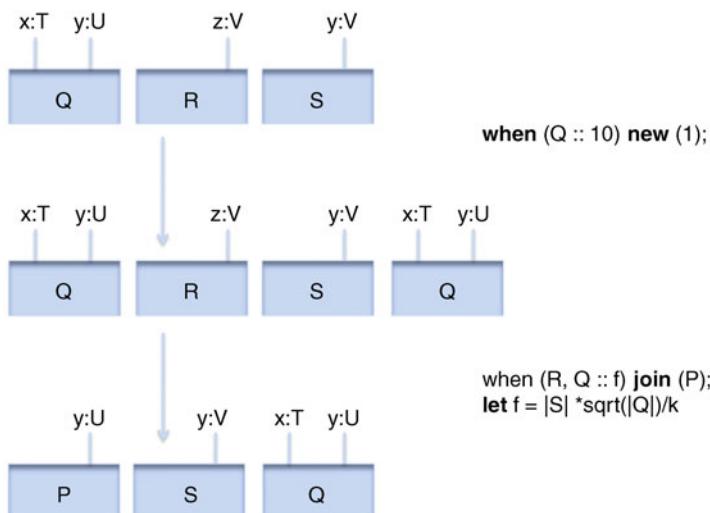


Fig. 10

Events in BlenX resemble conditional term rewriting systems. A condition on the structure of the system, on time, on the step of the simulation, or on the quantities of the species is tested after the keyword **when**. If the condition is satisfied, the event specified after the condition is enabled and participates in the selection of the next action through the stochastic simulation algorithm. The execution of a **new** or a **delete** event introduces or removes as many copies of boxes equal to the one in the condition as specified by their arguments (see first event in the figure for a **new** example). The execution of a **join** or a **split** event removes the boxes specified in the condition from the system and introduces the ones specified in the event action. For instance, the second event in the figure removes one **Q** and one **R** and introduces one box **P**. Conditions are also equipped with kinetic information. Finally, an **update** event action is provided to assign values to continuous or discrete variables associated with states of the system.



Events are very powerful modeling tools that also enable the execution of perturbative experiments over the models as is usually done in a laboratory. It can be specified that whenever a global condition, possibly time-dependent, is satisfied, a gene can be knocked-out, a drug injected in the system to study its interactions, or some components removed to see how they affect the overall behavior of the systems or, analogously, how robust a system is. Note that any event is associated with a rate and enters the race with all the other actions for selection by the simulation algorithm. Therefore, even if specified differently from the other interaction mechanisms, an event is a normal action of the system that does require modifications of the stochastic engine.

Note that the second event in [Fig. 10](#) is specified by a general mathematical function that does not only depend on the concentrations of the boxes affected by the event, but also on the concentrations of other components. Actually BlenX permits the specification of general kinetics by relying on mathematical functions. This is a useful generalization because it easily implements an interoperability with ODEs by translating them into BlenX models automatically. In this case, boxes coincide with variables and the dynamics emerge by simply copying the ODEs into functions associated with a very limited class of special events (see Mura et al. (2009)). As a consequence, the BlenX modeling approach can also handle smoothly hybrid

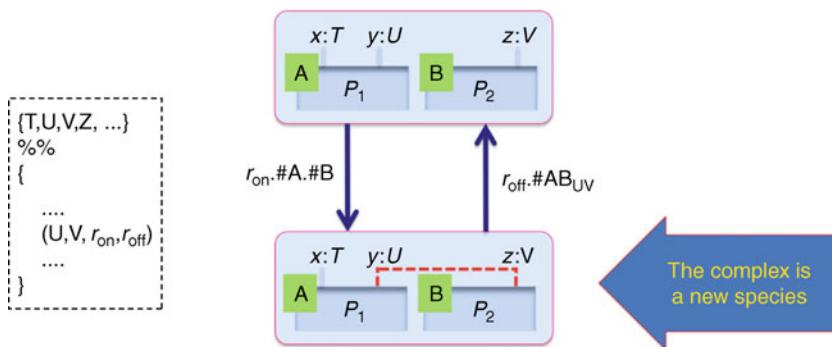
situations in which continuous and discrete aspects are fused. Finally, note that if one uses general distribution functions, the exactness of Gillespie's stochastic simulation algorithm is lost. However, this should not be a concern because most of Gillespie's hypotheses are not satisfied, being, for example, the interior of a cell crowded, not homogeneous and not well-stirred. Actually, the design of a good simulation algorithm for biological systems is an open issue.

Interoperability with mathematical modeling is also ensured in BlenX by the possibility of associating states of the system with discrete and continuous mathematical variables. These variables can be manipulated by events whose conditions are on the values of the state variables and the event action *update* provides the values for the new assignments. This feature can be used, for instance, to easily change the rate of a reaction dynamically depending on some conditions over time, space, or the structure of the system. The same can apply to the reaction volume that can change dynamically (e.g., cell growth).

BlenX includes the management of complexes in a way that specific actions of formation or destruction do not need to be specified (see Fig. 11). Because the formation of complexes and their destruction is a very primitive mechanism of interaction in biological systems, BlenX provides it in its runtime support so that the modeler can think of the basic biological components and their sensitivity without the need to program the interactions in detail. The positive result is an easier modeling approach on one side and a better predictive power of the models on the other. Here, predictive power refers to the set of observable behaviors by executing the systems. In fact, many actions are determined by the runtime of the language according to the compatibility of types and structure of boxes and to binders status. These conditions are dynamically checked and produce potential transitions at runtime that have

Fig. 11

In BlenX, it is not necessary to specify the interactions that form or destroy complexes. It is still the affinity between interfaces that determines whether a couple of boxes can complex or not. The parameters for the dynamics of complex formation are specified again in the separate file expressing compatibility. For instance, the tuple (U, V, r_{on}, r_{off}) in the figure means that two boxes having interfaces with types U and V can complex with a rate r_{on} and, once complexed, can detach each other with a rate r_{off} . The stochastic simulation algorithms also consider, at any step, the whole set of possible complexation and decomplexation among the enabled actions. As far as species are concerned, one should note that once two boxes are complexed, they form a new species in which a link between the interfaces complexed is created and visualized (the dashed red line in the figure).



not been coded by the programmer in the flow of control. The degree of freedom added to the runtime of the language increases the potential of observing behaviors, not considered at modeling time.

A BlenX model is made up of a static description of the system obtained by listing the entities of the initial configuration with their amount and specificity for interaction. Then, the execution of the model and the support of the language are in charge of determining, at any step, the possible actions to be performed. This approach enables a drag-and-drop library-based modeling: the modeler simply selects the components that must be considered in the system and then defines the sensitivity of interaction between them. After this step, only a little is needed to equip the boxes with their relevant internal behavior, if particular actions need to be implemented. The BlenX modeling process described above allows modelers to build scalable, modular, and compositional models. Finally, all the combinatorial effects are ruled out at modeling level, because the very same box represents all the possible states through which the corresponding biological entity can pass.

Almost all the concerns expressed in the previous sections for the first two generations of calculi for biology have been greatly reduced by the BlenX language. Some efforts are still ongoing to simplify the management of space and to improve the efficiency of the execution of systems both by parallelizing algorithms (Dematté and Mazza 2008) and by improving the efficiency of them, once contextual information is available (Kuwahara and Mura 2008) (Fig. 12).

Fig. 12

The figure summarizes the concerns and the characteristics of the calculi of the three generations described in the paper.

	1st generation π - calculus	2nd generation Beta-binders	3rd generation BlenX
Entities	Represented by processes, it is difficult to trace their identity	Encapsulated in boxes and hence easy to trace and count	Encapsulated in boxes and hence easy to trace and count
Complexes	Implicitly represented by scope of names and tricky manipulated through the restriction operator	Explicitly represented through the join/split of boxes or through dedicated links	Improved manipulation of complexes with conditions on their structure and state
Low-level programming	Thought for computing systems, all biological events must be encoded in the available primitives	Thought for biology, primitives closer to biological processes makes model simpler	Enriched and more natural set of primitives. A real programming language
Interaction	Exact complementarity of channel names	Compatibility of the types of the interfaces of boxes	Compatibility of the types of the interfaces of boxes
Incremental model building	Difficult: interconnection structure and affinity of entities hard-coded in the model.	Medium: affinity separated from model behavior and boxes allow easy identification of the connections between models	High: Affinity and complex formation separated from model behavior and handled through types. Boxes and different syntactic components identify connections between models
Implementation	Gillespie algorithm, counting through structural congruence	Gillespie algorithm, counting through structural congruence	Many simulation algorithms, efficient counting due to the design of the language
Qualitative versus quantitative	Exponential distributions with rates associated to channel names	Exponential distributions with rates associated both to typed interfaces and channel names	General distributions, rates as general functions, hybrid models with continuous variables
Space	Extremely limited support	Space can be encoded through tricks	Preliminary notions of space
Standards	Very limited import/export capabilities	Some import/export capabilities	Connection with SBML and ODE
Predictive power	Every interaction must be programmed	Many interactions are inferred by the execution engine	Many interactions are inferred by the execution engine

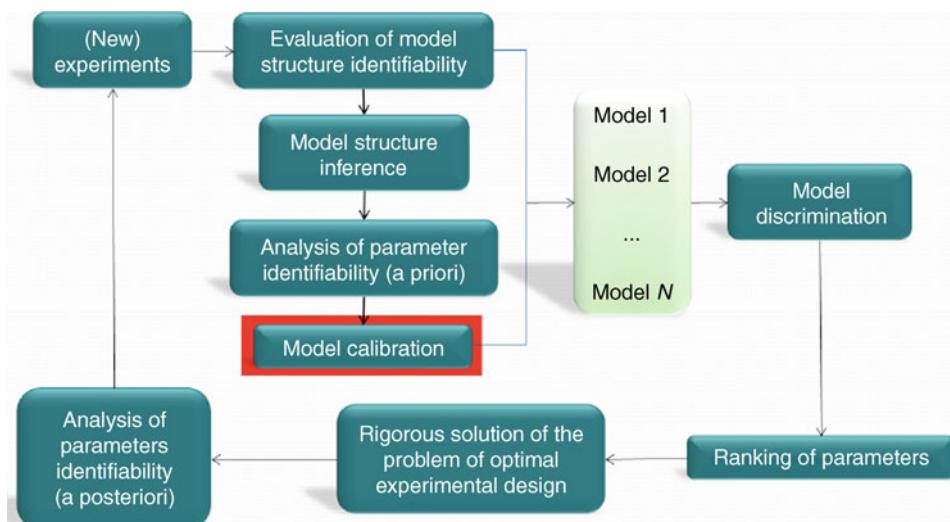
3 Knowledge Inference

Many efforts have been made toward the refinement of the modeling languages, and a reasonable level of easiness of use and generality to address most of the relevant biological problems has been reached. However, in order to execute these models, kinetic information is needed, which is often difficult to find. In particular, rates of interactions can be inferred from literature or database mining and sometimes directly by manipulating the outcome of wet laboratory experiments (e.g., time-course concentrations from microarrays or outcomes of NMR analyses). Therefore, it is essential, for a conceptual framework that aims at supporting biological research, to cover inference activities, both from a theoretical and computational perspective. Kinetics information inference is referred to here, although network structure inference is an important field as well. The tight integration of inference capabilities with a modeling framework is absolutely mandatory to make the framework practical. [Figure 13](#) shows how to integrate the inference of knowledge within a modeling cycle.

Here, one concentrates on model calibration (parameter estimation) since this is the most critical task in building an executable stochastic model following the algorithmic systems biology approach. Interaction networks can be estimated as well, but much more information on this aspect is available through various interactome projects and databases.

Fig. 13

A biological problem is usually modeled by describing the qualitative behavior constrained by quantitative information used to drive the dynamics. Starting from wet laboratory experiments, one needs to infer the parameters governing the kinetics of the reactions as well as the topology of the interaction network of the entities of the system at hand. By varying the parameters and the interaction network, one obtains a class of different models for the same phenomenon and one needs to discriminate between them according to some phenotypical fitness or statistic measures that allows one to rank the relevance of parameters. Once the main parameters are identified, it is possible to provide hints on the design of new experiments to gain the minimum knowledge one needs from the lab, in order to maximize the discrimination between the models. (Figure prepared by Alida Palmisano.)



Recent literature reports many examples of methodologies of parameter estimation both in deterministic and stochastic models, including Polisetti et al. (2006), Moles et al. (2003), Tian et al. (2007), and Chou et al. (2006), for example. Tools for parameter fitting, through regression or maximum likelihood methods, can be found as an integral part of simulation tools (e.g., Copasi (Hoops et al. 2006)), but there also exists stand-alone software, exclusively designed for that purpose, such as Splendid (Bashi et al. 2005) and PET (<http://mpf.biol.vt.edu/pet/contact.php>). Finally, Boys et al. (2008) and Golightly and Wilkinson (2008) developed Bayesian model-based inference techniques. Bayesian schemes offer some advantages over the maximum likelihood methods, such as when the volume of data is limited or the analytic form of the kinetic model makes the maximization of the likelihood difficult.

Most of the current tools for parameter estimation lack robustness to the noise as well as the absence of any estimates of experimental error in their outcome. Experimental uncertainties on parameters propagate from the measurements of the concentrations of the species. Inferring the parameters with an estimate of their uncertainty is essential if one wants to use the tool in the context of optimal experimental design. Moreover, most of the current tools based on optimization techniques suffer from the problem of univocally finding the solution by global optimization, and ask the user to provide a priori the optimization algorithm with the region of parameter space in which to perform the search for the global max/minimum.

A new approach has been recently proposed that enhances the state of the art in parameter estimation by relying on a completely new mathematical method (Lecca et al. 2009). The method is based on a probabilistic, generative model of the variations in reactant concentrations. The time-series concentrations of n reactant species are collected in n state vectors x_1, \dots, x_n . The method discretizes the law of mass action and provides a tool to predict the values of the variables x_i at time t , conditioned by their values at the previous time point. The variations of the concentration of the species, at different time points, are conditionally independent by the Markov nature of the discrete model of the law of mass action. Assuming the observation noise to be Gaussian with variance σ^2 , the probability of observing a variation for the concentration of species, between time t_{k-1} and t_k , is a Gaussian with variance depending on σ and the expectation value of the law of mass action under the noise distribution. The discretization of the law of mass action provides a model for the variations of species concentrations, rather than a model for the time-trajectory of species concentrations. This makes the evaluation of the expectation value of the law of mass action function more easily and analytically tractable. The rate coefficients and the level of noise are then obtained by maximizing the likelihood function defined by the observed variations.

This method produces the rate coefficients, the level of noise, and an error range on the estimates of rate constants. Its probabilistic formulation is key to a principled handling of the noise inherent in biological data, and it allows for a number of further extensions, such as a fully Bayesian treatment of the parameter inference and automated model selection strategies based on the comparison between marginal likelihoods of different models.

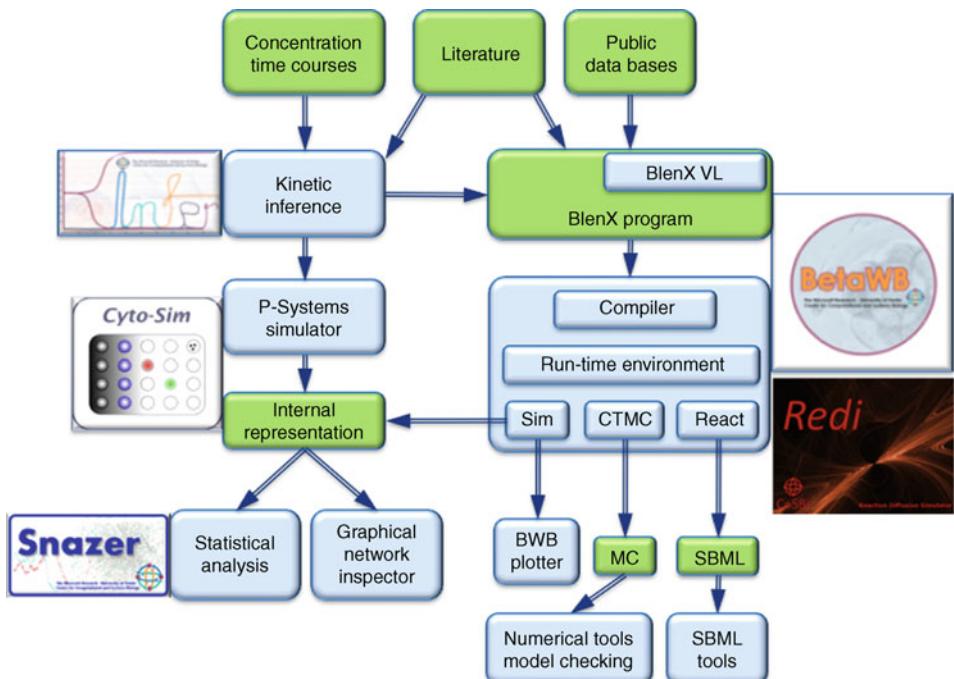
The implementation of this method in the tool KInfer (see next section) is used as an interface tool, connecting the outcomes of the wet lab activity for the concentration measurements and the software for the simulation of chemical kinetics. Finally, KInfer can also be used as a validation tool for models. If given a model, it is not possible to infer kinetic parameters that allow the model to reproduce the observed behavior in wet labs, it means that something is missing from the model or something is wrong.

4 CoSBI Lab: An Algorithmic Systems Biology Platform

The conceptual framework of algorithmic systems biology can become effective only if a suitable computational support that assists all phases in the life cycle of the modeling, simulation, and analysis of biological systems is implemented. At the time of writing, the algorithmic systems biology platform that covers most of the issues is CoSBI Lab, illustrated in Fig. 14, where the components of the platform and their connections are highlighted. CoSBI Lab is a software platform that implements a new conceptual modeling, analysis, and simulation approach to quantitative, dynamic systems primarily inspired by algorithmic systems biology. CoSBI Lab is intended to become a complete artificial laboratory in which it is possible to replicate, *in-silico*, all the activities that are usually performed in real wet labs. CoSBI Lab is centered around the new programming language, BlenX, that is the core of the formal modeling facilities. BlenX is directly derived from process calculi and hence has concurrency as a primitive feature that helps simplify models to avoid the combinatorial explosion of variables. CoSBI Lab

Fig. 14

CoSBI Lab, the algorithmic systems biology platform in the figure, provides support for quantitative model definition through both textual and graphical formalisms within the BetaWB set of tools. It allows for inference of kinetic rates via KInfer. The platform provides connection to simulators via the BetaWB, Cyto-sim, and Redi. Other kind of analyses on the models are allowed through exporting into Markov chains and SBML descriptions. Finally, Snazer provides support to visualize networks of reactions and annotate them. Snazer also implements a rich set of statistical analyses.



supports Gillespie-based simulations as well as spatial diffusion of entities. Export of models in the SBML format is allowed. CoSBi Lab implements inference procedures for quantitative parameters needed to feed the BlenX models and to drive the simulation engines. The inference starts directly from various sources of data, including the outcome of wet lab experiments that can be mapped (directly or through preprocessing transformations) into time courses of concentrations. CoSBi Lab offers a set of features to analyze the outcome of the experiments (simulations) that include the main statistical techniques as well as the visualization of networks of reactions and plots of concentrations.

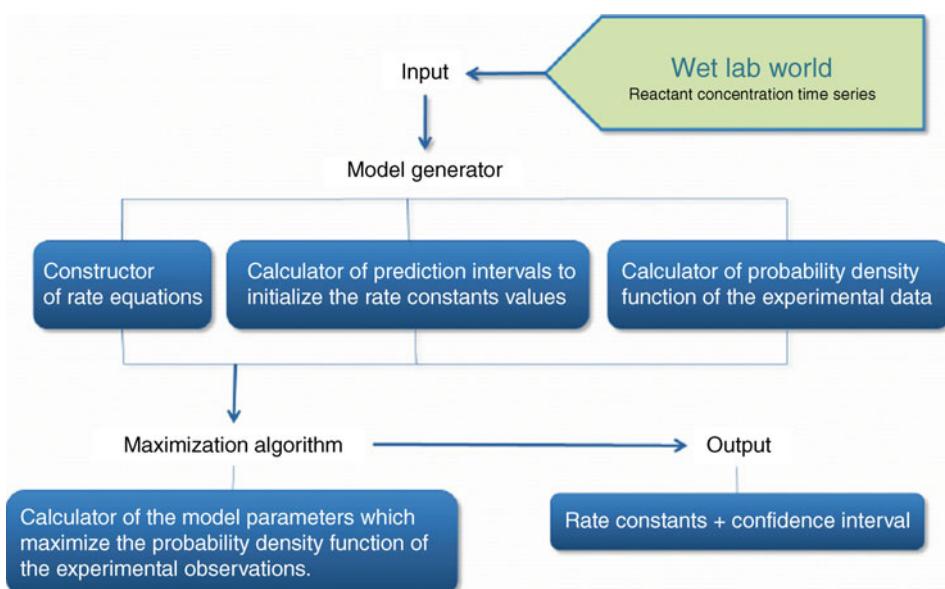
The main algorithmic tools populating the platform are now described.

BetaWB (http://www.cosbi.eu/Rpty_Soft_BetaWB.php) (Dematté et al. 2008b) is a collection of tools based on the programming language BlenX, explicitly designed to represent biological entities and their interactions. The BetaWB includes the BetaWB simulator, a stochastic simulator based on an efficient variant of the Gillespie stochastic simulation algorithm (SSA); the BetaWB designer, a graphical editor for developing models; and the BetaWB plotter, a tool to analyze the results of a stochastic simulation run. Generation of Markov chains and export into SBML is allowed.

KInfer (http://www.cosbi.eu/Rpty_Soft_KInfer.php) is a tool for estimating rate constants of biochemical network models from concentration data measured, with error, at discrete time points. A schema of KInfer is reported in **Fig. 15**. KInfer is inspired by the maximum

Fig. 15

KInfer starts from a time series concentration of entities and a list of reactions and automatically builds the mathematical model to be solved in order to estimate the rate constants describing the dynamics. Parameters are selected according to a fitting of the overall model toward some experimental observed data. KInfer also provides the user with a measure of the error performed in the calculation of the parameters, that is, the strength of the noise in the system that can be introduced by the experimental production of the time series. (Figure prepared by Alida Palmisano.)



likelihood estimation and assumes a discretized version of the law of mass action as a rate equation. The discretization of the rate equation makes the evaluation of its expectation value analytically tractable. The probabilistic formulation of the KInfer algorithm guarantees the noise-robustness and the possibility of extending it to a Bayesian treatment of the parameters. The principal features of the tools are:

1. Automatic generation of a generalized mass action model
2. Automatic estimation of the initial guesses and bounds for the parameter values
3. Estimation of the experimental errors on the inferred parameters
4. Estimation of the strength of noise in the input data

Redi (http://www.cosbi.eu/Rpty_Soft_RedI.php) (Lecca et al. 2008) is a reaction-diffusion simulator, built to test new diffusion models and algorithms. Redi simulates biochemical systems at the mesoscopic scale of interaction, employing a space discretized variant of the Gillespie SSA. Diffusion coefficients are not fixed, but are computed dynamically in a state-dependent way, inspired by the Maxwell–Stefan model of transport phenomena.

Cyto-Sim (http://www.cosbi.eu/Rpty_Soft_CytoSim.php) (Sedwards and Mazza 2007) is a stochastic simulator of biochemical processes in hierarchical compartments based on P-systems. The compartments may be isolated or may communicate via peripheral and integral membrane proteins. The native syntax is designed to be a compact and intuitive way of describing chemical systems. Arbitrary kinetic rate functions are permitted, allowing seamless import and export to SBML. Export to Matlab is also facilitated.

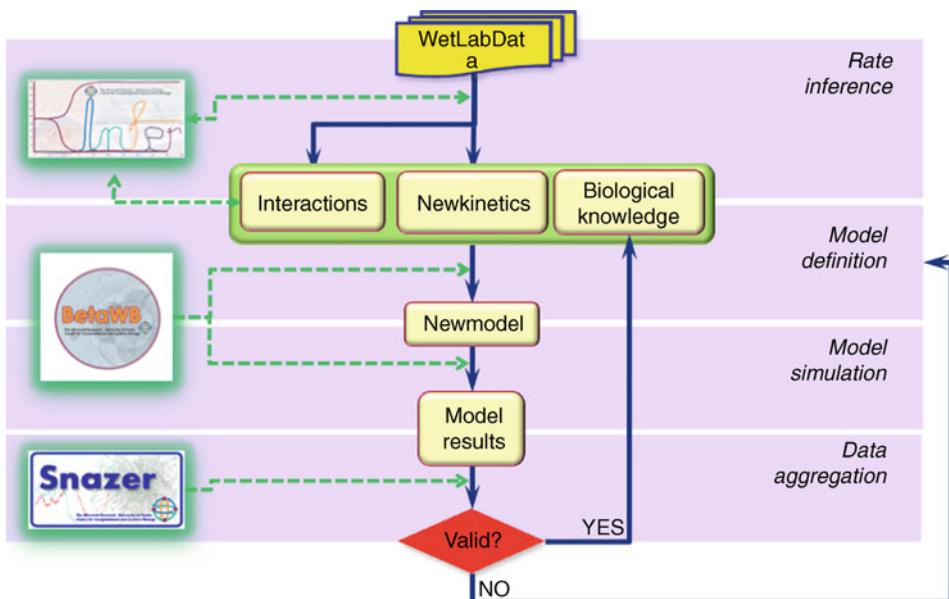
Snazer (http://www.cosbi.eu/Rpty_Soft_Snazer.php) (Mazza et al. 2010) is a modular tool designed to aid the processes of visualizing and manipulating reactive models, as well as to share and interpret time-course data produced by stochastic simulators or by any other means. Snazer upgrades the viewer of Beta Workbench and interfaces its output format. It loads biological networks encoded in SBML as well, and stores graph layouts in standard GraphML files. Moreover, it loads time-course data exported in CSV, and also compresses and prepares them for remote sharing and statistical processing. Finally, to enhance interoperability, it has been equipped with a public, XML-based schema to allow other tools both to encapsulate and structure their time-course data and to package their interaction networks, if any. A number of features are provided:

1. Import from Beta Workbench output, CSV, SBML, and (proprietary) XML file formats
2. Export to CSV, GraphML, and (proprietary) XML file formats
3. Interacting view of the chemical reactive networks
4. Color auto-tuning for color-blind users
5. Node importance highlighting (degree distribution)
6. Custom and MIRIAM-compliant annotation (see <http://www.ebi.ac.uk/miriam/>) support
7. Statistical analysis of simulated multi-traces
8. Statistical outcome export in support vector graphics (SVG)

The possible workflow depicted in  Fig. 16 highlights a relevant aspect of in-silico laboratories based on algorithmic descriptions of the dynamics of systems produced through computing languages. Most biological research is mainly driven by the availability of reactants, by technical capabilities of performing specific perturbation to the modeling organisms, and by the time needed to get the outcome of an experiment. In-silico replica of biological systems can instead exploit the power of algorithmic approaches to speed up experiments and to make thousands of them at a very low cost. The comparison of these in-silico experiments can

Fig. 16

A typical workflow supported by the algorithmic systems biology platform starts from the collection of the available biological knowledge and the inference from it of the relevant parameters through KInfer. The outcome of this step is used to feed the quantitative part of the models within BetaWB, that together with the algorithmic description of the behavior allows the simulation of the dynamics of the system. The result of the simulation can be inspected by Snazer and can suggest modifications to the model or new experiments to acquire new knowledge by iterating the cycle. Eventually this process can lead to the discovery of new biological insights. (Figure prepared by Ivan Mura.)



then drive the optimal design of new wet experiments. To make the approach described above practical, it is necessary to interpret the outcome of the execution of a model exactly as the outcome of a wet experiment and to make the same kind of inference on it that is usually done on the output of the bench. This is why the statistical analyses implemented in Snazer, directly on the outcome of multi-run simulations, provide an important added value to the algorithmic systems biology platform described so far.

This section concludes by noting that the described platform is the only integrated environment based on process calculi available. Other implementations are limited to simulators and various graphical tools to visualize the outcome of simulations. The first implementation of a biochemical stochastic π -calculus simulator was BIOSPI (Priami et al. 2001), followed by SPiM (Phillips and Cardelli 2007). A stochastic κ -calculus simulator is described elsewhere (Danos et al. 2007). The Kappa Factory (<http://www.lix.polytechnique.fr/krivine/kappaFactory.html>) is a graphical platform for the design, analysis, and simulation of biomolecular systems. Different kinds of analysis methods are supported, for example, static dependencies on rules and analysis of traces. Recently, an effort to integrate stochastic simulation with other analyses techniques, like model checking, has been done for BioPEPA (Ciocchetta et al. 2008a).

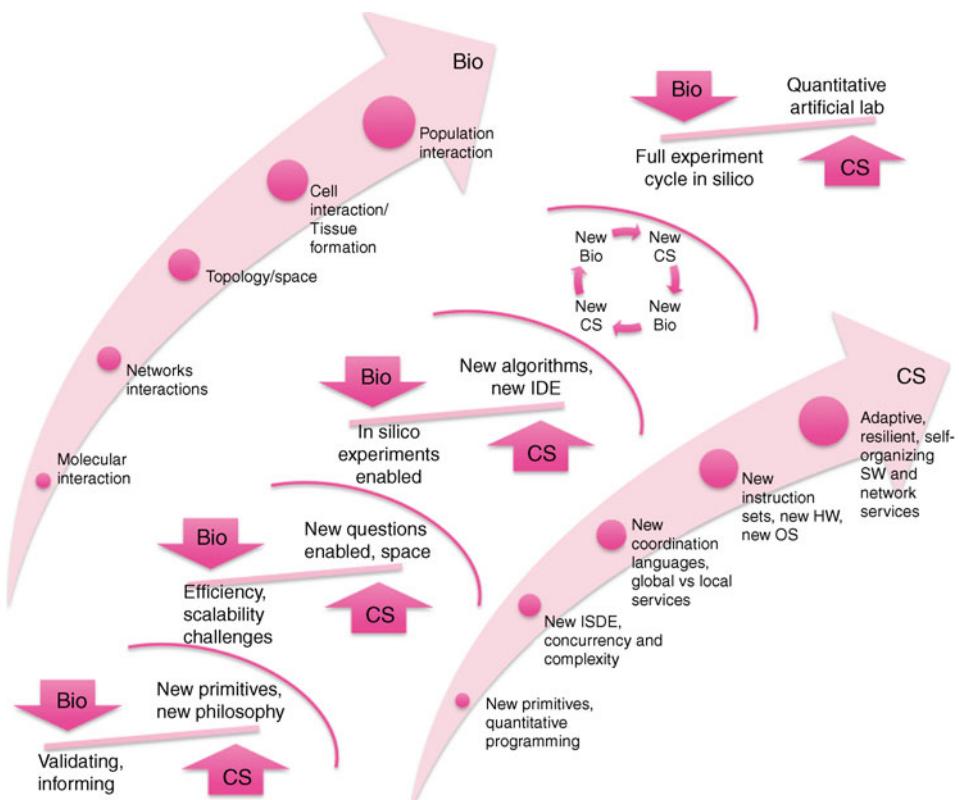
5 Conclusions

The integration of computer science and systems biology into algorithmic systems biology is a win-win strategy that impacts both disciplines from a scientific and technological perspective (Fig. 17).

The scientific impact of accomplishing the vision and the feedback from the biological understanding will be the definition of new quantitative theoretical frameworks that address the challenge of overcoming the increasing concurrency and complexity that is observed in the asynchronous, heterogeneous, and decentralized (natural/artificial) systems in a verifiable, modular, incremental, and composable way. Furthermore, the definition of novel quantitative

Fig. 17

Algorithmic systems biology can propel both biological and computer science research. The conceptual and computational framework developed to enhance biological capabilities of understanding basic principles allows biologists to address new and challenging questions (left arrow) whose answers provide the requirements for developing new and better computer science frameworks (right arrow). At the same time, the environment developed to address the biological challenges can turn computer science into a more quantitative science and can enable simulation-based research in all fields where the dynamics are driven by the interaction of entities (economic markets, social networks, etc.).



coordination and orchestration mechanisms of the loosely coupled components of the kind of complex systems being studied will produce new conceptual frameworks to cope with the growing paradigm of distributing the logics of applications between local software and global services. The definition of new logical and conceptual schemas to store data related to the dynamics of systems and the new query languages needed to retrieve and examine this new kind of records will create completely new perspectives to build the increasingly valuable data centers that provide added value to global services.

Another major scientific impact will be the definition of a new philosophical foundation of systems biology, which is algorithmic in nature and allows scientists to raise new questions that are out of reach for the current conceptual and computational support. An example of this would be the cross-talks (interaction) between pathways (cascades of interactions of biological entities). Actually the current interpretation of pathways (that do not exist per se in nature) is a reductionist approach (imposed by technological limitations) with respect to the system-level understanding of network behavior (a collection of interwoven pathways working simultaneously and interacting).

The technological impact of accomplishing the full merging of computer science and systems biology will be the design and implementation of artificial biology laboratories that allow many more experiments than the ones currently feasible in wet labs, at lower costs in terms of human and financial resources as well as time. These labs will allow biologists to design, execute, and analyze experiments to generate new hypotheses in a shorter time than needed in wet labs. The new findings in hierarchical networks of biological elements (molecules, cells, organs, and organisms) inform new technology actions coordinated by research requirements to develop novel, massive high-throughput tools. These actions should mainly address advances in experimental design, documentation, and interpretation as well as a deeper integration between dry and wet research. Furthermore, the artificial laboratories will be a main technology to move from single gene diseases to multifactorial diseases that are more than 90% of those affecting society. A deeper look at the causes of multifactorial diseases can positively impact their diagnosis and management. Health is not the only practical application of algorithmic systems biology; the comprehension of the basic mechanisms of life coupled with engineered design environments for synthetic biology will lead one toward the use of ad hoc bacteria to repair environmental damages as well as to produce energy.

Another major technological impact will be allowing computer scientists to properly address and master the challenges originated by the new HW design, which mainly exploit parallelism via many/multi-core architectures rather than the speed of single processors, through new integrated programming environments suitable to address concurrency and complexity.

Unraveling the very basic mechanisms adopted by living organisms to compute and manipulate information leads to the heart of computer science: computability. Life underwent billions of years of tests and was optimized during this time, and thus one can learn new computational paradigms to enhance the field. The same arguments apply to hardware architectures as well. Starting from the very basics, one can further build on top of them to enhance resource management and hence operating systems, primitives to instruct highly parallel systems and consequently (concurrent) programming languages, and also software development environments that ensure higher quality and better properties than current software applications. As a consequence, algorithmic systems biology can contribute to the future of computer science.

Algorithmic systems biology can also contribute to the future of natural and life sciences through connecting models and experiments by means of new conceptual and computational

tools integrated in a user-friendly environment equipped with templates of major biological components for drag-and-drop modeling of (artificial) organisms or populations and used by many life scientists to predict the behavior of multi-level, multi-scale biological systems in a modular, compositional, scalable, and executable manner.

Acknowledgments

The author thanks the CoSBi team for many inspiring discussions.

References

- Bashi K, Forrest A, Ramanathan M (2005) SPLINDID: a semi-parametric, model-based method for obtaining transcription rates and gene regulation parameters from genomic and proteomic expression profiles. *Bioinformatics* 21(20):3873–3879
- Bergstra J, Ponse A, Smolka S (eds.) (2001) *Handbook of process algebras*. Elsevier, Amsterdam, The Netherlands
- Bernardo M, Degano P, Zavattaro G (eds) (2008) *Formal methods for computational systems biology*. Lecture notes in computer science, vol 5016. Springer, New York
- Boys RJ, Wilkinson DJ, Kirkwood TB (2008) Bayesian inference for a discretely observed stochastic kinetic model. In: *Statistics and computing*. Springer, The Netherlands
- Cardelli L (2005) Brane calculi-interactions of biological membranes. In: CMSB 2004, Paris, France, May 2004. Lecture notes in computer science, vol 3082. Springer, Berlin, pp 257–280
- Cardelli L, Zavattaro G (2008) On the computational power of biochemistry. In: AB08, Austria, July–August 2008
- Cassman M, Arkin A, Doyle F, Katagiri F, Lauffenburg D, Stokes C (2005) International research and development in systems biology. WTEC Panel on Systems Biology final report
- Chou IC, Martens H, Voit EO (2006) Parameter estimation in biochemical systems models with alternating regression. *Theor Biol Med Model* 3(25)
- Ciocchetta F, Hillston J (2008) Calculi for biological systems. In: SFM-08, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, pp 265–312
- Ciocchetta F, Gilmore S, Guerriero M, Hillston J (2008a) Integrated simulation and model-checking for the analysis of biochemical systems. In: Proceedings of PASM 2008. Electr Notes Theor Comput Sci 238:17–38
- Ciocchetta F, Priami C, Quaglia P (2008b) An automatic translation of SBML into beta-binders. *IEEE/ACM Trans Comput Biol Bioinform* 5(1):80–90
- Danos V, Laneve C (2004) Formal molecular biology. *TCS* 325(1):69–110
- Danos V, Feret J, Fontana W, Krivine J (2007) Scalable simulation of cellular signaling networks. In: *Proceedings of APLAS'07*, Singapore, November–December 2007
- de Alfaro L, Henzinger TA, Jhala R (2001) Compositional methods for probabilistic systems. In: CONCUR 2001, Aalborg, Denmark, August 2001. Lecture notes in computer science, vol 2154
- Dematté L, Mazza T (2008) On parallel stochastic simulation of diffusive systems. In: *Proceedings of CMSB2008*, Rostock, Germany, October 2008. Lecture notes in computer science, vol 5307. Springer, pp 191–210
- Dematté L, Priami C, Romanel A (2008a) The beta workbench: a tool to study the dynamics of biological systems. *Brief Bioinform* 9(5):437–449
- Dematté L, Priami C, Romanel A (2008b) The BlenX language: a tutorial. In: *Formal methods for computational systems biology*, SFM 2008, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, pp 313–365
- Denning P (2007) Computing is a natural science. *Commun ACM* 50(7):13–18
- Eccher C, Priami C (2006) Design and implementation of a tool for translating sbml into the biochemical stochastic π -calculus. *Bioinformatics* 22(24): 3075–3081
- Golightly A, Wilkinson DJ (2008) Bayesian inference for nonlinear multivariate diffusion models observed with error. *Comput Stat Data anal* 52(3):1674–1693
- Guerriero M, Prandi D, Priami C, Quaglia P (2008) Process calculi abstractions for biology. In: Condon A, Harel D, Kok J, Salomaa A, Winfree E (eds) *Algorithmic bioprocesses*. Springer, Berlin

- Gutowitz H (1990) Introduction (to cellular automata). *Physica D* 45:vii
- Harel D (2007) Statecharts in the making: a personal account. In: Proceedings of 3rd ACM SIGPLAN history of programming languages conference (HOPL III), San Diego, CA, June 2007
- Hood L, Galas D (2003) The digital code of DNA. *Nature* 421:444–448
- Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. (2006) COPASI – a COmplex PAthway SImulator. *Bioinformatics* 22:3067–3074
- Kitano H (2002) Systems biology: a brief overview. *Science* 295:1662–1664
- Kuwahara H, Mura I (2008) An efficient and exact stochastic simulation method to analyze rare events in biochemical systems. *J Chem Phys* 129 (16):165101
- Lecca P, Dematté L, Priami C (2008) Modeling and simulating reaction-diffusion systems with state-dependent diffusion coefficients. In: Proceedings of world academy of science, engineering and technology – international conference on bioinformatics and biomedicine, vol 35
- Lecca P, Palmisano A, Priami C, Sanguinetti G (2009) A new probabilistic generative model of parameter inference in biochemical networks. In: Proceedings of the 2009 ACM Symposium on Applied Computing, Honolulu, HI, March 2009
- Mazza T, Iaccarino G, Priami C (2010) Snazer: the simulations and networks analyzer. *BMC Syst Biol* 4:1
- Moles GC, Mendes P, Banga JR (2003) Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res* 13:2467–2474
- Mura I, Palmisano A, Priami C (2009) From ODEs to language-based, executable models of biological systems. In: PSB09: Proceedings of the Pacific symposium on biocomputing, Kohala Coast, HI, January 2009
- Phillips A, Cardelli L (2007) Efficient, correct simulation of biological processes in stochastic pi-calculus. In: Proceedings of CMSB07, Edinburgh, Scotland, September 2007. Lecture notes in computer science, vol 4695. Springer, pp 184–199
- Polisetty PK, Voit EO, Gatzke EP (2006) Identification of metabolic system parameters using global optimization methods. *Theor Biol Med Model* 3(4):1–15
- Priami C (1995) Stochastic π -calculus. *Comput J* 38 (6):578–589
- Priami C (2009) Algorithmic systems biology. *Communications of the ACM* 52(5):80–88
- Priami C, Quaglia P (2004) Modeling the dynamics of biosystems. *Brief Bioinform* 5:259–269
- Priami C, Quaglia P (2005) Beta binders for biological interactions. In: Proceedings of CMSB04, Paris, France, May 2004. Lecture notes in bioinformatics, vol 3082. Springer, pp 21–34
- Priami C, Regev A, Shapiro E, Silvermann W (2001) Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inform Process Lett* 80:25–31
- Regev A, Shapiro E (2003) Cells as computation. *Nature* 419:343
- Regev A, Panina EM, Silverman W, Cardelli L, Shapiro E (2004) BioAmbients: an abstraction for biological compartments. *TCS* 325(1):141–167
- Roos D (2001) Bioinformatics – trying to swim in a sea of data. *Science* 291:1260–1261
- Rosen R (1998) Essays on life itself. Columbia University Press, New York
- Searls D (2002) The language of genes. *Nature* 420:211–217
- Sedwards S, Mazza T (2007) Cyto-sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. *Bioinformatics* 23 (20):2800–2802
- Spengler S (2000) Bioinformatics in the information age. *Science* 287:1221–1223
- The Kappa Factory. <http://www.lix.polytechnique.fr/~krivine/kappaFactory.html>
- Tian T, Xu S, Burrage K (2007) Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics* 23(1):84–91
- Welch P, Barnes F (2005) Communicating mobile processes: introducing occam-pi. In: Proceedings of CSP25, London, July 2004. Lecture notes in computer science, vol 3525. Springer, pp 175–210
- Wing J (2006) Computational thinking. *Commun ACM* 49(3):33–35

55 Process Calculi, Systems Biology and Artificial Chemistry

Pierpaolo Degano¹ · Andrea Bracciali²

¹Dipartimento di Informatica, Università di Pisa, Italy
degano@di.unipi.it

²Department of Computing Science and Mathematics,
University of Stirling, UK
braccia@cs.stir.ac.uk

1	<i>Introduction</i>	1864
2	<i>Process Calculi for Computational Models of Bio-systems</i>	1867
3	<i>Artificial Chemistry</i>	1881
4	<i>Formal Analysis of Models</i>	1888
5	<i>Case Studies</i>	1892

Abstract

Knowledge about life processes develops through the interplay of theoretical speculation and experimental investigation. Both speculation and experiments present several difficulties that call for the development of faithful and accessible abstract models of the phenomena investigated. Several theories and techniques born in computer science have been proposed for the development of models that rely on solid formal bases and allow virtual experiments to be carried out computationally *in silico*.

This chapter surveys the basics of *process calculi* and their applications to the modeling of biological phenomena at a system level. Process calculi were born within the theory of concurrency for describing and proving properties of distributed interacting systems. Their application to biological phenomena relies on an interpretation of systems as made of interacting components exhibiting a computational kind of behavior, “cells as computation.”

The first seminal proposals and the subsequent enhancements for best adapting computer science theories to the domain of biology (with particular reference to chemical, biochemical, and cellular phenomena) are surveyed.

1 Introduction

Knowledge about life processes and ultimately about oneself as a human being develops through the interplay of theoretical speculation and experimental investigation. Experiments have high costs, as materials and techniques are not easily available, may be lengthy, have speed that is determined by the speed of natural processes, pose ethical issues (for example, because they may be invasive), and have available investigation techniques that are not always sufficient to study the phenomena of interest.

A long-standing way of addressing such difficult investigations is to revert to models of the studied problem. As in any other scientific field, a model focuses on the essence of what is studied, abstracting away the unnecessary details so as to reduce complexity. Furthermore, a rigorous, formal description of natural phenomena permits reasoning on them at an abstract level. This is crucial for assessing the adequacy of the proposed model, which can then be used for formally checking aspects and properties of the modeled objects. Particularly relevant for an assessed model is its prescriptive capability, which in the realm of life sciences means that new facts can be discovered and hypotheses tested without resorting to actual experiments. This reduces the costs of the experimental work and drives those tests that remain essential for demonstrating these findings.

The use of formal methods in biology has been advocated for some time. A seminal exploration appeared in Schrödinger’s “*What is Life?*” (Schrödinger 1946), which contains *in nuce* a quest for a description of living matter in terms of the mathematics and the techniques used in physics. Biophysics is a fascinating and effective discipline, originally influenced by that book.

Beyond physics and mathematics, computer science also contributes to the development of biology, which becomes more and more a multi-disciplinary research field. On the one hand, computer science has traditionally provided mankind with the information technology infrastructure needed to store the vast amount of data we acquire about biological phenomena and to organize it, as well as with several methods and effective tools to search, retrieve, and reason on these data. On the other hand, more recently, computer science has been

fostering the development of *executable* formal models of biology, for which several theories born in computer science are being exploited.

One of the most successful and widely known examples of this is the modeling of DNA as a string and the application of several string-based algorithms in the quest for DNA sequencing. A large body of research has been done on algorithms for mining recurrent patterns for a better understanding of the *structure* of the DNA double chain.

Broadly speaking, these results in representing and searching genomic structures follow from a reductionist approach, which focuses on the single aspects of the components of a biological system, namely the sequential structure of the DNA. However, these efforts seem inadequate for explaining the overall *functioning* of a complex system and the properties emerging from its behavior when considered in its dynamical evolution. Clearly, the complete knowledge of the DNA structure is not enough to describe its metabolic and regulatory functionalities, which rather depend on the interactions among genes, the proteins they code, and also the environment in which these interactions happen.

In the words of Kitano:

- ▶ To understand biology at the system level, we must examine the structure and dynamics of cellular and organismal function, rather than the characteristics of isolated parts of a cell or organism. . . many breakthroughs in experimental devices, advanced software, and analytical methods are required. (Kitano 2002)

Such a systemic approach has been embraced by the recently developing *systems biology* research field.

This chapter focuses on the role that theories, models, and analysis techniques born in computer science may have within systems biology. More specifically, theories from the field of *concurrency*—that is, the study and analysis of the interactions happening within computer systems—are considered. This approach relies on interpreting biological systems as interactive computational systems. In both contexts, a potentially vast number of components (for example, biochemical molecules vs. software applications) cooperate together in a similar manner (for example, by chemical reactions vs. communications over a computer network) toward a common goal (for example, sustaining a cell vs. managing a flight control system). This approach was clearly suggested by Regev and Shapiro in their seminal paper (Regev and Shapiro 2002):

- ▶ We believe that computer science can provide the much-needed abstraction for biomolecular systems. . .the ‘molecule-as-computation’ abstraction, in which a system of interacting molecular entities is described and modelled by a system of interacting computational entities. Abstract computer languages, . . . enable simulation of the behaviour of biomolecular systems, . . . supporting qualitative and quantitative reasoning on these systems’ properties.

The mentioned abstract computer languages considered here are *process calculi*. In the form of an algebraic representation, they describe the dynamics of an interacting system. This representation is *formal*, that is, mathematically precise, and the modeler can choose the desired *level of abstraction*; it is largely *compositional*, that is, separate subparts of a system can easily be assembled together; it is *executable*, that is, it supports automated simulation; and it offers a repertoire of tools and techniques for a mechanical *verification* of properties.

Process calculi offer their users the possibility of representing both *qualitative* information, (for example, checking whether a metabolite is or is not produced by a cell) and *quantitative* information (for example, the stochastic time-course of specific metabolites along a pathway).

In recent years, various process calculi designed for modeling network distributed applications and their analysis tools have been used to describe the dynamics of biological systems (for example, biochemical pathways and even whole cells). However, it became clear that, as they are, these calculi lack descriptive power under several pragmatic aspects. Then, new process calculi were defined in order to properly take into account biological features, such as the specific structures or the complex interactions that occur within and on membranes and compartmentalized systems (Regev et al. 2004; Cardelli 2004; Priami and Quaglia 2004). These extensions also call for related enhancements in the formal theory that supports them, and so started a new research subfield across concurrency theory and life sciences. In particular, one has to rethink the way the dynamics of systems is specified (that is, their semantics), so as to comprise relevant biological aspects at different levels of detail (for example, the stochastic nature of various phenomena), starting from Hillston (1996) and Priami et al. (2004). This task is not easy at all, and also raises the formal question of the expressivity of the resulting calculi from a theoretical point of view, typically whether they express all the computable functions.

Beyond the mentioned features and despite its early stage, the process calculi for biology often are adequately descriptive and computationally efficient compared with other traditionally adopted modeling techniques, such as those based on the differential calculus. A crucial distinction is that a model in this approach turns out to be, roughly, a computer program that can be executed to *generate* a formal representation of the biological behavior, rather than *being* itself a description of the behavior, as offered, for example, by a system of differential equations. So, it might be easier to build the model, as well as to study the model under different conditions (for example, by varying and tuning parameters, looking for emergent properties, etc.) because each run can be seen as a *virtual* experiment.

Process calculi and the algebraic approach lie in a much bigger effort of defining formal (and executable) models of living matter and to reason on them, ultimately aiming at defining new formal languages for biology. References are only provided for some of these approaches, as completeness would be impossible. Among the various rule-based formalisms presented in the literature, one can find the λ -calculus (Fontana and Buss 1994), Biocham (Calzone et al. 2006), the κ -calculus (Danos et al. 2007), Moleculizer (Larry and Roger 2005), the pathway logic (Eker et al. 2002), the calculus of looping sequences (Milazzo 2008), and closer to formal languages, P-systems (Paun et al. 2007). Formalisms introduced for reactive systems have also been used to model biological entities (Damm and Harel 2001; Sadot et al. 2008). Petri nets offer a graphical framework for formally describing biological networks, and have been widely used (Reddy et al. 1993). The surveys collected in Bernardo et al. (2008) can help to complete the vast picture of the numerous contributions to this field, which cannot be properly summarized in the limits of this chapter.

Synopsis. Process calculi are illustrated in [Sect. 2](#), where their main features are recapitulated, along with some simple examples of applications taken from biology. In particular, the syntax and the semantics of a simple calculus are introduced, then there is a discussion on how to extend them with quantitative information and illustrate some features that make the modeling of membranes easier. In [Sect. 3](#), a representation of chemical facts through process calculi is discussed and some expressivity results are surveyed. Classical techniques for analyzing the behavior of processes are in [Sect. 4](#). Finally, [Sect. 5](#) surveys a few examples of biological systems that have been modeled and investigated within the approach based on process calculi.

2 Process Calculi for Computational Models of Bio-systems

Process calculi have been defined within concurrency theory to formally represent the behavior of a system made of interacting components, also called *processes*. The represented behavior is an abstraction of the actual behavior of the components at the desired level of abstraction. This abstraction is also referred to as *observable behavior*, suggesting that the aspects of interest that an external observer can perceive are considered, neglecting internal mechanisms or details deemed irrelevant. Process calculi are based on an algebraic notation whose main ingredients are

- (i) The basic *actions* that components can perform together with a set of operators for defining *processes* by composing basic actions and simpler processes;
- (ii) A *semantics* describing the possible dynamics of a process according to the actions it is enabled to do. Often the semantics is operational and consists of a *labelled transition system (LTS)*. LTS have *states*, roughly processes themselves, and *transitions*, that is, a relation between a state and the next one the process can move to. A transition may have *labels* recording its observable behavior, to keep track of the relevant information about the step. LTS are usually defined compositionally by inductive rules over the structure of the components of a process (SOS rules (Plotkin 2004)). A *trace* is a sequence of transitions, such that the target state of a transition in the sequence is the source of the next one. A trace represents a possible computation of the process in the source of its first transition;
- (iii) *Properties*, *methodologies*, and (mechanical) *tools* to analyze process behavior. Examples of properties can be equivalence relations, such as bisimulations, or reachability properties, such as checking whether a process can reach a given state of interest. Examples of mechanical analysis tools are model checkers, used for verifying whether an LTS can fulfill a logical formula expressing the property of interest, and static analysis tools, which check properties of an LTS by reasoning on an abstraction of it.

Below a simple calculus, called BCCS after basic CCS, will be introduced. It consists of a (restriction- and value passing-free) subset of CCS, an exemplary calculus (Milner 1989). This introduction to process calculi aims at presenting their use in systems biology and is definitely not exhaustive. The interested reader is referred to Milner (1980, 1999) and Bergstra et al. (2001) and citations therein.

Two extensions that have been demonstrated to be relevant for systems biology are then presented. The first one takes into consideration quantitative data. Many biological phenomena need to be understood in terms of the quantities, and their variations, of the elements involved. Often, these are continuous variations derived from deterministic laws. However, at certain scales and levels of abstraction, the phenomena are essentially discrete, as each single element may cause specific behavior to emerge globally. Furthermore, they are better described in a stochastic manner, as the dynamics of the system cannot be fully reduced to deterministic mechanisms (Wilkinson 2006). Process calculi have been equipped with stochastic semantics suitable for studying the performance of computer systems in the presence of events with a probabilistic distribution over time, for example, user requests and queued incoming messages (Hillston 1996; Priami 1995). Stochastic process calculi offer then natural means for formal quantitative modeling of biological phenomena.

The second extension concerns the development of calculi designed to account for a certain form of interaction that peculiarly happens within biological systems. In particular, the focus will be on a few calculi designed to express the notion of membraned components

and the behavior that can be expressed by the membranes themselves. The capability of describing intra- and extracellular processes show the versatility of the approach in modeling phenomena at different scales with the desired level of abstraction.

2.1 Process Calculi for Systems Biology

BCCS is illustrated here. Essentially, the processes may perform basic actions that are inputs and outputs over a communication channel. Channels may be seen as the interface of a process with its environment.

Formally, one can assume a set of channels Ch , upon which one can define the set of actions $Act = \{a, \bar{a} : a \in Ch\}$ (one can feel free to use the same letter to denote a channel and an action along it). Actions a and \bar{a} represent complementary input/output actions over channel a (with $\bar{\bar{a}} = a$). Additionally, $\tau \notin Act$ is a distinguished action called *internal action*.

Basic actions can be put in sequence within a process, and their execution will take place in the prescribed order. More interestingly, processes can be built by composing (sub)processes, running in parallel. Two (sub)processes in parallel may proceed independently of each other, or they can synchronize when ready to perform a complementary pair of input/output actions over a channel they share. Furthermore, (sub)processes may also be composed so as to represent mutually exclusive behavior chosen non-deterministically.

More precisely, BCCS is defined by its syntax and its SOS operational semantics, given in Fig. 1. The syntax is in the first line and defines (through the symbol $::=$) the various forms a process P may assume, separated by $|$. The semantics is specified in a logical style, through inference rules that define the transitions of the calculus, actually its LTS – technically, the minimal transition relation that shall be taken among processes fulfilling the rules in Fig. 1. A transition has the form $P \xrightarrow{\alpha} Q$, where P is its source process, Q its target, and $\alpha \in Act \cup \{\tau\}$ is its label, recording the activity performed by the transition.

The inference rules are now briefly commented upon. Each of them consists of some premises and a conclusion, written above and below the line, respectively.

The empty process 0 does nothing, so there is no need for any rule governing its behavior.

The rule *pref* specifies the temporal ordering of actions to occur. It has no premise, and directly justifies the transition for any process in the form $\alpha.P$, in words, P prefixed by action α (P plays here the role of process variable, to be instantiated to an actual process). The process $\alpha.P$ becomes P by performing α , which is recorded in the label of the transition.

The parallel composition of two processes is represented by the $|$ operator. Rule *par* says that if a process P can perform an action α , then $P|R$ can also perform the same action

Fig. 1

Syntax and operational semantics of BCCS.

$$P, Q ::= 0 | \alpha.P | P | Q | P+Q | N \doteq P$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} (\textit{pref}) \quad \frac{P \xrightarrow{\alpha} Q}{P|R \xrightarrow{\alpha} Q|R} (\textit{par}) \quad \frac{P \xrightarrow{\alpha} Q}{P+R \xrightarrow{\alpha} Q} (\textit{cho})$$

$$\frac{P_1 \xrightarrow{\alpha} Q_1 \quad P_2 \xrightarrow{\bar{\alpha}} Q_2}{P_1|P_2 \xrightarrow{\tau} Q_1|Q_2} (\textit{comm}) \quad \frac{A \doteq P \quad P \xrightarrow{\alpha} Q}{A \xrightarrow{\alpha} Q} (\textit{def})$$

with Q staying idle. More generally, if the (instantiated) premises of the rule (top row) are themselves justified, possibly by applying other rules, the bottom transition is justified as well. A further comment is in order. As is often the case, processes are considered up to some structural properties of the operators, so the same rule governs the behavior of any process whose structure matches (up to the structural properties) the source of the transition in the conclusion. Here, it is assumed that the parallel operator is associative (i.e., $(P|Q)|R = P|(Q|R)$), commutative ($P|Q = Q|P$), and has the inactive process 0 as the neutral element ($P|0 = P$). Then, the rule *par* also applies to $P|R = R|P$, with R moving and P staying idle.

The second rule governing the parallel operator is *comm*. It states that whenever any two subprocesses P_1 and P_2 in parallel are ready to perform complementary actions, they can synchronize: $P_1|P_2$ becomes $Q_1|Q_2$ (note that this rule has two premises, and both need to be justified to deduce the conclusion). The observable behavior of a synchronization is τ . This use of the internal action is a means of modular information hiding, not disclosing to the environment the details of the synchronization that happened among P_1 and P_2 . Technically, it also forces synchronization to be binary, because τ has no complementary action to be used in further synchronizations.

Rule *cho* allows $P + R$ to move as P does, and R is discharged. Also $+$ is associative, commutative, and with 0 as neutral element, and so this rule also prescribes that $R + P$ moves as R and P is discharged. Note that the choice among the two alternatives is purely nondeterministic.

The last syntactic definition permits a process to be denoted by a name, taken from a given set $N = \{A, B, \dots\}$. A process name can be used to identify a specific fragment of behavior: rule *def* specifies that the process A , standing for P , can move to any process Q to which P moves. Actually, using this feature one can define recursive processes accounting for the repetition of the same behavior. In a recursive definition, the name of the process being defined occurs within the definition itself. For example, the process $N = a.N + 0$ can perform an unbound number of a actions and then become the process 0.

It is worth underlining the *compositional* flavor of these rules. For instance, the behavior of $P|Q$ is defined in terms of the behavior of its components P and Q . This is *abstractly represented* in terms of the mere observable behavior of P and Q respectively, without any reference to their internal structure.

Example 1 (LTS) The simple process $a.0|b.0|\bar{a}.\bar{b}.0$ consists of three parallel sub-processes, the last of which can communicate in a precise order with the other two. This behavior is specified by the following trace:

$$a.0|b.0|\bar{a}.\bar{b}.0 \rightarrow_{\tau} b.0|\bar{b}.0 \rightarrow_{\tau} 0.$$

The first transition is justified by recalling that $a.0|b.0|\bar{a}.\bar{b}.0 = a.0|\bar{a}.\bar{b}.0|b.0$; hence rule *comm* can be applied since both its premises hold: $a.0 \rightarrow_a 0$ (by rule *pref*) and $\bar{a}.\bar{b}.0|b.0 \rightarrow_{\bar{a}} \bar{b}.0|b.0$ (by rules *par* and *pref*). Being 0 the neutral element for $|$, it can be elided. The last state 0 has no further possible transition.

Differently, the process $P = a.0|b.0|(\bar{a}.0 + \bar{b}.0)$ has the two traces

$$a.0|b.0|(\bar{a}.0 + \bar{b}.0) \rightarrow_{\tau} b.0 \rightarrow_b 0 \quad \text{and} \quad a.0|b.0|(\bar{a}.0 + \bar{b}.0) \rightarrow_{\tau} a.0 \rightarrow_a 0$$

both making available to the environment an action to synchronize with.

Finally, an example of (mutually) recursive processes can be represented by a quite abstract specification of the acquisition/release of energy by the *ATP/ADP* molecules (one can omit here the involved inorganic *ortho*-phosphate). The process representing *ATP* transforms into *ADP* by performing an internal action τ , which needs no synchronization and is executed autonomously. This action represents the mechanisms for the exchange of energy by actually abstracting them away. Analogously, *ADP* performs the inverse process through another τ action.

$$ATP = \tau.ADP \quad ADP = \tau.ATP$$

A trace of a system made of the simple process *ATP* is

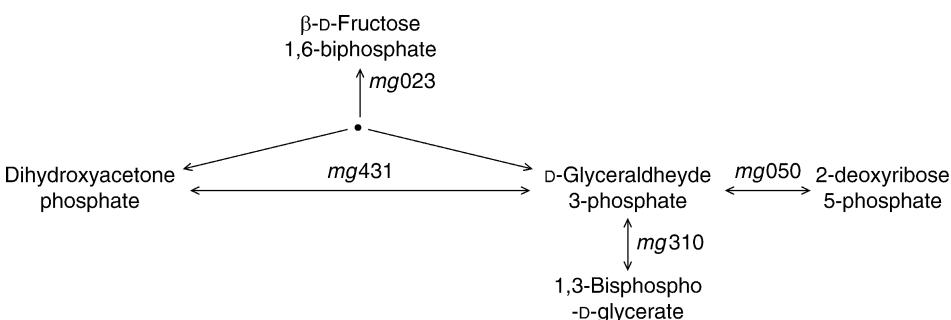
$$ATP \rightarrow_{\tau} ADP \rightarrow_{\tau} ATP \rightarrow_{\tau} \dots$$

The basic calculus introduced comprises several of the ingredients needed to model relevant biological phenomena. Below, another example is considered at the biochemical level and model a fragment of the *Glycolysis* pathway. Recalling the described approach, “cells as computation” metabolites are represented as BCCS processes. The interaction points of metabolites are abstractly represented by channels. Two molecules that may react are then modeled as two processes sharing a channel a , and the reaction between them as a synchronization along a . Indeed, at this level of abstraction, reactions are often understood as binary phenomena after the consideration that the probability that more than two reactants interact at the very same instant is practically negligible. Hence, calculi based on basic binary interaction operators, such as the one introduced, have been often used. Also, reactions are represented that consider the number of reactants present in the system, rather than the more usual concentrations; this point will be addressed later.

Example 2 Consider the small fraction of glycolysis shown in Fig. 2. We use long names for metabolites (IUPAC convention) and codes (as in Fraser et al. (1995), to which we refer the reader) for enzymes: (i) *mg023* for fructose-bisphosphate aldolase, (ii) *mg431* for phosphoglycerate mutase, (iii) *mg301* for glyceraldehyde 3-phosphate dehydrogenase, and (iv) *mg050* for 2-deoxyribose-5-phosphate aldolase. We represent the catalytic effects of enzymes in an abstract way: their contribution is simply recorded with a label attached to the τ actions. In this way we do not explicitly represent enzymes as processes, although this would be possible with a specification at a finer level of detail. Note, however, that the labels of transitions are enough to trace the enzymatic activities along the dynamics of the pathway. The freedom

Fig. 2

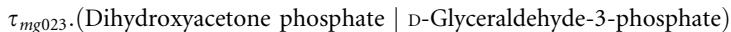
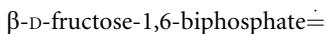
A fragment of the glycolysis pathway.



in selecting what will be the focus and what will be represented more abstractly is a feature of process calculi.

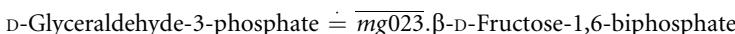
The metabolite β -D-fructose-1,6-biphosphate, reacting with the enzyme *mg023*, splits into Dihydroxyacetone phosphate and D-Glyceraldehyde 3-phosphate. In turn, Dihydroxyacetone phosphate, catalyzed by the enzyme *mg431*, becomes D-Glyceraldehyde 3-phosphate. This can become (i) the metabolite 1,3-Bisphospho-D-glycerate via *mg301*, or (ii) Dihydroxyacetone phosphate via *mg431*, or (iii) 2-deoxyribose 5-phosphate via *mg050*, or finally, (iv) it can reconstitute β -D-fructose-1,6-biphosphate via a reverse reaction with respect to the one that has generated it.

The topmost reaction of Fig. 2 can be described by defining the behavior of the involved metabolite as follows:

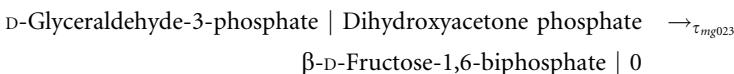


where the prefix consisting of the internal action represents the biochemical mechanisms driven by the *mg023* enzyme that triggers the reaction. In this case, the enzyme action is independent of any metabolite other than β -D-fructose-1,6-biphosphate.

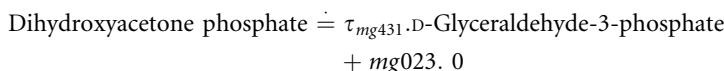
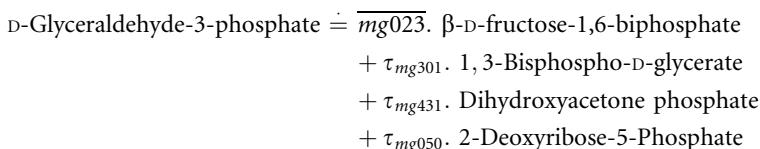
The reverse interaction arises from the specification of the interaction capabilities of D-Glyceraldehyde-3-phosphate and of Dihydroxyacetone phosphate. The first offers a synchronization on the channel (representing the) enzyme *mg023*, and then becomes β -D-Fructose-1,6-bP; similarly, the second is willing to accept a synchronization on *mg023*, and will then disappear, becoming 0.



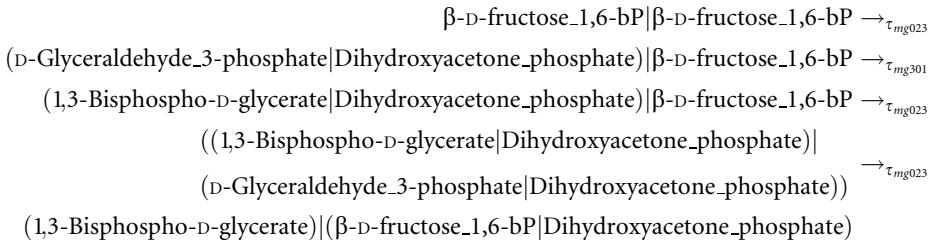
The *adolase* reaction then spawns from the parallel composition of the two metabolites, modeled by the synchronization “catalyzed” by *mg023*:



The one specified above is not the only interaction in which D-Glyceraldehyde-3-phosphate is involved. Its behavior can be incrementally specified by adding the other alternatives to its nondeterministic behavior. Beyond interacting with Dihydroxyacetone-phosphate through the *mg023* channel/enzyme, it can: (i) execute a τ_{mg301} becoming 1,3-Bisphospho-D-glycerate, or (ii) execute a τ_{mg431} becoming Dihydroxyacetone phosphate, or (iii) execute a τ_{mg050} becoming 2-Deoxyribose 5-phosphate. Similarly, the behavior of Dihydroxyacetone phosphate can be completed by its possibility of becoming D-Glyceraldehyde-3-phosphate



The following is a trace of a system made of two copies of β -Fructose 1,6-bisphosphate:



2.2 Adding Quantitative Information

The need to embed quantitative information into process calculi has been put forward for studying the performance of systems. Time was the first quantitative dimension that came into play, although it is easy to recognize that other quantities also have to be considered. Indeed, in a distributed system several of the aspects, typically task completion time, that affect performance have a probabilistic nature, as they depend on the probability distribution of the incoming requests, among other factors.

A probabilistic formal model for the study of system performance, depending on the possible occurrence of events, was first defined in Hillston (1996) and Priami (1995) by embedding stochastic aspects into process calculi. This is realized by associating a random variable to each action. The random variable models the delay of action execution, according to the idea that, for each action, it is necessary to consider a completion time that is probabilistic in nature. In other words, the random variable represents the probability, drawn from a given distribution, that the action will be executed within a delay.

Often, the adopted distribution is a negative exponential distribution $1 - e^{-rt}$, meaning that the probability of completing the considered action before time t is $1 - e^{-rt}$. This distribution guarantees that the probability of a system to go from a state to the next via a transition only depends on the current state and not on the way this state has been reached, namely the system enjoys the Markov, or memory-less property. This property has a good correspondence with the modeled reality, because the current state of a distributed system is generally sufficient to determine the next one. Additionally, the memory-less property fosters good semantic definitions.

More formally, the actions Act of the calculus are annotated with a rate r , the parameter of the probability distribution associated with the random variable relative to the action. Basic actions hence consist of pairs (α, r) . Compositional operators and process construction work as standard. The stochastic machinery clearly reflects on the semantics. The inference rules label the transitions with actions and rates, for example,

$$\frac{}{(\alpha, r). P \xrightarrow{(\alpha, r)} P} (pref)$$

Noticeably, the meaning of choice is no longer purely nondeterministic. Rather, we have a probabilistic choice between weighted alternatives:

$$\frac{P \xrightarrow{(\alpha, r)} Q}{P + R \xrightarrow{(\alpha, r)} Q} (cho)$$

Furthermore, synchronization also needs to be rethought, as it is clearly related to the probability of the completion time of actions. This point has been heavily debated; see for example, Hillston (1994) and Bradley (1999). A natural choice would be to consider the slowest of the distributions of two synchronizing actions as the distribution of the synchronization, but min/max operations on exponential distributions do not yield an exponential distribution. A suitable solution to compute the rate r is using a function proposed by Hillston (2005), which is written as a side-condition of the rule below:

$$\frac{P_1 \xrightarrow{(\alpha, r_1)} Q_1 \quad P_2 \xrightarrow{(\bar{\alpha}, r_2)} Q_2}{P_1 | P_2 \xrightarrow{(\tau, r)} Q_1 | Q_2} \text{ with } r = \frac{r_1}{r_\alpha(P_1)} \frac{r_2}{r_\alpha(P_2)} \min(r_\alpha(P_1), r_\alpha(P_2))$$

Above, it is written $r_\alpha(P_i)$, namely the *apparent rate* of α within P_i , $i = 1, 2$, to denote the sum of all the rates associated with the α 's in P_i ready to be performed. The apparent rate takes into account the many possible different ways in which the subprocess P_i offers to its partner a synchronization, which clearly influences the probability of which to occur.

The other rules are straightforward extensions of the usual ones, for example, of (*par*) and (*def*) in  Fig. 1 for BCCS.

As mentioned above, the use of exponential distributions helps in defining and in exploiting the semantics of a stochastic process calculus. Actually, the operational semantics originates a continuous time Markov chain (CTMC), on which several statistical analyses can be made; for more details, the interested reader should refer to Hillston (2005).

One of the most influential calculi with the stochastic features mentioned above is PEPA (Hillston 1993), which was specifically designed for generating Markov processes that could be solved numerically for performance evaluation. This calculus combines the CCS operations discussed above with others, taken from csp. As above, PEPA has actions with a duration, represented by a random variable with a negative exponential distribution; in the choice $P + Q$, the first process completing its activity is taken and the other is discarded. Different from the above, processes are composed in parallel through a cooperation operator, indexed by a set of cooperation activities L , much in the csp style (Hoare 1985). Roughly, the set L determines those activities on which the cooperands are forced to synchronize, and the observable behavior is the activity itself, rather than a τ . This provides one with the possibility of a multi-way synchronization between components: further processes, performing the same actions, can be engaged in the synchronization. Asynchrony occurs when a process performs an action not in L . From csp, PEPA also inherits the additional operator of hiding, also indexed by a set L , written P/L . When such a process performs an activity belonging to L , it becomes a τ , thus preventing other processes from synchronizing and making the activity internal or private to the process P . Starting from an operational definition, the semantics of PEPA is defined as a CTMC, which can be exploited for extracting performance measures.

Besides being used in the performance evaluation of computer system, Bio-PEPA (Ciocchetta and Hillston 2006) has been recently proposed, which extends PEPA to model various biochemical aspects of living matter, including general kinetic laws.

As a matter of fact, stochastic process calculi have been fruitfully exploited for describing biological phenomena, which generally require quantitative models to be better understood. Consider for instance the biochemical level. System dynamics defines and is affected by the amounts of the reactants produced and consumed. Other measurable quantities, such as temperature and time, play a determinant role in the overall picture. Furthermore, the dynamics of biochemical reactions present a stochastic behavior, just like most natural phenomena.

Although CTMCs exhaustively represent the semantics of interest, they pose computational difficulties, because the interplay of states and transitions over time exponentially increases the size of the model, making the approach computationally intractable. This is particularly important for the models taken into consideration, because they often formalize rather complex biological entities, thus leading to systems with huge dimensions.

A more efficient, computationally successful alternative is using stochastic process calculi for simulation purposes. Instead of considering the whole model of all the possible states, transitions and their probabilities, a single trace, called the *trajectory*, is simulated. This consists of one of the possible time courses of the system whose steps are chosen according to the given probability distribution, obviously obeying the semantics of the calculus.

Pushing further the “cells as computation” analogy, one can see a trajectory as an experiment *in silico*. A simulation starts from the description of the initial state of the system in terms of the number of processes that are initially present (see [Sect. 3](#) for a discussion on the number of components vs. their concentrations). Recall that processes represent at the desired level of abstraction the components of the biological system specified, and that the models are discrete (and abstract from temperature, pressure, etc.). The initial state above represents then the initial conditions of the biological entity under experimentation.

Trajectory simulation can be repeated, yielding possibly different outcomes: different experiments are carried out, and various behavioral or statistical analysis are possible on the collected “virtual” results.

Technically, trajectories are computed by including Gillespie’s stochastic simulation algorithms (SSA) in the semantics of the calculus, which will be outlined in [Sect. 3](#). Roughly, SSA stochastically selects the pair of complementary actions in the whole process, which are ready to fire. Also, it determines the time at which the resulting synchronization will occur, namely its apparent rate. Note that synchronizations correspond to reactions, so the actions that occur along a channel a will have all the same rate, often called the *basal rate*. As will be clear later on, SSA then suitably manipulates basal rates to compute the apparent rates of transitions, taking into account all the synchronizations possible (besides a couple of random numbers).

Several simulation tools have been developed, such as the Stochastic Pi-calculus Machine (Phillips and Cardelli 2007), to cite one. These tools represent a viable computational support for the simulation of phenomena at a sufficiently detailed level of description. See [Sect. 5](#) for details.

Next, the results of an experiment carried out by using a tool that implements (a variant of) the stochastic version of BCCS are illustrated.

Example 3 Consider again the fragment of Glycolysis in [Example 2](#), which is taken from the specification of almost all the metabolic pathways of a virtual cell, called VICE (Chiarugi et al. 2007), which is discussed in more detail in [Sect. 5](#). As said, to obtain a quantitative description of a metabolic pathway, it is necessary to link each transition, and in particular its apparent rate, to a measurable biological parameter. The underlying idea is to associate a channel with a basal rate drawn from the specific (microscopic) rate constant of the described biochemical reaction.

In the original paper (Chiarugi et al. 2004), these basal rates were estimated starting from the constants of Michaelis–Menten kinetics (Fersht 1999; Hammes and Shimmel 1970), and taking care of the control strength of the enzymes involved in a pathway (Fell 1997). These values proved accurate enough, as confirmed also by recent findings (Zhao et al. 2008). The

simulation *in silico* of VICE showed, for example, that it reaches its steady state, which is resistant to nonshocking changes in the external environment, for example, in the feeding regimen. This property mimics the homeostatic capability, typical of real cells, which require it in order to regulate their internal medium. Homeostatic biological systems oppose external environment change to maintain their internal equilibrium and succeed in reestablishing their balance, while nonhomeostatic ones eventually stop functioning.

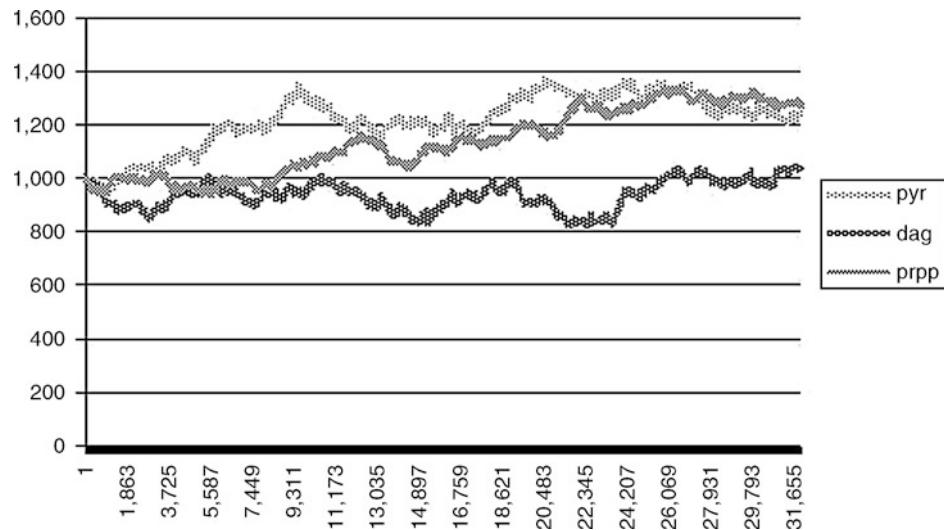
The steady state is manifest when the time course distribution of metabolites reaches a plateau after a certain initial period of time, *in silico* after some transitions. This is shown for VICE in **Fig. 3** for three selected metabolites, whose behavior is rather informative, because they represent critical nodes in the entire metabolic network. Note that **Fig. 3** is affected by white Gaussian noise, because the whole model is stochastic, as it embodies Gillespie's SSA.

The first model of a biological system based on a stochastic process calculus, namely the Stochastic Pi-calculus, is in Priami et al. (2004). It extends the Pi-calculus (Milner 1999), associates rates with channels, and enhances its semantics as sketched above (Hillston 1996; Priami 1995).

The Pi-calculus is a nominal calculus that models systems of concurrent communicating processes whose communication network may change in time. Nominal calculi have the distinguished notion of *name*, roughly an identifier that denotes a shared resource, possibly a channel itself. A process that possesses a name can communicate it to another process across the system, but disclosing a name is constrained by *visibility rules*. Typically, a name, say n , is often private to a group of processes, so the resource it denotes can be referred to only by them; for example, when n denotes a channel, it is the mean for each member of the group to communicate with the others. Instead, a process P not belonging to the group cannot use n , unless the group is enlarged and n communicated to P . Technically, this is done by carefully handling the visibility scope of the name n , so to avoid any possible mismatch or ambiguity or leaking of n to other processes. A name n is declared private to a (group of processes) Q by writing $(vn)Q$; in words,

Fig. 3

Time course distribution of pyruvate (pyr), diacylglycerol (dag), phosphoribosylpyrophosphate (prpp). The concentration of metabolites is plotted vs. the number of transitions.



n is restricted within Q (more precisely, v acts as a binder for n , whose scope is Q , with some *caveat* (Milner et al. 1992), among which α -conversions to avoid name captures).

Clearly, communications are not simple synchronizations, like in BCCS, but also exchange information. As the semantics of the calculus is very rich, instead of giving the details, only a couple of examples are shown.

Notationally, $\bar{n}\langle m \rangle$ is the output action of the name m , through the channel denoted by n ; with $n(x)$ one can represent reading a name that replaces the place-holder x . So, a process $\bar{n}\langle m \rangle.Q \mid n(x).P$ will perform a communication that also has the effect of binding within P (suitable occurrences of) the place-holder x to the received name m . If m denotes a channel, the communication additionally alters the interconnection topology of the system. As a matter of fact, now P , rather P with x instantiated to m , is linked to channel m as well, even though m originally was not a channel in the interface of P .

To exemplify the disclosure of a restricted name, consider the process $(vm)(\bar{\ell}\langle m \rangle.R \mid m(x).P(x))$ in which the name m is restricted, so both processes in parallel can use it (as a short-hand, one can write $P(x)$ to indicate that the place-holder x occurs within P). Consider now the following trace, starting from the above process, in parallel with two other processes $T = \ell(y).\bar{y}\langle n \rangle.0$ and Q , neither of which can know m . The first transition is a communication of m to T , and it enlarges the scope of the restriction to include $\bar{m}\langle n \rangle.0$ (the *extruded* name is highlighted), but not Q . A further communication on the channel denoted by m is now possible, between processes under the restriction, that is, *within* the same group.

$$\begin{aligned} & (vm)(\bar{\ell}\langle m \rangle.R \mid m(x).P(x)) \mid \ell(y).\bar{y}\langle n \rangle.0 \mid Q \xrightarrow{\tau} (vm)(R \mid m(x).P(x) \mid \bar{m}\langle n \rangle.0) \mid Q \xrightarrow{\tau} \\ & (vm)(R \mid P(\bar{n})) \mid Q \end{aligned}$$

The system described in Priami et al. (2004) consists of a gene regulatory network with positive feedback. It involves two genes, RNA transcription and degradation, and positive feedback. Name machinery plays an important part in the positive feedback of protein A onto protein TF , both coded by the genes of the system. Actually, a complex is formed by the disclosure of the private channels (backbones) of A to TF , basically through the same extrusion mechanism mentioned above. Then, A and TF can interact along newly shared, highly efficient channels, boosting the role of TS in the reciprocal promotion, thus originating the positive feedback. The results of computational simulations clearly show the expected uprise of A in the presence of TF , confirming experimental data.

Sharing restricted channels can be conceived as a natural mechanism for implementing a notion of compounded complex, as is often the case in computer system modeling. However, this seems to be neither completely satisfactory in biological systems modeling, nor expressive enough to faithfully and directly describe compounds, compartmentalized cells, and membranes. In the next subsection, a few calculi will be illustrated with operators explicitly for these features.

2.3 Membrane Systems

Distributed computational paradigms often consider software applications that can move around through the interconnection network, crossing environmental boundaries and interacting at different locations – think of the Internet applications that move along the net and enter/exit from sites where they are executed. The concept of *distributed ambient* has been

formalized in the AMBIENT calculus (Cardelli and Gordon 1998). A process P is wrapped within a named environment, called *ambient*, resulting in $n[P]$. Within the ambient n , the process P may contain other (nested) ambients and standard communications actions. Processes can move along the nested ambients by executing the actions *in* n and *out* n , and also dissolve through *open* n . The ambient name referred to by the actions implements a capability discipline: The name of an ambient must be known in order to interact with that ambient. Communication happens only within the same ambient. The premiseless rules in the upper part of Fig. 4 are part of the semantics of the calculus.

Quite naturally, ambients may be interpreted as membraned components, for example, cells. Indeed, a system of membraned components has a nested structure and its components often move, interact, and also transform, for instance by dissolving or inglobing membranes.

A first proposal in this direction is the BIOAMBIENT calculus (Regev et al. 2004), which extends the ideas of AMBIENT in several directions to better suit biological interaction; a stochastic version of it is in Brodo et al. (2007), which exploits the SSA algorithm. In BIOAMBIENT, communications can cross ambients in a controlled way, as happens in intra-membrane interaction. The actions for ambient interaction occur synchronously through different channels/modalities, and merging of ambients/membranes has been introduced. Two rules of the operational semantics are in Fig. 4, which may help in understanding its peculiarities. Just as in nominal calculi, communication actions have “directions,” for example, $\bar{a}(m)$ represents sending the name m through channel a , while $a(x)$ is the complementary action of receiving, but they are controlled to mimic membrane permeability. See, for example, the parent-to-child communication rule ($p2c$), where the output with modality $p2c$ matches the input with modality $c2p$. Two ambients can also merge, giving rise to a new ambient, by modeling the analogous activity on membranes. This is specified by the rule (mrg) in Fig. 4, where the name c associated with the actions $merge+ c$ and $merge- c$ identifies the two membranes that will fuse together. The “direction” $+/-$ (yet another usage of $+ !$) names the resulting membrane m , as $merge +$ occurs within m .

The pragmatics of BIOAMBIENT have been illustrated through several proof of concept examples, such as transport layers, protein complexation, and reversible enzyme activity, (Regev et al. 2004). The last one will be briefly surveyed below.

Example 4 Suppose one has the following parallel composition of membraned components corresponding to the enzyme and the substrate molecule:

$$\text{enzyme}[E] | \dots | \text{enzyme}[E] | \text{molecule}[S] | \dots | \text{molecule}[S]$$

The enzyme E either accepts within its ambient the substrate membrane or an already

Fig. 4

An extract of AMBIENT and BIOAMBIENT semantics.

$$\begin{array}{c}
 \frac{m[in\ n.Q|R]\ | \ n[P] \rightarrow_{\tau} n[m[Q|R]|P]}{n[m[out\ n.Q|R]\ | \ P] \rightarrow_{\tau} m[Q|R]\ | \ n[P]} \quad (in) \qquad (out) \\
 \frac{n[P]\ | \ open\ n.Q \rightarrow_{\tau} P\ | \ Q}{m[merge+\ c.P|Q]\ | \ n[merge-\ c.R|S] \rightarrow_{\tau} m[P|Q|R|S]} \quad (open) \qquad (mrg) \\
 \frac{n[p2c\ \bar{a}(b).P\ | \ m[c2p\ a(x).Q(x)]] \rightarrow_{\tau} n[P]\ | \ m[Q(b)]}{n[P]\ | \ m[Q|R|S]} \quad (p2c)
 \end{array}$$

transformed product molecule. Accordingly, it can then either expel the product, or the original substrate and revert to the original state:

$$\begin{aligned} E &::= \text{accept e-s-bind}.ES + \text{accept e-p-bind}.ES \\ ES &::= \text{expel unbind}.E + \text{expel react}.E \end{aligned}$$

The substrate S can initially only enter the enzyme environment (becoming X), and then, synchronizing with the enzyme, it can either revert to the substrate or exit as the product P . In turn, the product can again be re-processed by the enzyme.

$$\begin{aligned} S &::= \text{enter e-s-bind}.X \\ X &::= \text{exit unbind}.S + \text{exit react}.P \\ P &::= \text{enter e-p-bind}.X \end{aligned}$$

Given a single enzyme and substrate molecule, a possible trajectory (abstracting from the stochastic machinery) follows, in which the substrate transforms into the product:

$$\begin{aligned} &\text{enzyme}[E] \mid \text{molecule}[S] \rightarrow \\ &\text{enzyme}[\text{accept e-s-bind}.ES + \text{accept e-p-bind}.ES] \mid \text{molecule}[\text{enter e-s-bind}.X] \rightarrow \\ &\text{enzyme}[(\text{expel unbind}.E + \text{expel react}.E) \mid \text{molecule}[\text{exit unbind }.S + \text{exit react}.P]] \rightarrow \\ &\text{enzyme}[E] \mid \text{molecule}[P] \end{aligned}$$

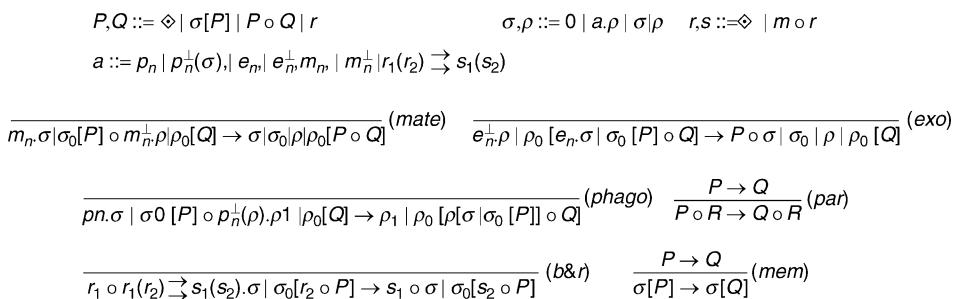
A further step in the direction of modeling membranous components and the interaction mediated by membranes are the BRANE calculi (Cardelli 2004; Miculan and Bacci 2006). These are a modular set of process calculi expressing different details of membrane activity, and associated logics for describing system properties. An important novelty is that membranes themselves exhibit a behavior.

For brevity, a reduced instance of BRANE, namely sBRANE, is considered below. Its (simplified) syntax is in the topmost part of Fig. 5, which also displays the rules of its qualitative semantics (neither quantities of reactants nor stochastic rates are considered).

A few comments are in order. In sBRANE, membrane complexes (P, Q, \dots) and membrane behavior (σ, ρ, \dots) are kept apart. The complex $\sigma[P]$ consists of an (active) external membrane layer σ and of a complex P contained inside the membrane. The symbol \diamond denotes the null membrane complex.

Fig. 5

Syntax and operational semantics of sBRANE, a simplified BRANE calculus.



The complex $P \circ Q$ is obtained by putting in parallel P and Q , while $m_1 \circ \dots \circ m_k$ is the complex consisting of a multiset of molecules (r, s, \dots) .

Membrane complexes interact through the membrane layer σ . This can either perform no action if it is 0, the empty one; or perform an action prefixing a layer a . σ ; or behave as two parallel layers $\sigma|\rho$ (other structure can also be added). Here, one can consider the actions $phago_n$ (in brief p_n) and exo_n (in brief e_n), together with the coactions $p_n^\perp(\sigma)$ and e_n^\perp ; the name n therein constrains the occurrence of these actions in communications, like in BIOAMBIENTS. Moreover, action m_n and coaction m_n^\perp model *cell merging* (although this can be expressed by (*phago*) and (*exo*), only (Cardelli 2004)).

A membrane complex $\sigma[P]$ can enter another complex $\rho[Q]$, if σ can execute a phago p_n action and ρ the corresponding coaction $p_n^\perp(\rho)$ with the same n (see rule (*phago*)). As a result, $\sigma[P]$ is enclosed within a membrane with part ρ . A nested membrane complex $\rho[\sigma[P] \circ Q]$ allows the subsystem P to leave the $\rho[\dots]$ membrane complex if σ and ρ are ready to execute e_n and e_n^\perp , respectively, as dictated by rule (*exo*). Membranes and their contents merge according to rule (*mate*). A membrane complex can also proceed autonomously in a system or within a membrane (rules (*par*) and (*mem*)). As standard, $|$ and \circ are associative and commutative with 0 and \Diamond as neutral elements.

Finally, rule (*b&r*) illustrates a kind of membrane permeability. Being molecules r_1 and r_2 outside and inside the complex, respectively, and being the membrane sensitive to them (they appear in the membrane specification as arguments of the \Rightarrow operator), then the rule can be applied and its effect is to bring inside the complex the molecule s_2 (also an argument of \Rightarrow), while r_1 and r_2 are consumed.

The next example illustrates the use of the chosen calculus in modeling the behavior of systems with membranous components. One can consider (the first steps of) a reference model of viral infection.

Example 5 Via phagocytosis, the virus enters the cell wrapped by a membrane. The external membrane of the virus then merges with a component of the cell, the *endosome*. Finally, through an exocytosis, the viral *nucleocapsid* and the viral RNA it contains are directly released in the *cytosol* of the cell. These first steps on the viral infection can be modeled as follows:

$$\begin{array}{lll} virus = p_a.e_a [nucap] & nucap = capsid[vRNA] & capsid = p_b \mid bud \mid disasm \\ cell = p_a^\perp(m_a) \mid e_b^\perp [cytosol] & cytosol = endosome \circ CC & endosome = m_a^\perp \mid e_a^\perp [\Diamond] \end{array}$$

The virus membrane is ready to execute a phago p_a and an exo e_a action. Its content *nucap* will take part in later stages, and in turn has a membrane *capsid* that contains the RNA of the virus. The membrane *capsid* is ready to execute a phago action p_b , and two further actions *disasm* and *bud* that are not relevant for the initial steps of the viral infection. The cell membrane is ready for a phago $p_a^\perp(m_a)$ and an exo e_b^\perp action, with the latter modeling the reproduced virus that eventually leaves the cell. What is inside the membrane, *cytosol*, consists of two parts. The first is the endosome, which is a membrane complex that can merge, via an action m_a^\perp , with what has been “eaten” by the cell through a phago action. The endosome can also uncoat its content, via e_a^\perp , as soon as the virus provides a suitable coaction. The second component, *CC*, is not relevant here.

The initial configuration of the system is a complex composed by the virus and the cell. The following trace shows an endocytic pathway, consisting of phagocytosis, merging, and exocytosis. As expected, its last state shows that the genetic material of the virus floats within the cell (recall that *nucap* is defined as *capsid*[*vRNA*]).

$$\begin{aligned}
 p_a \cdot e_a[nucap] &\circ p_a^\perp(m_a) | e_b^\perp[m_a^\perp | e_a^\perp[\Diamond] \circ CC] \rightarrow \\
 e_b^\perp[m_a[e_a[nucap]]] &\circ m_a^\perp | e_a^\perp[\Diamond] \circ CC \rightarrow \\
 e_b^\perp[e_a^\perp[e_a[nucap]]] &\circ CC \rightarrow \\
 e_b^\perp[0[\Diamond] \circ nucap \circ CC]
 \end{aligned}$$

The last membrane and compartment oriented calculus that is surveyed is BETA BINDERS (Priami and Quaglia 2004), which builds over the Pi-calculus. It has a notion of ambient, called *box*, so a process has the form $B[P]$; boxes cannot be nested. The component B is a set of so-called *BETA BINDERS*, and represents the possibly modifiable interface of the box. The process P is a Pi-calculus process with standard communication actions, and with some additional actions that manipulate the interfaces. The description of a biological system is then rendered as a set of boxes in parallel.

More technically, a BETA BINDER $\beta(x : \Gamma)$ associates with the name x a set of channels through which the box can communicate with another box, and represents the interaction sites of the modeled membrane. A binder can be either active $\beta^h(x : \Gamma)$ or hidden $\beta^h(x : \Gamma)$, that is, either ready for or temporarily unavailable for interaction. The interfaces are manipulated through the actions $hide(x)$, $unhide(x)$, and $expose(x, \Gamma)$, the semantics of which is in Fig. 6 (the rule for *unhide* and some details are omitted). The first action hides the name x , rather the binder for x in the interface; the second activates a hidden x ; the third action makes x available for interaction by creating a binder for it in the interface (name captures are avoided).

A process P is essentially a Pi-process, and communication within a box is standard. Interaction between boxes is instead peculiar: it still occurs on a given channel, but it is not necessary to explicitly say on which. Suppose that one box has in its interface the BETA BINDER $\beta(x : \Gamma)$ and the other has the BETA BINDER $\beta(y : \Delta)$. The two boxes communicate along x and y provided that the sets Γ and Δ are compatible, according to a given notion. The simplest requires that Γ and Δ contain at least one common channel, that is, $\Gamma \cap \Delta \neq \emptyset$. This simple definition of compatibility contains the rule (*inter*) of the operational semantics in Fig. 6; besides this, others conditions can be given.

Here two further actions that can be used to implement a variety of membrane interactions of interest for biological modeling, namely splitting and joining boxes, are not discussed, and the interested reader is directed to refer to the original presentation (Priami and Quaglia 2004) for a discussion on their definition and usage.

The interested reader can find a stochastic semantics for BETA BINDERS in Degano et al. (2006).

Examples of virus infection, similar to the one presented for the sBRANE calculus, have also been modeled in Priami and Quaglia (2004). These illustrate a natural and expressive way of using the loose communication mechanism offered by BETA BINDERS to define which virus can interact with which cells.

Fig. 6

Some rules of BETA BINDERS.

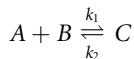
$$\overline{B\beta(x : \Gamma)[hide(x).P|Q] \rightarrow B\beta^h(x : \Gamma)[P|Q]}^{(hide)} \quad \overline{B[\mathbf{expose}(x : \Gamma).P|Q] \rightarrow B\beta(x : \Gamma)[P|Q]}^{(exp)}$$

$$\overline{B\beta(x : \Gamma)[x(w).P|Q] \mid B\beta(y : \Delta)[y(z).R(z)|S] \rightarrow B\beta(x : \Gamma)[P|Q] \mid B\beta(y : \Delta)[R(w)|S]}^{(inter)} \text{ with } \Gamma \cap \Delta \neq \emptyset$$

3 Artificial Chemistry

In this section, the focus is on a process calculi-based representation of chemistry. First, some classical notions, in particular those connected with quantitative and stochastic modeling, are briefly discussed.

The dynamics of a chemical system are usually described by transformation rules annotated with quantities, known as Generalized Mass Action Laws (GMA) (Segel 1987). These rules define and are affected by the amounts, traditionally measured as concentrations, of the reactants produced and consumed. Remarkably, GMA deterministically govern the changes in a chemical system. For instance, the reversible interaction of molecules of species A and B generating molecules of species C is described as



This stands for the fact that quantities of A and B transform themselves into quantities of C with a rate k_1 (note the different use of $+$ with respect to the nondeterministic operator of the algebra of processes). The rate represents here a proportional constant that determines the amount of quantity transformed in the time unit, in proportion to the joint amount of A and B . Vice versa, C decays back into its constituents, proportionally to the amount of C and the rate k_2 . This reaction happens by the spontaneous activity of C . Generally, GMA model systems in terms of variations of continuous *concentrations* of species A , often represented as $[A]$.

Being $[A(t)]$, $[B(t)]$, and $[C(t)]$ the molecular concentrations (numbers of molecules per unit volume) at time t , the following system of ordinary differential equations (ODEs) corresponds to the above kinetic schema:

$$\frac{d[C]}{dt} = k_1[A][B] - k_{-1}[C] \quad \frac{d[A]}{dt} = -k_1[A][B] + k_{-1}[C] \quad \frac{d[B]}{dt} = -k_1[A][B] + k_{-1}[C]$$

Given initial conditions, the solution of the above system provides $[A(t)]$, $[B(t)]$, and $[C(t)]$. In the case of the spontaneous decay $C \xrightarrow{k} 0$, one can have $\frac{d[C]}{dt} = -k[C(t)]$, hence $[C(t)] = e^{-kt}[C_0]$, which is the well-known exponential law of decay, with $[C_0]$ as the initial concentration.

An extensive treatment of ODEs modeling chemical reactions is dealt with in Voit (2000). Although a valuable and widespread description tool, ODEs present methodological difficulties and rely on assumptions that do not suit all possible scenarios of interest.

Among the first ones, ODEs rarely can be solved analytically; numerical solutions are typically computationally expensive (beyond few tens of variables); and models can hardly be constructed in a compositional manner (the introduction of a new variable/molecule/reaction typically requires the redefinition of the whole model).

As for the necessary assumptions, in many phenomena it happens that extremely low concentrations, up to few molecules in a given volume unit, are responsible for relevant behavior emerging at the whole system level. Moreover, such limited amounts perturb the system so that it behaves in a way that is difficult to interpret deterministically, as is done for concentration variations. In the words of Wilkinson (2006), the deterministic approach fails to capture the nature of chemical kinetics, which at low concentrations is discrete and stochastic. (See also Wolkenhauer for a critical analysis on the adequacy of deterministic ODEs descriptions and the relationships between the continuous deterministic approach and the stochastic discrete one.)

A model accounting for the stochastic aspects of system dynamics can be defined analogously to GMA; its result is particularly suitable for modeling small volumes where only few

molecules interact (Gillespie and Petzold 2006; Van Kampen 1992; Gardiner 2001; Wilkinson 2006; Gillespie 1977). The model describes the *stochastic* time-course of *discrete* quantities, that is, the actual number of involved molecules, where rates represent the distribution probability that a selected reaction occurs at a given time-point. This approach leads to the definition of the chemical master equation (CME) (Voit 2000), which describes the interactions of N well-stirred chemical species S_1, \dots, S_N . By naming $x_i(t)$ the number of molecules of S_i at time t , one wants to model the time evolution of the vector $X(t) = (x_1(t), \dots, x_N(t))$ (sometime written X) from a given initial condition $X(t_0) = X_0$. Species interact by M uni- or bimolecular chemical reactions R_1, \dots, R_M . Each reaction R_i is characterized by two quantities: $a_i(X)$ and v_i , namely the *propensity function* and the *state-change vector*, respectively. The system changes because of the reaction R_i with a probability defined by $a_i(X)dt$. This probability is defined on top of a probability rate constant c_i , such that c_idt represents the probability that any two, or one, molecules that are randomly chosen among those reacting according to R_i will react in the time interval dt . By knowing c_i and summing over all the possible distinct combinations of R_i reacting molecules (addition law of probability), we can immediately see that if R_i is a unimolecular reaction $S_j \rightarrow products$, then $a_i(X) = c_i x_j$, while if R_i is a bimolecular reaction $S_j + S_p \rightarrow products$, then $a_i(X) = c_i x_j x_p$. When the reaction R_i occurs, the state vector $X(t)$ becomes $X(t + dt) = X(t) + v_i$.

The CME defines the variations of the probability associated with state changes of the system, which basically corresponds to the probability of reaching the state of interest X from any other possible state minus the probability, being in X , of leaving the state. A possible formalization is the following (see Voit (2000) for further details):

$$\frac{\partial P(X, t | X_0, t_0)}{\partial t} = \sum_{j=1}^M [a_j(X - v_j)P(X - v_j | X_0, t_0) - a_j(X)P(X, t | X_0, t_0)]$$

The CME consists of as many coupled differential equations as all the possible molecular reactions within the system, a number that is almost always prohibitively large. Additionally, even for very simple systems, the numerical solution of the CME is computationally hard, given that the transition probability matrix becomes rapidly untractable because of its dimensions (Smith 2005; Ermentrout 2002).

The stochastic formulation of the biochemical intracellular processes of the CME can be simulated by a Monte Carlo method, as proposed by Daniel Gillespie with his stochastic simulation algorithm (SSA) (Gillespie and Petzold 2006; Gillespie 1977). The algorithm determines a numerical realization of $X(t)$ as a function of t , called the *trajectory*, according to the probability values in the system. This realization has been proven to give an exact simulation of the CME. The SSA relies on homogeneous volume hypotheses and models reactions, but not diffusion.

Roughly, the idea behind Gillespie's algorithm can be recapitulated as follows (see provided references for technical details). Assume $P(t, \mu)dt$ is the probability that R_μ will be the next reaction and it will occur in the interval $(t_0 + t, t_0 + t + dt)$, with t_0 the current time. Then, one can get $P(t, \mu) = P_1(t) \times P_2(\mu, t)$, where $P_1(t)dt$ is the probability that the next reaction will occur in the interval $(t_0 + t, t_0 + t + dt)$, and $P_2(\mu, t)$ is the probability that the next reaction will be of type R_μ . Gillespie showed that $P(t, \mu) = a_\mu(X)^{-a_0(X)t}$, where $a_0(X) = \sum_{k=1}^M a_k(X)$ represents the sum of the probabilities of all possible reactions. Then the time of the next reaction is $t = (1/a_0(X)) \times \ln(1/r_1)$ and the next occurring reaction R_μ is identified by μ as the smallest integer satisfying the relation $\sum_{k=1}^\mu a_k(X) > r_2 a_0(X)$. Here r_1 and r_2 are two random numbers that make the system stochastic. Summing up the application of SSA consists of repeating the steps (2–4) below as many times as needed, after the initialization step (1):

- (1) Initialize the initial time t_0 and initial conditions of the system X_0
- (2) Evaluate all the $a_k(X)$ and $a_0(X)$
- (3) Generate t and μ as indicated above
- (4) Replace the time t_0 with $t_0 + t$ and the number of molecules X with $X + v_\mu$

Since $P(t, \mu)dt$ follows an exponential distribution, system dynamics is memory-less. Importantly, the next reaction and time determined are *exactly* coherent with the probability distributions considered. Once the time and type of the next reaction are determined, time and quantities are updated as prescribed by the reaction and the algorithmic selection of the next reaction and time can be iterated.

As mentioned in [Sect. 2.2](#), the stochastic extensions of process calculi are generally based on SSA and its variations. As said, in those settings SSA determines, beyond the elapsed time, the processes that will interact next. The operational semantics then rules the steps of the system. This amounts to saying that SSA and the operational semantics select a trajectory, exactly mimicking the probability distributions described by the rates of the model.

A thorough investigation about the feasibility of a computational modeling of chemistry, and also of biochemistry, is in the work by Cardelli ([2009](#), [2008](#)). A shallow survey follows, in particular considering some examples showing how to model typical chemical reactions, which are often taken literally from the above papers. These examples also describe how some phenomena are best captured by stochastic approaches. Then, some results about the relations between stochastic and deterministic modeling are mentioned. Also, the use of some new features added to the calculi seen so far, which are needed for modeling certain forms of complexation and decomplexation typical of biochemistry, will be illustrated. Finally, the computational expressiveness of the calculi adopted shall be discussed from a theoretical point of view. This allows one to characterize which class of behavior can be expressed by the programs in that formalism and which cannot, and which properties can be mechanically proved about that class.

Cardelli introduces *interacting stochastic automata* to represent species and their capabilities in a graphical notation. For brevity, some of their features will be discarded here, and their *simple* version is expressed as stochastic BCCS processes, each modeling a (state of) a molecule (in their full form interacting stochastic automata exhibit a dynamical interface of communication capabilities, similarly to nominal calculi, mentioned in [Sect. 2.2](#)).

The reactions of kind $A \xrightarrow{r} B$ are *first order*, those like $A + B \xrightarrow{r} C$ are *second order*. Concentrations follow the mass action law $\frac{d[A]}{dt} = -r[A]$ in first-order reactions, and $\frac{d[A]}{dt} = -r[A][B]$ in second-order reactions, under the assumption of working with a fixed interaction volume and a *well-stirred* solution (diffusion effects are then neglected because they are immaterial; refer also to the assumptions of SSA). Once the volume is fixed, the conversion between numbers of molecules and concentration is mediated by Avogadro's number. Three-way reactions are not considered here because they are unlikely to occur.

A molecule performing a first-order reaction waits for a delay, according to a negative exponential distribution, and then reacts. Technically, in this setting, such an action cannot reduce to a synchronization between two processes, because it is performed by a process in isolation. Usually, one assumes there is an internal action annotated with a suitable rate, representing an *ad hoc* stochastic delay construct (note that rates are here associated also with internal actions, and not only with channels). For instance, the decaying process $C \xrightarrow{r} 0$ can be modeled by the process

$$C \xrightarrow{\cdot} \tau_r.0$$

Second-order reactions, for example, $A + B \xrightarrow{r} C$, follow the above mentioned law of mass action, and can be represented as standard processes, for example, $A \doteq a.0$ and $B \doteq \bar{a}.C$ with the rate r associated to the channel a . Note that the choice of A becoming 0 and B becoming C is arbitrary. A case that requires a bit of care is a second-order reaction within a homogeneous population, for example, $A + A \xrightarrow{r} C$. In this case, each A can interact in two symmetric ways with other A 's, for example,



Since two A 's are transformed by each reaction, one can have $\frac{d[A]}{dt} = -2r[A]^2$.

Other examples of reproducible behavior at different levels of abstraction are oscillators, feedbacks, and even operators of the Boolean algebra (see Cardelli (2009) for details). Also zero-order reactions are considered, which obey a law of mass action independent of concentrations, for example, $\frac{d[A]}{dt} = r$. These represent a sort of constant growth of a population, and have no straightforward chemical correspondence, although they show a dynamic similar to that of saturated enzymes.

Some examples facilitate the intuitive understanding of how far one can go with the process-based representation of chemistry. By analyzing the *collective* behavior of large populations of processes/reactants, it is easy to observe stochastically emerging behaviors due to a small number of specific processes spawned in the large populations. One can also observe emerging behavior that can be assimilated to continuous and deterministic variations, as they concern the fluctuations of large populations.

Consider the so-called *celebrity* kind of behavior:



As soon as any two A 's synchronize on a , one of the two decides to become B , snobbishly “differentiating” itself. Analogously, as soon as any two B 's synchronize, one of them differentiates by becoming A again. The overall behavior of a system of an equal number of A and B , with equals rates for channels a and b , exhibits a dynamical equilibrium, with the expected stochastic fluctuations. The results are in [Fig. 7](#), and were obtained using SPiM, the Stochastic Pi-calculus Machine (Phillips and Cardelli 2007).

Similarly, *groupies* try to maintain their state as far as they are able to synchronize with others in the same state:



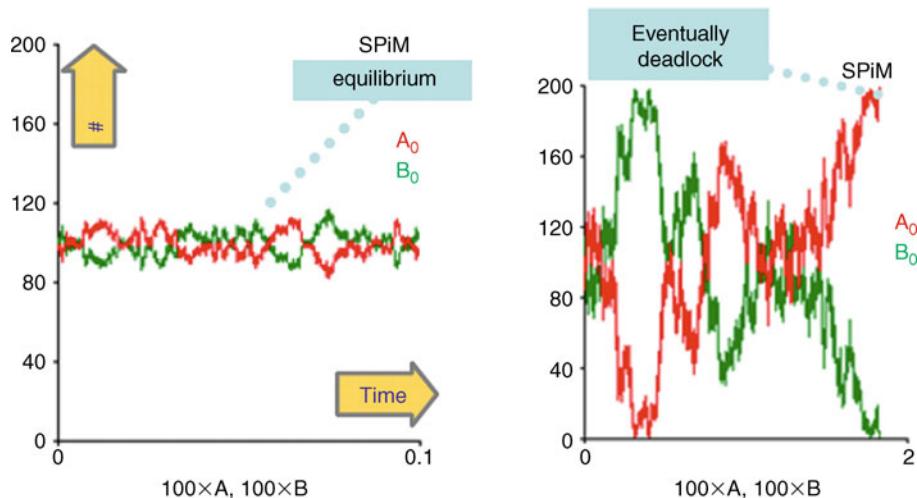
Under the quantities assumed, rates and number of molecules, the overall behavior tends to uniformity where eventually all the groupies end up in the same state, either A or B , where no further reaction is possible.

It is quite instructive to observe the behavior of the same population of groupies with just one celebrity, [Fig. 8](#). The presence of a *single* molecule influences the macroscopic behavior of the system that passes through a deadlock-free random walk: Whenever all groupies get uniform, the celebrity distinguishes itself, followed by some imitating groupie.

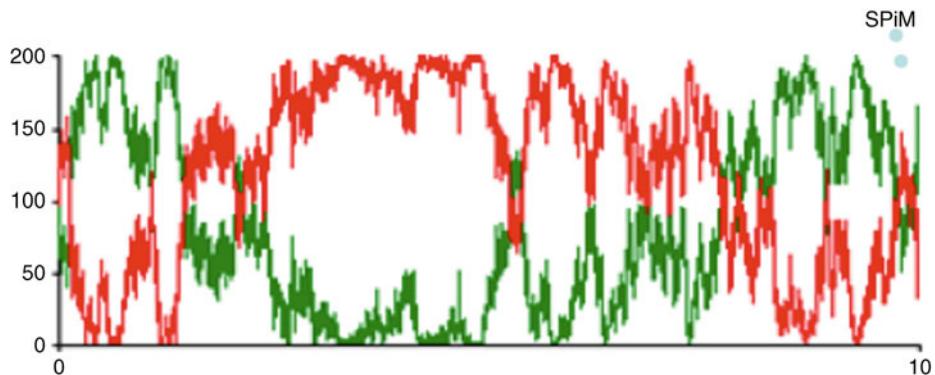
In order not to underestimate the differences between stochastic and deterministic models, it is worth comparing ODEs and process calculi-based models on systems with minimal quantities of “destabilizing” molecules. [Figure 9](#) compares some ODEs (top row) with processes (bottom row). The leftmost column refers to the example of [Fig. 8](#), which exhibits the most evident differences. The other two cases deal with similar examples, in which the number of molecules increases moving rightward, and the groupies offer growing resistance

Fig. 7

Celebrities (left) and groupies (right) (from Cardelli 2009).

**Fig. 8**

Groupies and one celebrity (from Cardelli 2009).

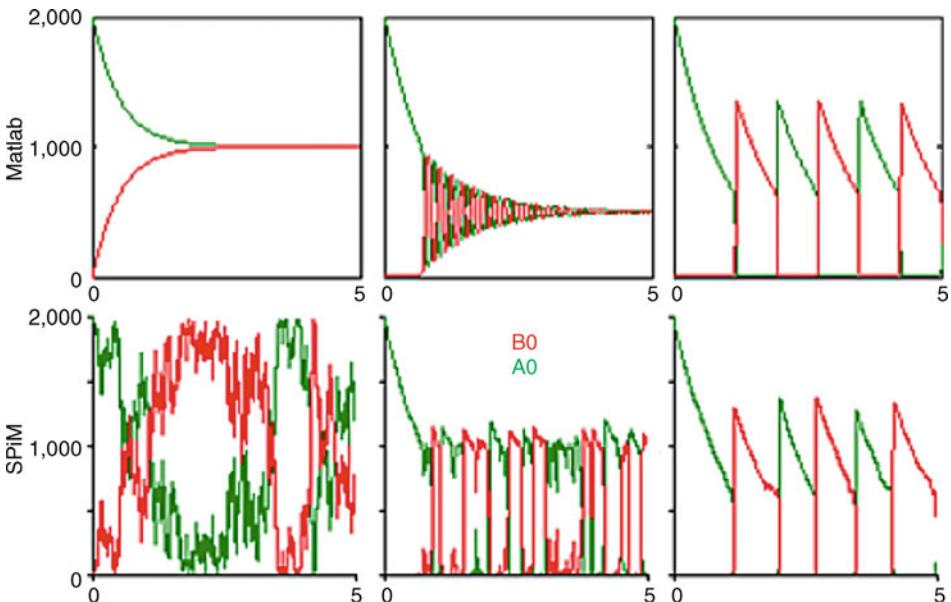


to changes. Consequently, the stochastic behavior tends to assimilate to the continuous case, and the differences between the two approaches blur moving towards the right. As a matter of fact, some phenomena are inherently discrete and stochastic and cannot be faithfully modeled by continuous and deterministic methods.

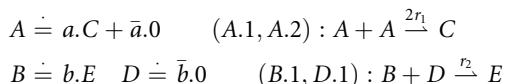
It is now interesting to link the usual representation of chemical reactions with the process-based representation, through a correspondence between the semantics of the two representations. This involves giving both deterministic (ODEs) and probabilistic (CTMC) semantics to chemistry, as done in Cardelli (2008), starting from a formalization of the language of chemical reactions. Note that while the chemical notation is “reaction centric,” that is, the single reactions are listed, the process notation is “reactant centric,” as it consists of a list of reactants with an associated behavior. Then, a mapping is needed from processes to the

Fig. 9

Stochastic and deterministic models are not the same (from Cardelli 2009).



chemical reactions they represent, and backward. We illustrate it through the following example. Consider the processes in the left column below, which will be put in correspondence with the second-order reactions in the right column:



Here r_1 is the rate of channel a and r_2 the rate of channel b (recall that $+$ has a different meaning in reactions and in processes). As part of the mapping, reactions are annotated with information about the reactants/processes that generate them: here the label $(A.1, A.2)$ denotes the two different actions of the same process A . This defines the reaction as occurring among homogeneous reactants, and hence the rate can properly be determined as $2r$. For passing from processes to reactions, these can be labeled with a name that, when translating to processes, may well become the name of the channel representing that specific reaction. This implies that each channel occurs within two processes only (and hence the set of processes is somewhat larger than the set of reactions).

Having defined a language for reactions and one for processes and a way for passing from one to another, one is obliged to prove a correspondence between them. This requires some technical machinery and will not be detailed here. The relevant steps are only sketched here.

For the discrete interpretation of the models, both reactions and processes can be given a CTMC as semantics. This can be straightforwardly derived once the initial conditions are fixed, that is, the number and the species of reactants/processes initially present in a system. For processes, the CTMC semantics follows from the LTS (which turns out to be a graph) associated with the initial state, that is, the initial set of processes composed in parallel. For

reactions, the CTMC semantics follows from a graph representing the possible dynamics of the system of reactions. Relying on a sufficiently analogous structure, the two semantics are then proven equivalent under a suitable equivalence notion. This guarantees that the translation from reactions to processes and vice versa is correct with respect to discrete semantics.

The semantics of the continuous interpretation of the models is the set of ODEs determined according to the Generalized Mass Action Laws. As seen above, once a standard interaction volume is fixed, it is possible to read the number of interacting processes and of reactants as (continuous) concentrations. This requires one to consider the initial conditions as concentrations and to suitably convert rates. Other technical considerations are needed for the hypothesis of continuity in the variation of concentrations. Finally, it has been proven that the continuous semantics for processes is equivalent to the ODEs one.

One can stress again that the often neat correspondence between processes and ODEs models outlined above should not make one believe that the two approaches are freely interchangeable. Indeed, it has been seen that examples of phenomena are intrinsically discrete and stochastic, while others are better modeled by continuous approaches.

We now have a very quick overview of biochemistry and the features required to model it, following Cardelli (2009). Typical biochemical reactions include the creations of complexes, which are best modeled with calculi more expressive than BCCS. A *complex* is obtained by two molecules that interact through complementary reaction sites, a shared channel in these terms, and bind together. As a consequence, those reaction sites are no longer available for further complexations. Furthermore, a complex can split back into the original components, making reaction sites available again. Relevant examples of complexation and decomplexation came from enzymatic activity. Typically, this consists of bounding the enzyme with the substrate, and then splitting the obtained compound either back into enzyme and substrate or, much more likely, into enzyme and product.

Such a notion of composition that preserves the identities of the components cannot be easily represented by BCCS, as so far defined. However, it is sufficient to add a naming mechanism, similar to that of Pi-calculus mentioned in [Sect. 2.2](#), to obtain the needed expressiveness. Processes of such a calculus are a basic ingredient of the so-called *polyautomata*. These have explicit complexation and decomplexation channels and rates, and they also maintain a state with information about performed complexation actions. When two processes synchronize on a complexation channel, the channel is recorded in a binding list that prevents us reusing it. Moreover, these communications associate a brand new name, say n , with the synchronizing processes, so n exactly identifies the two. Then, only the two processes that have been bound together can further synchronize through the unique name n , by performing a decomplexation action that restores the original components.

Other than complexation and detailed enzymatic activity, processes corresponding to polyautomata have been used to model polymerizations as well.

From the computer science point of view, a natural question to ask now is whether the additional expressive power obtained through the naming machinery is only a matter of pragmatics or if it has an intrinsic computational meaning. More precisely, it is interesting to know whether BCCS is Turing-complete, and if not which extensions are required. Roughly, a Turing-complete, or *universal*, formalism can compute through its programs all the functions that the programs of any other formalism can, often called *the computable* functions (typically, a function is such if it is computed by a Turing machine). Note that the programs are all finite, are made of a fixed finite number of symbols, and can be mechanically executed, with no limitations on the number of steps or on the size of the data they operate on.

This is the case with the programs, viz. the processes. A well-known result of computability theory is that there exist functions that are not computable in this sense. In particular, one cannot mechanically decide, in a finite amount of time, if the execution of a program will never halt, by only looking at the text of the program itself (the so-called *halting problem*).

The question of the formal expressive power of chemistry was first addressed by Magnasco (1997), and then studied by Busi and Gossieri (2006) for the BRANE calculus, by Soloveichik et al. (2008) using the different formalism of Minsky's register machine, and by Cardelli and Zavattaro (2008) for calculi such as those discussed above. Briefly, a first answer is that (a variant of) BCCS is *not* universal, in spite of the intuitions of Magnasco (1997). In chemical terms, this can be rephrased by saying that chemistry can implement, with a finite number of species, no Turing-complete formalism. It cannot thus be used to compute all the computable functions. Intuitively, the reason is that one can decide in finite time whether a certain molecule will be produced, so contradicting the undecidability of the halting problem. (Instead, chemistry with an *unbound* number of species is Turing-complete, but clearly its "programs" are no longer finite, nor do they use a finite number of symbols – nor are they very realistic, either!)

When enhanced with the association and dissociation actions discussed above, BCCS becomes universal (Cardelli and Zavattaro 2008). (Note in passing, these two actions cannot therefore be encoded in BCCS, while they can be encoded in the full CCS, which is known to be Turing-complete.) In other words, biochemistry, which includes operations of complexation and decomplexation formalized as polyautomata, is universal.

4 Formal Analysis of Models

One of the distinguishing features of the presented calculi is that a process can be *executed*, and more generally, *automatically processed*. An example of the first modality is the *simulation* of the modeled reality, of which a few examples have been shown. An example of the second kind is *reasoning* on a process, so as to investigate, prove, or disprove some of its properties, which hopefully are in correspondence with some properties of the modeled reality.

A few techniques that can be fully automated or supported by mechanical tools are briefly reported here. These techniques have been largely investigated in computer science in order to provide means for proving properties of software and hardware systems, and they can usually be smoothly applied also to the formalisms for modeling biological entities. Traditionally, these are techniques based on qualitative models, for example, for checking whether a given condition, a deadlock, or an error condition can ever occur in a system. More recently, and particularly following the development of the quantitative models for systems biology, there are proposals that take into account quantitative models, too.

Theorem proving consists in devising a *proof* for a given fact within a formal system. It often requires human supervision and has been generally applied to well-defined contexts, like restricted topics of mathematics and other sciences. It requires a precise domain knowledge, as well as nontrivial search and heuristic strategies implementing some forms of reasoning, for example, induction. It has not generally been applied to system biology and it will not be discussed any further; for an example, see Paulson (1989).

Model checking is instead a technique that can be carried out in a largely automated way. It was defined independently by Clarke and Emerson (Clarke et al. 1986) and Sifakis (1982) in the early 1980s. In its general formulation, model checking consists of verifying whether a

(finite) structure is a *model* for a given property, expressed in a suitable logic. The result can either be positive or negative, in which case a counterexample is returned.

In this case, it consists of checking whether the semantics of a process—that is, its LTS—fulfills a property of interest.

Generally, the logic-expressing properties predicate over LTS; hence, they can also have predicates referring to events that may happen during system evolution, besides the classical Boolean operators such as conjunction, disjunction, and the like. A few operators of the *computation tree logic (CTL*)* (Emerson and Sistla 1983), a well-known logic of this kind, will be considered. A formula f can be an atomic proposition p (which usually depends on the application context), the negation $\neg f$ or the conjunction $f_1 \wedge f_2$ of formulas. A formula holds for a specific state s of the given LTS (the semantics of the logic is a Kripke structure):

$$s \models f$$

Formulas then grow by using modal operators: X is a “next state operator” and is used together with the quantifiers A , always, and E , exists. For example, one can write the formula

$$s \models A X f$$

which is true if and only if f holds in all the states s' reachable after a single transition from s , in symbols if and only if $\forall s'. s \rightarrow s' \ s' \models f$.

The method to verify if the above simple formula is true suggests how to automatize the process of checking whether a given structure is a model for the formula at hand. One mechanically builds the relevant LTS, exploiting the operational semantics. The formula is then verified, often by proceeding by structural induction on it and by exploring the LTS, as done for the formula above. The need to visit the LTS (or to incrementally generate pieces of it when needed) is even more evident for formulas with an “eventually”-like operator $\Diamond f$: there will be a state in the future fulfilling f (see below for a formal definition).

Different working hypotheses, like the finiteness of the LTS and the choice of the logical operators available, lead to different decidability results. This kind of logic is generally nonconclusive in its full formulation (there is not a general algorithmic procedure to check every possible formula against every possible structure).

In Micalan and Bacci (2006), a modal logic for reasoning over qualitative BRANE CALCULI models was introduced, and states are direct processes. Beyond classical Boolean operators, it also consists of operators that predicate over the structure and, like CTL, over the behavior of BRANE (see [Sect. 2.3](#)) processes. Skipping a few technical details, there are structural formulae, like $f_1[f_2]$, the intuition of which is that a complex consists of a membrane fulfilling f_1 and of an internal complex fulfilling f_2 ; the already seen $\Diamond f$, a process can evolve to a state where f is fulfilled; and a modal formula over the structure $\star f$, a process has a sub-term that fulfills f :

$$P \models f_1[f_2] \text{ iff } P = \sigma[Q] \wedge \sigma \models f_1 \wedge Q \models f_2$$

$$P \models \Diamond f \text{ iff } \exists Q. P \rightarrow^* Q \wedge Q \models f$$

$$P \models \star f \text{ iff } \exists Q. P \downarrow^* Q \wedge Q \models f$$

with $P \rightarrow^* Q$ indicating a state Q reachable in zero or more transitions, and \downarrow^* defining a suitable notion of sub-term, typically Q can occur anywhere within P .

An example of use of the logic follows. In settings similar to those of [Example 5](#) about viruses and cells, the following statement can be verified by model checking:

$$\text{Infectable_cell} \models \text{Virus} \triangleright (\diamond \text{Infected_cell})$$

Here *Infectable_cell* is a process modeling a cell that can be infected, *Virus* is a formula fulfilled by processes modeling a given virus (e.g., a formula representing its structure, say), and *Infected_cell* is a formula satisfied by processes modeling infected cells.

Overall, the above formula expresses the fact that an infectable cell of the specified type can eventually become infected if exposed to a virus of the specified type. (Intuitively, the formula $f_1 \triangleright f_2$ is fulfilled by all the processes that, whenever composed with a process fulfilling f_1 , form together a complex fulfilling f_2 .)

The full logic is not decidable, but subsets of it can be chosen that are decidable, hence allowing for fully automated model checking (under the hypothesis of finiteness of the LTS).

Other approaches involving model checking for biological systems are those based on a combination of qualitative and quantitative properties, where quantities are understood as probability values and time. One example is in Kwiatkowska et al. (2008): the reference LTS has associated probabilities, and properties are expressed in suitable temporal logics with probabilities, for example, Aziz et al. (2000) and Baier et al. (2003). The kind of expressible properties talk about the “probability that an event *eventually* occurs” or the “probability that an event occurs within t time units.” Experiments can be done using the PRISM probabilistic model checker (Hinton et al. 2006).

It is worth pointing out that model checking appears to be a mature verification technique, although a full quantitative verification is still an open research topic. Thus, this technique appears adequate to be fruitfully exploited in systems biology. A successful example is shown by the work of Fages and Soliman (2008), which however is carried on a rule-based and reaction-centric approach, rather than the reactant-centric approach surveyed in this paper. A system is given three different semantics, based on ODEs, discrete stochastic processes and Boolean. The last semantics is a sort of “flat” quantitative semantics that only admits the presence or the absence of a component. It is fruitfully used, for example, in checking causal relations among the metabolites of a pathway. Properties can also be expressed by a kind of temporal logic that can express standard qualitative properties (so considering the Boolean semantics) and semiquantitative properties (exploiting discretized ODE-based semantics). The properties are then model checked by verifying the model in hand. Remarkably, the results of model checking permit its user to determine unknown parts of the model, such as reaction rates.

Static techniques refrain from considering the state space a system/process can pass through, as done by model checking. Rather, they analyze “statically” the text of the system, without actually executing it. A typical use of static analysis is in optimizing the code of computer programs, for example, determining from the program code that a variable is dead, that is, it would never be used. In general, these techniques are efficient, being low polynomial in time, but they only give approximate results that err on the safe side. Back to the example above, it may happen that a variable predicted to be used will be never used at execution time (approximation), but it will never happen that a variable is predicted to be dead when it will actually be used (sound).

An example of a static technique used in systems biology is *control flow analysis*, originally proposed for BioAMBIENT in Nielson et al. (2004a) and further developed in Nielson et al. (2004b),

and used for BETA BINDERS in Bodei (2009). For example, properties concerning the nesting of membranes are investigated by considering the set of “possible” interactions as they emerge from process definitions. The approximation comes from the fact that some interactions are statically predicted, but they may be mutually exclusive in the dynamics of the system.

Another problem largely investigated within concurrency is that of establishing the *equivalence* of (the behavior of) processes (Sangiorgi 2004). Behavioral equivalence is at the basis of component interchangeability. Indeed, one can replace a part of a system with an equivalent one, which additionally enjoys a property of interest, or also proceed in incrementally building systems, by replacing (abstract) components with equivalent, more detailed ones. So far, most of the studies have been carried out on qualitative semantics, but proposals for addressing quantitative aspects have been put forward, like notions of stochastic equivalence (see, for example, Doberkat (2007)). Although not yet widely explored, the application of notions of equivalence for biological systems appears of interest, especially for defining equivalences that reflect interchangeability of molecules or of pathways.

The interest in the automated processing of process calculi models has called for the development of a range of computational support, from simulation tools for calculi, like the cited SPiM (Phillips and Cardelli 2007), to extended modeling environments that integrate various analysis tools, both quantitative and qualitative, and support various semantics.

BioPEPA (Ciocchetta and Hillston 2006) builds upon the theory of PEPA (● Sect. 2.2). The essential algebraic representation of calculi is substituted by a richer language. Relevant biological aspects can be explicitly represented, for example, the stoichiometric matrix that summarizes quantitative information about reactions, which is implicitly defined by the overall process interaction in PEPA. Interaction rates are functions that dynamically determine their values according to the current state of the modeled system. Furthermore, the level of detail of the representation can be varied upon according to need, such as by modeling abstract (or non-fully known) interactions involving more than two reactants. Typically, this is done by explicitly associating quantities to reactants, and by assigning a role to actions, such as *activator*, *product*, *reactant*, etc. The semantics consists of an LTS enriched with quantitative information, which has a direct correspondence with CTMCs, as usual. A range of diverse kinetic laws can be applied to model the system. Furthermore, the analysis can be carried out through different approaches, like ODE-based descriptions or Gillespie’s simulations (with a bit of care depending on the kinetic adopted).

BlenX (Dematté et al. 2008) is explicitly presented as a programming language, with a stochastic execution support based on an efficient variant of Gillespie SSA. It builds upon BETA BINDERS (● Sect. 2.3) and consists of a modular architecture, among the modules of which there are the CTMC generator, the one that traces reactions and reagents involved, statistical tools, as well as a graphical user interface and a graphical output module. The programming language includes some mathematics, declarations, binder representations, and its control part relies on (an extended representation of) BETA BINDER processes. Process behavior can also be determined by events, for example, environmental conditions, that may trigger actions, like “no molecules are left of a given species.” BlenX supports modeling of a range of phenomena, such as enzymatic reactions, Michaelis–Menten kinetics, oscillatory behaviors, and self-assembly. As far as analysis is concerned, it features the capability of keeping trace of the species that have played an active role during a system simulation and the specific reactions in which they have taken part.

5 Case Studies

In this section, some case studies that show how suitable process calculi help in constructing *in silico* models of biological phenomena will be reported.

The already mentioned virtual cell experiment VICE (Chiarugi et al. 2004) addresses the biological problem of determining a genome as minimally as possible, and checking whether a hypothetical organism possessing it is able to live or not. The starting point was a hypothetical minimal genome (minimal gene set, MGS) (Mushegian and Koonin 1996), obtained by functionally comparing the genomes of two simple bacteria and eliminating duplicated or functionally identical genes.

To check the viability problem of a MGS-based prokaryote, Chiarugi et al. first specified *in silico* its metabolic pathways, using a process calculus very similar to BCCS. Then, they performed dynamic simulations, that is, virtual experiments of cellular metabolic activities, in order to check whether the MGS-prokaryote was able to reach some equilibrium state and to produce the necessary biomass, as these two properties are enjoyed by any living organism. The simulations clearly showed that MGS does not express an organism able to live, as it cannot reach any equilibrium.

Next, some genes were discarded, others added or replaced upon a functional analysis, and the various genomic sets obtained in this way checked for viability. The virtual experiments were made of about 10^8 transitions/reactions and involved up to 10^7 processes. Using a matrix-based interpreter (Chiarugi et al. 2008), a few minutes each were enough on a medium-sized PC.

After several iterations, a genome composed of 187 elements was selected. It expresses a virtual organism that exhibits homeostatic capabilities and produces biomass in the expected manner. Moreover, the steady-state distribution of the concentrations of virtual metabolites that resulted was similar to that experimentally measured in bacteria, as anticipated in **Example 3**. At least *in silico*, the virtual cell VICE is able to live.

It might be interesting to compare the above results with those of some authors who recently used a wet-lab approach to characterize the minimal gene set necessary to sustain bacterial life (Glass et al. 2006). Their experimentation consisted of knocking out *Mycoplasma genitalium* genes, which are supposed to form a set close to a minimal one. The difference between the genome proposed after *in silico* experimentations and the one obtained *in vitro* is about 10% (while the difference is about 30% with MGS).

A process-based model of pharmacodynamics was proposed in Dematté et al. (2007), to study the effects of drugs controlling the hypertension. A typical therapy consists of acting on the vascular tone, the degree of constriction experienced by a blood vessel relative to its maximally dilated state. The vascular tone is primarily dependent on a protein called myosin light chain kinase (MLCK), which is deeply influenced by intracellular calcium concentration.

Dematté et al. fully specified the nitric oxide-cGMP pathway, through which it is possible to modulate several signal transduction mechanisms that control the level of active MLCK in a cell. To study its effects, the considered drug is also represented as a set of processes.

A main contribution of Dematté et al. (2007) is the definition of the *effective index*, which formally gives sensible measures of the effects of a drug, as well as of the dose-response curve. More precisely, an effective index gives the expectation to reach a state, in the transition system

of the pathway and the drug, which is safe with respect to an observable property, defined as a logical formula. Remarkably, effective indexes are mechanically computed by relying on Markov process theory (Norris 1970).

Summing up, this case study proves the feasibility of an automatic and system-level decision support in drug discovery and analysis, based on process calculi.

The illustrated framework Bio-PEPA (Ciocchetta and Hillston 2006) has been used to investigate a number of phenomena. A repressilator, that is, a synthetic gene regulatory network with oscillating behavior, is analyzed in Ciocchetta and Hillston (2008). Three genes are connected in a feedback loop, where the protein coded by one inhibits the transcription of another one. The model comprises mRNA transcription, protein synthesis and degradation (the first is regulated by the Hill kinetics and the others by the GMA law). Starting from data in the literature, the biological parameters and initial conditions of the experiments are set. Then, exploiting the facilities of Bio-PEPA, both deterministic and stochastic analysis are carried out. The first consists of ODE-based simulations and reveals the expected oscillatory behavior of the three proteins considered. The second consists of a SSA-based simulation that, when averaged over a number of trials, tends to attenuate the oscillations. This phenomenon is explained by a stochastic difference of phases in the oscillations of each trial, which hence tend to cancel each other. Such a behavior underlines how analysis results may require further interpretations and the utility of frameworks like Bio-PEPA, which smoothly allow a phenomenon to be observed under different viewpoints, especially when there is no privileged viewpoint for all the situations.

Gene regulation networks and their feedback-determined behavior have been long appreciated as test-bed examples of the adequacy of the defined models. It is worth recalling here that the first phenomenon modeled within the process calculus approach is the gene regulatory network described and discussed in [Sect. 2.2](#).

In cooperation with two neuroscientists, Bracciali et al. (2008a, b) developed the first stochastic and discrete model of the calyx of Held, a big glutamatergic neuronal synapse in the auditory pathway of the mammalia. The synapses are the places of functional contacts between neurons, where the information is stored and transmitted from one neuron to another. A particular feature of the calyx of Held is its sensitivity to very small quantities of calcium ions, and this makes the continuity assumptions of its ODE-based models not very realistic.

We specified the pre- and the postsynaptic phases in the Stochastic Pi-calculus, starting from existing kinetic models based on ODEs of some sub-components of the synapse, integrating other data from the literature and making some assumptions about non-fully understood processes. A delicate point was determining the stochastic rates of the channels used in processes, starting from the deterministic ones; a similar problem was addressed and solved in Ihewkawa et al. (2007).

In silico experiments have confirmed the coherence of the model with known biological data, also validating the assumptions made. Sensitivity analysis over several parameters of the model has provided results that help to clarify the dynamics of synaptic transmission and explain short-term plasticity mechanisms that are supposed to be at the basis of memory.

It is worth remarking that the compositionality typical of process calculi has permitted us to specify separately, experiment on, and tune the models of the pre- and postsynaptic traits, and then to connect them in a plain and straightforward way.

Acknowledgments

The authors are deeply indebted to Luca Cardelli who kindly gave them permission to reuse here parts of his work, as well as Davide Chiarugi and Roberto Marangoni for joint previous work on VICE. The authors thank Enrico Cataldo for helpful comments and suggestions. The first author wishes to thank all the people at the Microsoft Research—University of Trento Centre for Computational and Systems Biology.

References

- Aziz A, Sanwal K, Singhal V, Brayton R (2000) Model checking continuous time Markov chains. *ACM Trans Comput Logic* 1(1):162–170
- Baier C, Haverkort B, Hermanns H, Katoen J-P (2003) Model-checking algorithms for continuous-time Markov chains. *IEEE Trans Software Eng* 29(6):524–541
- Bergstra JA, Ponse A, and Smolka SA (2001) *Handbook of process algebra*. North-Holland, Amsterdam, The Netherlands
- Bernardo M, Degano P, Zavattaro G (eds) (2008) Formal methods for computational systems biology. In: SFM 2008: 8th international school on formal methods for the design of computer, communication, and software systems, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, Berlin
- Bodei C (2009) A control flow analysis for beta-binders with and without static compartments. *Theor Comput Sci* 410(33–34):3110–3127
- Bracciali A, Brunelli M, Cataldo E, Degano P (2008a) Stochastic models for the *in silico* simulation of synaptic processes. *BMC Bioinform* 9(4):S7
- Bracciali A, Brunelli M, Cataldo E, Degano P (2008b) Synapses as stochastic concurrent systems. *Theor Comput Sci* 408(1):66–82, 2008
- Bradley J (1999) Towards reliable modelling with stochastic process algebras. PhD thesis, Department of Computer Science, University of Bristol
- Brodo L, Degano P, Priami C (2007) A stochastic semantics for BioAmbients. In: Proceedings of PaCT, Pereslavl-Zalessky, Russia, September 2007. Lecture notes in computer science, vol 4671. Springer, Heidelberg
- Busi N, Gorrieri R (2006) On the computational power of Brane calculi. In: Transactions on computational systems biology VI. Lecture notes in computer science, vol 4220. Springer, Heidelberg, pp 16–43
- Calzone L, Fages F, Soliman S (2006) BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics* 22(14):1805–1807
- Cardelli L (2009) Artificial biochemistry. In: Condon A, Harel D, Kok JN, Salomaa A, Winfree E (eds) *Algorithmic bioprocesses*. Springer, New York
- Cardelli L (2008) On process rate semantics. *Theor Comput Sci* 391(3):190–215
- Cardelli L (2004) Brane calculi-interactions of biological membranes. In: Danos V, Schachter V (eds) *Proceedings of computational methods in systems biology*, Paris, France, May 2004. Lecture notes in computer science, vol 3082. Springer, Berlin
- Cardelli L, Gordon A (1998) Mobile ambients. In: Nivat M (ed) *Proceedings of FoSSaCS'98*, Lisbon, Portugal, March–April 1998. Lecture notes in computer science, vol 1378. Springer, Berlin, pp 140–155
- Cardelli L, Zavattaro G (2008) On the computational power of biochemistry. In: *Proceedings of algebraic biology*, Castle of Hagenberg, Austria, July–August 2008. Lecture notes in computer science, vol 5147. Springer, Berlin
- Chiarugi D, Curti M, Degano P, Marangoni R (2004) ViCe: a Virtual CCell. In: *Proceedings of 2nd international W/S computational methods in systems biology*, Paris, France, May 2004. Lecture notes in computer science, vol 3082. Springer, Berlin
- Chiarugi D, Degano P, Marangoni R (2007) A computational approach to the functional screening of genomes. *PLoS Comput Biol* 3(9):1801–1806
- Chiarugi D, Degano P, Bert Van Klinken J, Marangoni R (2008) Cells *in silico*: a holistic approach. In: *Formal methods for computational systems biology*, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, Berlin, pp 366–386
- Ciocchetta F, Hillston J (2006) Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. In: *Proceedings of FBTC 2007*, Lisbon, Portugal, September 2007. Electr Notes Theor Comput Sci 194(3):101–117
- Ciocchetta F, Hillston J (2008) Process algebras in systems biology. In: Bernardo M, Degano P, Zavattaro G (eds) *SFM 2008: Formal methods for computational systems biology*, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, Berlin, pp 265–312
- Clarke EM, Emerson EA, Sistla AP (1986) Automatic verification of finite-state concurrent systems using

- temporal logic specifications. *ACM Trans Program Lang Syst* 8(2):244–263
- Damm W, Harel D (2001) LSCs: breathing life into message sequence charts. *Formal Methods Syst Des* 19(1):45–80
- Danos V, Feret J, Fontana W, Harmer R, Krivine J (2007) Rule-based modelling of cellular signalling. In: Proceedings of CONCUR, Lisbon, Portugal, September 2007. Lecture notes in computer science, vol 4703. Springer, Berlin, pp 17–41
- Degano P, Prandi D, Priami C, Quaglia P (2006) Beta-binders for biological quantitative experiments. In: Proceedings of QAPL06, Vienna, Austria, April 2006. *Electr Notes Theor Comput Sci* 164(3): 101–117
- Dematté L, Prandi D, Priami C, Romanel A (2007) Effectiveness Index: A formal measure of drug effects. In: Proceedings of the 2nd Conference Foundations of Systems Biology in Engineering (FOSBE). Stuttgart, Germany, September 2007, pp 485–490
- Dematté L, Priami C, Romanel A (2008) The BlenX language: a tutorial. In: Bernardo M, Degano P, Zavattaro G (eds) SFM 2008, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, Berlin, pp 313–365
- Doberkat E-E (2007) Stochastic relations. Chapman & Hall/CRC, Boca Raton, FL
- Eker S, Knapp M, Laderoute K, Lincoln P, Meseguer J, Sönmez MK (2002) Pathway logic: symbolic analysis of biological signaling. In: Altman RB, Dunker AK, Hunter L, Lauderdale K, Klein TE (eds) Pacific symposium on biocomputing. Kauai, HI, 3–7 January 2002, pp 400–412
- Emerson EA, Sistla AP (1983) Deciding branching time logic: a triple exponential decision procedure for CTL*. In: Clarke EM, Kozen D (eds) Proceedings logic of programs, Pittsburgh, PA, June 1983. Lecture notes in computer science, vol 164. Springer, Berlin, pp 176–192
- Ermentrout B (2002) Simulating, analyzing, and animating dynamical systems. SIAM, Philadelphia, PA
- Fages F, Soliman S (2008) Formal cell biology in Bioc-ham. In: Bernardo M, Degano P, Zavattaro G (eds) SFM 2008: Formal methods for computational systems biology, Bertinoro, Italy, June 2008. Lecture notes in computer science, vol 5016. Springer, Berlin, pp 265–312
- Fell DA (1997) Understanding the control of metabolism. Portland Press, London
- Fersht A (1999) Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. Freeman, New York
- Fontana W, Buss IW (1994) The arrival of the fittest: toward a theory of biological organization. *Bull Math Biol* 56:1–64
- Fraser CM et al. (1995) The minimal gene complement of mycoplasma genitalium. *Science* 270(1):397–403
- Gardiner CW (2001) Handbook of stochastic methods for physics, chemistry and the natural sciences. Springer, Berlin
- Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* 81: 2340–2361
- Gillespie DT, Petzold LR (2006) Numerical simulation for biochemical kinetics. In: Szallasi Z, Stelling J, Perival V (eds) System modeling in cellular biology, 1st edn. MIT Press, Cambridge, MA, pp 331–354
- Glass J, Assad-Garcia N, Alperovich N (2006) Essential genes of a minimal bacterium. *PNAS* 103:425–430
- Hammes GG, Shimmel PR (1970) In: Boyer PD (ed) The enzymes, vol 2. Academic Press, New York
- Hillston J (1993) PEPA – performance enhanced process algebra. PhD thesis, University of Edinburgh, Computer Science Department
- Hillston J (1994) The nature of synchronisation. In: Herzog U, Rettelbach M (eds) Proceedings of 2nd workshop on Process Algebras and Performance Modelling (PAPM'92). Erlangen, Germany, July 1994, pp 51–70
- Hillston J (2005) Process algebras for quantitative analysis. In: LICS 2005: Proceedings of the 20th annual symposium on logic in computer science, Chicago, IL, USA, June 2005. IEEE Computer Society, Washington DC, pp 239–248
- Hillston J (1996) A compositional approach to performance modelling. Cambridge University Press, Cambridge
- Hinton A, Kwiatkowska M, Norman G, Parker D (2006) PRISM: a tool for automatic verification of probabilistic systems. In: Hermanns H, Palsberg J (eds) Proceedings 12th international conference on tools and algorithms for the construction and analysis of systems, Vienna, Austria. Lecture notes in computer science, vol 3920. Springer, Heidelberg
- Hoare CAR (1985) Communicating sequential processes. Prentice-Hall, Englewood Cliffs, NJ
- Ihekweaba A, Larcher R, Mardare R, Priami C (2007) BetaWB – a language for modular representation of biological systems. In: Proceedings of ICSB 2007, Long Beach, CA, October 2007
- Kitano H (2002) Systems biology: a brief overview. *Theor Comput Sci* 295(5/6):1662–1664
- Kwiatkowska MZ, Norman G, Parker D (2008) Using probabilistic model checking for systems biology. *SIGMETRICS Performance Evaluation Review* 35(4):14–21
- Larry L, Roger B (2005) Automatic generation of cellular reaction networks with molecuLizer 1.0. *Nat Biotechnol* 23:131–136
- Magnasco MO (1997) Chemical kinetics is Turing universal. *Phys Rev Lett* 78:1190–1193
- Miculan M, Bacci G (2006) Modal logics for Brane calculus. In: Priami C (ed) CMSB06: Computational

- methods in systems biology, Trento, Italy, October 2006. Lecture notes in computer science, vol 4210. Springer, Heidelberg, pp 1–16
- Milazzo P (2008) Formal modeling in systems biology. An approach from theoretical computer Science. VDM - Verlag Dr. Müller, Saarbrücken, Germany
- Milner R (1980) A calculus of communicating systems. Lecture notes in computer science, vol 92. Springer, Berlin
- Milner R (1989) Communication and concurrency. Prentice-Hall, Englewood Cliffs, NJ
- Milner R (1999) Communicating and mobile systems: the π -calculus. Cambridge University Press, Cambridge
- Milner R, Parrow J, Walker D (1992) A calculus of mobile processes, I-II. *Inform Comput* 100(1):1–77
- Mushegian AR, Koonin EV (1996) A minimal gene set for cellular life derived by comparison of complete bacterial genome. *PNAS* 93:10268–10273
- Nielson F, Riis Nielson H, Schuch-Da-Rosa D, Priami C (2004a) Static analysis for systems biology. In: Proceedings of workshop on systematics - dynamic biological systems informatics, Cancun, Mexico, 2004. Computer Science Press, Trinity College Dublin, pp 1–6
- Nielson HR, Nielson F, Pilegaard H (2004b) Spatial analysis of BioAmbient. In: Proceedings of static analysis symposium, Verona, Italy, August 2004. Lecture notes in computer science, vol 3148. Springer, Berlin, pp 69–83
- Norris JR (1970) Markov chains. Cambridge University Press, Cambridge, MA
- Paulson LC (1989) The foundation of a generic theorem prover. *J Automated Reasoning* 5(3):363–397
- Paun G, Pérez-Jiménez MJ, Salomaa A (2007) Spiking neural P systems: an early survey. *Int J Found Comput Sci* 18(3):435–455
- Phillips A, Cardelli L (2007) Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Calder M, Gilmore S (eds) Proceedings of computational methods in systems biology, Edinburgh, Scotland, September 2007. Lecture notes in computer science, vol 4695. Springer, Heidelberg, pp 184–199
- Plotkin GD (2004) A structural approach to operational semantics. *J Log Algebr Program* 60–61:17–139
- Priami C (1995) Stochastic π -calculus. *Comput J* 36 (6):578–589
- Priami C, Quaglia P (2004) Beta binders for biological interactions. In: Proceedings of CMSB, Paris, France, May 2004. Lecture notes in computer science, vol 3082. Springer, Berlin, pp 20–32
- Priami C, Regev A, Shapiro E, Silvermann W (2004) Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Theor Comput Sci* 325(1):141–167
- Reddy VN, Mavrounouiotis ML, Lieberman MN (1993) Qualitative analysis of biochemical reduction systems. *Comput Biol Med* 26(1):9–24
- Regev A, Shapiro E (2002) Cellular abstractions: cells as computation. *Nature* 419:343
- Regev A, Panina E, Silverman W, Cardelli L, Shapiro E (2004) BioAmbients: an abstraction for biological compartments. *Theor Comput Sci* 325(1):141–167
- Sadot A, Fisher J, Barak D, Admanit Y, Stern MJ, Hubbard EJA, Harel D (2008) Toward verified biological models. *IEEE/ACM Trans Comput Biol Bioinform* 5(2):223–234
- Sangiorgi D (2004) Bisimulation: from the origins to today. In: LICS 2004: Proceeding of 19th IEEE symposium on logic in computer science, Turku, Finland, July 2004. IEEE Computer Society, Washington DC, pp 298–302
- Schrödinger E (1946) What is life? Macmillan, New York
- Segel LA (1987) Modeling dynamic phenomena in molecular and cellular biology. Cambridge University Press, Cambridge
- Sifakis J (1982) A unified approach for studying the properties of transition systems. *Theor Comput Sci* 18:227–258
- Smith GD (2005) Modeling the stochastic gating of ion channels. In: Fall CP, Marland ES, Wagner JM, Tyson JJ (eds) Computational cell biology, 2nd edn. Springer, New York, pp 285–319
- Soloveichik D, Cook M, Winfree E, Bruck J (2008) Computation with finite stochastic chemical reaction networks. *Nat Comput.* doi: 10.1007/s11047-008-9067-y (2008)
- Van Kampen NG (1992) Stochastic processes in physics and in chemistry. Elsevier, Amsterdam, The Netherlands
- Voit EO (2000) Computational analysis of biochemical systems – a practical guide for biochemists and molecular biologists. Cambridge University Press, Cambridge
- Wilkinson DJ (2006) Stochastic modelling for systems biology. Chapman & Hall – CRC Press, London
- Wolkenhauer O (2008) Systems biology – Dynamic pathway modelling. Manuscript, available at http://www.sbi.uni-rostock.de/dokumente/t_sb.pdf
- Zhao J, Ridgway D, Broderick G, Kovalenko A, Ellison M (2008) Extraction of elementary rate constants from global network analysis of *E. Coli* central metabolism. *BMC Syst Biol* 2:41

56 Reaction–Diffusion Computing

Andrew Adamatzky¹ · Benjamin De Lacy Costello²

¹Department of Computer Science, University of the West of England, Bristol, UK

andrew.adamatzky@uwe.ac.uk

²Centre for Research in Analytical, Material and Sensor Sciences, Faculty of Applied Sciences, University of the West of England, Bristol, UK
ben.delacycostello@uwe.ac.uk

1	<i>Introduction</i>	1898
2	<i>History and Milestones of Reaction–Diffusion Computing</i>	1899
3	<i>Classification of Reaction–Diffusion Processors</i>	1901
4	<i>Recipes of the Chemical Processors</i>	1902
5	<i>Specialized Processors</i>	1904
6	<i>Universal Processors</i>	1914
7	<i>Conclusion</i>	1918

Abstract

A reaction–diffusion computer is a spatially extended chemical system, which processes information by transforming an input concentration profile to an output concentration profile in a deterministic and controlled manner. In reaction–diffusion computers, the data are represented by concentration profiles of reagents, information is transferred by propagating diffusive and phase waves, computation is implemented via the interaction of these traveling patterns (diffusive and excitation waves), and the results of the computation are recorded as a final concentration profile. Chemical reaction–diffusion computing is among the leaders in providing experimental prototypes in the fields of unconventional and nature-inspired computing. This chapter provides a case-study introduction to the field of reaction–diffusion computing, and shows how selected problems and tasks of computational geometry, robotics, and logics can be solved by encoding data within transient states of a chemical medium and by programming the dynamics and interactions of chemical waves.

1 Introduction

The field of natural computation (sometimes “unconventional computation,” “nonclassical computing,” and “nonstandard computation” can be used as well, depending on the context and personal preferences) is concerned with the design of computing paradigms, architectures, and experimental implementations of novel computing devices. These devices are based around the principles of information processing in chemical, physical, and biological systems and the implementation of conventional algorithms in nonstandard, non-silicon substrates (Adamatzky and Teuscher 2006; Adamatzky et al. 2007; Akl et al. 2007). The majority of the results published in natural computation (up to 99% of papers related to natural computation are theoretical) deal with algorithms, methodologies, or theories of computing based on paradigms or, occasionally very distant, analogies with biological, physical, and chemical processes. Only a few experimental prototypes of unconventional natural computers have been implemented in laboratories over the last few decades. These prototypes include examples of reaction–diffusion chemical processors (Adamatzky et al. 2005b; Adamatzky and De Lacy Costello 2002b; Tóth and Showalter 1995; Steinbock et al. 1996; Gorecki et al. 2003; 2005; Gorecka and Gorecki 2003; Motoike et al. 2001; Ichino et al. 2003), extended analog computers (Mills 2008), micro-fluidic circuits (Fuerstman et al. 2003), and plasmodium computers (Nakagaki et al. 2001; Shirakawa et al. 2009). This tendency is worrying because it looks like living in a world populated with combustion scientists but no car mechanics, aerodynamic engineers but no pilots. Bearing in mind that there are plenty of “talkers” but on the evidence just a few “doers” in this developing field we decided to focus this chapter entirely on experimental prototypes developed by the authors and coworkers/collaborators (just a couple of illustrations assisted by numerical experiments are included).

The chapter is structured as follows. ➤ [Section 2](#) is a short excursion into the history of reaction–diffusion computing and lists the main achievements in the field. A classification of reaction–diffusion computers is provided in ➤ [Sect. 3](#). In ➤ [Sect. 4](#), chemical recipes are provided for the preparation of the basic types of reaction–diffusion processors: precipitating palladium processors and excitable chemical media in the form of the Belousov–Zhabotinsky (BZ) reaction. Specialized chemical processors for computational geometry and robot control are discussed in ➤ [Sect. 5](#). In ➤ [Sect. 6](#), how to implement basic logical gates

in a sub-excitable chemical medium is shown. A few thoughts on the future developments in the field are outlined in [Sect. 7](#).

This chapter is only able to give a brief introduction to the field of reaction-diffusion computing; for this reason, the existence of working prototypes and some notable practical implementations are mainly stated without going far below the surface. Therefore, the interested reader is strongly encouraged to consult the book by Adamatzky et al. ([2005b](#)) for further details on the realization of chemical processors and also the implementation of the reaction-diffusion paradigm in silicon devices.

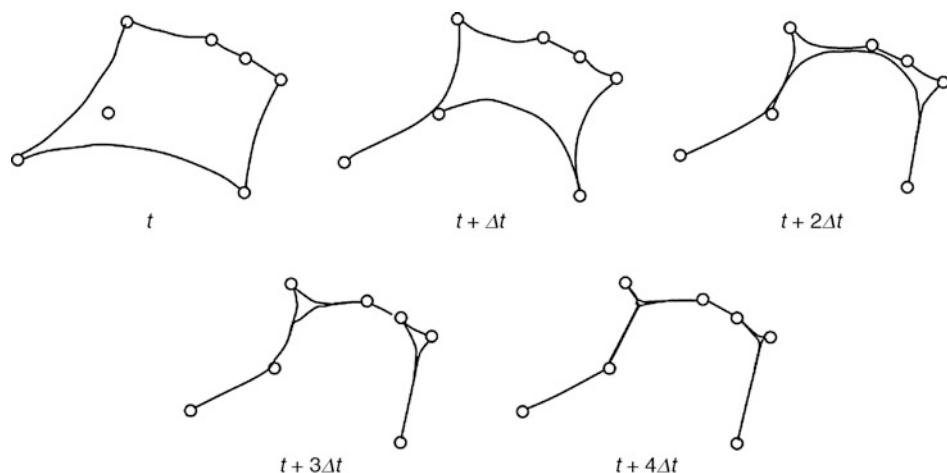
2 History and Milestones of Reaction-Diffusion Computing

The idea of physics-based computation can be attributed to Joseph Plateau, see references in Courant and Robbins ([1941](#)), who thought about methods to devise an experimental solution to the problem of calculating the surface of smallest area bounded by a given closed contour in space. Given a set of planar points, connect the points by a graph with the minimal sum of edge lengths (it is allowed to add more points, however the number of additional points should be minimal). Plateau's solution is simple yet nontrivial: mark the given planar points on a flat surface, insert pins in the points, place another sheet on top of the pins, briefly immerse the device in a soap solution, wait till the soap film dries, then record the topology of the dried soap film. The dried soap film represents a minimal Steiner tree spanning the given planar points ([Fig. 1](#)).

Due to surface tension, the soap film between the pins, representing points, naturally tries to minimize the total surface area. A length-minimizing curve enclosing a fixed-area region consists of circular arcs of positive outward curvature and line segments (Saltenis [1999](#)). The curvature of the arcs is inversely proportional to the pressure. By gradually increasing the pressure ([Fig. 1](#)) arcs can be transformed to straight lines, and thus a spanning tree is calculated.

Fig. 1

Several steps of spanning tree constructions by soap film (Saltenis [1999](#)).



The soap film does not involve diffusion or reaction; however, the operation is somewhat analogous to chemical reaction–diffusion computers. Initial data sites are represented by local perturbations of the system’s state. Information is transferred between the sites by propagating patterns. The computation is implemented via interaction of the propagating patterns and the result of the computation is represented by a final spatial configuration of the system. After the work of Plateau, it would be another hundred years until the Belousov–Zhabotinsky (BZ) reaction (Zaikin and Zhabotinsky 1970) became widely known and Kuhnert published his pioneering results (Kuhnert 1986a, b; Kuhnert et al. 1989) on implementation of memory devices and basic image processing procedures using a light-sensitive BZ medium. These first BZ processors did not employ the propagation of excitation waves but relied on global switching of the medium between excited and nonexcited states termed phase shifts. In a light sensitive analogue of the BZ reaction, the timing of these phase shifts could be coupled to the projected light intensity due to the light sensitivity of the catalyst used in the reaction. However, the publications encouraged other researchers to experiment with information processing in excitable chemical systems. The image processing capabilities of BZ systems were further explored and the research results were enhanced in Agladze et al. (1995), Rambidi (1997, 1998, 2003), Rambidi and Yakovenchuk (2001), and Rambidi et al. (2002).

In the mid-1990s, the first results concerning the directed, one-way, propagation of excitation waves in geometrically constrained chemical media was announced (Agladze et al. 1996). This led to a series of experimental studies concerning the construction of Boolean (Tóth and Showalter 1995; Steinbock et al. 1996; Motoike and Yoshikawa 2003) and three-valued (Motoike and Adamatzky 2004) logical gates and eventually led to the construction of more advanced circuits (Gorecka and Gorecki 2003; Gorecki et al. 2003; Ichino et al. 2003; Motoike and Yoshikawa 1999) and dynamical memory (Motoike et al. 2001) in the BZ reaction. The BZ medium was also used for one optimization task – shortest path computation (Steinbock et al. 1995; Agladze et al. 1997; Rambidi and Yakovenchuk 2001).

The studies in reaction–diffusion computing were boosted by algorithms of spatial computation in cellular automata, when the first automaton model for Voronoi diagram construction was designed (Adamatzky 1994, 1996). The algorithm was subsequently implemented under experimental laboratory conditions using a precipitating chemical processor (Tolmachiev and Adamatzky 1996; Adamatzky and Tolmachiev 1997), and later a variety of precipitating chemical systems were discovered to be capable of Voronoi diagram approximation (De Lacy Costello 2003; De Lacy Costello and Adamatzky 2003; De Lacy Costello et al. 2004a, b). The precipitating systems also proved to be efficient in the realization of basic logical gates, including many-valued gates (Adamatzky and De Lacy Costello 2002b).

In the early 2000s, the first ever excitable chemical controller mounted on-board a wheeled robot was constructed and tested under experimental laboratory conditions (Adamatzky and De Lacy Costello 2002a; Adamatzky et al. 2003, 2004), and also a robotic hand was interfaced and controlled using a Belousov–Zhabotinsky medium (Yokoi et al. 2004). These preliminary experiments opened the door to the future construction of embedded robotic controllers and other intelligent reaction–diffusion processors.

As for computational universality, sub-excitable analogues of chemical media have been proved by experiment to be powerful collision-based computers, capable of the implementation of universal and arithmetical logical circuits via the collision of propagating wave-fragments (Adamatzky 2004; Adamatzky and De Lacy Costello 2007; Toth et al. 2007).

3 Classification of Reaction–Diffusion Processors

In reaction–diffusion processors, both the data and the results of the computation are encoded as concentration profiles of the reagents. The computation *per se* is performed via the spreading and interaction of wave fronts.

This can be summarized using the following relationships between architecture or operation and implementation:

- Component base, computing substrate → Thin layer of reagents
- Data representation → Initial concentration profile
- Information transfer, communication → Diffusive and phase waves
- Computation → Interaction of waves
- Result representation → Final concentration profile

Reaction–diffusion computers are parallel because the chemical medium’s micro-volumes update their states simultaneously, and molecules diffuse and react in parallel. The architecture and operations of reaction–diffusion computers are based on three principles (Margolus 1984; Adamatzky 2001):

- Computation is dynamical: physical action measures the amount of information
- Computation is local: physical information travels only a finite distance
- Computation is spatial: their nature is governed by waves and spreading patterns

Potentially, a reaction–diffusion computer is a super-computer in a goo. Liquid-phase chemical computing media are characterized by

- Massive parallelism: millions of elementary – 2–4 bit – processors in a small chemical reactor
- Local connectivity: every micro-volume of the medium changes its state depending on the states of its closest neighbors
- Parallel I/O: optical input – control of initial excitation dynamics by illumination masks, output is parallel because the concentration profile representing the results of computation is visualized by indicators
- Fault tolerance and automatic re-configuration: because if some quantity of the liquid phase is removed, the topology is restored almost immediately

As far as reusability is concerned, there are two main classes of reaction–diffusion processors: reusable and disposable. Excitable chemical media are one of a family of reusable processors because given an unlimited supply of reagents, unlimited lifetime of the catalyst and the removal of any by-products, the system can be excited many times and at many simultaneous points on the computing substrate. The excitation waves leave almost no trace a certain time after the propagation. Precipitating chemical systems on the other hand are disposable processors. Once a loci of the substrate is converted to precipitate, it stays in the precipitate state indefinitely and therefore this processor cannot be used again within a realistic timeframe.

With regard to the communication between the domains of the computing medium the chemical media can be classified into either broadcasting or peer-to-peer architectures. Excitable chemical media and most precipitating systems are broadcasting. When a single site of a medium is perturbed the perturbation spreads in all directions, as a classical circular wave. Thus, each perturbed site broadcasts. However, when diffusion or excitation in the medium is limited, for example, the sub-excitable Belousov–Zhabotinsky system, no circular waves are

formed but instead self-localized propagating wave fragments are formed (chemical relatives of dissipative solitons). These wave-fragments propagate in a predetermined direction, therefore they can be seen as data packets transferred between any two points of the computing medium.

Broadcasting chemical processors are good for image processing tasks, some kinds of robot navigation and path computation. They are appropriate for all tasks when massive spatially extended data must be analyzed, modified, and processed. The circular-wave-based communication becomes less handy when one tries to build logical circuits: It is preferable to route the signal directly between two gates without affecting or disturbing nearby gates.

When one uses an excitable medium to implement a logical circuit, the propagation of the excitable waves must be restricted, for example, by making conductive channels surrounded by nonconductive barriers, or via the use of localized excitations in a sub-excitatory chemical medium. Reaction-diffusion processors where the computing space is geometrically inhomogeneous are called geometrically constrained processors. Sub-excitable chemical media, where wave-fragments propagate freely are called collision-based (because the computation is implemented when solution-like wave-fragments collide) or architecture-less/free-space computers (because the medium is homogeneous, regular, and uniform).

Further, examples of architecture-less chemical processors acting in either broadcasting or peer-to-peer modes are provided.

4 Recipes of the Chemical Processors

In this chapter, only three kinds of chemical reaction-diffusion processors are described: the precipitating palladium processor, and two chemical processors: an excitable Belousov-Zhabotinsky (BZ) medium and a light-sensitive sub-excitable analogue of the BZ reaction. Below we show how to prepare the reaction-diffusion processors in laboratory conditions.

4.1 Palladium Processor

A gel of agar (1.5% by weight, agar select Sigma-Aldrich Company Ltd., Poole, Dorset, UK) containing palladium chloride (Palladium (II) chloride 99%, Sigma-Aldrich Company Ltd.) in the range 0.2–0.9% by weight (0.011–0.051 M) is prepared by mixing the solids in warm deionized water. The mixture is heated with a naked flame and is constantly stirred until it boils to ensure full dissolution of the palladium chloride and production of a uniform gel (on cooling). The boiling liquid is then transferred to Petri dishes or alternatively spread on acetate sheets to a thickness of 1–2 mm and left to set. One can favor the use of Petri dishes because the reaction process is relatively slow and drying of the gel can occur if kept open to the atmosphere. The unreacted gel processors are then kept for 30 min although they remained stable for in excess of 24 h provided drying was controlled. A saturated solution (at 20°C) of potassium iodide (ACS reagent grade, Aldrich Chemical Co.) is used as the outer electrolyte for the reactions. Drops of outer electrolyte are applied to the surface of the gel to initiate the reaction process.

4.2 BZ Processor

A thin layer Belousov-Zhabotinsky (BZ) medium is usually prepared using a recipe adapted from Field and Winfree (1979): an acidic bromate stock solution incorporating potassium

bromate and sulfuric acid ($[BrO_3^-] = 0.5\text{ M}$ and $[H^+] = 0.59\text{ M}$) (solution A); solution of malonic acid (solution B) ($[CH_2(CO_2H)_2] = 0.5\text{ M}$); and sodium bromide (solution C) ($[Br^-] = 0.97\text{ M}$). Ferroin (1,10-phenanthroline iron-II sulfate, 0.025 M) is used as a catalyst and a visual indicator of the excitation activity in the BZ medium. To prepare a thin layer of the BZ medium, one can mix solutions A (7 ml), B (3.5 ml), C (1.2 ml), and finally when the solution becomes colorless, ferroin (1 ml) is added and the mixture is transferred to a Petri dish (layer thickness 1 mm). Excitation waves in the BZ reaction are initiated using a silver colloid solution.

4.3 Light-Sensitive Sub-excitatory BZ Processor

The recipe is quoted from Toth et al. (2007). A light-sensitive BZ processor consists of a gel impregnated with catalyst and a catalyst-free solution pumped around the gel. The gel is produced using a sodium silicate solution prepared by mixing 222 ml of the sodium silicate solution with 57 ml of 2 M sulfuric acid and 187 ml of deionized water (Toth et al. 2007). The catalyst $Ru(bpy)_3SO_4$ is recrystallized from the chloride salt with sulfuric acid. Pre-cured solutions for making gels are prepared by mixing 2.5 ml of the acidified silicate solution with 0.6 ml of 0.025 M $Ru(bpy)_3SO_4$ and 0.65 ml of 1.0 M sulfuric acid solution. Using capillary action, portions of this solution were quickly transferred into a custom-designed 25 cm long, 0.3 mm deep Perspex mold covered with microscope slides. The solutions are left for 3 h in the mold to permit complete gelation. After gelation, the adherence to the Perspex mold is negligible leaving a thin gel layer on the glass slide. After 3 h, the slides with the gel on them are carefully removed from the mold and placed into 0.2 M sulfuric acid solution for an hour. Then they are washed in deionized water at least five times to remove by-products. The gels are 26 mm \times 26 mm, with a wet thickness of approximately 300 m. The gels are stored under water and rinsed just before use.

The catalyst-free reaction mixture is freshly prepared in a 30 ml continuously fed stirred tank reactor, which involves the *in situ* synthesis of stoichiometric bromomalonic acid from malonic acid and bromine generated from the partial reduction of sodium bromate. This reactor is continuously fed with fresh catalyst-free BZ solution in order to maintain a nonequilibrium state. The final composition of the catalyst-free reaction solution in the reactor is the following: 0.42 M sodium bromate, 0.19 M malonic acid, 0.64 M sulfuric acid, and 0.11 M bromide.

A light projector is used to illuminate the computer-controlled image. Images are captured using a digital camera. The open reactor is surrounded by a water jacket thermostated at 22°C. Peristaltic pumps are used to pump the reaction solution into the reactor and remove the effluent.

In some cases, the regular structure of illumination is applied onto the BZ chemical reactor. Four light levels can be used: black (zero light level), at which level the reaction BZ oscillates; dark (0.035 mW cm^{-2}), which represents the excitable medium in which a chemical wave is able to propagate; white (maximum light intensity: 3.5 mW cm^{-2}), the inhibitory level; and finally the light level (1.35 mW cm^{-2}) corresponding to the weakly excitable medium, where excitation just manages to propagate.

Waves were initiated by setting the light intensity to zero within a small square at specific points on the gel surface. A black oscillating square is used to initiate waves periodically. The waves are then directed from source into a weakly excitable area (controlled by projecting a light intensity of 1.35 mW cm^{-2}) such that only small fragments are able to propagate.

5 Specialized Processors

As their name suggests, specialized processors are designed to solve just one possible computational task (or family of very similar tasks). In this section, examples of specialized processors for computational geometry, image processing, and robotics are provided.

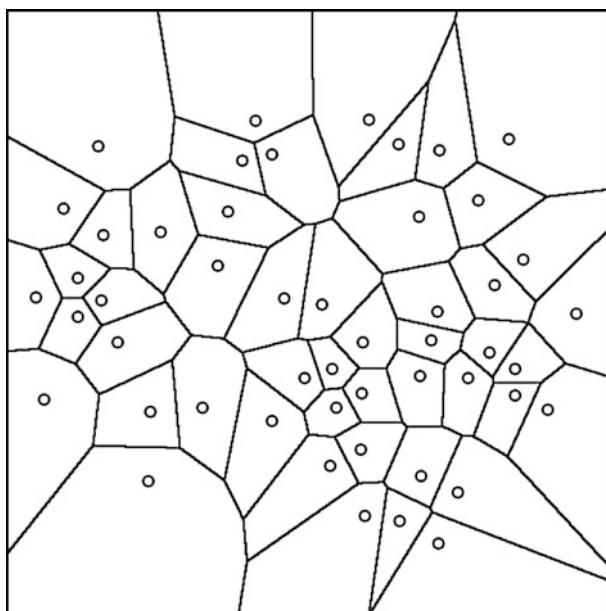
5.1 Voronoi Diagram and Skeletonization

Let \mathbf{P} be a nonempty finite set of planar points. A planar Voronoi diagram of the set \mathbf{P} is a partition of the plane into such regions, so that for any element of \mathbf{P} a region corresponding to a unique point p contains all those points of the plane which are closer to p than to any other node of \mathbf{P} . A unique region $vor(p) = \{z \in \mathbb{R}^2 : d(p, z) < d(p, m) \forall m \in \mathbf{P}, m \neq p\}$ assigned to point p is called a Voronoi cell of the point p . The boundary of the Voronoi cell of a point p is built of segments of bisectors separating pairs of geographically closest points of the given planar set \mathbf{P} . A union of all boundaries of the Voronoi cells determines the *planar Voronoi diagram*: $VD(\mathbf{P}) = \cup_{p \in \mathbf{P}} \partial vor(p)$. A variety of Voronoi diagrams and algorithms of their construction can be found in Klein (1990). An example of a Voronoi diagram is shown in [Fig. 2](#).

Voronoi cells of a planar set represent the natural or geographical neighborhood of the set's elements. Therefore, the computation of a Voronoi diagram based on the spreading of some "substance" from the data points is usually the first approach of those trying to design massively parallel algorithms in chemical nonlinear systems.

Fig. 2

Voronoi diagram of 50 planar points randomly distributed in the unit disc.



The “spreading substance” can be represented by a potential or an oscillatory field, or diffusing or phase waves, or simply by some traveling inhomogeneities or disturbances in the physical computing medium. This idea was first explored by Blum, Calabi, and Hartnett in their “grass-fire” transformation algorithm (Blum 1967, 1973; Calabi and Hartnett 1968). A fire is started at planar points of a given data set, the fire spreads from the data points on a substrate (that constitutes the computing medium) and the sites where the fire fronts meet represent the edges of the Voronoi diagram of the planar data set.

Quite similarly, to construct the diagram with a potential field one puts the field generators at the data points and then detects sites where two or more fronts of the field waves meet. This technique is employed in the computation of a Voronoi diagram via repulsive potential and oscillatory fields in homogeneous neural networks (Hwang and Ahuja 1992; Lemmon 1991). These approaches are “natural” and intuitive and their simplicity of implementation made them an invaluable tool for massively parallel image processing. The weakness of the approach is that a stationary structure is formed as a result of the computation and the meeting points of the colliding wave fronts must be detected by some “artificial” or extrinsic methods. This disadvantage is eliminated in the reaction–diffusion technique as demonstrated.

To compute a Voronoi diagram of planar points, each point is represented by a unique drop of outer electrolyte. The configuration of the outer electrolyte drops corresponds to the configuration of the data to be subdivided by the Voronoi diagram. Data objects (to be separated) are represented using a clear solution of potassium iodide. The potassium iodide diffuses from the sites of the drops or from the edges of planar shapes into the palladium chloride loaded gel. The potassium iodide reacts with the palladium chloride to form iodo-palladium species. The palladium chloride gel is light yellow colored while iodo-palladium species are dark brown. Thus, during the process of computation, the growth of dark-brown patterns emanating from the initial data sites can be observed. At sites where two or more diffusive fronts meet each other almost no precipitate is formed; these sites therefore remain uncolored, and they represent the bisectors of a Voronoi diagram, generated by the initial geometrical configuration of data objects (☞ Fig. 3a).

Voronoi diagrams of other geometric shapes can be constructed by substituting the drops of the outer electrolyte for pieces of absorbent materials soaked in the electrolyte solution and applying these to the gel surface (☞ Fig. 3b, c).

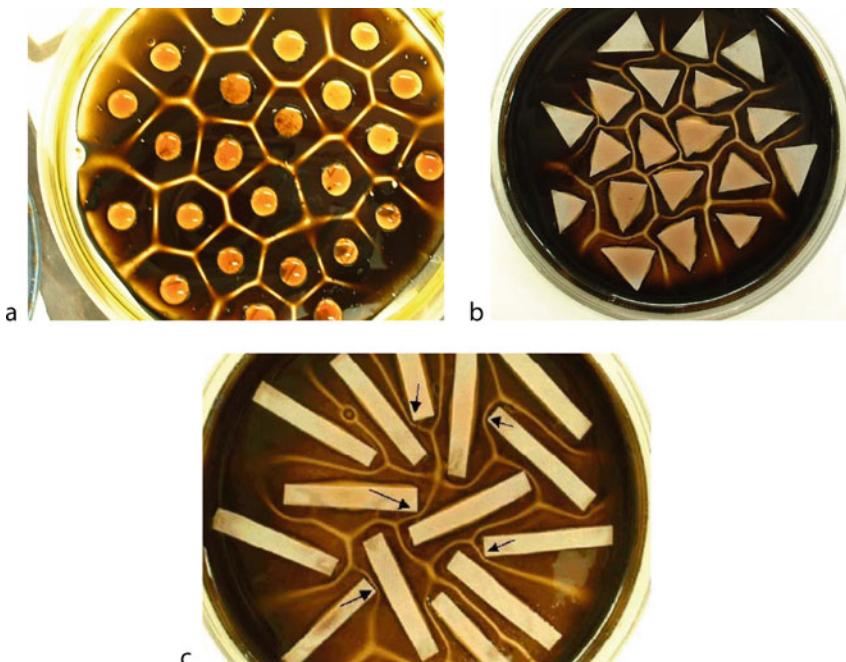
The processor starts its computation as soon as drops/shapes with potassium iodide are applied; the computation is finished when no more precipitate is formed. Configurations of drops/filter-paper templates of potassium iodide correspond to the input states of the processor; the resulting two-dimensional concentration profile of the precipitate (iodo-palladium species) is the output state of the processor. The processor can be seen as a preprogrammed or hardwired device because the way in which the chemical species react cannot be changed.

It should be noted that the “palladium processor” is just one of a huge number of chemical-based processors capable of Voronoi diagram calculation. Some others, particularly those based on gels containing potassium ferrocyanide and mixed with various metal salts, are discussed in Adamatzky et al. (2005b). Reagents with differing diffusion on the same substrate and differing chemical reactivities to the gel substrate can be used to form a number of generalized weighted Voronoi diagrams.

It is also interesting to note that in these chemical systems, the more data inputted in a given area the faster the computational outcome. The limit of the processors is currently determined by the gel thickness and the integrity and the ability to accurately input data points – but these are just design issues that can realistically be overcome and do not limit the

Fig. 3

Voronoi diagram of planar points/discs (a), triangles (b) and rectangles (c) computed in the palladium processor (from Adamatzky et al. 2005b).



theoretical capabilities of the systems. Also, these systems can be practically used to implement three-dimensional Voronoi tessellations. For example, algorithms of classical computation of Voronoi diagram of planar points and planar objects are very different because when arbitrary-shaped objects are involved, one needs to employ algebraic curves to produce bisectors, which increases complexity. In chemical processors, it does not matter what shape data objects have, the time of computation is determined only by the maximum distance between two geographically neighboring objects.

Skeletonization ([Fig. 4](#)) of the planar shapes in a reaction-diffusion chemical processor is yet further proof of the correctness of the proposed mechanism, namely, the wave-velocity-dependent bisector formation. During skeletonization, there are no interactions of wave fronts emanating from different sources; there is only one front, generated at the edges of the closed planar shape. Concave parts of the shape produce bisectors; however, the further the bisecting sites are from the given shape the larger the width of the bisectors and the more precipitate they contain (this is because the velocity of the fronts corresponding to the concave parts reduces with time) (Adamatzky and De Lacy Costello [2002c](#)).

5.2 Collision-Free Path Calculation

A BZ parallel processor (Adamatzky and De Lacy Costello [2002a](#)) that computes a collision-free path solves the following problem. Given a space with a set of obstacles and two selected

Fig. 4

Computation of the skeleton of a planar shape in the palladium processor (from Adamatzky and De Lacy Costello 2002c).



sites, find the shortest path between the source and the destination such that every site of the path is as far from the obstacles as possible. This is not only a classical problem in mathematical optimization but one heavily relied on in robotics, logistics, electronic design, etc.

The problem has already been tackled in a framework of wave-based computing. Three experimental prototypes exist, where BZ processors have been designed to compute the shortest paths:

- extraction of an optimal path in a labyrinth from excitation wave-front dynamics in the labyrinth (Steinbock et al. 1995); the path is extracted from time-lapsed snapshots of the wave-front motion;
- collision-free path using the BZ medium, where obstacles are represented by drops of KCl or strong illumination (Agladze et al. 1997); a path is extracted from the motion of excitation waves;
- approximation of a path on a tree in a light-sensitive BZ-medium (Rambidi and Yakovenchuk 2001); this technique is rather complicated because a data tree is represented by gradients of the medium and the image-processing routines were implemented at every step of the medium's development.

A BZ processor coupled with a two-dimensional cellular automaton has been designed, the hybrid system computes the shortest path by first approximating a distance field generated by the obstacles (BZ medium) and then calculating the shortest path in the field (cellular automaton) (Adamatzky and De Lacy Costello 2003).

A configuration of obstacles (☞ Fig. 5a) is represented by an identical configuration of silver wires parallel to each other and perpendicular to the surface of the BZ medium. Each circular obstacle is represented by a unique wire positioned exactly at the center of the obstacle. To start the computation in the BZ processor, the tips of all the wires are briefly immersed into the BZ mixture (in the reduced steady state). Immersing silver wires at specific points of the BZ mixture reversibly removes bromide ions from these sites. Bromide ions act

Fig. 5

Converting obstacles to excitations: (a) mapping experimental arena (30 m in diameter) to Belousov–Zhabotinsky medium (9 cm in diameter). (b) Approximation of a distance field (from Adamatzky and De Lacy Costello 2003).

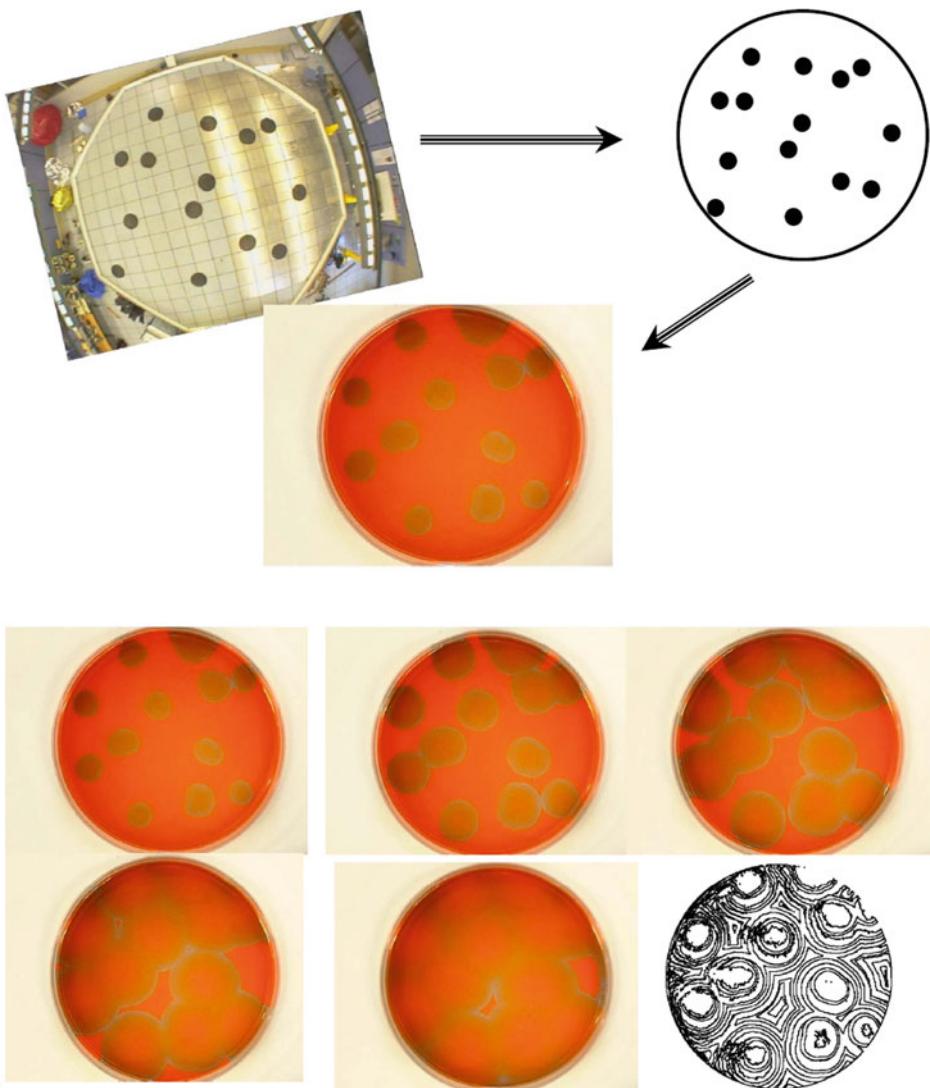
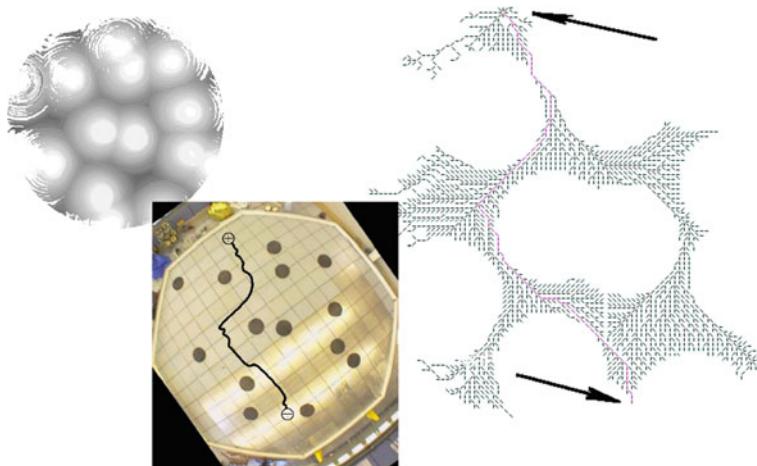


Fig. 6

Extraction of the shortest collision-free path (from Adamatzky et al. 2005b).



as primary inhibitors of the autocatalytic reaction in the BZ medium, therefore local removal of bromide ions stimulates wave generation. The digital images of the BZ medium were taken at certain intervals until all primary wave fronts had been annihilated (► Fig. 5b). Then we could transform a series of color snapshots taken to a gray-level matrix, which represents a distance scalar field derived from a configuration of obstacles (► Fig. 6, on the left).

The distance matrix is mapped to a two-dimensional excitable cellular automaton, which calculates the shortest path between any two points of the experimental arena, labeling the path with the local pointers (► Fig. 6, on the right). The configuration of the local pointers, representing the shortest collision-free path, can be communicated to a mobile robot that navigates the experimental arena (► Fig. 6, center).

5.3 Taxis

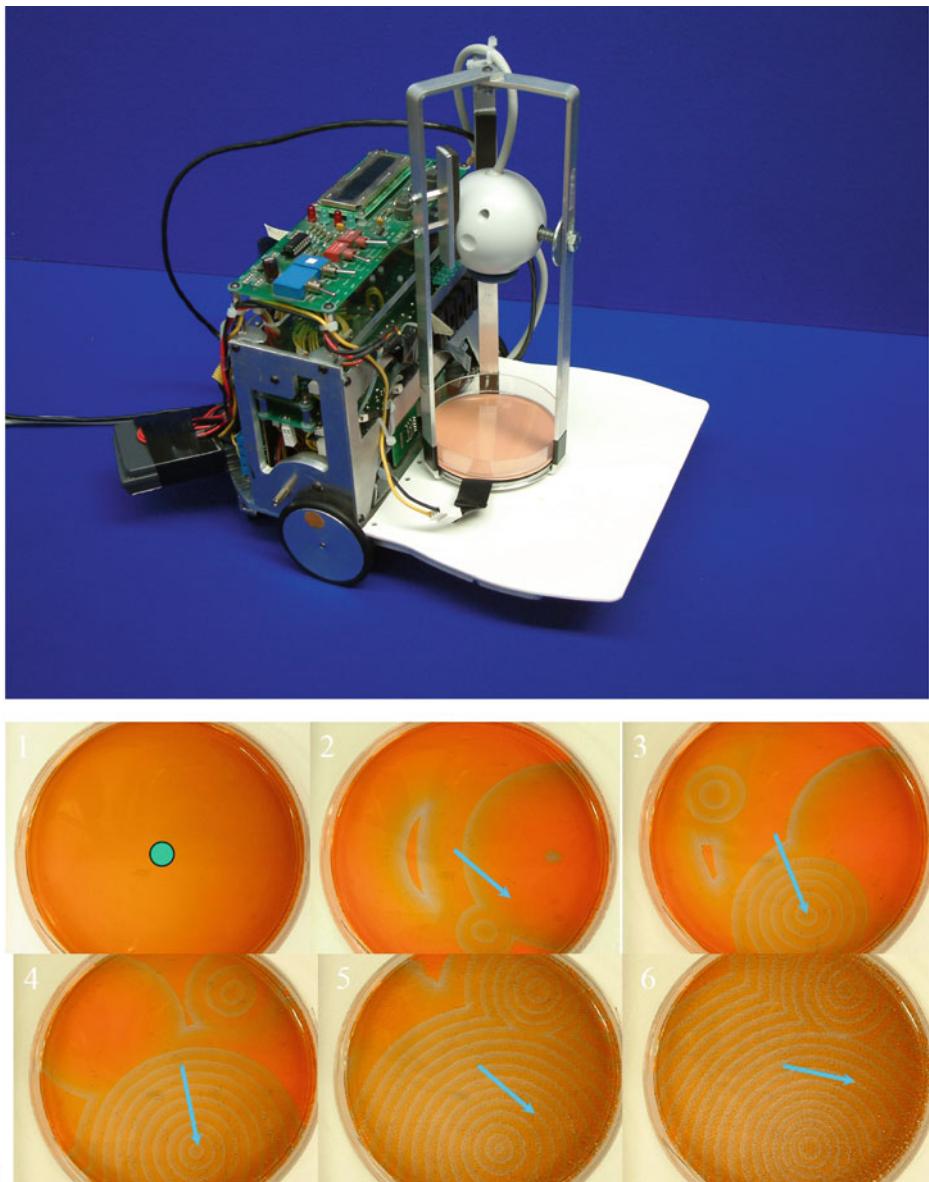
The Belousov–Zhabotinsky reaction is capable of assisting in the navigation of a mobile robot in a real-life environment (Adamatzky et al. 2004). Given an onboard thin-layer chemical reactor with the liquid-phase Belousov–Zhabotinsky medium, one can apply intermittent stimulation of the medium to sensibly guide the robot.

To make a BZ robotic taxis controller, a thin layer Belousov–Zhabotinsky (BZ) reaction is prepared using a recipe adapted from Field and Winfree (1979). Excitation waves in the BZ reaction are initiated using a silver colloid solution, which is added to the medium by a human operator. The chemical controller is placed onboard a wheeled mobile robot (► Fig. 7a). The robot is about 23 cm in diameter and able to turn on the spot; wheel motors are controlled by a Motorola 68332 onboard processor. The robot features a horizontal platform, where the Petri dish (9 cm in diameter) is fixed, and a stand with a digital camera Logitech QuickCam (in 120 × 160 pixels resolution mode), to record excitation dynamics. The robot controller and the camera are connected to a PC via serial port RS-232 and USB 1.0, respectively.

Because vibrations affect processes in BZ reaction systems, the movements of the robot are made as smooth as possible, thus in the experiments the robot moves with a speed

Fig. 7

Wheeled robot controlled by BZ medium (a) and snapshots of excitation dynamics with robot's velocity vector extracted (b) (from Adamatzky et al. 2004).



of circa 1 cm s^{-1} . It was found that rotational movements were particularly disruptive and caused spreading of the excitation waves from the site of the existing wave fronts faster than would be seen if the process was just diffusion. Therefore, to minimize this effect, the robot was made to rotate with a very low speed of around 1°s^{-1} .

To enhance the images of the excitation dynamics, the Petri dish is illuminated from underneath by a flexible electroluminescent sheet (0.7 mm thick) cut to the shape of the dish. The sheet, powered through an inverter from the robot's batteries, produces a cool uniform blue light not affecting (at least in conditions of the experimental setup) the physicochemical processes in the reactor. It was found preferable to implement experiments in dark areas to avoid interference of daylight and lamps with the light from the luminescent sheet.

When the medium is excited with a silver wire, a local generator of target waves is formed (Fig. 7b). The robot extracts the position of the stimulation point, relative to the center of the chemical reactor, from the topology of the excitation target waves. The robot adjusts its velocity vector to match the vector toward the source of stimulation. To guide the robot more precisely one can excite the chemical medium in several points (Fig. 7b). It should be noted that eventually it was intended that environmental interaction would become the source of stimulation for the excitation waves. The forced stimulation was simply to test the algorithms on board with real chemical controllers and to learn about any problems coupling chemical reactors and robots. Something that was identified as a problem, physical disruption of chemical waves, could actually turn out to be advantageous as it may feed back both direction, velocity, and even rotational movement of the robot.

5.4 Control of Robotic Hand

In the previous sections, we described the designs of chemical controllers for robots, which can calculate a shortest collision-free path in the robotic arena and guide the robot toward the source of stimulation (taxis). However, the controllers described lacked feedback from the robot. In a set of remarkable experiments undertaken by Hiroshi Yokoi and Ben De Lacy Costello (Yokoi et al. 2004), it was demonstrated that when a closed-loop interface between the chemical controller and the robot is established, the behavior of the hybrid systems becomes even more intriguing.

In the chemical controller of the robotic hand (Yokoi et al. 2004) the excitation waves propagating in the BZ reactor are sensed by photodiodes, which in turn trigger finger motion. When the bending fingers touch the chemical medium glass, “nails” fitted with capillary tubes release small quantities of colloidal silver into the solution and this triggers additional circular waves in the medium (Adamatzky et al. 2005b) (Fig. 8). Starting from any initial configuration, the chemical-robotic system always reaches a coherent activity mode, where fingers move in regular, somewhat melodic patterns, and just a few generators of target waves govern the dynamics of the excitation in the reactor (Yokoi et al. 2004).

5.5 Parallel Actuators

How can a reaction-diffusion medium manipulate objects? To find out the answer, a simulated abstract parallel manipulator (Adamatzky et al. 2005a) can be coupled with an experimental Belousov-Zhabotinsky (BZ) chemical medium. The simulated manipulator is a two-dimensional array of actuating units, each unit can apply a force vector of unit length to the manipulated objects. The velocity vector of the object is derived via the integration of the results of all local force vectors acting on the manipulated object.

Fig. 8

Robotic hand interacts with Belousov–Zhabotinsky medium (from Yokoi et al. 2004).



Coupling an excitable chemical medium with a manipulator allows one to convert experimental snapshots of the BZ medium to a force vector field and then simulate the motion of manipulated objects in the force field, thus achieving reaction–diffusion medium controlled actuation. To build an interface between the recordings of space–time snapshots of the excitation dynamics in the BZ medium and simulated physical objects, one can calculate the force fields generated by mobile excitation patterns and then simulate the behavior of an object in this force field (Skachek et al. 2005).

The chemical medium used to perform the actuation is prepared following the typical recipe, see Field and Winfree (1979), based on a ferroin catalyzed BZ reaction. A silica gel plate is cut and soaked in a ferroin solution. The gel sheet is placed in a Petri dish and the BZ solution is added. The dynamics of the chemical system is recorded at 30 s intervals using a digital camera.

The cross-section profile of the BZ wave-front recorded on a digital snapshot shows a steep rise of red color values in the pixels at the wave-front's head and a gradual descent in the pixels along the wave-front's tail. Assuming that excitation waves push the object, local force vectors generated at each site – a pixel of the digitized image – of the medium should be oriented along local gradients of the red color values. From the digitized snapshot of the BZ medium, an array of red components is extracted from the snapshot's pixels and then the projection of a virtual vector force at the pixel is calculated.

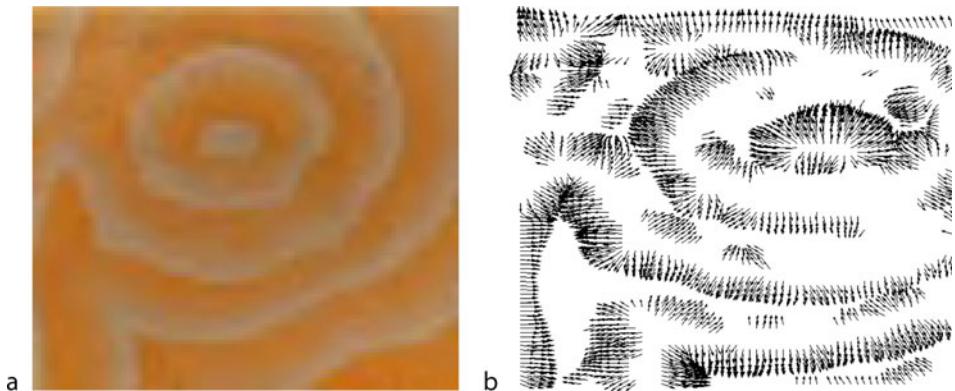
Force fields generated by the excitation patterns in a BZ system (Fig. 9) result in tangential forces being applied to a manipulated object, thus causing translational and rotational motions of the object (Adamatzky et al. 2005a).

It was demonstrated that the BZ medium controlled actuators can be used for sorting and manipulating both small objects, comparable in size to the elementary actuating unit, and larger objects, with lengths of tens or hundreds of actuating units.

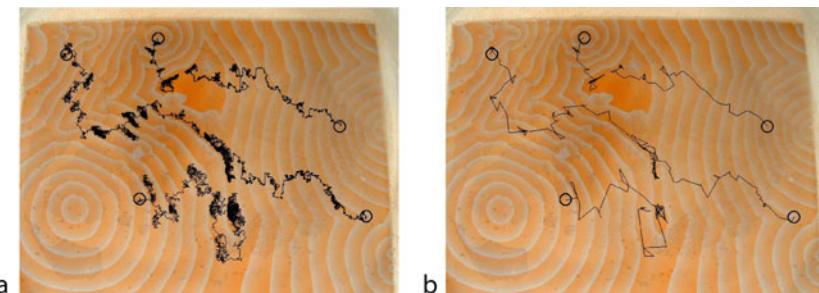
Pixel-objects, due to their small size, are subjected to random forces, caused by impurities of the physical medium and imprecision of the actuating units. In this case, no averaging of forces is allowed and the pixel-objects sensitively react to a single force vector. Therefore, one can adopt the following model of manipulating a pixel-object: If all force vectors at the 8-pixel neighborhood of the current site of the pixel-object are nil then the pixel-object jumps to

Fig. 9

Force vector field (**b**) calculated from BZ medium's image (**a**) (Adamatzky et al. 2005a).

**Fig. 10**

Examples of manipulating five pixel-objects using the BZ medium: (a) trajectories of pixel-objects. (b) Jump-trajectories of pixel-objects recorded every 100th time step. Initial positions of the pixel-objects are shown by circles (Adamatzky et al. 2005a).



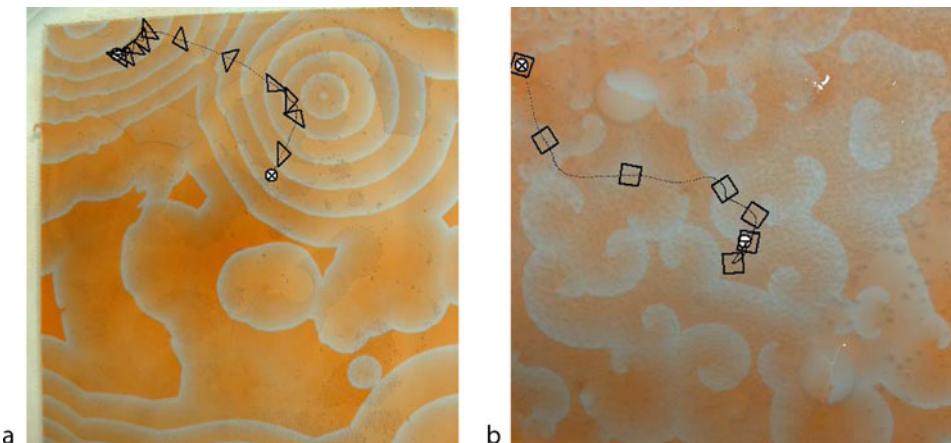
a randomly chosen neighboring pixel of its neighborhood, otherwise the pixel-object is translated by the maximum force vector in its neighborhood.

When placed on the simulated manipulating surface, pixel-objects move at random in the domains of the resting medium; however, by randomly drifting each pixel-object eventually encounters a domain of co-aligned vectors (representing the excitation wave front in the BZ medium) and is translated along the vectors. An example of several pixel-objects transported on a “frozen” snapshot of the chemical medium is shown in [Fig. 10](#). Trajectories of pixel-objects ([Fig. 10a](#)) show distinctive intermittent modes of random motion separated by modes of directed “jumps” guided by traveling wave fronts. Smoothed trajectories of pixel-objects ([Fig. 10b](#)) demonstrate that despite a very strong chaotic component in manipulation, pixel-objects are transported to the sites of the medium where two or more excitation wave fronts meet.

The overall speed of pixel-object transportation depends on the frequency of wave generations by sources of target waves. As a rule, the higher the frequency the faster the objects are transported. This is because in parts of the medium spanned by low-frequency target waves

Fig. 11

Manipulating planar object in BZ medium. (a) Right-angled triangle moved by fronts of target waves. (b) Square object moved by fronts of fragmented waves in sub-excitatory BZ medium. Trajectories of center of mass of the square are shown by the dotted line. Exact orientation of the objects is displayed every 20 steps. Initial position of the object is shown by \ominus and the final position by \otimes (Adamatzky et al. 2005a).



there are lengthy domains of resting states, where no force vectors are formed. Therefore, a pixel-sized object can wander randomly for a long time until climbing the next wave front (Adamatzky et al. 2005a).

Spatially extended objects follow the general pattern of motion observed for the pixel-sized objects. However, due to the integration of many force vectors, the motion of planar objects is smoother and less sensitive to the orientation of any particular force vector.

The outcome of the manipulation depends on the size of the object, with increasing size of the object – due to larger numbers of local vector forces acting on the object – the objects become more controllable by the excitation wave fronts (Fig. 11).

6 Universal Processors

Certain families of thin-layer reaction–diffusion chemical media can implement sensible transformation of the initial (data) spatial distribution of the chemical species concentrations to the final (result) concentration profile (Adamatzky 2001; Sienko et al. 2003). In these reaction–diffusion computers, computation is realized via the spreading and interaction of diffusive or phase waves. Specialized, intended to solve a particular problem, experimental chemical processors implement basic operations of image processing (Kuhnert 1986b; Rambidi 1998; Rambidi et al. 2002; Adamatzky et al. 2005b), computation of optimal paths (Steinbock et al. 1995; Agladze et al. 1997; Adamatzky et al. 2005b), and control of mobile robots (Adamatzky et al. 2005b).

A device is called computationally universal if it implements a functionally complete system of logical gates, for example, a tuple of negation and conjunction, in its space–time dynamics.

A number of computationally universal reaction–diffusion devices were implemented, the findings include logical gates (Tóth and Showalter 1995; Sielewiesiuk and Gorecki 2001) and diodes (Kusumi et al. 1997; Dupont et al. 1998; Motoike and Yoshikawa 1999) in the Belousov–Zhabotinsky (BZ) medium, and an XOR gate in a palladium processor (Adamatzky and De Lacy Costello 2002b).

So far, most known experimental prototypes of reaction–diffusion processors exploit the interaction of wave fronts in a geometrically constrained chemical medium. The computation is based on a stationary architecture of the medium’s inhomogeneities. Constrained by stationary wires and gates, reaction–diffusion chemical universal processors provide little computational novelty and no dynamical reconfiguration ability because they simply imitate the architectures of conventional silicon computing devices. To appreciate in full the inherent massive-parallelism of thin-layer chemical media and to free the chemical processors from the imposed limitations of fixed computing architectures, an unconventional paradigm of architecture-less, or collision-based, computing has been adopted. An architecture-based, or stationary, computation implies that a logical circuit is embedded into the system in such a manner that all elements of the circuit are represented by the system’s stationary states. The architecture is static. If there is any kind of “artificial” or “natural” compartmentalization, the medium is classified as an architecture-based computing device. Personal computers, living neural networks, cells, and networks of chemical reactors are typical examples of architecture-based computers.

A collision-based, or dynamical, computation employs mobile compact finite patterns, mobile self-localized excitations or simply localizations, in active nonlinear medium. The essentials of collision-based computing are as follows.

Truth values of logical variables are given by either the absence or presence of the localizations or other parameters of the localizations. The localizations travel in space and perform computation when they collide with each other. There are no predetermined stationary wires, a trajectory of the traveling pattern is a momentary wire. Almost any part of the medium space can be used as a wire. Localizations can collide anywhere within the overall space of the medium, there are no fixed positions at which specific operations occur, nor location-specified gates with fixed operations. The localizations undergo transformations, form bound states, annihilate, or fuse when they interact with other mobile patterns. The information values of the localizations are transformed as a result of the collisions and thus a computation is implemented (Adamatzky et al. 2005b).

The paradigm of collision-based computing originates from the technique of proving the computational universality of the Game of Life (Berlekamp et al. 1982), conservative logic and the billiard-ball model (Fredkin and Toffoli 1982) and their cellular automaton implementations (Margolus 1984).

Solitons, defects in tubulin microtubules, excitons in Scheibe aggregates, and breathers in polymer chains are most frequently considered candidates for the role of information carrier in nature-inspired collision-based computers, see the overview in Adamatzky (2001). It is experimentally difficult to reproduce all these artifacts in natural systems; therefore, the existence of mobile localizations in an experiment-friendly chemical media opens new horizons for the fabrication of collision-based computers.

The basis for the material implementation of the collision-based universality of reaction–diffusion chemical media was discovered by Sendiña-Nadal et al. (2001). They experimentally proved the existence of localized excitations – traveling wave fragments that behave like quasiparticles – in a photosensitive sub-excitable Belousov–Zhabotinsky medium.

6.1 Basic Gates in Excitable Chemical Medium

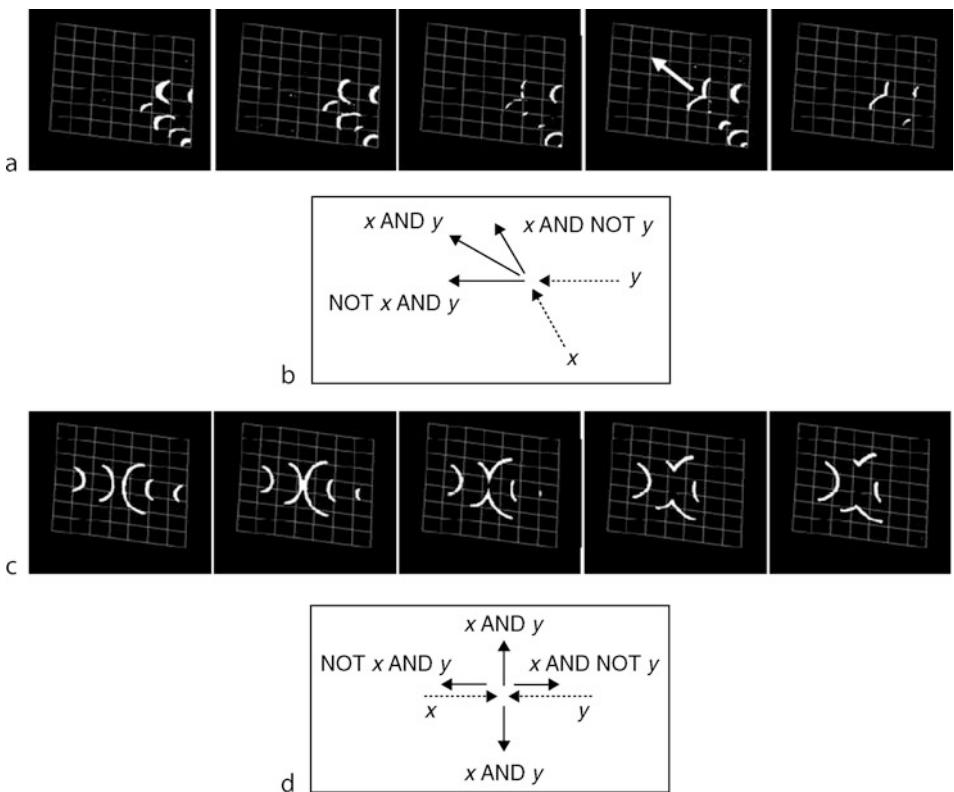
Most collisions between wave-fragments in a sub-excitatory BZ medium follow the basic rules of the Fredkin–Toffoli billiard ball model but not always conservatively. In most cases, particularly when chemical laboratory experiments are concerned, the wave-fragments do not simply scatter as a result of collisions but rather fuse, split or generate additional new mobile localizations. Two examples of logical gates realized in the BZ medium are shown in [Fig. 12](#).

In the first example ([Fig. 12a, b](#)) the wave-fragment traveling north-west collides with the wave-fragment traveling west, a new wave-fragment traveling north-west-west is produced as a result of the collision ([Fig. 12a](#)). This new wave-fragment represents the conjunction of the Boolean variables represented by the colliding wave-fragments ([Fig. 12b](#)).

In the second example of experimental implementation ([Fig. 12c, d](#)), one can see two wave-fragments, traveling east and west, colliding “head-on.” Two new wave-fragments are produced as a result of the collision. One wave-fragment travels north, another wave-fragment travels south. The corresponding logical gate is shown in [Fig. 12d](#).

Fig. 12

Two logical gates implemented in experimental laboratory conditions (Toth et al. 2007). Time lapsed snapshots of propagating wave-fragments are shown in (a) and (c), schemes of the logical gates are shown in (b) and (d).



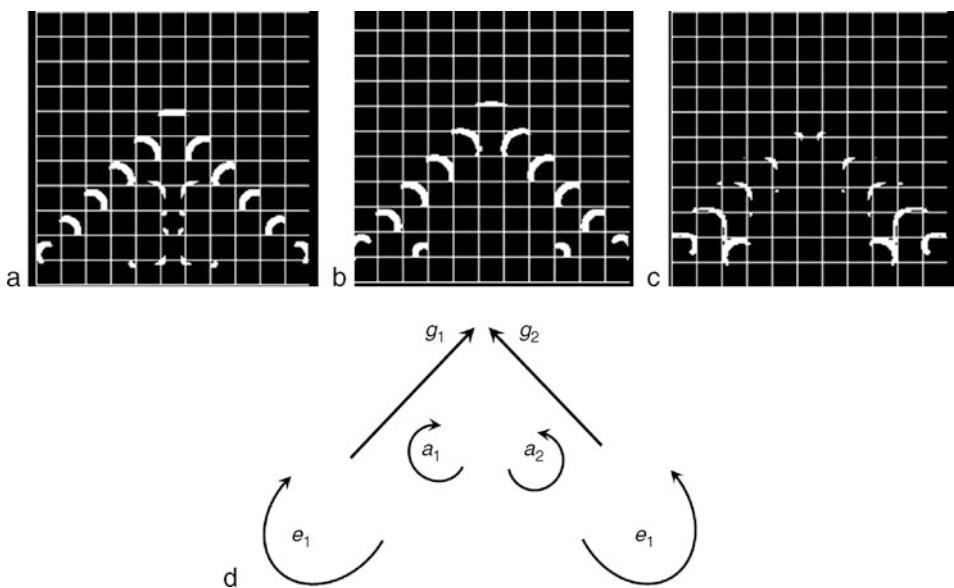
6.2 Constant Truth Generators

In the previous section, the design of logical gates in theoretical and experimental implementations of a sub-exitable BZ medium was demonstrated. The negation operator was the only thing missing in the construction, without negation and just one AND or OR gate a functional completeness could not be achieved. In cellular automata models of collision-based computers, see overview in Adamatzky (2002), negation is implemented with the help of glider guns – an autonomous generator of gliders. Streams of gliders emitted by the glider gun are collided with streams of data gliders to perform the negation operation. Until recently, there was no experimental proof that this missing component of current collision-based circuits could be obtained via experiments with the BZ medium. The gap was filled in Adamatzky et al. (2008), analogues of glider guns were realized in chemical laboratory conditions.

To produce the guns, a BZ light-sensitive sub-exitable processor was used with a slight modification, see Adamatzky et al. (2008) for details. A controllable regular configuration of excitable and non-exitable cells is created by projecting a 10×10 checkerboard pattern comprising low (0.394 mW cm^{-2}) and high (9.97 mW cm^{-2}) light intensity cells onto the gel surface using a data projector. In the checkboard pattern, the wave propagation depends on junctions between excitable and non-exitable cells. Junction permeability is affected by the excitability of the cell, the excitability of the neighboring cells, the size and symmetry of the network, the curvature of the wave fragments in the excitable cells, and the previous states of the junctions. When a junction fails, it acts as an impurity in the system, which leads to the

Fig. 13

Two glider guns where streams of fragments are in collision. (a–c) Snapshots of excitation dynamics in the BZ medium. (d) Scheme of the glider guns: e_1 and e_2 spiral wave fragments generating streams, g_1 and g_2 of localized excitations, a_1 and a_2 auxiliary wave fragments interacting with the wave streams (from De Lacy Costello et al. 2008).



formation of a spiral wave. In certain conditions a tail of the spiral wave becomes periodically severed. Thus, independently moving localized excitations are produced. The spiral generates trains of wave fragments. An example of two glider guns, generating trains of mobile self-localized excitations, is shown in Fig. 13.

7 Conclusion

Major achievements in the field of reaction-diffusion computing were listed, and using selected examples of chemical laboratory prototypes it was shown that chemical reaction-diffusion computers can solve a variety of tasks. They can perform image processing, calculate Voronoi diagrams of arbitrary-shaped planar objects and skeletons of planar shapes, approximate collision-free shortest paths, guide robots toward sources of stimulation, manipulate several small objects in parallel, and implement logical functions. Recipes for the preparation of precipitating and excitable chemical media capable of carrying out such computations were provided. It is hoped that this chapter is sufficient in order for the reader to obtain a basic understanding of reaction-diffusion computing; however, readers inspired to discover and design their own chemical reaction-diffusion computers are advised to consult a more specific and thus more detailed text (Adamatzky et al. 2005b).

References

- Adamatzky A (1994) Reaction-diffusion algorithm for constructing discrete generalized Voronoi diagram. *Neural Netw World* 6:635–643
- Adamatzky A (1996) Voronoi-like partition of lattice in cellular automata. *Math Comput Modelling* 23:51–66
- Adamatzky A (2001) Computing in nonlinear media and automata collectives. IoP Publishing, Bristol
- Adamatzky A (ed) (2002) Collision-based computing. Springer, London
- Adamatzky A (2004) Collision-based computing in Belousov–Zhabotinsky medium. *Chaos Solitons Fractals* 21:1259–1264
- Adamatzky A, De Lacy Costello BPJ (2002a) Collision-free path planning in the Belousov–Zhabotinsky medium assisted by a cellular automaton. *Naturwissenschaften* 89:474–478
- Adamatzky A, De Lacy Costello BPJ (2002b) Experimental logical gates in a reaction-diffusion medium: the XOR gate and beyond. *Phys Rev E* 66:046112
- Adamatzky A, De Lacy Costello BPJ (2002c) Experimental reaction-diffusion pre-processor for shape recognition. *Phys Lett A* 297:344–352
- Adamatzky A, De Lacy Costello BPJ (2007) Binary collisions between wave-fragments in a sub-excitable Belousov–Zhabotinsky medium. *Chaos Solitons Fractals* 34:307–315
- Adamatzky A, Teuscher C (Eds) (2006) From utopian to genuine unconventional computers. Luniver Press, Beckington, UK
- Adamatzky A, Tolmachiev D (1997) Chemical processor for computation of skeleton of planar shape. *Adv Mater Opt Electron* 7:135–139
- Adamatzky A, De Lacy Costello B, Melhuish C, Ratcliffe N (2003) Experimental reaction-diffusion chemical processors for robot path planning. *J Intell Robot Syst* 37:233–249
- Adamatzky A, De Lacy Costello B, Melhuish C, Ratcliffe N (2004) Experimental implementation of mobile robot taxis with onboard Belousov–Zhabotinsky chemical medium. *Mater Sci Eng C* 24:541–548
- Adamatzky A, De Lacy Costello B, Skacheck S, Melhuish C (2005a) Manipulating objects with chemical waves: open loop case of experimental Belousov–Zhabotinsky medium. *Phys Lett A*
- Adamatzky A, De Lacy Costello B, Asai T (2005b) Reaction diffusion computers. Elsevier, New York
- Adamatzky A, Bull L, De Lacy Costello B, Stepney S, Teuscher C (eds) (2007) Unconventional computing 2007. Luniver, Beckington, UK
- Agladze K, Obata S, Yoshikawa K (1995) Phase-shift as a basis of image processing in oscillating chemical medium. *Physica D* 84:238–245

- Agladze K, Aliev RR, Yamaguchi T, Yoshikawa K (1996) Chemical diode. *J Phys Chem* 100:13895–13897
- Agladze K, Magome N, Aliev R, Yamaguchi T, Yoshikawa K (1997) Finding the optimal path with the aid of chemical wave. *Physica D* 106:247–254
- Akl SG, Calude CS, Dinneen MJ, Rozenberg G (2007) In: Unconventional computation: 6th international conference, Kingston, Canada, August 2007. Lecture notes in computer science, vol 4618
- Berlekamp ER, Conway JH, Guy RL (1982) Winning ways for your mathematical plays, vol 2. Academic Press, New York
- Blum H (1967) A transformation for extracting new descriptors of shape. In: Wathen-Dunn W (ed) Models for the perception of speech and visual form. MIT Press, Cambridge, MA, pp 362–380
- Blum H (1973) Biological shape and visual science. *J Theor Biol* 38:205–287
- Calabi L, Hartnett WE (1968) Shape recognition, prairie fires, convex deficiencies and skeletons. *Am Math Mon* 75:335–342
- Courant R, Robbins H (1941) What is mathematics? Oxford University Press, New York
- De Lacy Costello BPJ (2003) Constructive chemical processors – experimental evidence that shows that this class of programmable pattern forming reactions exist at the edge of a highly non-linear region. *Int J Bifurcat Chaos* 13:1561–1564
- De Lacy Costello BPJ, Adamatzky A (2003) On multi-tasking in parallel chemical processors: experimental findings. *Int J Bifurcat Chaos* 13:521–533
- De Lacy Costello BPJ, Hantz P, Ratcliffe NM (2004b) Voronoi diagrams generated by regressing edges of precipitation fronts. *J Chem Phys* 120 (5):2413–2416
- De Lacy Costello BPJ, Adamatzky A, Ratcliffe NM, Zanin A, Purwins HG, Liehr A (2004a) The formation of Voronoi diagrams in chemical and physical systems: experimental findings and theoretical models. *Int J Bifurcat Chaos* 14(7):2187–2210
- De Lacy Costello B, Toth R, Stone C, Adamatzky A, Bull L (2008) Implementation of glider guns in the light-sensitive Belousov–Zhabotinsky medium. *Phys Rev E* 79:026114
- Dupont C, Agladze K, Krinsky V (1998) Excitable medium with left-right symmetry breaking. *Physica A* 249:47–52
- Field RJ, Winfree AT (1979) Travelling waves of chemical activity in the Zaikin–Zhabotinsky–Winfree reagent. *J Chem Educ* 56:754
- Fredkin F, Toffoli T (1982) Conservative logic. *Int J Theor Phys* 21:219–253
- Fuerstman MJ, Deschatelets P, Kane R, Schwartz A, Kenis PJA, Deutch JM, Whitesides GM (2003) *Langmuir* 19:4714
- Hwang YK, Ahuja N (1992) A potential field approach to path planning. *IEEE Trans Robot Autom* 8:23–32
- Gorecka J, Gorecki J (2003) T-shaped coincidence detector as a band filter of chemical signal frequency. *Phys Rev E* 67:067203
- Gorecki J, Yoshikawa K, Igarashi Y (2003) On chemical reactors that can count. *J Phys Chem A* 107: 1664–1669
- Gorecki J, Gorecka JN, Yoshikawa K, Igarashi Y, Nagahara H (2005) *Phys Rev E* 72:046201
- Ichino T, Igarashi Y, Motoike IN, Yoshikawa K (2003) Different operations on a single circuit: field computation on an excitable chemical system. *J Chem Phys* 118:8185–8190
- Klein R (1990) Concrete and abstract Voronoi diagrams. Springer, Berlin
- Kuhnert L (1986b) Photochemische manipulation von chemischen Wellen. *Naturwissenschaften* 76:96–97
- Kuhnert L (1986a) A new photochemical memory device in a light sensitive active medium. *Nature* 319:393
- Kuhnert L, Agladze KL, Krinsky VI (1989) Image processing using light-sensitive chemical waves. *Nature* 337:244–247
- Kusumi T, Yamaguchi T, Aliev R, Amemiya T, Ohmori T, Hashimoto H, Yoshikawa K (1997) Numerical study on time delay for chemical wave transmission via an inactive gap. *Chem Phys Lett* 271:355–360
- Leemmon MD (1991) 2-degree-of-freedom robot path planning using cooperative neural fields. *Neural Comput* 3:350–362
- Margolus N (1984) Physics-like models of computation. *Physica D* 10:81–95
- Mills J (2008) The nature of the extended analog computer. In: Teuscher C, Nemenman IM, Alexander FJ (eds) *Physica D Special issue: Novel Comput Paradigms Quo Vadis*. *Physica D* 237:1235–1256
- Motoike IN, Adamatzky A (2004) Three-valued logic gates in reaction-diffusion excitable media. *Chaos Solitons Fractals* 24:107–114
- Motoike IN, Yoshikawa K (1999) Information operations with an excitable field. *Phys Rev E* 59:5354–5360
- Motoike IN, Yoshikawa K (2003) Information operations with multiple pulses on an excitable field. *Chaos Solitons Fractals* 17:455–461
- Motoike IN, Yoshikawa K, Iguchi Y, Nakata S (2001) Real-time memory on an excitable field. *Phys Rev E* 63:036220
- Nakagaki T, Yamada H, Toth A (2001) *Biophys Chem* 92:47
- Rambidi NG (1997) Biomolecular computer: roots and promises. *Biosyst* 44:1–15
- Rambidi NG (1998) Neural network devices based on reaction-diffusion media: an approach to artificial retina. *Supramol Sci* 5:765–767

- Rambidi NG (2003) Chemical-based computing and problems of high computational complexity: the reaction-diffusion paradigm. In: Seinko T, Adamatzky A, Rambidi N, Conrad M (eds) Molecular computing. MIT Press, Cambridge, MA
- Rambidi NG, Yakovenchuk D (2001) Chemical reaction-diffusion implementation of finding the shortest paths in a labyrinth. *Phys Rev E* 63:026607
- Rambidi NG, Shamayaev KR, Peshkov G Yu (2002) Image processing using light-sensitive chemical waves. *Phys Lett A* 298:375–382
- Saltenis V (1999) Simulation of wet film evolution and the Euclidean Steiner problem. *Informatica* 10:457–466
- Sendiña-Nadal I, Mihaliuk E, Wang J, Pérez-Muñozuri V, Showalter K (2001) Wave propagation in subexcitable media with periodically modulated excitability. *Phys Rev Lett* 86:1646–1649
- Shirakawa T, Adamatzky A, Gunji Y-P, Miyake Y (2009) On simultaneous construction of Voronoi diagram and Delaunay triangulation by Physarum polycephalum. *Int J Bifurcat Chaos* 19(9):3109–3117
- Sielewiesiuk J, Gorecki J (2001) Logical functions of a cross junction of excitable chemical media. *J Phys Chem A* 105:8189–8195
- Seinko T, Adamatzky A, Rambidi N, Conrad M (eds) (2003) Molecular computing. MIT Press, Cambridge, MA
- Skachek S, Adamatzky A, Melhuish C (2005) Manipulating objects by discrete excitable media coupled with contact-less actuator array: open-loop case. *Chaos Solitons Fractals* 26:1377–1389
- Steinbock O, Tóth A, Showalter K (1995) Navigating complex labyrinths: optimal paths from chemical waves. *Science* 267:868–871
- Steinbock O, Kettunen P, Showalter K (1996) *J Phys Chem* 100(49):18970
- Tolmachev D, Adamatzky A (1996) Chemical processor for computation of Voronoi diagram. *Adv Mater Opt Electron* 6:191–196
- Toth R, Stone C, Adamatzky A, de Lacy Costello B, Bull L (2009) Experimental validation of binary collisions between wave-fragments in the photosensitive Belousov-Zhabotinsky reaction. *Chaos Solitons Fractals* 41(4):1605–1615
- Tóth A, Showalter K (1995) Logic gates in excitable media. *J Chem Phys* 103:2058–2066
- Yokoi H, Adamatzky A, De Lacy Costello B, Melhuish C (2004) Excitable chemical medium controlled by a robotic hand: closed loop experiments. *Int J Bifurcat Chaos* 14:3347–3354
- Zaikin AN, Zhabotinsky AM (1970) Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. *Nature* 225:535

57 Rough–Fuzzy Computing

Andrzej Skowron

Institute of Mathematics, Warsaw University, Poland

skowron@mimuw.edu.pl

1	<i>Introduction</i>	1922
2	<i>Rough Sets</i>	1924
3	<i>Fuzzy Sets</i>	1933
4	<i>Rough/Fuzzy Hybridization</i>	1940
5	<i>Wisdom Technology</i>	1942
6	<i>Conclusions</i>	1945

Abstract

In recent years, a rapid growth of interest in rough set theory, fuzzy set theory, and their hybridization and applications has been witnessed worldwide. In this chapter, the basic concepts of rough/fuzzy computing are presented. The role of rough/fuzzy computing in the development of Wisdom Technology (Wistech) is also emphasized.

1 Introduction

Gottfried Wilhelm Leibniz, one of the greatest of mathematicians, discussed *calculi of thoughts*. In particular, he wrote

- ▶ *If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, and say to each other: "Let us calculate."*

(Dissertio de Arte Combinatoria (Leipzig, 1666)).

- ▶ *...Languages are the best mirror of the human mind, and that a precise analysis of the signification of words would tell us more than anything else about the operations of the understanding.*

(New Essays on Human Understanding (1705), translated and edited by Peter Remnant and Jonathan Bennett, Cambridge University Press, 1982).

Only much later was it possible to recognize that new tools are necessary for developing such calculi, e.g., due to the necessity of reasoning under uncertainty about objects and vague concepts. Fuzzy set theory (Zadeh 1965) and rough set theory (Pawlak 1982) represent two complementary approaches to vagueness and, in a more general sense, to imperfect knowledge. Both of these approaches aim to approximate vague concepts. Fuzzy set theory addresses the “gradualness” of knowledge and expresses this in terms of fuzzy membership, whereas rough set theory addresses the granularity of knowledge. This granularity is manifest in the classes (knowledge granules) contained in the partition of each set of sample data defined by the indiscernibility relation.

In the rough/fuzzy approach, computations aim to construct granules satisfying a given specification, often vague and expressed in a natural language, to a satisfactory degree (Polkowski and Skowron 1996; Zadeh 2006). Granular computing (GC), introduced by Zadeh (1973, 1998) (Zadeh 2007; Bargiela and Pedrycz 2003; Pedrycz et al. 2008), may now be regarded as a unified framework for theories, methodologies, and techniques for the modeling of calculi of thoughts based on objects called granules. In the generalized theory of uncertainty, introduced by Zadeh (2006), reasoning under uncertainty is treated as a generalized constraint propagation. The generalized constraint language is used to express propositions, commands, and questions expressed in natural language.

One of the issues discussed in connection with the basic notion of a set is *vagueness*. Mathematics requires that all mathematical notions (including that of a set) must be exact, otherwise precise reasoning would be impossible. However, philosophers (Keefe 2000; Read 1994; Russell 1923; Black 1937) and recently computer scientists (see references in Pawlak and Skowron 2007c) as well as other researchers have become interested in *vague* (imprecise) concepts.

In classical set theory, a set is uniquely determined by its elements. In other words, this means that every element must be uniquely classified as belonging to the set or not. That is

to say, the notion of a set is a *crisp* (precise) one. For example, the set of odd numbers is crisp because every number is either odd or even. In contrast to odd numbers, the notion of a beautiful painting is vague, because all paintings cannot be classified uniquely into two classes: beautiful and not beautiful. For some paintings, it cannot be decided whether they are beautiful or not, and these paintings thus remain in the “doubtful” area. Hence, *beauty* is not a precise concept, but a vague concept. Almost all concepts in natural language are vague. Therefore, commonsense reasoning based on natural language must be based on vague concepts and not on classical logic. An interesting discussion on this issue can be found in, for example, Read (1994).

The idea of vagueness can be traced back to the ancient Greek philosopher Eubulides of Megara (ca. 400 BC) who first formulated the so-called “sorites” (heap) and “falakros” (bald man) paradoxes (see, e.g., Keefe 2000). The bald man paradox goes as follows: suppose a man has 100,000 hairs on his head; removing one hair from his head surely cannot make him bald. Repeating this step, it can be concluded that a man without any hair is not bald. A similar reasoning can be applied to a heap of stones.

Vagueness is usually associated with the boundary region approach (i.e., the existence of objects that cannot be uniquely classified relative to a set or its complement), which was first formulated in 1893 by the father of modern logic, the German logician, Frege (1848–1925) (Frege 1903).

According to Frege, a concept must have a sharp boundary. To a concept without a sharp boundary there would correspond an area that would not have any sharp boundary – i.e., no line all around the concept that clearly distinguishes the objects to which that concept applies. This means that mathematics must use crisp concepts and not vague ones; otherwise it would be impossible to reason precisely.

However, it should be noted that the modern understanding of the notion of a vague (imprecise) concept has a quite firmly established meaning in context, involving the following issues (Keefe 2000):

1. The presence of borderline cases
2. Boundary regions of vague concepts are not crisp
3. Vague concepts are susceptible to sorites paradoxes

Let it be noted that it is usually assumed that the understanding (approximation) of vague concepts (their semantics is determined by the satisfiability relation) depends, e.g., on the agent’s knowledge, which is often changing dynamically. Hence, the approximation of vague concepts by an agent should also be considered changing in time (this is known as concept drift). In the twentieth century, it became obvious that new specialized logic tools needed to be developed to investigate and implement practical problems involving vague concepts. Such tools can be based on rough sets and fuzzy sets. For example, it was shown that the rough set approach makes it possible to deal with vague concepts (see, e.g., Pawlak and Skowron 2007c; Bazan et al. 2006; Skowron 2005).

Vague complex concepts are very often related to one another through a hierarchy induced by the abstraction levels of these concepts. Such hierarchies occur, for instance, when some concepts are components of other concepts. In such a context, there is a great interest in investigating the relation *being a part-of*. This is a different approach from the ontology of modern mathematics (also known as *Cantor ontology*), which is based on the relation *being an element-of*. Such an alternative ontology for mathematics was proposed by Leśniewski (1929) and became the inspiration for an important research area within *rough mereology* (see, e.g., Polkowski and Skowron 1996). Within this approach, the notion of a *rough inclusion relation*

plays a central role. The rough inclusion relation describes to what degree some concepts are parts of other concepts. A rough mereological approach is based on the relation “to be a part of to a degree.”

Certainly, the mereological approach is not the only attempt at establishing links between vague concepts and rough sets. Among others are links based on treating vague concepts by means of logical values. In this case, one can build an algebra of such vague concepts as an algebra of logical values. Usually, this kind of algebra is a pseudo-Boolean algebra and the relationships between vague concepts can be expressed as relationships of logical values of an intermediate logic (see, e.g., Jankowski and Skowron 2008a).

Summing up, vagueness is (1) not allowed in mathematics, (2) interesting for philosophy, and (3) necessary for natural language, cognitive science, artificial intelligence, machine learning, and computer science.

It is worth mentioning that a large amount of knowledge related to natural computing (Rozenberg 2008; Cooper et al. 2008) is expressed in terms of vague concepts. Hence, natural computing seems to be the domain for future potential applications of rough/fuzzy computing.

The remainder of this chapter is structured as follows. ➤ [Section 2](#) presents the basic concepts of rough computing. ➤ [Section 3](#) reports the basic issues of fuzzy computing. Then, the combination of rough computing and fuzzy computing is surveyed in ➤ [Sect. 4](#). Finally, conclusions together with some comments on future applications of rough/fuzzy computing in Wistech (Jankowski and Skowron 2007, 2008a) are presented.

2 Rough Sets

The first paper on rough sets was published by Professor Pawlak in 1982 (Pawlak 1982, 1991). During recent years, numerous publications on the advances of the foundations of fuzzy sets and applications in many different areas have been published (see, e.g., the survey papers, Pawlak and Skowron 2007a, b, c and the bibliography included in these papers, as well as information about recent publications on rough set theory and applications on Web pages, e.g., <http://www.roughsets.org>, <http://logic.mimuw.edu.pl>, <http://rsds.wsiz.rzeszow.pl>).

The rough set philosophy is founded on the assumption that some information (data, knowledge) is associated with every object in the universe of discourse. For example, if objects are patients suffering from a certain disease, symptoms of the disease form information about patients. Objects characterized by the same information are indiscernible (similar) in view of the available information about them. The indiscernibility relation generated in this way is the mathematical basis of rough set theory. This understanding of indiscernibility is based on the idea of Gottfried Wilhelm Leibniz that objects are indiscernible if and only if all available functionals take identical values on them (Leibnizian indiscernibility).

Any set of all indiscernible (similar) objects is called an elementary set and forms a basic granule (atom) of knowledge about the universe. Any union of some elementary sets is referred to as a crisp (precise) set – otherwise the set is rough (imprecise, vague).

Consequently, each rough set has boundary-line cases, i.e., objects that can neither be classified with certainty as members of the set nor of its complement. Obviously, crisp sets have no boundary-line elements at all. This means that boundary-line cases cannot be properly classified by employing the available knowledge.

Thus, the assumption that objects can be *seen* only through the information available about them leads to the view that knowledge has a granular structure. Due to the granularity

of knowledge some objects of interest cannot be discerned and appear as the same (or similar). As a consequence, vague concepts, in contrast to precise concepts, cannot be characterized in terms of information about their elements. Therefore, in the proposed approach, it can be assumed that any vague concept is replaced by a pair of precise concepts – these are called the lower and the upper approximations of the vague concept. The lower approximation consists of all objects that surely belong to the concept and the upper approximation contains all objects that possibly belong to the concept. The difference between the upper and the lower approximation constitutes the boundary region of the vague concept. These upper and lower approximations are two basic operations in rough set theory.

In the following, the basic concepts are presented more formally.

Suppose two finite, nonempty sets U and A are given, where U is the *universe of objects*, and A is a set of *attributes*. The pair (U, A) is called an *information table*. With every attribute $a \in A$, a set V_a of its *values* is associated, called the *domain* of a . Any subset B of A determines a binary relation $I(B)$ on U , called an *indiscernibility relation*, defined by

$$xI(B)y \text{ if and only if } a(x) = a(y) \text{ for every } a \in B \quad (1)$$

where $a(x)$ denotes the value of attribute a for object x .

Obviously, $I(B)$ is an equivalence relation. The family of all equivalence classes of $I(B)$, i.e., the partition determined by B , will be denoted by $U/I(B)$, or simply U/B ; an equivalence class of $I(B)$, i.e., the block of the partition U/B containing x , will be denoted by $B(x)$ (or $[x]_B$).

If $(x, y) \in I(B)$, it can be said that x and y are *B-indiscernible*. Equivalence classes of the relation $I(B)$ (or blocks of the partition U/B) are referred to as *B-elementary sets* or *B-elementary granules*. In the rough set approach, the elementary sets are the basic building blocks (concepts) of our knowledge about reality. The unions of *B-elementary sets* are called *B-definable sets*.

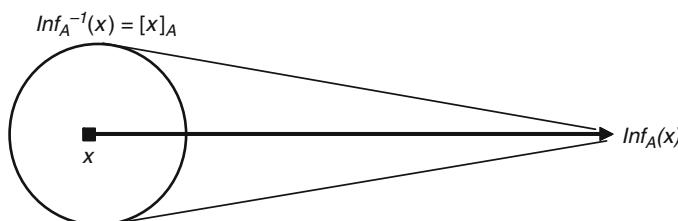
Let $\text{Inf}_B(x)$ denote the *B-signature* of $x \in U$, i.e., the set $\{(a, a(x)) : a \in B\}$. The signature of any object represents incomplete information about this object. The above definition of the indiscernibility relation can be rewritten in the following way:

$$xI(B)y \text{ if and only if } \text{Inf}_B(x) = \text{Inf}_B(y) \quad (2)$$

Hence, one can see that any *B-elementary granule* is defined by the signature of any of its objects, i.e., $B(x) = \text{Inf}_A^{-1}(x)$, where x is an arbitrary object from the granule (see [Fig. 1](#)). Any object from a given elementary granule is labeled by the same signature.

The equality on the right-hand side in the definition in (2) may be substituted by a similarity (tolerance) relation. This leads to the tolerance rough set approach (see, e.g., Skowron and Stepaniuk [1996](#)). It should be noted that the signatures of objects may be incomplete

Fig. 1
Elementary granule.



(e.g., when some values of attributes are missing) or can be influenced by noise. Many papers on rough sets are related to missing values (see the bibliography in Pawlak and Skowron 2007a, c). Recently, it was observed that the signature of objects can be extended to the relevant object contexts by hierarchical modeling (Skowron and Szczuka 2010).

The indiscernibility relation will be further used to define basic concepts of rough set theory. The two operations on sets can now be defined as

$$B_*(X) = \{x \in U : B(x) \subseteq X\} \quad (3)$$

$$B^*(X) = \{x \in U : B(x) \cap X \neq \emptyset\} \quad (4)$$

assigning to every subset X of the universe U two sets, $B_*(X)$ and $B^*(X)$, called the *B-lower* and the *B-upper approximation* of X , respectively. The set

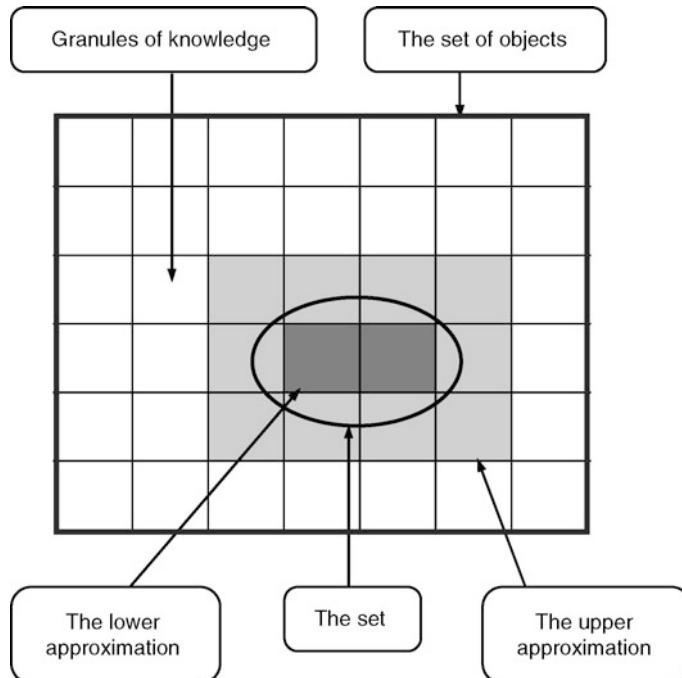
$$BN_B(X) = B^*(X) - B_*(X) \quad (5)$$

will be referred to as the *B-boundary region* of X .

If the boundary region of X is the empty set, i.e., $BN_B(X) = \emptyset$, then the set X is *crisp (exact)* with respect to B ; in the opposite case, i.e., if $BN_B(X) \neq \emptyset$, the set X is referred to as *rough (inexact)* with respect to B .

Approximations are illustrated in  Fig. 2.

 **Fig. 2**
A rough set.



The approximations have the following properties:

$$\begin{aligned}
 & B_*(X) \subseteq X \subseteq B^*(X) \\
 & B_*(\emptyset) = B^*(\emptyset) = \emptyset, B_*(U) = B^*(U) = U \\
 & B^*(X \cup Y) = B^*(X) \cup B^*(Y) \\
 & B_*(X \cap Y) = B_*(X) \cap B_*(Y) \\
 & X \subseteq Y \text{ implies } B_*(X) \subseteq B_*(Y) \text{ and } B^*(X) \subseteq B^*(Y) \\
 & B_*(X \cup Y) \supseteq B_*(X) \cup B_*(Y) \\
 & B^*(X \cap Y) \subseteq B^*(X) \cap B^*(Y) \\
 & B_*(-X) = -B^*(X) \\
 & B^*(-X) = -B_*(X) \\
 & B_*(B_*(X)) = B^*(B_*(X)) = B_*(X) \\
 & B^*(B^*(X)) = B_*(B^*(X)) = B^*(X)
 \end{aligned} \tag{6}$$

A rough set can also be characterized numerically by the following coefficient:

$$\alpha_B(X) = \frac{|B_*(X)|}{|B^*(X)|} \tag{7}$$

called the *accuracy of approximation*, where $|X|$ denotes the cardinality of $X \neq \emptyset$. Obviously, $0 \leq \alpha_B(X) \leq 1$. If $\alpha_B(X) = 1$, then X is *crisp* with respect to B (X is *precise* with respect to B); otherwise, if $\alpha_B(X) < 1$, then X is *rough* with respect to B (X is *vague* with respect to B).

Several generalizations of the classical rough set approach based on approximation spaces defined by (U, R) , where R is an equivalence relation (called indiscernibility relation) in U , have been reported in the literature. Two of the more prominent of these generalizations will now be presented.

A generalized approximation space can be defined by a tuple $AS = (U, I, v)$ where I is the *uncertainty function* defined on U with values in the powerset $P(U)$ of U ($I(x)$ is the *neighborhood* of x) and v is the *rough inclusion function* defined on the Cartesian product $P(U) \times P(U)$ with values in the interval $[0, 1]$ measuring the degree of inclusion of sets. The lower AS_* and upper AS^* approximation operations can be defined in AS by

$$AS_*(X) = \{x \in U : v(I(x), X) = 1\} \tag{8}$$

$$AS^*(X) = \{x \in U : v(I(x), X) = 0\} \tag{9}$$

In the standard case, $I(x)$ is equal to the equivalence class $B(x)$ of the indiscernibility relation $I(B)$; in case of tolerance (similarity) relation $\tau \subseteq U \times U$, $I(x) = \{y \in U : x \tau y\}$, i.e., $I(x)$ is equal to the tolerance class of τ defined by x . The standard rough inclusion relation is defined for $X, Y \subseteq U$ by

$$v(X, Y) = \begin{cases} \frac{|X \cap Y|}{|X|} & \text{if } X \neq \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

For applications, it is important to have some constructive definitions of I and v .

One can consider another way to define $I(x)$. Usually together with AS one can consider some set F of formulae describing sets of objects in the universe U of AS defined by semantics $\|\cdot\|_{AS}$, i.e., $\|\alpha\|_{AS} \subseteq U$ for any $\alpha \in F$. Now, one can take the set

$$N_F(x) = \{\alpha \in F : x \in \|\alpha\|_{AS}\} \quad (11)$$

and $I(x) = \{\|\alpha\|_{AS} : \alpha \in N_F(x)\}$. Hence, more general uncertainty functions having values in $P(P(U))$ can be defined. Usually there are considered families of approximation spaces which are labeled by some parameters. By tuning of such parameters one can search for the optimal, under chosen criteria (e.g., the minimal description length), approximation space for a given concept description.

The approach based on rough inclusion functions has been generalized to the *rough mereological approach* (Polkowski and Skowron, 1996). The inclusion relation $x\mu_r y$, with the intended meaning x is a part of y to a degree at least r , has been taken as the basic notion of rough mereology, being a generalization of the Leśniewski mereology (Leśniewski, 1929). Research on rough mereology has shown the importance of another notion, namely, the *closeness* of complex objects (e.g., concepts). This can be defined by $xcl_{r,r'}y$ if and only if $x\mu_r y$ and $y\mu_{r'}x$.

Rough mereology offers a methodology for the synthesis and analysis of objects in a distributed environment of intelligent agents, in particular, for the synthesis of objects satisfying a given specification to a satisfactory degree, or for control in such complex environments. Moreover, rough mereology has been used for developing the foundations of the *information granule calculi*, aimed at formalizing the “Computing with Words” paradigm, recently formulated by Zadeh (2001). More complex information granules are defined recursively using already defined information granules and their measures of inclusion and closeness. Information granules can have complex structures like classifiers or approximation spaces. Computations on information granules are performed to discover relevant information granules, e.g., patterns or approximation spaces for complex concept approximations.

It should be noted that rough sets can also be defined by employing the rough membership function instead of approximation. That is, consider

$$\mu_X^B : U \rightarrow \langle 0, 1 \rangle$$

defined by

$$\mu_X^B(x) = v(B(x), X) \quad (12)$$

where $x \in X \subseteq U$ and v is the standard rough inclusion (see Eq. 10).

The value $\mu_X^B(x)$ can be interpreted as the degree to which x belongs to X in view of knowledge about x expressed by B , or the degree to which the elementary granule $B(x)$ is included in the set X . This means that the definition reflects subjective knowledge about elements of the universe, in contrast to the classical definition of a set.

The rough membership function can also be interpreted as the conditional probability that x belongs to X given B . It is worth mentioning that set inclusion to a degree has been considered by Łukasiewicz (1970) in studies on assigning fractional truth values to logical formulae.

It can be shown that the rough membership function has the following properties:

1. $\mu_X^B(x) = 1$ iff $x \in B_*(X)$
2. $\mu_X^B(x) = 0$ iff $x \in U - B^*(X)$
3. $0 < \mu_X^B(x) < 1$ iff $x \in BN_B(X)$
4. $\mu_{U-X}^B(x) = 1 - \mu_X^B(x)$ for any $x \in U$
5. $\mu_X^B \cup Y(x) \geq \max(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$
6. $\mu_X^B \cap Y(x) \leq \min(\mu_X^B(x), \mu_Y^B(x))$ for any $x \in U$

From these properties, it follows that the rough membership function differs in essential ways from the fuzzy membership function (Zadeh 1965), since properties (5) and (6) show that membership for the union and intersection of sets, in general, cannot be computed – as in the case of fuzzy sets – from their constituents' membership. Thus, formally, rough membership is more general than fuzzy membership. Moreover, the rough membership function depends on available knowledge (represented by attributes from B). Besides, the rough membership function, in contrast to the fuzzy membership function, has a probabilistic flavor.

Sometimes one can distinguish, in an information table (U, A) , a partition of A into two classes $C, D \subseteq A$ of attributes, called *condition* and *decision (action)* attributes, respectively. The tuple $\mathcal{A} = (U, C, D)$ is called a *decision table*.

Let $V = \bigcup\{V_a | a \in C\} \cup V_d$. Atomic formulae over $B \subseteq C \cup D$ and V are expressions $a = v$ called *descriptors (selectors)* over B and V , where $a \in B$ and $v \in V_a$. The set $\mathcal{F}(B, V)$ of formulae over B and V is the smallest set containing all atomic formulae over B and V and closed with respect to the propositional connectives \wedge (conjunction), \vee (disjunction), and \neg (negation).

By $\|\varphi\|_{\mathcal{A}}$ we denote the meaning of $\varphi \in \mathcal{F}(B, V)$ in the decision table \mathcal{A} , which is the set of all objects in U with the property φ . These sets are defined by $\|a = v\|_{\mathcal{A}} = \{x \in U | a(x) = v\}$, $\|\varphi \wedge \varphi'\|_{\mathcal{A}} = \|\varphi\|_{\mathcal{A}} \cap \|\varphi'\|_{\mathcal{A}}$; $\|\varphi \vee \varphi'\|_{\mathcal{A}} = \|\varphi\|_{\mathcal{A}} \cup \|\varphi'\|_{\mathcal{A}}$; $\|\neg\varphi\|_{\mathcal{A}} = U - \|\varphi\|_{\mathcal{A}}$. The formulae from $\mathcal{F}(C, V)$, $\mathcal{F}(D, V)$ are called *condition formulae of \mathcal{A}* and *decision formulae of \mathcal{A}* , respectively.

Any object $x \in U$ belongs to a *decision class* $\|\bigwedge_{a \in D} a = a(x)\|_{\mathcal{A}}$ of \mathcal{A} . All decision classes of \mathcal{A} create a partition of the universe U .

A *decision rule* for \mathcal{A} is any expression of the form $\varphi \Rightarrow \psi$, where $\varphi \in \mathcal{F}(C, V)$, $\psi \in \mathcal{F}(D, V)$, and $\|\varphi\|_{\mathcal{A}} \neq \emptyset$. Formulae φ and ψ are referred to as the *predecessor* and the *successor* of decision rule $\varphi \Rightarrow \psi$. Decision rules are often called “*IF … THEN …*” rules.

The decision rule $\varphi \Rightarrow \psi$ is *true* in \mathcal{A} if and only if $\|\varphi\|_{\mathcal{A}} \subseteq \|\psi\|_{\mathcal{A}}$. Otherwise, one can measure its *truth degree* by introducing some inclusion measure of $\|\varphi\|_{\mathcal{A}}$ in $\|\psi\|_{\mathcal{A}}$. It is important to note that an inclusion measure expressed in terms of the confidence measure, widely used in data mining, was considered by Łukasiewicz (1970) a long time ago in studies on assigning fractional truth values to logical formulae. Given two unary predicate formulae $\alpha(x)$, $\beta(x)$ where x runs over a finite set U , Łukasiewicz proposes to assign to $\alpha(x)$ the value $\|\alpha(x)\|/|U|$, where $\|\alpha(x)\| = \{x \in U : x \text{ satisfies } \alpha\}$. The fractional value assigned to the implication $\alpha(x) \Rightarrow \beta(x)$ is then $\|\alpha(x) \wedge \beta(x)\|/|\alpha(x)|$ under the assumption that $|\alpha(x)| \neq \emptyset$.

Each object x of a decision table determines a *decision rule* $\bigwedge_{a \in C} a = a(x) \Rightarrow \bigwedge_{a \in D} a = a(x)$.

Decision rules corresponding to some objects can have the same condition parts but different decision parts. Such rules are called *inconsistent (nondeterministic, conflicting, possible)*; otherwise the rules are referred to as *consistent (certain, sure, deterministic, nonconflicting)* rules. Decision tables containing inconsistent decision rules are called *inconsistent (nondeterministic, conflicting)*; otherwise the table is *consistent (deterministic, nonconflicting)*.

Numerous methods have been developed within the rough sets literature for generating decision rules. They typically involve searching for decision rules that are (semi) optimal with respect to some optimization criteria describing the quality of decision rules in concept approximations.

In the case of searching for concept approximations in an extension of a given universe of objects (sample), the following steps are typically used. When a set of rules has been induced from a decision table containing a set of training examples, they can be inspected to see if they reveal any novel relationships between attributes that are worth pursuing for further research.

Furthermore, the rules can be applied to a set of unseen cases in order to estimate their classification power. For a systematic overview of rule application methods the reader is referred to the literature (Bazan 1998; Grzymała-Busse 1998; Bazan et al. 2000; Nguyen 2002; Triantaphyllou and Felici 2006) (see also references in Pawlak and Skowron 2007a, b, c).

Another important issue in data analysis is that of discovering dependencies between attributes. Intuitively, a set of attributes D depends totally on a set of attributes C , denoted $C \Rightarrow D$, if the values of attributes from C uniquely determine the values of attributes from D . In other words, D depends totally on C , if there exists a functional dependency between values of C and D . Formally, dependency can be defined in the following way. Let D and C be subsets of A .

We say that D depends on C to a degree k ($0 \leq k \leq 1$), denoted $C \Rightarrow_k D$, if

$$k = \gamma(C, D) = \frac{|\text{POS}_C(D)|}{|U|} \quad (13)$$

where

$$\text{POS}_C(D) = \bigcup_{X \in U/D} C_*(X) \quad (14)$$

called a *positive region* of the partition U/D with respect to C , is the set of all elements of U that can be uniquely classified to blocks of the partition U/D , by means of C .

If $k = 1$, we say that D depends totally on C , and if $k < 1$, we say that D depends partially (to a degree k) on C .

The coefficient k expresses the ratio of all elements of the universe, which can be properly classified to blocks of the partition U/D , employing attributes C and will be called the *degree of the dependency*.

It can be easily seen that if D depends totally on C then $I(C) \subseteq I(D)$. This means that the partition generated by C is finer than the partition generated by D . Notice that the concept of dependency discussed above corresponds to that considered in relational databases.

Summing up: D is *totally (partially)* dependent on C , if *all (some)* elements of the universe U can be uniquely classified to blocks of the partition U/D , employing C .

The question as to whether or not some data from a data-table can be removed while preserving its basic properties, i.e., whether a table contains some superfluous data, is a common one. This idea can be expressed more precisely.

Let $C, D \subseteq A$, be sets of condition and decision attributes, respectively. We say that $C' \subseteq C$ is a *D-reduct* (reduct with respect to D) of C , if C' is a minimal subset of C such that

$$\gamma(C, D) = \gamma(C', D) \quad (15)$$

The intersection of all *D-reducts* is called a *D-core* (core with respect to D). Because the core is the intersection of all reducts, it is included in every reduct, i.e., each element of the core belongs to some reduct. Thus, in a sense, the core is the most important subset of attributes, since none of its elements can be removed without affecting of the classification power of attributes.

Many other kinds of reducts and their approximations are discussed in the literature. It turns out that they can all be efficiently computed using heuristics based on a Boolean reasoning approach.

Tasks collected under the labels of data mining, knowledge discovery, decision support, pattern classification, approximate reasoning, and so on, require tools aimed at discovering *templates (patterns)* within the data, and classifying them into certain *decision classes*. Templates are in many cases the most frequent sequences of events, the most probable events,

regular configurations of objects, the decision rules of highest quality, and so on. Tools for discovering and classifying templates are based on *reasoning schemes* rooted in various paradigms (Hastie et al. 2001). Such patterns can be extracted from data by means of methods based on Boolean reasoning and discernibility.

The discernibility relation is one of the most important relations considered in rough set theory.

The ability to discern between perceived objects is important for constructing many entities like reducts, decision rules, or decision algorithms. In the classical rough set approach, the discernibility relation $\text{DIS}(B) \subseteq U \times U$ is defined by $x \text{ DIS}(B)y$ if and only if $\text{non}(xI(B)y)$. However, this is in general not the case for the generalized approximation spaces (one can define indiscernibility by $x \in I(y)$ and discernibility by $I(x) \cap I(y) = \emptyset$ for any objects x, y).

The idea of Boolean reasoning is based on the construction of, for a given problem P , a corresponding Boolean function f_P with the following property: the solutions for the problem P can be decoded from the prime implicants of f_P . It should be noted that to solve real-life problems it is necessary to deal with Boolean functions having a large number of variables.

A successful methodology based on the discernibility of objects and Boolean reasoning has been developed for computing many application-critical entities such as reducts and their approximations, decision rules, association rules, discretization of real-value attributes, symbolic value grouping, searching for new features defined by oblique hyperplanes or higher-order surfaces, pattern extraction from data, as well as conflict resolution or negotiation (Nguyen and Skowron 1997; Nguyen and Nguyen 1998; Nguyen 1998, 2006; Skowron 2002; Pawlak and Skowron 2007a).

Most of the problems related to generating the above mentioned entities are NP-complete or NP-hard. However, it has been found possible to develop efficient heuristics yielding suboptimal solutions. The results of experiments on many data sets are very promising (Nguyen and Skowron 1997; Nguyen and Nguyen 1998; Nguyen 1998, 2006; Skowron 2002; Pawlak and Skowron 2007a). They show very good quality of solutions generated by the heuristics in comparison with other methods reported in the literature (e.g., with respect to the classification quality of unseen objects). Moreover, they are very efficient from the point of view of the time necessary for computing the solution.

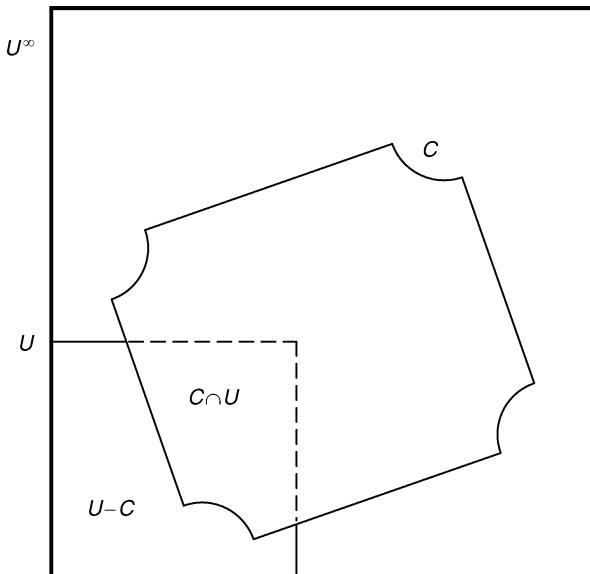
It is important to note that the methodology makes it possible to construct heuristics having a very important *approximation property*, which can be formulated as follows: expressions generated by heuristics (i.e., implicants) close to prime implicants define approximate solutions for the problem (Skowron 2002; Nguyen 2006).

The problem of approximation of concepts over a universe U^∞ (concepts that are subsets of U^∞) is now outlined. It is assumed that the concepts are perceived only through some subsets of U^∞ , called samples. This is a typical situation in the machine learning, pattern recognition, or data-mining approaches (Hastie et al. 2001). The rough set approach to induction of concept approximations is explained using the generalized approximation spaces of the form $AS = (U, I, v)$, defined earlier.

Let $U \subseteq U^\infty$ be a finite sample. By Π_U we denote a perception function from $P(U^\infty)$ into $P(U)$ defined by $\Pi_U(C) = C \cap U$ for any concept $C \subseteq U^\infty$. $\Pi_U(C)$ represents partial information relative to the given sample U about the concept C (see Fig. 3).

Let $AS = (U, I, v)$ be an approximation space over the sample U . The problem as to how to extend the approximations of $\Pi_U(C)$ defined by AS to the approximation of C over U^∞ is considered. It is shown that the problem can be described as searching for an extension $AS_C = (U^\infty, I_C, v_C)$ of the approximation space AS , relevant for the approximation of C .

Fig. 3
Concept sample.



This requires us to show how to extend the inclusion function v from subsets of U to subsets of U^∞ that are relevant for the approximation of C . Observe that for the approximation of C , it is enough to induce the necessary values of the inclusion function v_C without knowing the exact value of $I_C(x) \subseteq U^\infty$ for $x \in U^\infty$.

Let AS be a given approximation space for $\Pi_U(C)$ and let a language L be considered in which the neighborhood $I(x) \subseteq U$ is expressible by a formula $\text{pat}(x)$, for any $x \in U$. This means that $I(x) = \|\text{pat}(x)\|_U \subseteq U$, where $\|\text{pat}(x)\|_U$ denotes the meaning of $\text{pat}(x)$ restricted to the sample U . In the case of rule-based classifiers, patterns of the form $\text{pat}(x)$ are defined by feature value vectors.

It is assumed that for any new object $x \in U^\infty \setminus U$ (e.g., as a result of sensor measurement) a pattern $\text{pat}(x) \in L$ with semantics $\|\text{pat}(x)\|_{U^\infty} \subseteq U^\infty$ can be obtained. However, the relationships between information granules over U^∞ like sets: $\|\text{pat}(x)\|_{U^\infty}$ and $\|\text{pat}(y)\|_{U^\infty}$, for different $x, y \in U^\infty$, are, in general, known only if they can be expressed by relationships between the restrictions of these sets to the sample U , i.e., between sets $\Pi_U(\|\text{pat}(x)\|_{U^\infty})$ and $\Pi_U(\|\text{pat}(y)\|_{U^\infty})$.

The set of patterns $\{\text{pat}(x) : x \in U\}$ is usually not relevant for approximation of the concept $C \subseteq U^\infty$. Such patterns are too specific or not sufficiently general, and can directly be applied only to a very limited number of new objects. However, by using some generalization strategies, one can search, within a family of patterns definable from $\{\text{pat}(x) : x \in U\}$ in L , for such new patterns that are relevant for the approximation of concepts over U^∞ . Let a subset $\text{PATTERNS}(AS, L, C) \subseteq L$ chosen as a set of pattern candidates for the relevant approximation of a given concept C be considered. For example, in the case of rule-based classifiers, one can search for such candidate patterns among sets definable by subsequences of feature value vectors corresponding to objects from the sample U . The set $\text{PATTERNS}(AS, L, C)$

can be selected by using some quality measures checked on the meanings (semantics) of its elements restricted to the sample U (such as the number of examples from the concept $\Pi_U(C)$ and its complement that support a given pattern). Then, on the basis of properties of sets definable by these patterns over U , we induce approximate values of the inclusion function v_C on subsets of U^∞ definable by any such pattern and the concept C .

Next, the value of v_C on pairs (X, Y) is induced where $X \subseteq U^\infty$ is definable by a pattern from $\{\text{pat}(x) : x \in U^\infty\}$ and $Y \subseteq U^\infty$ is definable by a pattern from PATTERNS(AS, L, C).

Finally, for any object $x \in U^\infty \setminus U$, the approximation of the degree $v_C(\|\text{pat}(x)\|_{U^\infty}, C)$ is induced applying a conflict resolution strategy *Conflict_res* (a voting strategy, in the case of rule-based classifiers) to two families of degrees:

$$\{v_C(\|\text{pat}(x)\|_{U^\infty}, \|\text{pat}\|_{U^\infty}) : \text{pat} \in \text{PATTERNS(AS, L, C)}\} \quad (16)$$

$$\{v_C(\|\text{pat}\|_{U^\infty}, C) : \text{pat} \in \text{PATTERNS(AS, L, C)}\} \quad (17)$$

Values of the inclusion function for the remaining subsets of U^∞ can be chosen in any way – they do not have any impact on the approximations of C . Moreover, observe that for the approximation of C , the exact values of the uncertainty function I_C need not be known – it is enough to induce the values of the inclusion function v_C . Observe that the defined extension v_C of v to some subsets of U^∞ makes it possible to define an approximation of the concept C in a new approximation space AS_C.

In this way, the rough set approach to induction of concept approximations can be explained as a process of inducing a relevant approximation space.

It is worthwhile mentioning that for any formula α over descriptors one can consider its semantics $\|\alpha\|_U$ over U and semantics $\|\alpha\|_{U^\infty}$ over U^∞ , respectively: (1) $\|\alpha\|_U \subseteq U$ and (2) $\|\alpha\|_{U^\infty} \subseteq U^\infty$. In approximate reasoning, hypotheses on relationships of semantics of formulae over U^∞ are induced (estimated) from information on relationships between semantics of formulae over U .

Many tasks in rough computing are based on searching for (semi)optimal approximation spaces (Jankowski et al. 2008). There are numerous significant results and active current research directions on rough set theory and applications (Pawlak and Skowron 2007a, b, c).

3 Fuzzy Sets

Fuzzy set theory has entered its fifth decade of research and development efforts since the first paper on fuzzy sets was published by Zadeh (1965). During these decades, numerous publications on the advances of the foundations of fuzzy sets and applications in many different areas have been published (see, e.g., Dubois and Prade 2000; Hohle and Rodabaugh 1999; Bezdek et al. 1999a, b; Słowiński 1998; Nguyen and Sugeno 1998; Zimmermann 1999; Klir and Yuan 1995; Klir 2006; Pedrycz and Gomide 2007; Nikravesh et al. 2007, 2008).

In this section, some basic issues that are important for fuzzy computing are discussed.

Fuzzy sets differ from classical sets by rejecting the requirement that for any object it must be possible to determine whether the object is, or is not, a member of the set. More formally, in classical set theory, it can be assumed that for any set $X \subseteq U$, where U is the universe of objects, the following condition holds:

$$\forall x \in U(x \in X \vee x \notin X). \quad (18)$$

Contrary to classical sets, fuzzy sets are not required to have sharp boundaries distinguishing their members from other objects. The membership in a fuzzy set is a matter of degree. Due to their sharp boundaries, classical sets are usually referred to in the fuzzy literature as *crisp* sets.

Any fuzzy set over a given (crisp) universe of objects is defined by a function analogous to the characteristic function of crisp sets. This function is called a (fuzzy) membership function.

Let U be the universe of objects. A *fuzzy set* is defined uniquely by a *fuzzy membership function* of the form $\mu : U \rightarrow [0, 1]$. For any $x \in U$, the value $\mu(x)$ defines the degree of membership of the object x from U in the fuzzy set identified with the function μ .

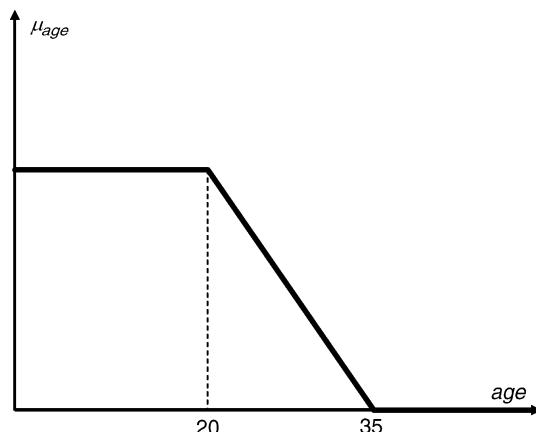
The membership of any object from the universe of objects in any fuzzy set is a matter of degree. By using degrees of membership, fuzzy sets make it possible to express gradual transitions from membership to nonmembership. This expressive capability is important for expressing, e.g., the meanings of vague expressions in natural language. Membership degrees in the fuzzy sets define compatibilities of relevant objects with the linguistic expression that the fuzzy sets attempt to approximate. Crisp sets are inadequate for this purpose.

Let an illustrative example of a fuzzy set corresponding to a linguistic hedge *young* be considered. The membership function corresponding to this hedge is presented in [Fig. 4](#).

In the example, the universe U consists of age values. One can see that if the age of a person is not greater than 20, then the membership degree is equal to 1, i.e., this person is classified with certainty as *young*. If the age of a person is greater than 35, then the membership degree is equal to 0, i.e., this person is classified with certainty as *not young*. If the age is between 20 and 35 years the certainty degree is greater than 0 but less than 1, and the higher the age the smaller the degree of membership. Definitely, this membership function is subjective. One can ask how such a function may be acquired from data or from experts, or computed from some other fuzzy sets. One can assume that in the discussed example, the membership function was drawn by an expert. Another solution may be to select the membership function from a parameterized family of functions that are candidates for the membership function so that

Fig. 4

A fuzzy set corresponding to the linguistic expression: *young*.



the selected function could match the expert opinions in the best way. One can consider, e.g., the following simple parameterized family of functions:

$$\mu_{a,b}(x) = \begin{cases} 1 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a < x < b \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

where a, b are parameters to be tuned to match the expert opinions. In the example, the selected parameters are $a = 20$ and $b = 35$. Certainly, in many applications, the process of construction of the relevant fuzzy membership function may not be so easy. The construction of fuzzy membership functions the most suited for the application under consideration is one of the main tasks in fuzzy computing.

In the above definition of fuzzy sets, membership degrees are expressed by real numbers from the interval $[0, 1]$. In generalizations of the concept of fuzzy sets, which are often referred to as nonstandard fuzzy sets, degrees of membership may be expressed by more complex objects such as intervals of reals or elements of lattices (Goguen 1967). For the sake of simplicity, our considerations are restricted mainly to the case of degrees of membership from the interval $[0, 1]$.

The set of all fuzzy sets over the universe U is denoted by $\mathcal{F}(U)$.

If $\mu, v \in \mathcal{F}(U)$, then the fuzzy set μ is a fuzzy subset of the fuzzy set v , in symbols $\mu \subseteq v$ iff $\mu(x) \leq v(x)$ for any $x \in U$.

When U is a finite set, then one can consider a fuzzy subsethood relation, Sub defined for any fuzzy subset μ, v over U by

$$\text{Sub}(\mu, v) = \frac{\sum_{x \in U} \mu(x) - \sum_{x \in U} \max\{0, \mu(x) - v(x)\}}{\sum_{x \in U} \mu(x)} \quad (20)$$

Many characteristics are defined for any fuzzy set $\mu \in \mathcal{F}(U)$. Some of them are considered here. When U is a finite set, then $|\mu|$ denotes the fuzzy cardinality of μ defined by $|\mu| = \sum_{x \in U} \mu(x)$. Among the most important concepts of fuzzy sets are the concepts of α -cut and strong α -cut, which are defined for any fuzzy set $\mu \in \mathcal{F}(U)$ and any $\alpha \in [0, 1]$ by ${}^{\alpha}\mu = \{x \in U : \mu(x) \geq \alpha\}$ and ${}^{>\alpha}\mu = \{x \in U : \mu(x) > \alpha\}$. The set ${}^{0+}\mu$ is called the *support* and the set ${}^{1+}\mu$ is called the *core* of the fuzzy set μ .

Another task in fuzzy computing is the task of searching for the construction of membership functions (e.g., this is highly relevant for applications) from given fuzzy sets. Some examples of operations that can be used in this process are presented here.

The counterparts of set theoretical operations (i.e., complement, union, and intersection of sets) are not unique for fuzzy sets. These operations for fuzzy sets are defined relative to some functions.

The complement operation for fuzzy sets is defined by *complementation operations*, i.e., functions $n : [0, 1] \rightarrow [0, 1]$ that satisfy the following conditions: (1) n is order reversing, (2) $n(0) = 1$, (3) $n(1) = 0$, and (4) $n(n(s))$ for all $s \in [0, 1]$ (n is an involution). An example of a complementation function is $n_{\lambda}(s) = (1 - s^{\lambda})^{1/\lambda}$, where $\lambda > 0$. For a given complementation operation and a fuzzy set $\mu \in \mathcal{F}(U)$ the n -complement of μ is defined by $\mu'(x) = n(\mu(x))$ for $x \in U$.

The operations of union and intersection on fuzzy sets are defined relative to functions referred to in the research literature as *triangular norms* (or *t-norms*) and *triangular conorms* (or *t-conorms*), defined in terms of functions from the product $[0, 1] \times [0, 1]$ into $[0, 1]$,

which are (1) commutative, (2) associative, (3) monotone nondecreasing, and (4) consistent with the characteristic functions of set theoretical intersections and unions, respectively. The standard example of a t -norm is *min*, and the standard example of a t -conorm is *max*. There are many other examples of t -norms and t -conorms such as

$$g(s, w) = 1 - \min\{1, [(1-s)^\lambda + (1-w)^\lambda]^{1/\lambda}\} \quad (21)$$

$$h(s, w) = \min\{1, (s^\lambda + w^\lambda)^{1/\lambda}\} \quad (22)$$

where $\lambda > 0$ is a parameter tuned for each application by knowledge acquisition techniques.

For given fuzzy sets $\mu, v \in \mathcal{F}(U)$, t -norm g , and t -conorm h , the g -union and h -intersection of μ and v are defined by

$$(\mu \cup v)(x) = g(\mu(x), v(x)) \quad (23)$$

$$(\mu \cap v)(x) = h(\mu(x), v(x)) \quad (24)$$

respectively, where $x \in U$.

There are other operations used for constructing fuzzy sets without counterparts in classical set theory. Among them are modifiers and averaging operations. For example, modifiers are unary operations preserving the order used for modifying fuzzy sets representing linguistic terms. They make it possible to account for linguistic hedges such as *very*, *fairly*, *extreme*, and *more or less*.

Relations are defined in classical set theory as subsets of Cartesian products of some universes of objects. One can also consider fuzzy relations as elements of $\mathcal{F}(U_1 \times \dots \times U_k)$, where U_1, \dots, U_k are universes of objects. Fuzzy relations involve additional concepts and operations due to their multidimensionality. Among these operations are projections, cylindric extensions, compositions, joins, and inverses or fuzzy relations.

Let two examples of operations on fuzzy relations, namely, projection and composition be considered. For simplicity, it can be assumed that $\mu \in \mathcal{F}(U_1 \times U_2)$. Then the projection of μ on U_2 is defined by $\mu_{U_2}(x) = \max_{y \in Y} \mu(x, y)$, where $x \in U_1$. In the example of composition of fuzzy relations, let it be assumed that two fuzzy relations $\mu_1 \in \mathcal{F}(U_1 \times U_2)$ and $\mu_2 \in \mathcal{F}(U_2 \times U_3)$ are given. Then the fuzzy composition μ of μ_1 and μ_2 is defined by

$$\mu(x, y) = \max_{z \in U_2} \min\{\mu_1(x, z), \mu_2(z, y)\} \quad (25)$$

where $x \in U_1$ and $y \in U_3$.

An illustrative operation transforming fuzzy sets into fuzzy sets can be defined by a crisp function $f: U_1 \rightarrow U_2$. Any such function defines an operation $F: \mathcal{F}(U_1) \rightarrow \mathcal{F}(U_1)$ by the *extension principle*, i.e., by

$$\mu'(y) = \begin{cases} \sup\{\mu(x) : x \in U_1 \text{ and } f(x) = y\} & \text{if } f^{-1}(y) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

Some properties of fuzzy sets can be derived from properties of standard sets. Such properties are called *cutworthy properties*. For example, a fuzzy set over reals is said to be a *fuzzy interval* if and only if all of its α -cuts are closed intervals of reals. The equivalence property of fuzzy binary relations can be considered as another example. A fuzzy relation $\mu \in \mathcal{F}(U \times U)$ is a fuzzy equivalence relation if and only if this relation is reflexive, symmetric, and max-min-transitive, i.e., $\mu(x, y) \geq \max_{z \in U} \min\{\mu(x, z), \mu(z, y)\}$ for all $x \in U \times U$.

The equivalence property of binary relation over U is also a cutworthy property. However, many operations on fuzzy sets are not cutworthy (e.g., when t -norms or t -conorms, different from \min and \max , are applied in definitions of fuzzy union or intersection).

Several measures of fuzziness of fuzzy sets have been introduced. The basic idea in defining such measures is based on the observation that the less the fuzzy set differs from its fuzzy complement, the fuzzier it is. This idea can be expressed by the following functional $f : \mathcal{F}(U) \rightarrow \mathcal{R}_+$ defined by

$$f(\mu) = \sum_{x \in U} (1 - |\mu(x) - n(\mu(x))|) \quad (27)$$

where $x \in U$, U is a finite universe of objects, and n is a fuzzy negation.

The degree of fuzzy membership $\mu(x)$, for a given fuzzy set $\mu \in \mathcal{F}(U)$ and $x \in U$ can be interpreted as the degree of truth of the proposition x is a member of μ .

The simplest fuzzy canonical propositional form is defined by

$$f_\mu(\mathbf{x}) : \mathbf{x} \text{ is } \mu \quad (28)$$

where \mathbf{x} is a variable with values in U and $\mu \in \mathcal{F}(U)$, which is called a fuzzy predicate. By substituting into the propositional form instead of the variable \mathbf{x} a particular object $x \in U$, a fuzzy proposition is obtained

$$f_\mu(x) : x \text{ is } \mu \quad (29)$$

where $f_\mu(x)$ also denotes the degree of truth of the proposition x is μ , which is equal to $\mu(x)$ for $x \in U$.

This simple correspondence is used to define the truth of more complex propositions constructed by using negations, conjunctions, disjunctions, or implications of fuzzy propositions. Let the case of implication be considered. The propositions constructed using fuzzy implication are called conditional fuzzy propositions. They are essential for knowledge-based systems (Nguyen and Sugeno 1998). Conditional fuzzy propositions are based on the propositional form

$$f_{\mu'|\mu} : \text{if } \mathbf{x} \text{ is } \mu \text{ then } \mathbf{y} \text{ is } \mu' \quad (30)$$

where \mathbf{x} and \mathbf{y} are variables with values in the universes U_1 and U_2 , respectively. This form may also be presented as

$$f_{\mu'|\mu} : (\mathbf{x}, \mathbf{y}) \text{ is } \xi \quad (31)$$

where $\xi \in \mathcal{F}(U_\infty \times U_\infty)$ is defined by

$$\xi(x, y) = \iota(\mu(x), \mu'(y)) \quad (32)$$

where $\iota : [0, 1] \times [0, 1] \rightarrow [0, 1]$ defines a relevant fuzzy implication in the given application context. The class of such functions defining fuzzy implications is large and the selection of the relevant implication is a challenging task in the given application. One can consider some natural conditions that should satisfy any function $\iota : [0, 1] \times [0, 1] \rightarrow [0, 1]$ defining fuzzy implications (Klir and Yuan 1995; Pedrycz and Gomide 2007). The basic conditions are the following: (1) (monotonicity) if $b \leq b'$ then $\iota(a, b) \leq \iota(a, b')$, (2) (dominance of falsity) $\iota(0, b) = 1$, and (3) (neutrality of truth) $\iota(1, b) = b$. In addition, also often some other conditions are added (4) if $a \leq a'$ then $\iota(a, b) \geq \iota(a', b)$, (5) $\iota(a, \iota(a', b)) = \iota(a', \iota(a, b))$,

(6) $\iota(a, a) = 1$, (7) $\iota(a, b) = 1$ iff $a \leq b$, (8) $\iota(a, b) = \iota(n(b), n(a))$, where n is a fuzzy negation, and (9) ι is a continuous function. In all conditions (1)–(9) $a, b, a', b' \in [0, 1]$.

In general, one class of implications is defined in terms of negation and disjunction. Another class of fuzzy implications may be defined by the residuation of continuous t -norms. Any t -norm g defines its residuation by

$$\iota(a, b) = \sup_{c \in [0, 1]} \{g(a, c) \leq b\} \quad (33)$$

where $a, b \in [0, 1]$.

An important subclass of fuzzy implications known as Łukasiewicz implications is defined by

$$\iota(a, b) = \min [1, 1 - a^\lambda + b^\lambda]^{1/\lambda} \quad (34)$$

where $\lambda > 0$ is a parameter to be tuned to obtain a relevant implication for the given application and $a, b \in [0, 1]$. There are many other fuzzy implications used in fuzzy set theory and its applications. Among them are the Gödel implication and the Kleene implication defined by

$$\iota(a, b) = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{otherwise} \end{cases} \quad (35)$$

and

$$\iota(a, b) = \max\{1 - a, b\} \quad (36)$$

respectively, where $a, b \in [0, 1]$.

Fuzzy propositions constructed by means of fuzzy connectives may also be quantified by means of fuzzy quantifiers. Fuzzy quantifiers are usually fuzzy intervals.

Reasoning based on fuzzy propositions is related to approximate reasoning. The most fundamental components of approximate reasoning are conditional fuzzy propositions that can be quantified in different ways. Various methods have been developed to deal with different kinds of fuzzy propositions attempting to emulate commonsense reasoning based on natural language. Let one simple example be considered that is related to the *generalized modus ponens* rule, defined by the following scheme:

Fuzzy rule: if x is μ then y is μ'

Fuzzy fact: x is v

Fuzzy conclusion: y is v'

where $\mu, v \in \mathcal{F}(U_1)$, $\mu', v' \in \mathcal{F}(U_2)$.

Assuming that the fuzzy rule has been presented in the relational form

$$(x, y) \text{ is } \xi \quad (37)$$

where ξ is represented by a fuzzy implication. The fuzzy conclusion v' is obtained by composition v and ξ , i.e., $v' = v \circ \xi$, where \circ denotes the fuzzy composition, i.e.,

$$v'(y) = \max_{x \in U_1} \{\min\{v(x), \xi(x, y)\}\} \quad (38)$$

where $y \in U_2$.

Observe that in any application, it is necessary to choose a relevant fuzzy implication and use it to describe the relation ξ . This requires searching, in particular, for relevant fuzzy implication making it possible to derive conclusions matching the empirical observations or expectations by human experts. There are several methods for the selection of the necessary parameters (Dubois and Prade 2000; Hoehle and Rodabaugh 1999; Bezdek et al. 1999a; Pedrycz and Gomide 2007).

Fuzzy systems are represented by a set of variables and dependencies among values (states) of the variables, which are fuzzy sets. Very often, the values are fuzzy intervals representing linguistic terms such as *small*, *medium*, *large*, *very large*, with the interpretation of these variables depending on the application under consideration. Each linguistic variable is defined by (1) a name of this variable; (2) a base variable with a given set of values, which is often the set of reals; (3) a set of linguistic terms referring to the base variable such as *small*, *medium*, *large*, *very large* and (4) a set of fuzzy intervals defining the semantics of each linguistic term. Fuzzy sets defining semantics of linguistic terms create a *fuzzy partition*, i.e., for any object x from the domain of the base variable the sum of membership degrees in these fuzzy sets is equal to 1.

In knowledge-based systems, relationships between variables are given by a set of *fuzzy inference rules*. These rules are conditional fuzzy propositional forms representing the relevant human expert knowledge, often expressed in natural language. Another class of fuzzy systems involves creating model-based fuzzy systems based on traditional modeling combined with the application of fuzzy sets. By combining knowledge-based systems and model-based systems, such hybrid systems are obtained.

The input–output relationship of such a knowledge-based system can be described by the following scheme of inference:

Rule 1: if x_1 is μ_{11} and x_2 is μ_{21} ...
then y_1 is μ'_{11} and y_2 is μ'_{21} and ...

Rule 2: if x_1 is μ_{12} and x_2 is μ_{22} ...
then y_1 is μ'_{12} and y_2 is μ'_{22} and ...

.....
Rule k: if x_1 is μ_{1k} and x_2 is μ_{2k} ...
then y_1 is μ'_{1k} and y_2 is μ'_{2k} and ...

Fact: x_1 is v_1 and x_2 is v_2 and ...

Conclusion: y_1 is v_1 and y_2 is v_2 and ...

Different inference methods based on the fuzzy inference rules presented above have been developed. A thorough consideration of all the details of those methods is beyond the scope of this overview. However, it is appropriate to make some general comments. Let it be assumed that (1) each fuzzy inference rule is based on input and output linguistic variables and (2) the values of variables on the left-hand side of rules are given, and the fuzzy intervals (over reals) corresponding to these variables are defined. Then, for each rule r , the inference is performed using the following steps:

1. Selection of aggregation methods of membership degrees given for fuzzy sets on the left-hand side of the rule r . This requires, e.g., the selection of some t -norms corresponding to conjunctions occurring on the left-hand side of the rule r . It is worth mentioning that interpretation of all conjunctions by the t -norm *min* may not be satisfactory for a given application and it may then be necessary to search for different t -norms for different conjunctions.

2. *Modification of fuzzy sets on the right-hand side of the rule r using the aggregated membership degree from the previous step.* As a consequence, the result of application of each fuzzy inference rule is a fuzzy interval for each output variable. Next, for each output variable, the fuzzy intervals obtained from different inference rules are aggregated and a new fuzzy interval is obtained. Finally, a defuzzification process is launched, i.e., for each variable, the obtained fuzzy interval is transformed into a single real number representing, in the context of the given application and in the best way, the fuzzy set. One of the basic defuzzification methods, which is called a *centroid method*, can be described for a finite universe U of objects by

$$d(\mu) = \frac{\sum_{x \in U} x \cdot \mu(x)}{\sum x \in U \mu(x)} \quad (39)$$

where $x \in U$.

Methods based on fuzzy logic have been applied to many real-life problems including control problems (Nguyen and Sugeno 1998). Fuzzy logic was inspired by a number of remarkable human capabilities. Among them Zadeh distinguishes (1) the ability to reason and make decisions in cases of imprecision, uncertainty, incompleteness of information and partial truth and (2) the capability to perform many tasks based on perceptions, without any measurements and any computations. In the case of fuzzy logic, we move away from numerical variables and toward linguistic variables or words in natural languages; we sacrifice precision in order to achieve important advantages down the line. This is what is called *the fuzzy gambit*. Let it be noted that in this case, the values of the variables are known precisely. Fuzzy logic is also applied when the values of variables are not known precisely (Zadeh 2007). There are three principal rationales that dictate when words should be used in preference to numbers: (1) when available information is not precise enough to use numbers; (2) when there is a tolerance for imprecision that can be exploited to achieve tractability, robustness, and low solution costs; and (3) when the expressive power of words is higher than the expressive power of numbers.

The problem of construction of fuzzy sets (fuzzy membership functions) is very important for applications. This problem is also related to knowledge acquisition. Methods for constructing fuzzy membership functions can be classified into direct and indirect. In direct methods, experts are expected to define the fuzzy membership function. In indirect methods, experts are asked to compare elements in pairs from the universe of objects relative to the order of the membership degrees. Methods for constructing fuzzy membership functions and relevant operations on fuzzy sets for a given application are crucial for successful applications of fuzzy sets.

There are numerous significant results and active current research directions on fuzzy set theory and applications (Dubois and Prade 2000; Hoehle and Rodabaugh 1999; Bezdek et al. 1999a, b; Słowiński 1998; Nguyen and Sugeno 1998; Zimmermann 1999; Klir and Yuan 1995; Klir 2006; Pedrycz and Gomide 2007; Nikravesh et al. 2007, 2008).

4 Rough/Fuzzy Hybridization

There are numerous theoretical results and real-life applications based on the combination of rough set theory and fuzzy set theory (see, e.g., Dubois and Prade 1987, 1988, 1990; Nanda

1992; Pal and Banerjee 1996; Greco et al. 1998; Pal and Skowron 1999; Greco et al. 1999, 2000, 2006; Pal 2003; Polkowski 2002; Wu et al. 2003; Inuiguchi et al. 2004; Lingras and Jensen 2007; Maji and Pal 2007, 2005; Pawlak and Skowron 2007a, c and references in these papers). These results show the fruitful complementarity of fuzzy set theory and rough set theory, which were both created to deal with imperfect knowledge, in particular, with vague concepts. In fuzzy set theory, any object is characterized by a grade of membership relative to a given concept (*gradualness of membership*), while rough set theory started from the assumption about the possible indiscernibility of perceived objects (*granularity of knowledge*).

Let some examples be considered illustrating the results of the hybridization of fuzzy sets and rough sets in the definition of concept approximation.

Let it be assumed that $\mu \in \mathcal{F}(U)$. The fuzzy set μ can be characterized by a set of α -cuts $\{\mu_\alpha\}_{\alpha \in [0, 1]}$. Let it also be assumed that $R \subseteq U \times U$ is an equivalence relation over the universe of objects U . Then, the rough/fuzzy set corresponding to μ is defined by two families of sets, the lower approximation family of cuts and the upper approximation family of cuts defined by

$$R(\mu) = \bigcup_{x \in U} \{[x]_R : [x]_R \subseteq^\alpha \mu\} \quad (40)$$

and

$$\overline{R}(\mu) = \bigcup_{x \in U} \{[x]_R : [x]_R \cap^\alpha \mu \neq \emptyset\} \quad (41)$$

respectively, where $x \in U$, and $[x]_R$ denotes the equivalence class of R defined by x .

Another approach can be used to define the so-called fuzzy/rough sets. In this case, it is assumed that $\mu \in \mathcal{F}(U)$ and $v = \{v_i\}_{i \in I}$ is a fuzzy partition of U . Then two fuzzy sets (over the universe of elements v_i of fuzzy partition v) are defined: the fuzzy/rough lower approximation of μ relative to v and the fuzzy/rough upper approximation of μ relative to v by

$$v\mu(v_i) = \inf_{x \in U} \max\{1 - v_i(x), \mu(x)\} \quad (42)$$

and

$$\overline{v}\mu(v_i) = \sup_{x \in U} \min\{v_i(x), \mu(x)\} \quad (43)$$

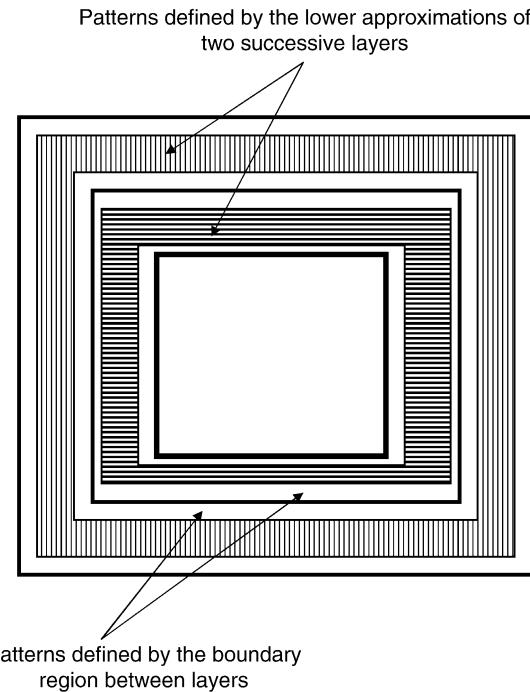
where $i \in I$.

There are many results on fuzzy/rough sets and rough/fuzzy sets including the generalizations of the above definitions (see, e.g., the references in Lingras and Jensen 2007).

Many methods have also been developed that are based on the combination of rough sets and fuzzy sets for real-life applications. In particular, these methods have been applied in supervised learning, information retrieval, feature selection, clustering, neurocomputing, and evolutionary computing (Lingras and Jensen 2007). There are different strategies for the combination of these two approaches. Let one possible application be illustrated, in which rough sets are used in the approximation of fuzzy sets by searching for *subconcepts* of a given fuzzy set that can be approximated by rough set methods with high quality. The layers defined by the differences of successive cuts are selected as subconcepts. The problem that should be solved is the selection of relevant cuts so that the patterns defined by approximations of layers are relevant for the application under consideration (Skowron and Stepaniuk 2003; Bazan 2008a, b).  **Figure 5** illustrates patterns corresponding to the lower approximations of layers and boundary regions. In applications, it is necessary to select the relevant number of

Fig. 5

Learning layers and their approximations.



layers and regions representing layers before the approximations can be constructed. Optimization strategies supported by domain knowledge are used in searching for layers to obtain the relevant patterns for the given application.

One of the very active research areas in which rough/fuzzy computing plays a very important role is GC (see, e.g., Bargiela and Pedrycz 2003; Pedrycz et al. 2008). In fuzzy logic, everything may be graduated and granulated. The concepts of graduation, granulation, and graduated granulation play a key role in GC. In GC, computations are performed on granules that are clumps of objects drawn together by indistinguishability, similarity, proximity, or functionality. Graduated granulation is inspired by the way in which humans deal with complexity and imprecision (Zadeh 2007). In searching for different relevant granules, rough set methods can be used making it possible to construct, e.g., complex classifiers of high quality. The goal of computations is to construct granules satisfying a given specification (generalized constraints), often expressed in natural language, to a satisfactory degree (Polkowski and Skowron 1996; Zadeh 2006).

5 Wisdom Technology

One of the potential areas for many further applications of rough/fuzzy computing is Wistech discussed in Jankowski and Skowron (2007, 2008a, b) as one of the main paradigms for the development of modern intelligent systems. It will be useful to present some basic ideas of

Wistech because in our opinion Wistech is also very important for developing new methods in natural computing (Rozenberg 2008; Cooper et al. 2008).

There are many indications that we are currently witnessing the onset of an era of radical technological changes. These radical changes depend on the further advancement of technology to acquire, represent, store, process, discover, communicate, and learn “wisdom.” We call this technology *wisdom technology* (or Wistech, for short). The term *wisdom* commonly means *rightly judging*. This common notion can be refined. By *wisdom*, we understand an adaptive ability to make judgments correctly to a satisfactory degree (in particular, correct decisions) having in mind real-life constraints. The intuitive nature of wisdom understood in this way can be expressed by the so-called *wisdom equation* (Jankowski and Skowron 2007), as shown metaphorically in (44):

$$\text{Wisdom} = \text{knowledge} + \text{adaptive judgment} + \text{interactions} \quad (44)$$

Wisdom can be treated as a special type of knowledge processing. To explain the specificity of this type of knowledge processing, let it be assumed that a control system of a given agent Ag consists of a society of agent control components interacting with the other agent Ag 's components and with the agent Ag 's environments. Moreover, there are special agent components called the agent coordination control components that are responsible for the coordination of control components. Any agent coordination control component mainly searches for answers to the following question: What to do next? Or, more precisely: Which of the agent Ag control components should be activated now? Of course, any agent control component has to process some kind of knowledge representation. In the context of agent perception, the agent Ag itself (by using, e.g., interactions, memory, and coordination among control components) is processing a very special type of knowledge reflecting the agent perception of the hierarchy of needs (objectives, plans, etc.) and the current agent or the environment constraints. This kind of knowledge processing mainly deals with complex vague concepts (such as risk or safety) from the point of view of the selfish agent needs. Usually, this kind of knowledge processing is not necessarily logical reasoning in terms of proving statements (i.e., labeling statements by truth values such as TRUE or FALSE). This knowledge processing is rather analogous to the judgment process in a court aiming at recognition of evidence that could be used as an argument for or against. Arguments for or against are used in order to make the final decision as to which one of the solutions is the best for the agent in the current situation (i.e., arguments are labeling statements by judgment values expressing the action priorities). The evaluation of current needs by agent Ag is realized by using a hierarchy of agent Ag life values/needs). Wisdom style knowledge processing by the agent Ag is characterized by the ability to improve the quality of the judgment process based on the agent Ag 's experiences. In order to emphasize the importance of this ability, we use the concept of adaptive judgment in the wisdom equation instead of just judgment. An agent who is able to perform adaptive judgment in the above sense is simply called a judge.

The adaptivity aspects are also crucial from the point of view of interactions (Goldin et al. 2006; Nguyen and Skowron 2008; Skowron 2008; Skowron and Szczyka 2010). The need for adaptation follows, e.g., from the fact that complex vague concepts on the basis of which the judgment is performed by the agent Ag are approximated by classification algorithms (classifiers), which should drift in time following changes in data and represented knowledge.

An important aspect of Wistech is that the complexity and uncertainty of real-life constraints mean that in practice we must reconcile ourselves to the fact that our judgment is based on non-crisp concepts (i.e., concepts with borderline cases) and also does not take into

account all the accumulated and available knowledge. This is why our judgments are usually imperfect. But, as a consolation, we also learn to improve the quality of our judgments via observation and analysis of our experience during interaction with the environment. Satisfactory decision-making levels can be achieved as a result of improved judgments.

Thus, wisdom is directly responsible for the focusing of an agent's attention on problems and techniques for their solution that are important in terms of the agent's judgment mechanism. This mechanism is based on the Maslow hierarchy of needs (Jankowski and Skowron 2008a) and agent perception of ongoing interactions with other agents and environments. In particular, the agent's wisdom can be seen as the control at the highest level of hierarchy of the agent's actions and reactions, and is based on concept processing in the metaphoric Aristotle tetrahedron. One can use the following conceptual simplification of agent wisdom: Agent wisdom is an efficient and an online agent judgment mechanism making it possible for agents to answer the following questions: (1) How to construct the most important priority list of problems to be solved? (2) How to solve the top priority problems under real-life constraints? (3) What to do next?

One of the main barriers hindering an acceleration in the development of Wistech applications lies in developing satisfactory computational models implementing the function of *adaptive judgment*. This difficulty primarily consists in overcoming the complexity of integrating the local assimilation and processing of dynamically changing, non-crisp, and incompletely specified concepts necessary to make correct judgments. In other words, we are only able to model tested phenomena using local (subjective) models and interactions between them. In practical applications, usually, we are not able to give perfect global models of analyzed phenomena. However, global models can only be approximated by integrating the various incomplete perspectives of problem perception.

Wistech is based on techniques of reasoning about knowledge, information, and data, which helps apply the current knowledge to problem solving in real-life highly unpredictable environments and autonomous multiagent systems. This includes such methods as identification of the current situation on the basis of interactions or dialogs, extraction of relevant fragments of knowledge from knowledge networks, judgment for prediction for relevant actions or plans in the current situation, or judgment of the current plan reconfiguration.

The concepts surrounding Wistech can be summarized as follows.

Wisdom technology (*Wistech*) is a collection of techniques aimed at the further advancement of technology to acquire, represent, store, process, discover, communicate, and learn *wisdom* in the design and implementation of intelligent systems. These techniques include approximate reasoning by agents or teams of agents about vague concepts concerning real-life dynamically changing, usually distributed, systems in which these agents are operating. Such systems consist of other autonomous agents operating in highly unpredictable environments and interacting with each other. Wistech can be treated as the successor of database technology, information technology, and knowledge management technologies. Wistech is the combination of the technologies represented in (44) and offers an intuitive starting point for a variety of approaches to designing and implementing computational models of Wistech in intelligent systems.

Knowledge technology in Wistech is based on techniques for reasoning about knowledge, information, and data, which helps apply the current knowledge in problem solving. This includes, e.g., extracting relevant fragments of knowledge from knowledge networks for making decisions or reasoning by analogy.

Judgment technology in Wistech covers the representation of agent perception and adaptive judgment strategies based on results of the perception of real-life scenes in environments and their representations in the agent's mind. The role of judgment is crucial, e.g., in adaptive planning relative to the Maslow hierarchy of the agent's needs or goals. Judgment also includes techniques used for perception, analysis of perceived facts, learning, and adaptive improving of approximations of vague complex concepts (from different levels of concept hierarchies in real-life problem solving) applied to modeling interactions in dynamically changing environments (in which cooperating, communicating, and competing agents exist) by using uncertain and insufficient knowledge or resources.

Interaction technology includes techniques for performing and monitoring actions by agents and environments. Techniques for planning and controlling actions are the result of the combination of judgment and interaction technologies.

Clearly, one of the promising possible ways to build Wistech computational models is by using methods of rough/fuzzy computing and other soft computing approaches.

6 Conclusions

Some basic concepts related to rough/fuzzy computing have been outlined. There are numerous research directions based on the foundations and applications of rough sets, fuzzy sets, and their combinations. The interested reader is referred, e.g., to Pawlak and Skowron (2007a, b, c), Dubois and Prade (2000), Hoehle and Rodabaugh (1999), Bezdek et al. (1999a, b), Słowiński (1998), Nguyen and Sugeno (1998), Zimmermann (1999), Klir and Yuan (1995), Klir (2006), Pedrycz and Gomide (2007), and Nikravesh et al. (2007, 2008), and the bibliographies in these papers and books.

A large part of knowledge related to natural computing is expressed by vague concepts. Hence, natural computing seems to be the domain for future potential applications of rough/fuzzy computing, especially in the framework of Wistech.

Acknowledgments

The author would like to express his gratitude to Professor Dave Corne for suggestions and corrections helping to improve this chapter.

This research has been supported by the grant N N516 368334 from the Ministry of Science and Higher Education of the Republic of Poland.

References

- Bargiela A, Pedrycz W (2003) Granular computing: an introduction. Kluwer, Dordrecht
- Bazan J (2008a) Hierarchical classifiers for complex spatio-temporal concepts. In: Transactions on rough sets IX. Lecture notes in computer science, vol. 5390. Springer, Berlin, pp 470–450
- Bazan J (2008b) Rough sets and granular computing in behavioral pattern identification and planning. In: Pedrycz W, Skowron A, Kreinovich V (eds) Handbook of granular computing. Wiley, New York, pp 777–800
- Bazan J, Skowron A, Swiniarski R (2006) Rough sets and vague concept approximation: from sample approximation to adaptive learning. In: Transactions on rough sets V. Lecture notes in computer science, vol 4100. Springer, Berlin, pp 39–62

- Bazan JG (1998) A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables. In: Polkowski L, Skowron A (eds) *Rough sets in knowledge discovery 1: methodology and applications*. Studies in fuzziness and soft computing, vol 18. Physica, Heidelberg, pp 321–365
- Bazan JG, Nguyen HS, Nguyen SH, Synak P, Wróblewski J (2000) Rough set algorithms in classification problems. In: Polkowski L, Lin TY, Tsumoto S (eds) *Rough set methods and applications: new developments in knowledge discovery in information systems*. Studies in fuzziness and soft computing, vol 56. Springer/Physica, Heidelberg, pp 49–88
- Bezdek J, Dubois D, Prade H (eds) (1999a) *Fuzzy sets in approximate reasoning and information systems*. In: Dubois D, Prade H (series eds) *Handbook of fuzzy sets series*, vol 2. Kluwer, Boston/Dordrecht
- Bezdek J, Pal N, Keller J, Krishnapuram R (eds) (1999b) *Fuzzy set models for pattern recognition and image processing*. In: Dubois D, Prade H (series eds) *Handbook of fuzzy sets series*, vol 2. Kluwer, Boston/Dordrecht
- Black M (1937) Vagueness: An exercise in logical analysis. *Philos Sci* 4(4):427–455
- Cooper SB, Löwe B, Sorbi A (eds) (2008) *New computational paradigms, changing conceptions of what is computable*. Springer, New York
- Dubois D, Prade H (1987) Twofold fuzzy sets and rough sets—some issues in knowledge representation. *Fuzzy Set Syst* 23(1):3–18
- Dubois D, Prade H (1988) Fuzzy rough sets. Note on Mult.-Valued Logic in Japan 9(8):1–8
- Dubois D, Prade H (1990) Rough fuzzy sets and fuzzy rough sets. *Int J Gen Syst* 17(2–3):191–209
- Dubois D, Prade H (eds) (2000) *Fundamentals of fuzzy sets*. In: Dubois D, Prade H (series eds) *Handbook of fuzzy sets series*, vol 1. Kluwer, Boston/Dordrecht
- Frege G (1903) *Grundgesetze der Arithmetik*, 2. Verlag von Hermann Pohle, Jena
- Goguen J (1967) L-fuzzy sets. *J Math Anal Appl* 18:145–174
- Goldin D, Smolka S, Wegner P (2006) Interactive computation: the new paradigm. Springer, Heidelberg
- Greco S, Matarazzo B, Slowinski R (1998) Fuzzy similarity relation as a basis for rough approximations. In: Polkowski L, Skowron A (eds) *Rough sets and current trends in computing*. Lecture notes in computer science, vol 1424. Springer, Berlin, pp 283–289
- Greco S, Matarazzo B, Slowinski R (1999) The use of rough sets and fuzzy sets in MCDM. In: Gal T, Stewart T, Hanne T (eds) *Advances in MCDM models, algorithms, theory, and applications*. Kluwer, Dordrecht, pp 14.1–14.59
- Greco S, Matarazzo B, Slowinski R (2000) Fuzzy extension of the rough set approach to multicriteria and multiattribute sorting. In: Fodor J, Baets BD, Perny P (eds) *Preferences and decisions under incomplete knowledge*. Physica, Heidelberg, pp 131–151
- Greco S, Inuiguchi M, Slowinski R (2006) Fuzzy rough sets and multiple-premise gradual decision rules. *Int J Approx Reason* 41(2):179–211
- Grzymała-Busse JW (1998) LERS—a knowledge discovery system. In: Polkowski L, Skowron A (eds) *Rough sets in knowledge discovery 2. Applications, case studies and software systems*. Studies in fuzziness and soft computing. Physica, Heidelberg, pp 562–565
- Hastie T, Tibshirani R, Friedman JH (2001) *The elements of statistical learning: data mining, inference, and prediction*. Springer, Heidelberg
- Hoehle U, Rodabaugh S (eds) (1999) *Mathematics of fuzzy sets: Logic, topology and measure theory*. In: Dubois D, Prade H (series eds) *Handbook of fuzzy sets series*, vol 2. Kluwer, Boston/Dordrecht
- Inuiguchi M, Greco S, Slowinski R (2004) Fuzzy-rough modus ponens and modus tollens as a basis for approximate reasoning. In: Tsumoto S, Slowinski R, Komorowski HJ, Grzymała-Busse JW (eds) *Rough sets and current trends in computing. Lecture notes in computer science*, vol 3066. Springer, Berlin, pp 84–94
- Jankowski A, Skowron A (2007) A Wistech paradigm for intelligent systems. In: *Transactions on rough sets VI. Lecture notes in computer science*, vol 4374. Springer, Berlin, pp 94–132
- Jankowski A, Skowron A (2008a) Logic for artificial intelligence: the Rasiowa-Pawlak school perspective. In: Ehrenfeucht A, Marek V, Srebrny M (eds) Andrzej Mostowski and foundational studies. IOS Press, Amsterdam, pp 106–143
- Jankowski A, Skowron A (2008b) Wisdom granular computing. In: Pedrycz W, Skowron A, Kreinovich V (eds) *Handbook of granular computing*. Wiley, New York, pp 329–346
- Jankowski A, Peters J, Skowron A, Stepaniuk J (2008) Optimization in discovery of compound granules. *Fund Inform* 85(1–4):249–265
- Keefe R (2000) *Theories of vagueness*. Cambridge University Press, Cambridge
- Klir G, Yuan B (1995) *Fuzzy logic: theory and applications*. Prentice-Hall, Englewood Cliffs, NJ
- Klir GJ (ed) (2006) *Uncertainty and information: foundations of generalized information theory*. Wiley, Hoboken, NJ
- Leśniewski S (1929) *Grundzüge eines neuen Systems der Grundlagen der Mathematik*. Fund Math 14:1–81
- Lingras P, Jensen R (2007) Survey of rough and fuzzy hybridization. In: *Preferences and decisions under*

- incomplete knowledge. FUZZ-IEEE 2007: Proceedings of 2007 IEEE international conference on fuzzy systems. Imperial College, London, 23–26 July, pp 125–130
- Lukasiewicz J (1970) Die logischen Grundlagen der Wahrscheinlichkeitsrechnung, Kraków 1913. In: Borkowski L (ed) Jan Łukasiewicz – selected works. North Holland, Amsterdam/London
- Maji P, Pal SK (2005) Rough-fuzzy c-medoids algorithm and selection of bio-basis for amino acid sequence analysis. *IEEE T Knowl Data Eng* 19(6): 859–872
- Maji P, Pal SK (2007) RFCM: A hybrid clustering algorithm using rough and fuzzy sets. *Fund Inform* 80(4):475–496
- Nanda S (1992) Fuzzy rough-sets. *Fuzzy Set Syst* 45:157–160
- Nguyen H, Sugeno M (eds) (1998) Fuzzy systems modelling and control. In: Dubois D, Prade H (series eds) Handbook of fuzzy sets series, vol 6. Kluwer, Boston/Dordrecht
- Nguyen HS (1998) From optimal hyperplanes to optimal decision trees. *Fund Inform* 34(1–2):145–174
- Nguyen HS (2002) Scalable classification method based on rough sets. In: Alpigini JJ, Peters JE, Skowron A, Zhong N (eds) Rough sets and current trends in computing. Lecture notes in computer science, vol 2475. Springer, Berlin, pp 433–440
- Nguyen HS (2006) Approximate Boolean reasoning: Foundations and applications in data mining. In: Transactions on rough sets V. Lecture notes in computer science, vol 4100. Springer, Berlin, pp 334–506
- Nguyen HS, Skowron A (1997) Boolean reasoning for feature extraction problems. In: Ras ZW, Skowron A (eds) ISMIS. Lecture notes in computer science, vol 1325. Springer, Berlin, pp 117–126
- Nguyen HS, Skowron A (2008) A rough granular computing in discovery of process models from data and domain knowledge. *J Chongqing Univ* 20(3):341–347
- Nguyen SH, Nguyen HS (1998) Pattern extraction from data. *Fund Inform* 34(1–2):129–144
- Nikravesh M, Kacprzyk J, Zadeh LA (eds) (2007) Forging new frontiers: Fuzzy pioneers I. In: Studies in fuzziness and soft computing, vol 217. Springer, Heidelberg
- Nikravesh M, Kacprzyk J, Zadeh LA (eds) (2008) Forging new frontiers: Fuzzy pioneers II. In: Studies in fuzziness and soft computing, vol 218. Springer, Heidelberg
- Pal S, Banerjee M (1996) Roughness of a fuzzy set. *Inform Sci* 93(3):235–246
- Pal SK (2003) Rough-fuzzy granular computing, case based reasoning and data mining. In: Gesù VD,
- Masulli F, Petrosino A (eds) WILF. Lecture notes in computer science, vol 2955. Springer, Berlin, pp 1–10
- Pal SK, Skowron A (eds) (1999) Rough fuzzy hybridization: a new trend in decision-making. Springer, Singapore
- Pawlak Z (1982) Rough sets. *Int J Comput Inf Sci* 11:341–356
- Pawlak Z (1991) Rough sets: theoretical aspects of reasoning about data. In: System theory, knowledge engineering and problem solving, vol 9. Kluwer, Dordrecht
- Pawlak Z, Skowron A (2007a) Rough sets and Boolean reasoning. *Inform Sci* 177(1):41–73
- Pawlak Z, Skowron A (2007b) Rough sets: some extensions. *Inform Sci* 177(1):28–40
- Pawlak Z, Skowron A (2007c) Rudiments of rough sets. *Inform Sci* 177(1):3–27
- Pedrycz W, Gomide F (2007) Fuzzy systems engineering toward human-centric computing. Wiley, Hoboken, NJ
- Pedrycz W, Skowron A, Kreinovich V (eds) (2008) Handbook of granular computing. Wiley, New York
- Polkowski L (ed) (2002) Rough sets: mathematical foundations. Advances in soft computing. Physica, Heidelberg
- Polkowski L, Skowron A (1996) Rough mereology: a new paradigm for approximate reasoning. *Int J Approx Reason* 51:333–365
- Read S (1994) Thinking about logic: an introduction to the philosophy of logic. Oxford University Press, Oxford
- Rozenberg G (2008) Computer science, informatics, and natural computing – personal reflections. In: Cooper SB, Löwe B, Sorbi A (eds) New computational paradigms changing conceptions of what is computable. Springer, New York, pp 373–379
- Russell B (1923) Vagueness. *Austral J Psychol Philos* 1:84–92
- Skowron A (2002) Rough sets in KDD – plenary talk. In: Shi Z, Faltings B, Musen M (eds) IFIP'00: 16-th world computer congress: IIP'00, Proceedings of conference on intelligent information processing. Publishing House of Electronic Industry, Beijing, pp 1–14
- Skowron A (2005) Rough sets and vague concepts. *Fund Inform* 64(1–4):417–431
- Skowron A (2008) Learning complex granules and their interactions. In: Nguyen HS, Huynh VN (eds) SCKT 2008: International workshop on soft computing for knowledge technology at the 10-th Pacific Rim international conference on artificial intelligence, 15–19 May 2008. Hanoi, Vietnam, pp 1–14
- Skowron A, Stepaniuk J (1996) Tolerance approximation spaces. *Fund Inform* 27:245–253
- Skowron A, Stepaniuk J (2003) Information granules and rough-neural computing. In: Pal SK, Polkowski L, Skowron A (eds) Rough-neural computing:

- techniques for computing with words. Cognitive technologies. Springer, Berlin, pp 43–84
- Skowron A, Szczuka M (2010) Toward interactive computations: a rough-granular approach. In: Koronacki J, Ras Z, Wierzchon S, Kacprzyk J (eds) Advances in machine learning II, Dedicated to the memory of Professor Ryszard S. Michalski. Studies in computational intelligence, vol. 263. Springer, Heidelberg, pp 23–42
- Słowiński R (ed) (1998) Fuzzy sets in decision analysis, operations research & statistics. In: Dubois D, Prade H (series eds) Handbook of fuzzy sets series, vol 5. Kluwer, Boston/Dordrecht
- Triantaphyllou E, Felici G (eds) (2006) Data mining and knowledge discovery approaches based on rule induction techniques. Springer, New York
- Wu WZ, Mi JS, Zhang WX (2003) Generalized fuzzy rough sets. *Inform Sci* 151:263–282
- Zadeh L (2007) Granular computing and rough set theory. In: Kryszkiewicz M, Peters JF, Rybiński H, Skowron A (eds) RSEISP 2007: International conference rough sets and intelligent systems paradigms, Warsaw, Poland, 28–30 June 2007. Lecture notes in artificial intelligence, vol 4585. Springer, Heidelberg, pp 1–4
- Zadeh LA (1965) Fuzzy sets. *Inform Control* 8:338–353
- Zadeh LA (2001) A new direction in AI – toward a computational theory of perceptions. *AI Mag* 22(1):73–84
- Zadeh LA (2006) Generalized theory of uncertainty (GTU)-principal concepts and ideas. *Comput Stat Data Anal* 51:15–46
- Zimmermann H (ed) (1999) Practical applications of fuzzy technologies. In: Dubois D, Prade H (series eds) Handbook of fuzzy sets series, vol 7. Kluwer, Boston/Dordrecht

58 Collision-Based Computing

Andrew Adamatzky¹ · Jérôme Durand-Lose²

¹Department of Computer Science, University of the West of England,
Bristol, UK

andrew.adamatzky@uwe.ac.uk

²LIFO, Université d'Orléans, France
jerome.durand-lose@univ-orleans.fr

1	<i>Introduction</i>	1950
2	<i>Principles of Collision-Based Computing</i>	1952
3	<i>Collision-Based Computing in Natural Systems</i>	1953
4	<i>One-Dimensional Cellular Automata</i>	1962
5	<i>Abstract Geometrical Computation</i>	1969

Abstract

Collision-based computing is an implementation of logical circuits, mathematical machines, or other computing and information processing devices in homogeneous, uniform and unstructured media with traveling mobile localizations. A quanta of information is represented by a compact propagating pattern (gliders in cellular automata, solitons in optical systems, wave fragments in excitable chemical systems). Logical truth corresponds to presence of the localization, logical false to absence of the localization; logical values can also be represented by a particular state of the localization. When two or more traveling localizations collide, they change their velocity vectors and/or states. Post-collision trajectories and/or states of the localizations represent results of logical operations implemented by the collision. One of the principal advantages of the collision-based computing medium – hidden in 1D systems but obvious in 2D and 3D media – is that the medium is architecture-less: nothing is hardwired, there are no stationary wires or gates, a trajectory of a propagating information quanta can be seen as a momentary wire. The basics of collision-based computing are introduced, and the collision-based computing schemes in 1D and 2D cellular automata and continuous excitable media are overviewed. Also a survey of collision-based schemes, where particles/collisions are dimensionless, is provided.

1 Introduction

Edward Fredkin, Tommaso Toffoli, Norman Margolus, Elwyn Berlekamp and John Conway are fully recognized founders of the field of collision-based computing. The term “collision-based computing” was first used in Adamatzky (2002a). There are, however, several equivalent but lesser-used terms such as “signal computing,” “ballistic computing,” “free space computing,” and “billiard ball computing.” The idea of collision-based computing is based on studies dealing with collisions of signals traveling along discrete chains, on a one-dimensional cellular automata. The interaction of signals traveling along one-dimensional conductors was among the famous problems in physics, biology, and physiology for centuries, and the problem of interaction was interpreted in terms of finite state machines in the 1960s. The earliest computer science-related results on signal interaction can be attributed to:

- Atrubin (Atrubin 1965), who designed the first-ever multiplier based on a one-dimensional cellular automaton in 1965;
- Fischer (Fischer 1965), who developed a cellular automaton generator of prime numbers in 1965; and
- Waksman (Waksman 1966), who initiated the very popular firing squad synchronization problem and provided an eight-state solution, in 1966.

Banks (1971) showed how to build wires and simple gates in configurations of a two-dimensional binary-state cellular automaton. This was not architecture-free computing, because a wire was represented by a particular stationary configuration of cell states (this was rather a simulation of a “conventional” electrical, or logical, circuit). However, Banks’s design was a huge influence on the theory of computing in cellular automata and beyond.

In 1982, Elwyn Berlekamp, John Conway, and Richard Gay proved that Game of Life “can imitate computers” (Berlekamp et al. 1982).

They mimicked electric wires by lines “along which gliders travel” and demonstrated how to do a logical gate by crashing gliders into one another. Chapter 25 of their “Winning Ways” (Berlekamp et al. 1982) demonstrates computing designs that do not simply look fresh 20 years later but are still rediscovered again and again by Game of Life enthusiasts all over the Net.

Berlekamp, Conway, and Guy employed a vanishing reaction of gliders – two crashing gliders annihilate themselves – to build a NOT gate. They adopted Gosper’s eater to collect garbage and to destroy glider streams. They used combinations of glider guns and eaters to implement AND and OR gates, and the shifting of a stationary pattern or block, by a mobile pattern, or glider, when designing auxiliary storage of information.

There is even the possibility that space–time itself is granular, composed of discrete units, and that the universe, as Edward Fredkin of M.I.T. and others have suggested, is a cellular automaton run by an enormous computer. If so, what we call motion may be only simulated motion. A moving particle in the ultimate microlevel may be essentially the same as one of our gliders, appearing to move on the macro-level, whereas actually there is only an alteration of states of basic space–time cells in obedience to transition rules that have yet to be discovered. – Berlekamp et al. (1982).

Meanwhile, in 1978, Edward Fredkin and Tommaso Toffoli submitted a 1-year project proposal to DARPA, which got funding and thus started a chain of remarkable events.

Originally, Fredkin and Toffoli aimed to “drastically reduce the fraction of” energy “that is dissipated at each computing step” (Fredkin and Toffoli 2002). To design a non-dissipative computer they constructed a new type of digital logic – conservative logic – that conserves both “the physical quantities in which the digital signals are encoded” and “the information present at any moment in a digital system” (Fredkin and Toffoli 2002).

Fredkin and Toffoli (1982) further developed these ideas in the seminal paper “Conservative Logic,” from which a concept of ballistic computers emerged. The Fredkin–Toffoli model of conservative computation – the billiard ball model – explores “elastic collisions involving balls and fixed reflectors.” Generally, they proved that “given a container with balls one can do any kind of computation.”

The billiard ball model became a masterpiece of cellular automaton theory, thanks to Norman Margolus who invented a cellular automaton (block cellular automata or partitioned cellular automata) implementation of the model. Norman published this result in 1984 (Margolus 1984). “Margolus neighborhood” and “billiard ball model cellular automata” are exploited widely nowadays.

A detailed account of collision-based computing is not provided. There are no excuses to avoid reading original sources (Berlekamp et al. 1982; Fredkin and Toffoli 1982; Margolus 1984). A comprehensive, self-contained report of the modern state of collision-based computing is provided in the book by Adamatzky (2002a). The present chapter rather discusses personal experience designing collision-based computing schemes in one- and two-dimensional cellular automata, and spatially extended nonlinear media. It also provides some hands-on examples of recently discovered collision-based computing devices, with the hope that the examples will help readers to experiment with their own designs.

The chapter is structured as follows. Principles of collision-based computing are outlined in [Sect. 2](#). [Section 3](#) shows how basic logical gates can be implemented by colliding localizations in natural systems – simulated reaction–diffusion medium ([Sect. 3.1](#)) and light-sensitive Belousov–Zhabotinsky medium ([Sect. 3.2](#)). The theoretical foundations of computing with signals in 1D cellular automata are presented in [Sect. 4](#), including

collision-based implementation of 1D Turing machine (☞ Sect. 4.2.1) and cyclic tag systems (CTS) (☞ Sect. 4.2.2). The excursion to architectureless computing is complete with abstract geometrical computation in ☞ Sect. 5, where time and space are continuous and particles/localizations are dimensionless.

2 Principles of Collision-Based Computing

This section attempts to summarize all types of collision-based computers. A collision-based computer is an empty space populated with mobile and stationary localizations. The mobile localizations used for computing so far are:

- Billiard balls (Fredkin and Toffoli 1982)
- Gliders in cellular automata models (Berlekamp et al. 1982; Adamatzky and Wuensche 2007; Delorme and Mazoyer 2002; Rennard 2002; Rendell 2002)
- Solitons in nonlinear media (Jakubowski et al. 1996, 2001; Steiglitz 2001; Anastassiou et al. 2001; Rand et al. 2005; Rand and Steiglitz 2009), and
- Localized wave fragments in excitable chemical media (Adamatzky 2004; Adamatzky and De Lacy Costello 2007)

Examples of stationary localizations are:

- “Still lives,” blocks, and eaters in Conway’s Game of Life (Berlekamp et al. 1982; Rendell 2002)
- Standing waves in automaton models of reaction–diffusion systems (Adamatzky and Wuensche 2007) and
- Breathers in computing devices based on polymer chains and oscillons in vibrating granular materials (Adamatzky 2002b)

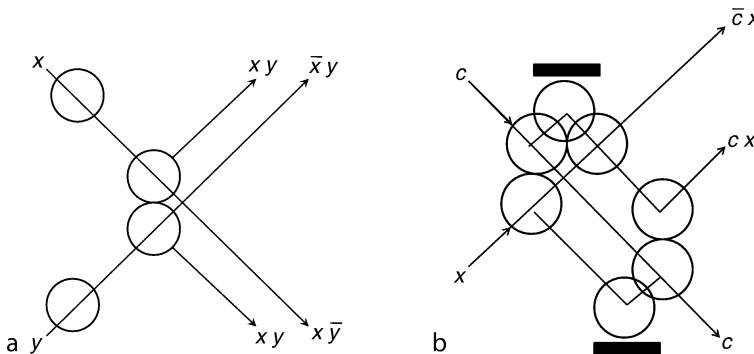
Usually mobile localizations represent signals and stationary localizations are used to route the signals in the space; however, by allowing balls and mirrors to change their states, in addition to velocity vectors, one can, in principle, build multivalued logic circuits.

Classical examples of collision-based gates are shown in ☞ Fig. 1. The interaction gate involves two balls to represent values of variables x and y (☞ Fig. 1a). If two balls are present at the input trajectories, this corresponds to both variables having TRUTH values that collide and deflect as a result of collisions. The deflected trajectories of the balls represent conjunctions of the input variables, xy . If only one ball—say the ball corresponding to the variable x —is present initially, then this ball travels along its original trajectories and does not change its velocity vector. The undisturbed trajectories of the balls represent logical functions $\bar{x}y$ and $x\bar{y}$ (☞ Fig. 1a).

The switch gate (☞ Fig. 1b) is another famous example, which also demonstrates the role of mirrors (stationary localizations). In the switch gate, the signal x is conditionally routed by a control signal c . If the signal is not present, the ball x continues along its original trajectory traveling southeast. The ball x is delayed and its trajectory shifts eastward if the signal c is present in the system. After collision, balls x and c are reflected but their further propagation is restricted by mirrors: the ball c collides with the Northern mirror and the ball x with the Southern mirror (☞ Fig. 1b).

Fig. 1

Basics of billiard ball model: Fredkin–Toffoli interaction gate (a) and switch gate (b). (From Fredkin and Toffoli 1982.)



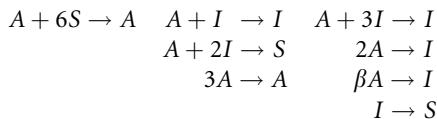
3 Collision-Based Computing in Natural Systems

Those focused on implementation issues may wonder how a scheme of collision-based computing can be implemented in natural, chemical, physical, or biological materials. This section provides two examples of computing with localizations in spatially extended quasi-chemical (reaction–diffusion cellular automata) and excitable chemical (Belousov–Zhabotinsky reaction) systems.

3.1 Computing Schemes in Reaction–Diffusion Cellular Automata: Spiral Rule

The reaction–diffusion cellular automaton Spiral Rule (Adamatzky and Wuensche 2007) exhibits a wide range of mobile and stationary localizations, thus offering unique opportunities to employ both traveling and still patterns in a computation process.

An automaton is designed that emulates nonlinearity of activator (A) and inhibitor (I) interaction for subthreshold concentrations of activator. The following quasi-chemical reaction was used to construct cell-state transition rules (Adamatzky and Wuensche 2007):



For subthreshold concentration of the inhibitor and threshold concentrations of activator, the activator is suppressed by the inhibitor. For critical concentrations of the inhibitor, both inhibitor and activator dissociate, producing the substrate.

The quasi-chemical reactions are mapped to cellular automaton rules as follows. Take a totalistic hexagonal cellular automaton (CA), where a cell takes three states – substrate S ,

activator A , and inhibitor I – and the cell updates its state depending on just the numbers of different cell-states in its neighborhoods. The update rule can be written as follows:

$$x^{t+1} = f(\sigma_I(x)^t, \sigma_A(x)^t, \sigma_S(x)^t)$$

where $\sigma_p(x)^t$ is the number of cell x 's neighbors (in seven cells neighborhood) with cell-state $p \in \{I, A, S\}$ at time step t . The rule is compactly represented as a matrix $M = (M_{ij})$, where $0 \leq i, j \leq 7$, $0 \leq i + j \leq 7$, and $M_{ij} \in \{I, A, S\}$ (Adamatzky et al. 2006). The output state of each neighborhood is given by the row-index i (the number of neighbors in cell-state I) and column-index j (the number of neighbors in cell-state A). One does not have to count the number of neighbors in cell-state S , because it is given by $7 - (i + j)$. A cell with a neighborhood represented by indices i and j will update to cell-state M_{ij} , which can be read off the matrix. In terms of the cell-state transition function, this can be presented as follows: $x^{t+1} = M_{\sigma_2(x)^t \sigma_1(x)^t}$. The exact structure of the transition matrix is as follows (Adamatzky and Wuensche 2007):

$$M = \left(\begin{array}{ccccccc} S & A & I & A & I & I & I \\ S & I & I & A & I & I & I \\ S & S & I & A & I & I & I \\ S & I & I & A & I & & \\ S & S & I & A & & & \\ S & S & I & & & & \\ S & S & & & & & \\ S & & & & & & \end{array} \right)$$

The entry $M_{01} = A$ symbolizes the diffusion of activator A , $M_{11} = I$ represents the suppression of activator A by the inhibitor I , and $M_{z2} = I$ ($z = 0, \dots, 5$) is the self-inhibition of the activator in particular concentrations. $M_{z0} = S$ ($z = 1, \dots, 7$) means that the inhibitor is dissociated in the absence of the activator, and that the activator does not diffuse in subthreshold concentrations; $M_{zp} = I$, $p \geq 4$ is an upper-threshold self-inhibition.

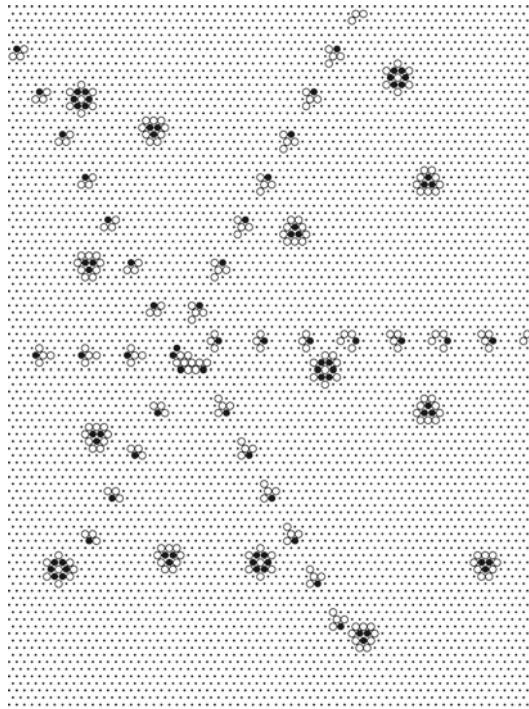
Starting in a random initial configuration, the automaton will evolve towards a quasi-stationary configuration, with typically two types of stationary localizations, and a spiral generator of mobile localizations, or gliders (Fig. 2). The core of a glider-gun is a discrete analog of a “classical” spiral wave. However, at some distance from the spiral wave tip, the wave front becomes unstable and splits into localized wave fragments. The wave fragments continue traveling along their originally determined trajectories and keep their shape and velocity vector unchanged unless disturbed by other localizations. So, the wave fragments behave as in sub-excitable Belousov–Zhabotinsky systems.

Basic gliders, those with one (activator) head, are found in five types (Fig. 3), which vary by the number of trailing inhibitors. Three types (G_{34}, G_{24}, G_{43}) alternate between two forms. Two types (G_4, G_5) have just one form. The spiral glider-gun in Fig. 2 emits G_{34} gliders. An alternative, low frequency, spiral glider-gun (Wuensche and Adamatzky 2006) (not shown) releases G_4 gliders. These basic gliders, and also a variety of more complicated gliders including mobile glider-guns, are also generated by many other interactions. Stationary localizations, or eaters (Fig. 3i, j), are another important feature of the CA.

The principal components of any computing device are the input interface, memory, routers, and logical gates. Readers are referred to the paper Adamatzky and Wuensche (2007) to study possible input functions.

Fig. 2

A typical quasi-stable configuration of the CA, which started its development in a random initial configuration (with 1/3 probability of each cell-state). Cell-state I (inhibitor) is shown by a black disk, cell-state A (activator) by a circle, and cell-state S (substrate) by a dot. One can see there are two types of stationary localizations (glider eaters) and a spiral glider-gun, which emits six streams of gliders, with a frequency of a glider per six time steps in each glider stream. (From Adamatzky and Wuensche 2007.)



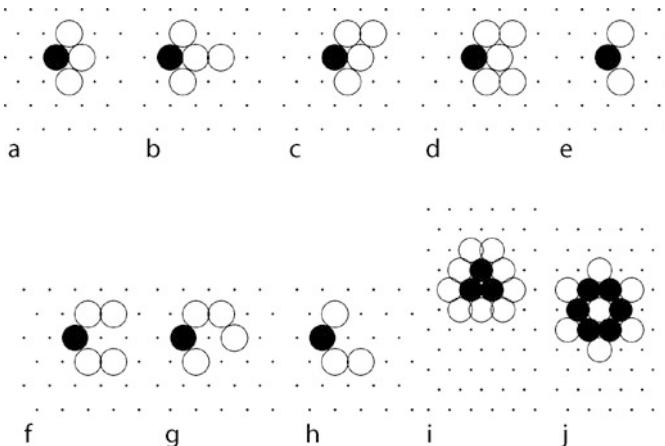
How does one implement a memory in the Spiral Rule CA? The eater E_6 can play the role of a 6-bit flip-flop memory device. The substrate-sites (bit-down) between inhibitor-sites ([Fig. 3i, j](#)) can be switched to an inhibitor-state (bit-up) by a colliding glider.

An example of writing one bit of information in E_6 is shown in [Fig. 4](#). Initially E_6 stores no information. The aim is to write one bit in the substrate-site between the northern and northwestern inhibitor-sites ([Fig. 4a](#)). A glider G_{34} is generated ([Fig. 4b, c](#)) that travels west. G_{34} collides with (or brushes past) the north edge of E_6 resulting in G_{34} being transformed to a different type of glider, G_4 ([Fig. 4g, h](#)). There is now a record of the collision – evidence that writing was successful. The structure of E_6 now has one site (between the northern and northwestern inhibitor-sites) changed to an inhibitor-state ([Fig. 4j](#)) – a bit was saved.

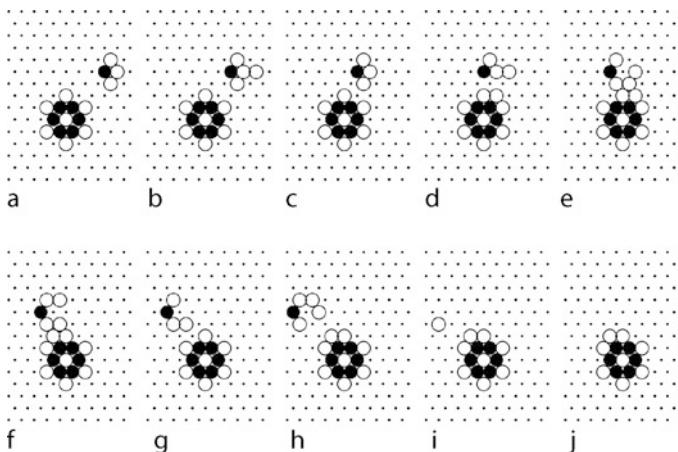
To read a bit from the E_6 memory device with one bit-up ([Fig. 5a](#)), one collides (or brushes past) with glider G_{34} ([Fig. 5b](#)). Following the collision, the glider G_{34} is transformed into a different type of basic glider, G_3 ([Fig. 5g](#)), and the bit is erased ([Fig. 5j](#)).

Fig. 3

The basic localizations in a Spiral Rule automaton: gliders (a–g) and eaters (h–i). (a–g) Five types of gliders, shown here traveling west, in the direction of their activator head (cell-state A), with a tail of trailing inhibitors made up of several cell-states I. The glider designator G_{ab} refers to the numbers of trailing inhibitors. (a) and (b) Two forms of glider G_{34} . (c) Glider G_4 . (d) Glider G_5 . (e) and (f) Two forms of glider G_{24} . (g) and (h) Two forms of glider G_{43} . (i) Eater E_3 . (j) Eater E_6 . (From Adamatzky and Wuensche 2007.)

**Fig. 4**

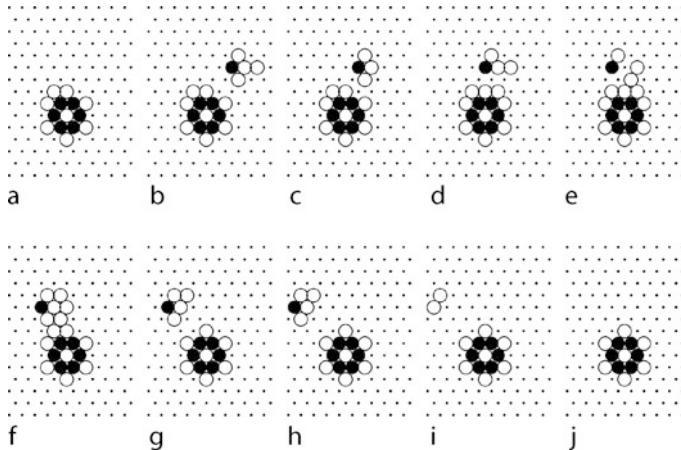
Write bit. (a) t. (b) $t + 1$. (c) $t + 2$. (d) $t + 3$. (e) $t + 4$. (f) $t + 5$. (g) $t + 6$. (h) $t + 7$. (i) $t + 8$. (j) $t + 9$. (From Adamatzky and Wuensche 2007.)



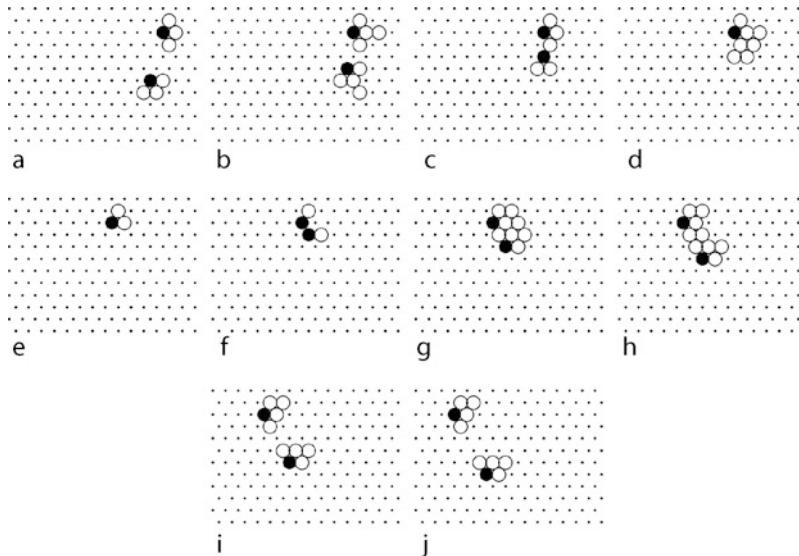
To route signals, one can potentially employ other gliders to act as mobile reflectors. **Figure 6** shows how a glider traveling northwest collides with a glider traveling west, and is reflected southwest as a result of the collision. However, both gliders are transformed to different types of gliders. This is acceptable on condition that both types of gliders represent the same signal, or signal modality.

Fig. 5

Read and erase bit. (a) t . (b) $t + 5$. (c) $t + 7$. (d) $t + 8$. (e) $t + 9$. (f) $t + 10$. (g) $t + 11$. (h) $t + 12$. (i) $t + 13$. (j) $t + 14$. (From Adamatzky and Wuensche 2007.)

**Fig. 6**

Glider reflection. (a) t . (b) $t + 1$. (c) $t + 2$. (d) $t + 3$. (e) $t + 4$. (f) $t + 5$. (g) $t + 6$. (h) $t + 7$. (i) $t + 8$. (j) $t + 9$. (From Adamatzky and Wuensche 2007.)



There are two more gates that are useful in designing practical collision-based computational schemes. They are the FANOUT gate and the ERASE gate. The FANOUT gate is based on glider multiplication. There are a few scenarios where one glider can be multiplied by another glider (for details see the original beehive rule [Wuensche 2005]); for example, one can make a FANOUT gate by colliding glider G_{34} with glider G_{24} . The gliders almost annihilate as a result of the collision, but recover into a complicated one, which splits into three G_5 gliders. To annihilate

a glider, one can collide it with the central body of an eater, or with another glider; for example, head-on collisions usually lead to annihilation.

The *asynchronous* XOR gate can be constructed from the memory device in [Fig. 4](#) and [Fig. 5](#), employing the eater E_6 and the glider G_{34} . The incoming trajectory of the gliders is an input $x = \langle x, y \rangle$ of the gate, and the state of the cell that is switched to the inhibitor state by the gliders is an output z of the gate (this cell is shown by \otimes in [Fig. 7a](#)). As seen in [Fig. 4](#), when glider G_{34} brushes by the eater E_6 it “adds” one inhibitor state to the eater configuration ([Fig. 4, t + 7](#)), and transforms itself into glider G_{43} . If glider G_{34} brushes by E_6 with an additional inhibitor state ([Fig. 5, t](#)), it “removes” this additional state and transforms itself into glider G_4 ([Fig. 5, t + 11](#)).

Assume that the presence of glider G_{34} symbolizes input logical TRUE and its absence – input FALSE, inhibitor state I in cell \otimes – output TRUE and substrate state S – output FALSE. The result of this logical operation can be read directly from the configuration of E_6 or by sending a control glider to brush by E_6 to detect how the glider is transformed. Then the structure implements exclusive disjunction ([Fig. 7b](#)). The gate constructed is asynchronous, because the output of the operation does not depend on the time interval between the signals but only on the value of signals: when the inhibitor state is added or removed from E_6 the configuration of E_6 remains stable and does not change till another glider collides into it.

The eater E_6 can take four different configurations resulting from the interactions of gliders brushing past, and there are seven types of gliders produced in collisions with the eater (including some basic types flipped). One therefore can envisage (Adamatzky and Wuensche 2007) that a finite state machine can be implemented in the eater-glider system. The internal state of such a machine is represented by the configuration of the eater, the type of the incoming glider symbolizes the input symbol of the machine, and the type of the outgoing glider represents the output state of the machine.

To construct the full state transition table of the eater-glider machine, seven types of gliders are collided into four configurations of the eater and the results of the collisions are recorded. For the sake of compact representation, the configurations of the eater are encoded as shown in [Fig. 8](#). The gliders are denoted as follows: G_{34} as a , G_{43} as b , G_5 as c , G_4 as d , G_{24} as e , G^4 (glider G_4 flipped horizontally) is f , and G^{43} (glider G_{43} flipped horizontally) is g . The state transition table is shown in [Fig. 9](#).

Consider the internal states of the eater-glider machine as unary operators on the set $\{a, b, c, d, e, f, g\}$, that is, the machine’s state is reset to its initial state after the collision with the glider. For example, the unary operator α implements the following transformation: $a \rightarrow b$, $b \rightarrow c$,

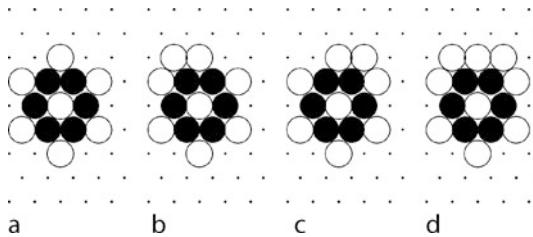
Fig. 7

Asynchronous XOR gate. (a) Position of output cell is shown by \otimes . (b) Operation implemented by the gate, input state G_{34} is logical TRUE, output state S is FALSE, output state I is TRUE. (From Adamatzky and Wuensche 2007.)

	x_1	x_2	y
G_{34}	0	0	S
G_{34}	0	G_{34}	I
G_{34}	G_{34}	I	
G_{34}	G_{34}	G_{34}	S

Fig. 8

Encoding the internal states of the glider-eater machine in the configuration of eater E_6 . (a) α . (b) β . (c) χ . (d) δ .

**Fig. 9**

The state transition table of the eater-glider machine. Tuple xy , a pair made up of an eater state x and glider state y , at the intersection of the i th row and j th column, signifies that being in state i while receiving input j the machine takes state x and generates output y .

	a	b	c	d	e	f	g
α	βb	δc	αb	αe	δd	αe	δc
β	αd	δe	βc	βc	χg	αa	χe
χ	χd	βe	δf	χa	βb	χa	βe
δ	δb	βc	χg	χe	αf	δe	αa

Fig. 10

Limit sets of unary operators a, \dots, g .

Operator	Limit set
a	$\{\alpha, \beta\}, \{\delta\}$
b	$\{\beta, \delta\}$
c	$\{\alpha\}, \{\beta\}, \{\delta, \chi\}$
d	$\{\alpha\}, \{\beta\}, \{\chi\}$
e	$\{\alpha, \delta\}, \{\beta, \chi\}$
f	$\{\alpha\}, \{\chi\}, \{\delta\}$
g	$\{\alpha, \delta\}, \{\beta, \chi\}$

$c \rightarrow a, d \rightarrow a, e \rightarrow d, f \rightarrow e, g \rightarrow e$. The operators have the following limit sets: operator α has the limit set $\{a, b, c\}$, β – set $\{c\}$, χ has two limit sets $\{a, d\}$ and $\{b, c\}$, and operator δ – two limit sets $\{a, b, c, d\}$ and $\{e, f\}$. Considering unary operators a, \dots, g operating on set $\{\alpha, \beta, \chi, \delta\}$, one obtains the limit sets shown in Fig. 10. Many of the operators have more than two limit sets, which may indicate a significant computational potential of the eater-glider machine.

To characterize the eater-glider machine in more detail, a study was conducted to find out what output strings are generated by the machine when the machine receives the uniform infinite string s^* , $s \in \{a, \dots, g\}$ on its input. These input string to output string transformations are shown in Fig. 11.

Input string $abcdefg$ evokes the following output strings when fed into the machine. The machine starting in state α generates string $begabac$, in state β string $dgcabac$, in state χ string $deccgac$, and in state δ string $bcccgae$.

Fig. 11

Input string to output string transformations implemented by the eater-glider machine. String s , at the intersection of the i th row and j th column, tells one that being initially in state i and receiving a uniform string j , the machine generates string s .

	a^*	b^*	c^*	d^*	e^*	f^*	g^*
α	$(bd)^*$	$c(ce)^*$	b^*	e^*	$(de)^*$	e^*	$(ca)^*$
β	$(db)^*$	$(ec)^*$	c^*	c^*	$(gb)^*$	ae^*	e^*
χ	d^*	$e(ec)^*$	$(fg)^*$	a^*	$b(gb)^*$	a^*	e^*
δ	b^*	$(ce)^*$	$(gf)^*$	ea^*	$(ed)^*$	e^*	$(ac)^*$

3.2 Collision-Based Computing in Excitable Chemical Media

This section gives a brief introduction to implementation of collision-based circuits in excitable chemical media, the Belousov–Zhabotinsky (BZ) system. Examples discussed here are based on numerical experiments; see the chapter [Reaction–Diffusion Computing](#) of this book for implementation of collision-based circuits in a BZ medium in chemical laboratory conditions. Now, computing is discussed with localized wave fragments in the Oregonator (Field and Noyes 1974; Tyson and Fife 1980) model adapted to a light-sensitive BZ reaction with applied illumination (Beato and Engel 2003; Krug et al. 1990):

$$\frac{\partial u}{\partial t} = \frac{1}{\varepsilon} \left(u - u^2 - (fv + \phi) \frac{u - q}{u + q} \right) + D_u \nabla^2 u, \text{ and}$$

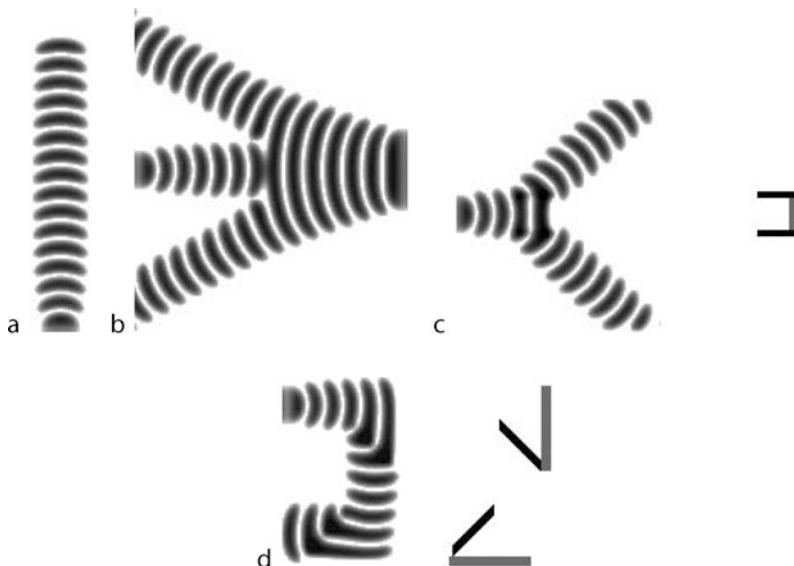
$$\frac{\partial v}{\partial t} = u - v$$

where variables u and v represent local concentrations of bromous acid HBrO_2 and the oxidized form of the catalyst ruthenium Ru(III), ε sets up a ratio of time scale of variables u and v , q is a scaling parameter depending on reaction rates, f is a stoichiometric coefficient, ϕ is a light-induced bromide production rate proportional to the intensity of illumination (an excitability parameter – moderate intensity of light will facilitate excitation process, higher intensity will produce excessive quantities of bromide which suppresses the reaction). It is assumed that the catalyst is immobilized in a thin layer of gel; therefore, there is no diffusion term for v . To integrate the system, one uses the Euler method with five-node Laplacian operator, time step $\Delta t = 10^{-3}$ and grid point spacing $\Delta x = 0.15$, with the following parameters: $\phi = \phi_0 + A/2$, $A = 0.0011109$, $\phi_0 = 0.0766$, $\varepsilon = 0.03$, $f = 1.4$, $q = 0.002$.

The chosen parameters correspond to a region of “higher excitability of the sub-excitability regime” outlined in Sendiña-Nadal et al. (2001), which supports propagation of sustained wave fragments ([Fig. 12a](#)). These wave fragments are used as quanta of information in the design of CB logical circuits. The waves were initiated by locally disturbing initial concentrations of species; for example, ten grid sites in a chain are given value $u = 1.0$ each. This generates two or more localized wave fragments, similar to counter-propagating waves induced by temporary illumination in experiments. The traveling wave fragments keep their shape for around $4 \cdot 10^3$ – 10^4 steps of simulation (4–10 time units), then decrease in size and vanish. The wave’s life-time is sufficient, however, to implement logical gates; this also allows one not to worry about “garbage collection” in the computational medium.

Fig. 12

Basic operations with signals. Overlay of images taken every 0.5 time units. Exciting domains of impurities are shown in black, inhibiting domains of impurities are shown in gray. (a) Wave fragment traveling north. (b) Signal branching without impurities: a wave fragment traveling east splits into two wave fragments (traveling southeast and northeast) when it collides into a smaller wave fragment traveling west. (c) Signal branching with impurity: wave fragment traveling west is split by impurity (shown on the right) into two waves traveling northwest and southwest. (d) Signal routing (U-turn) with impurities: wave fragment traveling east is routed north and then west by two impurities (shown on the right). An impurity-reflector consists of inhibitory (gray) and excitatory (black) chains of grid sites. (From Adamatzky 2004.)



Signals are modeled by traveling wave fragments (Sendiña-Nadal et al. 2001; Beato and Engel 2003): a sustainably propagating wave fragment (● Fig. 12a) represents the TRUE value of a logical variable corresponding to the wave's trajectory (momentarily wire). To demonstrate that a physical system is logically universal, it is enough to implement negation and conjunction or disjunction in the spatio-temporal dynamics of the system. To realize a fully functional logical circuit, one must also know how to operate input and output signals in the system's dynamics, namely to implement signal branching and routing; delays can be realized via appropriate routing.

One can branch a signal using two techniques. Firstly, one can collide a smaller auxiliary wave to a wave fragment representing the signal, the signal-wave will then split into two signals (these daughter waves shrink slightly down to stable size and then travel with constant shape a further $4 \cdot 10^3$ time steps of the simulation) and the auxiliary wave will annihilate (● Fig. 12b). Secondly, one can temporarily and locally apply illumination impurities on a signal's way to change properties of the medium and thus cause the signal to split (● Fig. 12c).

A control impurity, or reflector, consists of a few segments of sites for which the illumination level is slightly above or below the overall illumination level of the medium. Combining excitatory and inhibitory segments, one can precisely control the wave's trajectory, for example, determining a U-turn of a signal (● Fig. 12d).

A typical billiard ball model interaction gate (Fredkin and Toffoli 1982; Margolus 1984) has two inputs – x and y , and four outputs – $x\bar{y}$ (ball x moves undisturbed in the absence of ball y), $\bar{x}y$ (ball y moves undisturbed in the absence of ball x), and twice xy (balls x and y change their trajectories when they collide into each other). It was not possible to make wave fragments implement exact billiard-ball gates, because the interacting waves either fused or one of the waves was annihilated as a result of the collision with another wave.

However, a BZ (nonconservative) version of a billiard-ball gate with two inputs and three outputs is implemented, which is just one xy output instead of two. This BZ collision gate is shown in [Fig. 13](#).

The rich dynamic of the BZ medium allows one to also implement complicated logical operations just in a single interaction event. An example of a composite gate with three inputs and six outputs is shown in [Fig. 14](#). As one sees in [Fig. 14](#), some outputs, for example $\bar{x}yz$, are represented by gradually vanishing wave fragments. The situation can be dealt with by either using a very compact architecture of the logical gates or by installing temporary amplifiers made from excitatory fragments of illumination impurities.

As known from results of computer simulations and experimental studies, classical excitation waves merge or annihilate when they collide with one another. This may complicate the implementation of nontrivial logical circuits in classical excitable media. Wave fragments, however, behave a bit differently, more like quasi-particles.

In computational experiments with exhaustive analysis of all possible collisions between localized wave fragments (Adamatzky and De Lacy Costello 2007), all the collisions are classified as ([Fig. 15](#)): quasi-elastic reflection of wave fragments ([Fig. 15a](#)), pulling and pushing of a wave fragment by another wave fragment ([Fig. 15b, c](#)), sliding of one wave fragment along the refractory trail of another fragment ([Fig. 15d](#)), and translation of a wave fragment along one axis by another wave fragment ([Fig. 15e](#)). Examples of two types of collision are shown in [Fig. 16](#).

4 One-Dimensional Cellular Automata

This section deals with cellular automata in general and discrete signals in particular. It is shown both how they compute in the classical understanding and how they can be used to

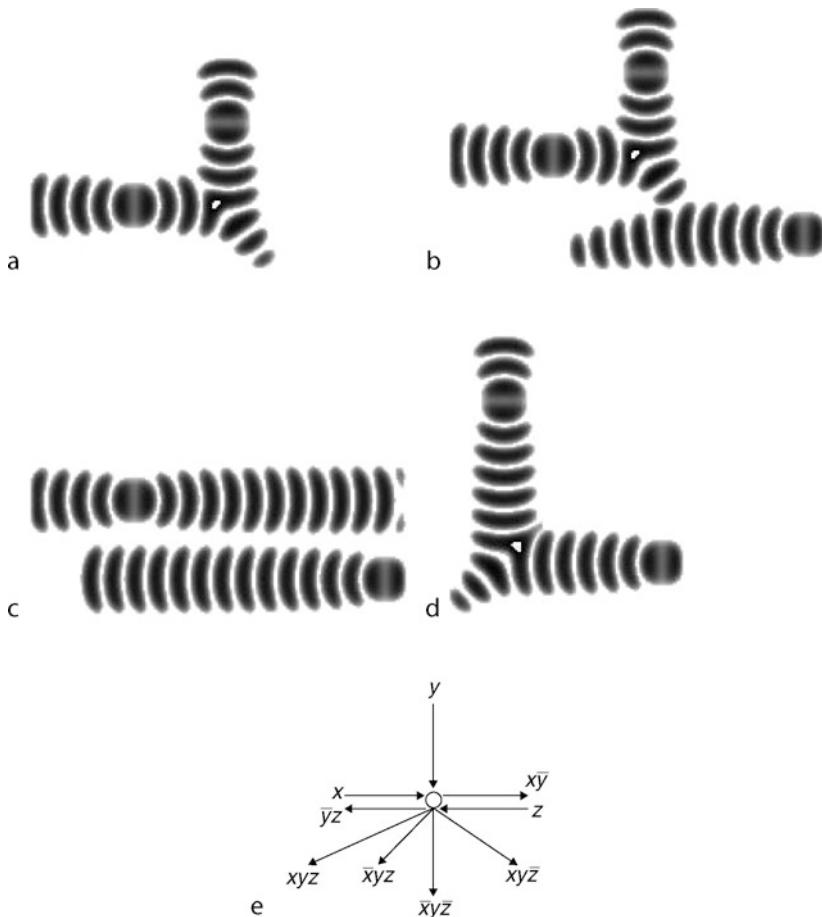
Fig. 13

Two wave fragments undergo angle collision and implement interaction gate $\langle x, y \rangle \rightarrow \langle x\bar{y}, xy, \bar{x}y \rangle$. (a) In this example $x = 1$ and $y = 1$, both wave fragments are present initially. Overlay of images taken every 0.5 time units. (b) Scheme of the gate. In upper left and bottom left corners of (a) one sees domains of wave generation, two echo wave fragments are also generated, they travel outward from the gate area and thus do not interfere with computation. (From Adamatzky 2004.)



Fig. 14

Implementation of $\langle x, y, z \rangle \rightarrow \langle x\bar{y}, \bar{y}z, xyz, \bar{x}yz, \bar{x}\bar{y}\bar{z}, xy\bar{z} \rangle$ interaction gate. Overlay of images of wave fragments taken every 0.5 time units. The following combinations of input configuration are shown: (a) $x = 1, y = 1, z = 0$, north-south wave collides with east-west wave. (b) $x = 1, y = 1, z = 1$, north-south wave collides with east-west wave, and with west-east wave. (c) $x = 1, y = 0, z = 1$, west-east and east-west wave fragments pass near each other without interaction. (d) $x = 0, y = 1, z = 1$, north-south wave collides with east-west wave. (e) Scheme of the gate. (From Adamatzky 2004.)



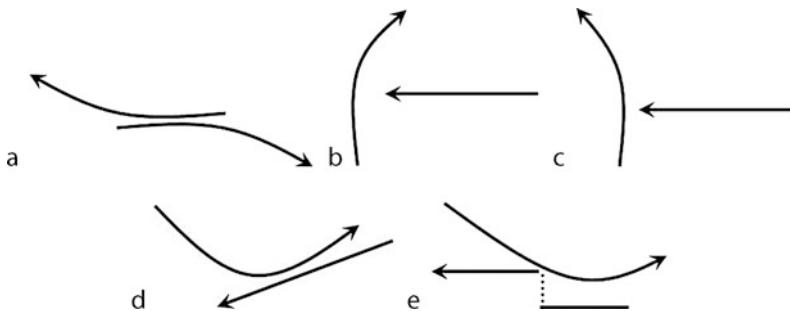
implement some phenomena relevant to massively distributed computing but without any sequential counterpart.

Cellular automata (CA) were introduced as a discrete model for parallel, local, and uniform phenomena, whether of engineering, physical, or biological nature. They often provide a medium where signals naturally appear and are thoroughly used to understand the global dynamics. On the other hand, a correct handling of such signals is the key to compute and to design special purpose CA.

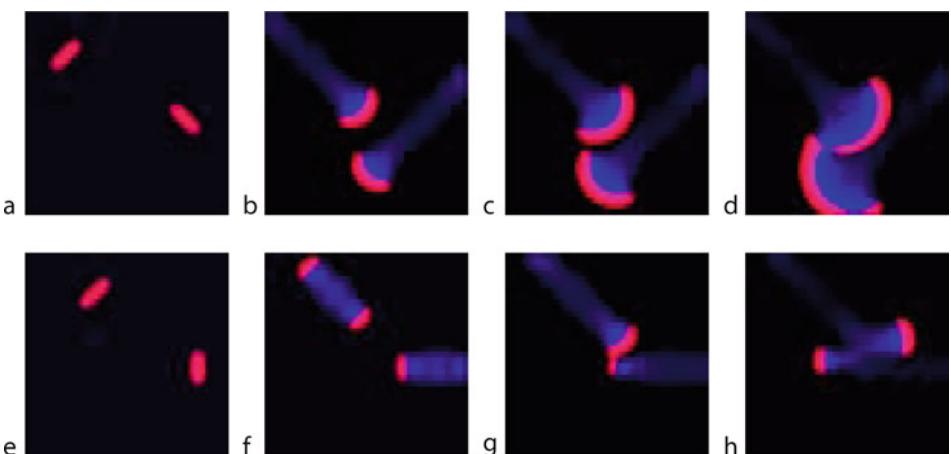
A cellular automaton works in the following way. The space is regularly partitioned in cells. All the cells are identical and are regularly displayed as an infinite array (having as many

Fig. 15

Schematic representation of interaction between wave-fragments. (a) Reflection. (b) Attraction. (c) Repulsion. (d) Sliding. (e) Shifting-sliding. (From Adamatzky and De Lacy Costello 2007.)

**Fig. 16**

(a–d) Sliding: snapshots of wave fragment moving southwest colliding with wave fragment traveling southeast. Center of the initial excitation of the southwest wave fragment is shifted southwards by 22 sites. (From Adamatzky and De Lacy Costello 2007.) (e–h) Slide-shift: snapshots of wave fragment moving southeast colliding with wave fragment traveling west.



dimensions as the modeled space). Each cell can be in finitely many states and evolves according to its state and the states of the surrounding cells – *locality*. All the cells are identical and behave similarly – *uniformity*. The cells are all updated *synchronously*, like a *massive parallel* process sharing a single clock. In essence, CA form a discrete model: discrete time, discrete space, and discrete values.

Generally, the array is considered to extend infinitely in all directions so that there are no boundaries to deal with. However, borders can be encoded using special states which are never changed and make the two sides independent. Another way to simulate a CA on a computer is to consider that outside of a finite range all cells are in the same stable state (called a *quiescent state*). Finite/periodic configuration can also be generated by displaying the cells to be on a ring (or torus): the last and first are then neighbors.

Previous collision-based systems are CA, or, more accurately, are simulated by CA, some of them work on hexagonal lattices. Computation in dimension 2 and above can be done straightforwardly as already presented by bit encoding and implementation of logic gates.

From now on only one-dimensional space is considered because, on the one hand, in higher dimensions it can – up to some point – be treated alike or correspond to what has been exemplified in previous sections, and, because, on the other hand, dimension 1 is particularly restrictive and needs special focus.

The reader interested in CA might consult Ilachinski (2001), Kari (2005), Sarkar (2000) for various topics not covered here.

4.1 Signals in CA

In space–time diagrams, or orbits, configurations are concatenated as iterations go. They are very important in order to comprehend the dynamics. Since the temporal coordinate is added, this leads to an array with one more dimension than the underlying space.

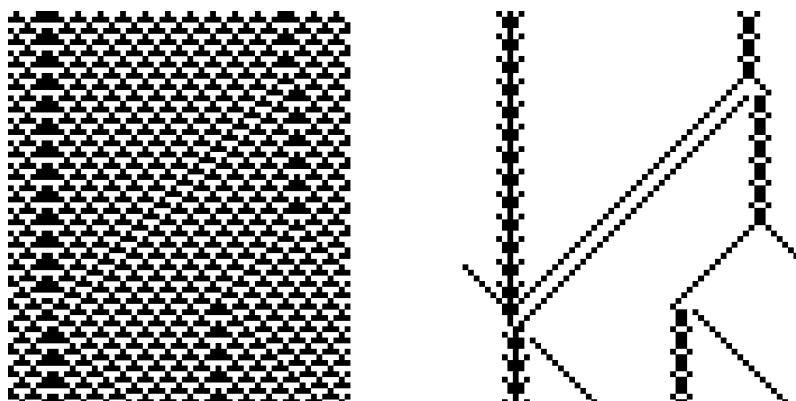
A signal is anything periodic in the space–time diagram. If the periodicity is in two directions, then it can fill the whole space–time and is referred as a *background*, the substrata upon which signals move. The frontier between two backgrounds is a signal as long as it is periodic. [Figure 17](#) provides an example with a complex background (alternatively 0111, 1000, 1101, and 0010 repeating) on the left. On the right, signals are revealed by an ad hoc filtering. This example illustrates the variety in width and pattern of signals. As can be seen, collisions between signals can be pretty complicated.

4.2 Computing in One-Dimensional Systems

In dimension two and above, as soon as it is possible to carry information around (with signal), to make information crossing and duplication, and to process it (with collision) in the proper way to yield a sufficiently large set of logical gates, computation is straightforwardly possible.

Fig. 17

Filtering to exhibit signals and backgrounds on rule 54 (inspired from Boccara et al. 1991, [Fig. 7](#)). Time is increasing upwards.



In dimension 1, this is not so easy, because there is no way for a signal to go around another one or some static artifact. This means that signal propagation is really an issue and, for example, handling garbage signals can be cumbersome. One way to cope with it is to have various kinds of signals not identically affected by artifacts (some would pass unaffected while others interact).

This means that to use the bits and gates scheme, one has to be very careful with available signals and have a very clever positioning of the logic circuitry, which is always thought of as two dimensional (time often provides the extra dimension for displaying it). For example, the construction of Ollinger (2002) provides a small CA that is able to compute with circuit implementation where displaying the circuit is not straightforward.

Another way to tackle computation is to use other formalisms. One classical way is to go back to Turing machines, another one is to implement minimal rewriting systems like cyclic tag systems.

4.2.1 Turing Machine

A Turing machine is a finite automaton acting on a potentially infinite array of symbols. The computation starts with the automaton in its initial state and the input written on the tape. The tape is accessed through a read/write head. At each iteration, the automaton reads the symbol under the head, rewrites a symbol, changes its state, and moves the head left or right. A Turing machine is basically as one dimensional as its tape, so that it naturally fits on a one-dimensional CA.

The simulation is rather simple: signals encoding the symbols are motionless and one signal amounting to the state of the automaton is moving round updating the symbols. Collisions are as follows: a moving state meets a stationary symbol, which leaves a new stationary symbol and a moving new state. (● [Figure 24](#) in the next section provides a clear illustration of this in a continuous setting.) This was implemented by, for example, Lindgren and Nordahl (1990).

4.2.2 Cyclic Tag Systems (CTS)

Another way to define computation is to rely on a word that is rewritten until the system halts. The input is given as the starting word and the output is the final word. Connexion with a Turing machine is straightforward when considering the tape as a finite word and the state of the automaton as an extra symbol at the head.

Cyclic tag systems (CTS) is a particular case of many Turing-universal rewriting systems. A CTS considers a binary word together with a circular list of binary words (*appendants*). At each iteration, the first bit is removed from the word; if it is 1, then the first appendant of the list is added at the end of the word, then the list is rotated. The computation stops when the word is empty or a special halting word (denoted *h* in the example) is activated. An example is provided in ● [Fig. 18](#).

Cyclic tag systems were proven to be able to perform any computation (Cook 2004) and even to do so in polynomial time (Woods and Neary 2006) and were used to build very small universal Turing machines (Neary and Woods 2009). They have been implemented in one-dimensional CA with collision/signal-based computing by Cook (2004) to produce

Fig. 18

Example of computation of a CTS (given on the first line).

1011	011:h:011:01
0110 11	h:011:01:011
11011	011:01:011:h
10110 11	01:011:h:011
0110110 1	011:h:011:01
1101101	h:011:01:011
101101	

computation universal CA with only two states (it is impossible to compute with less). The construction relies on signals moving and colliding that are sought and classified in order to work on a more abstract level.

As far as minimal CA are concerned, it is worth mentioning that four states are enough to get a one-dimensional CA, which is able to simulate any other one-dimensional CA (Richard and Ollinger 2008). This is not Turing universality since the simulation encompasses infinite configurations.

4.3 Signal-Specific Issues

Since CA form a model for physical phenomena and also for parallel computing architectures, other issues arise. One is to understand the underlying dynamics when used as a discrete model. The other is to design CA for special purposes.

4.3.1 Signals to Understand Dynamics

Once a simulation is running as a CA, one common way to understand the dynamics is to try to find regularities in the space–time diagram and observe them. They are often the key to the underlying dynamics and predictions. (Although it might happen that what is observed is nothing but an artifact of the model and has no counterpart in reality.) This is made clear by considering two examples.

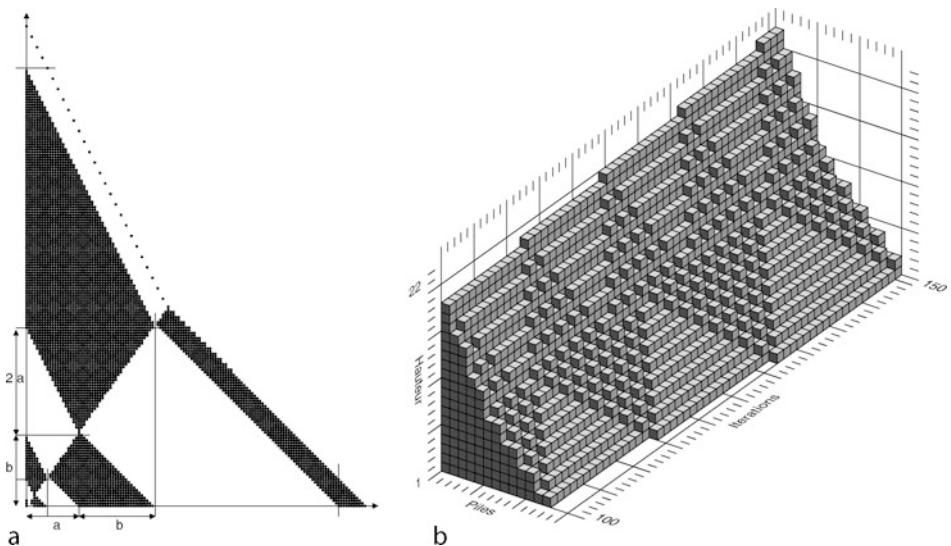
The first one models sand dripping in a one-dimensional space. The dynamics are quite simple: there is nothing in the initial configuration and at each iteration one grain is added to the first pile/cell. Each time a pile has at least two more grains than the next one, a grain falls.

If only grains dropping at odd time are colored in black, then their position is as in [Fig. 19a](#) after 10,000 iterations and only the top grains will ever move. This strange disposal, as well as the two different slopes and precise long-term behavior, are explained by signals (Durand-Lose 1996, 1998). These signals can be identified on [Fig. 19b](#) where the successive configurations (iteration 100–150) are set one after the other to form a volume. On this volume, triangles can be seen on the lower (as well as the upper parts), with their frontiers as signals (which can be revealed with an appropriate filtering).

Another example is provided by Das et al. (1995). The aim is to generate a CA with two states and a five-closest-cell neighborhood, such that, on a ring of any size, whatever configuration it is started on it always ends up blinking: all cells are 0 then all cells are 1, alternatively, forever. Instead of trying to build it straightaway, evolutionary programming is used: random

Fig. 19

One-dimensional sand dripping. (a) Dotting even grains (Durand-Lose 1996, Fig. 6) at iteration 10,000. (b) Successive configurations (Durand-Lose 1996, Fig. 3).



CA are generated; they are ranked according to their “blinking capability”; then the best are kept, recombined and mutated to form the next generation. It then cycles through ranking and the next generation until one “fully blinking” CA emerges.

The obtained CA is analyzed in terms of signal (again with some filtering), as illustrated in **Fig. 20**. (The apparent non-connectivity of some signals comes from the filtering and also because cells can influence one another up to distance 2.) The evolutionary process goes in steps where various intermediate levels of “blinking” appear. Analyzing typical members of each level reveals the progressive apparition of signals and collision rules.

It is very important to notice how, in a context where signals were not asked for by the evolutionary process, they indeed appear and are the key to both the desired dynamics and the evolutionary process.

The previous example does not compute in the classical understanding; nevertheless, it provides a relevant dynamical global property.

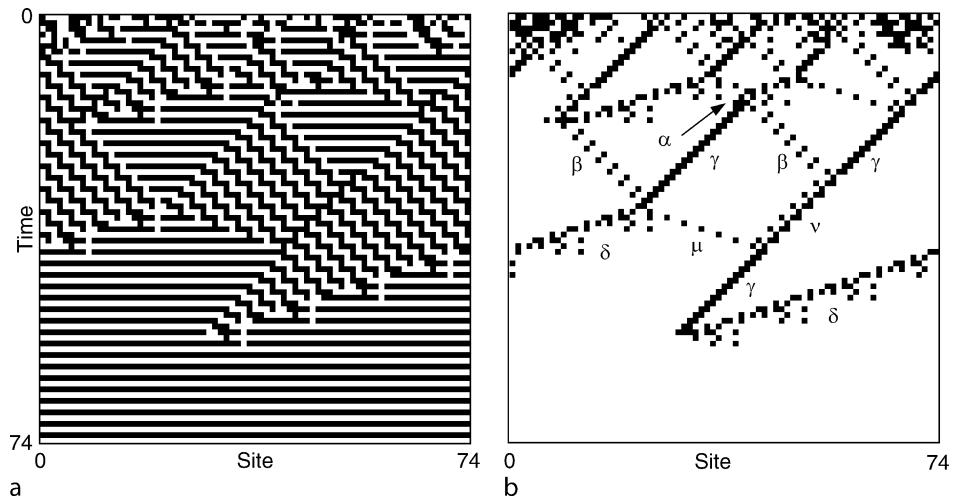
4.3.2 Signals to Generate a Particular Behavior

Computing, in the usual understanding, is a special behavior. But CA can also be thought of as a computing model on its own. Then primitives can be specially defined to take advantage of the huge parallelism. Signals are the key to managing it.

As already cited, the pioneering work of Fischer (1965) generates prime numbers using a parallel implementation of the Sieve of Eratosthenes where they are indicated as no signal on the first cell at the corresponding times. Prime numbers can then be read on the space–time diagram by observing the state of the first cell.

Fig. 20

Figure and filtering to highlight signals and analyze (from Das et al. 1995, Fig. 1). Time is increasing downward. (a) Space-time diagram. (b) Filtered space–time diagram.



Another complex behavior is the famous firing squad synchronization problem (FSS). Starting from all but one cell in a quiescent state, one wants all the cells to enter simultaneously the same state – which has not been used before. Like if they would all blink for the very first time synchronously.

Each example of [Fig. 21](#) shows the Euclidean conception and then the discrete implementation of a FSS solution. Both constructions rely on recursive cuts in half. In the discrete implementation, at some point, the granularity of space – a cell cannot be divided – is reached. Since CA are synchronous, this point is reached simultaneously everywhere, ensuring the global synchronization of the entire array of cells.

4.3.3 Signals as a Computational Model on Its Own

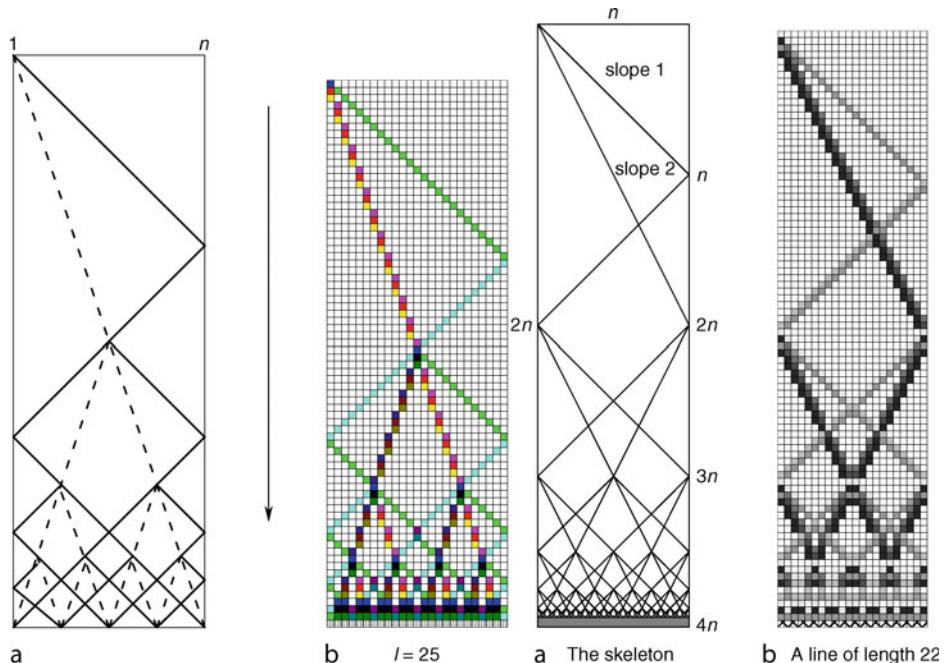
Mazoyer et al. developed a computing model where multiplication (as in [Fig. 22a](#)), composition (as in [Fig. 22b](#)), and even iteration are graphically achieved. The programming system is achieved by using a trellis of adaptable size that can be dynamically generated. The computation is carried out over the trellis. Composition is then achieved by having the next computation displayed on a new trellis. Recursion is provided by a scheme that dynamically starts another iteration providing each time an adapted trellis.

5 Abstract Geometrical Computation

Abstract geometrical computation (AGC) is an idealization of collision-based computing where particles/signals are dimensionless (time and space are continuous). Although the number of signals in a bounded space is expected to be finite, it is unbounded. Another way

Fig. 21

FSS implementations. (a) Divide and conquer in $3n$ -steps (Yunès 2007a, Fig. 2.3).
 (b) Eight-states and $4n$ -steps (Yunès 2007b, Fig. 1).



to consider AGC is as a continuous counterpart of CA as a limit when the size of the cells tends to zero. In CA, signals are almost always the key to understanding and designing, and, indeed, in the literature, the discreteness of CA and of their signals is often put aside to reason at a more abstract level and is left to technical details.

In abstract geometrical computation, dimensionless signals move at constant speed. When they meet, they are replaced by others according to some rewriting rules. A *signal machine* (SM) gathers the definition of the nature of available signals (called *meta-signals*), their speeds, and the collision rules. There are finitely many meta-signals and each one is assigned a constant speed (velocity and direction). The constant speed may seem surprising, but each discrete CA signal has indeed a constant speed.

The space–time diagrams generated are continuous in both space and time and the traces of signals form line segments. For a given meta-signal, all these segments are parallel. In this section, only one-dimensional AGC are considered, so that the space–time diagram is always two dimensional with time increasing upwards as illustrated by [Fig. 23](#).

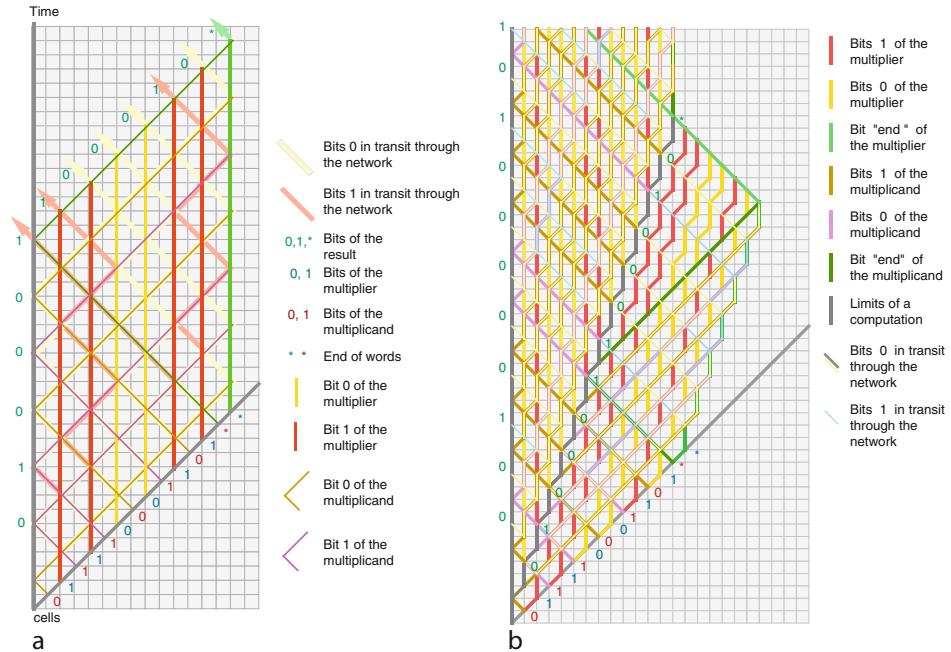
Space–time diagrams can be much more complicated and, as presented below, use continuity to implement the Zeno paradox and emulate the black hole model of computation.

5.1 Computing

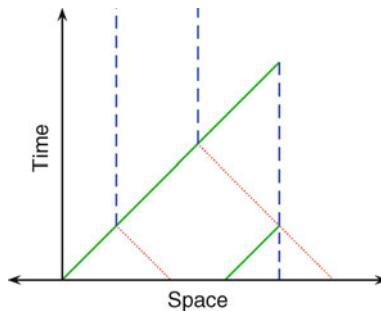
Computing, in the classical understanding, is pretty easy. In fact, many constructions for CA directly translate to ACG and are even easier, since discreteness does not have to be taken

Fig. 22

Mazoyer's system. Time is increasing upwards. (a) Multiplication (Mazoyer 1996, Fig. 4). (b) Composition of multiplications (Mazoyer 1996, Fig. 8).

**Fig. 23**

Example of a space–time diagram.

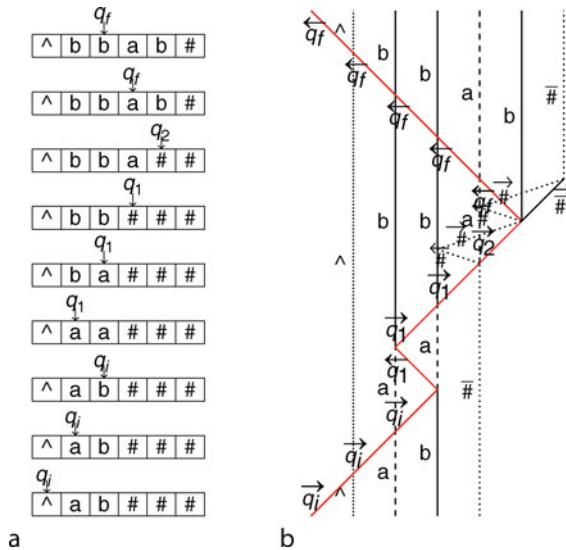


into account. Two ways to achieve computability are presented: Turing machines and cyclic tag systems.

For Turing machines (as presented in [Sect. 4.2.1](#)), the implementation is quite straightforward: static signals encode the tape – one signal per symbol – and one signal encodes the state of the automaton and the position of the head. The latter moves forth and back from symbol to symbol as the read/write head. This is depicted in [Fig. 24](#). The construction is detailed in Durand-Lose (2009).

Fig. 24

Simulating a Turing machine. (a) TM evolution. (b) TM simulation.



Cyclic tag systems are presented in [Sect. 4.2.2](#). The simulation is done by encoding, left to right, the word and then the circular list of appendants by parallel signals encoding the bits as shown in [Fig. 25a](#). At each iteration, the list is rotated to the right, but before that, if the erased bit of the word is 1, a copy is left. The copy is directly added at the right of the word as in [Fig. 25b](#), which presents one full iteration. The full simulation of the example of [Fig. 18](#) is given in [Fig. 25c](#). The rightward drifting corresponds to the erasure of the word from the left and to the rightward rotation of the list. Each group of oblique lines corresponds to one rotation of the list.

This simulation is detailed in Durand-Lose ([2008a](#)). The signal machine obtained is able to simulate any CTS, and is thus Turing universal. It is the smallest one known (it has only 13 meta-signals).

5.2 Geometric Primitives

As for CA, AGC can also be considered as a model on its own, with particular operators, which might not have any classical computation counterpart (like FSS) or discrete counterpart.

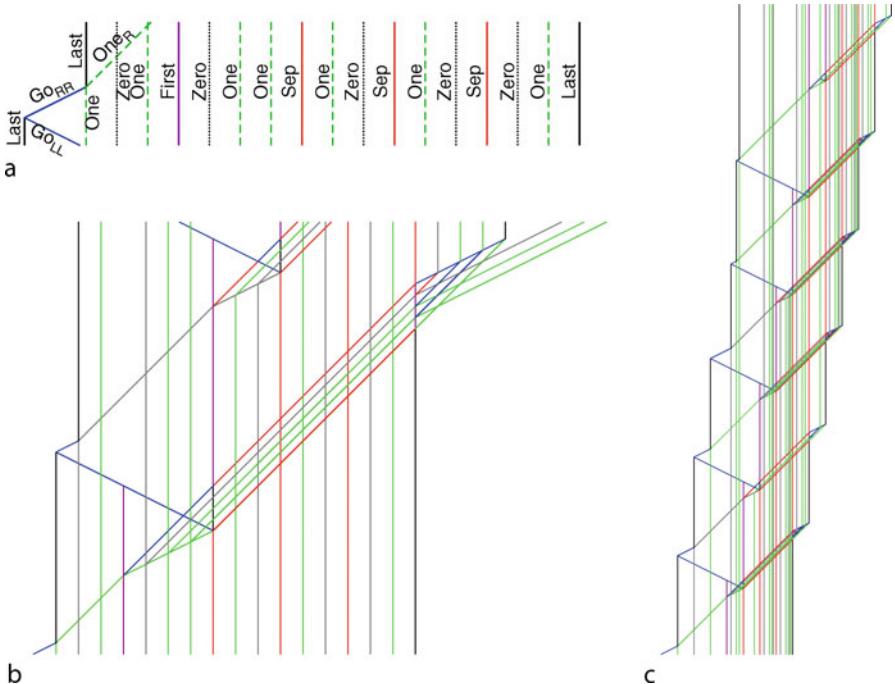
All the following constructions involve adding meta-signals and rules to a given SM, so that the desire capability exists. Signals have to be added to the initial configuration in order to fire the effect. Rules can also be modified in order to dynamically fire it.

Space and time are continuous and scaleless, and since signals have no dimension, there is no limit on the scalability. If all signals are scaled spatially by a given coefficient, the whole computation is also scaled temporally. So, the duration of a computation is meaningless and complexity measures, such as the maximal number of collisions, with a causal link, have to be used.

Spatial rescaling of the initial configuration is a static operation. It is also possible to do it during the computation. The construction relies on the ability to freeze a configuration.

Fig. 25

Simulating a cyclic tag system. (a) Initial configuration on 101 and 011 :: 10 :: 10 :: 01. (b) Initial configuration and first iteration on 101 and 011 :: 10 :: 10 :: 01. (c) Full simulation on 101 & 011 :: h :: 0110 :: 01011.



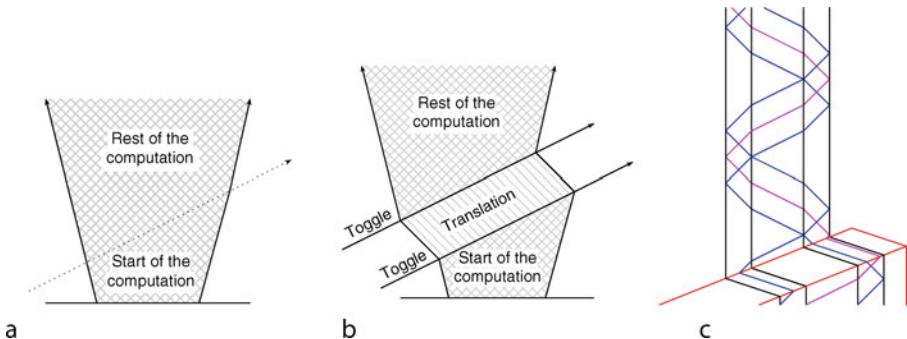
Freezing a computation is quite simple. One signal is added on one side and it crosses the entire configuration. Doing so, each time it meets a signal (or enters a previously existing collision), it replaces it with another signal encoding the meta-signal. All the encoding signals have the same speed: they are parallel and do not interact and, moreover, the distance between them is preserved. The configuration is frozen and shifts. To unfreeze it, a signal comes from the side, crosses the configuration, and replaces each encoding signal by the encoded one (or the result of the collision). Freezing and unfreezing signals must have the same speed so that the configuration is restored exactly as it was, up to a translation. (And indeed they correspond to the same meta-signal toggle.) This is depicted in [Fig. 26](#).

Meanwhile, when a configuration is frozen into parallel signals, it is possible to act on these signals. One simple trick is to change their direction. In [Fig. 27a](#), this is done twice so as to restore the original direction. (An extra signal is used on the right to delete the structure.) As a result, since different slopes are used to change direction, the distances between signals are scaled (here by one half). Adding freezing and unfreezing signals (automatically fired in due time), an artifact, to scale down a whole configuration, is generated as can be seen on the [Fig. 27c](#).

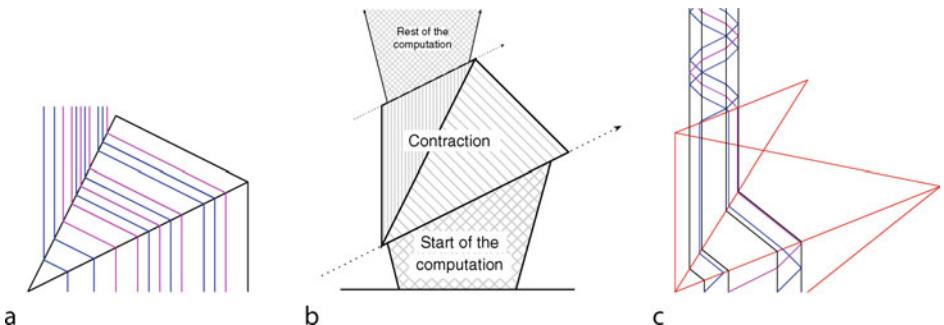
The initial configuration has been modified in order to start straightaway the effect. It is also possible to fire it dynamically during the computation when some special collision happens. This is used to iterate it.

Fig. 26

Freezing and unfreezing. (a) Normal computation. (b) Translated computation. (c) Example.

**Fig. 27**

Scaling. (a) Principle. (b) Scaled computation. (c) Example.



5.3 Infinite Acceleration and Non-recursive Function

Scaling can be automatically restarted ad infinitum; since both space and time are continuous, this is not a problem – at least before the singularity. In [Fig. 28](#), a computation producing an infinite trellis, extending indefinitely in both space and time, is folded into the structure. The right signal and the middle one are very important since the folded computations should be bounded in order to ensure that it is fully scaled.

A *singularity* refers to an accumulation of infinitely many collisions and signals to a given location. [Figure 28a](#) shows that the structure alone already creates a singularity. This illustrates the Zeno paradox. The leftmost signals of [Fig. 28a](#) form an infinite sequence with infinitely (yet countably) many collisions, although the time elapsed as well as the total distance crossed by signals is finite.

All the signals and collisions that would have existed if there would have been no folding indeed exist, but in a bounded portion of the space and time diagram. (The equivalence of the plane and a bounded part is somehow implemented.) So that up to the presence of the structure and of the frozen parts and the different scales, the computation is the same.

Any computation starting with finitely many signals can be folded. Inside the structure, the rescaling provides speedup (the closer they are, the faster they collide). This speedup is unbounded and goes to infinity. Inside the structure there is a space–time where time-lines are infinitely accelerated compared to the outside. This is the first step to emulate the black-hole model of computation (Hogarth 1994; Etesi and Németi 2002; Lloyd and Ng 2004).

The second step is to find a way for an atomic piece of information to leave the black hole, here the singularity. One meta-signal can be added and rules changed so that the new meta-signal is not affected by the structure but can be generated by the initial computation.

The final step is to add bounding signals on both sides of the folding (called here `horizonLe` and `horizonRi`). At their collision point, the folded computation is entirely in the causal past as displayed in Fig. 29. Any signal leaving the folding would have been collected.

This way the black-hole model is emulated: there are two time-lines, one for the machine and one for the observer. The machine one is infinite. On the observer one, after a finite duration, the whole machine one is entirely in the causal past. A single piece of information can be sent by the machine to the observer and it has to be sent after a finite (machine) duration. The ultimate date for the observer to receive anything from the machine is finite and known. To understand the power of this model, just imagine that the machine only sends a signal if its computation ends. The observer receives a signal only if it stops and after a duration the observer is aware that any signal sent would have been received. So, by just

Fig. 28

Folding or infinite rescaling. (a) Folding structure. (b) Example.

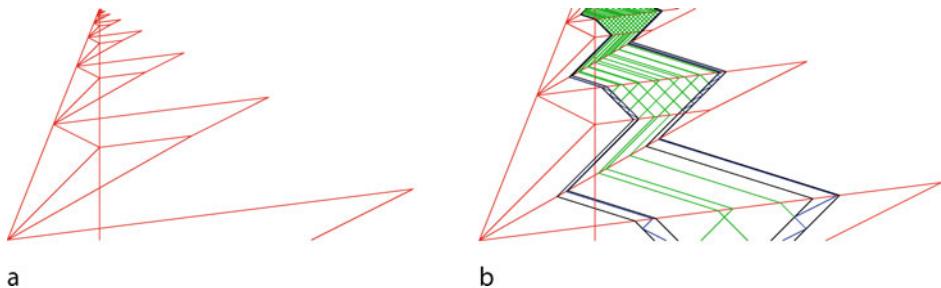
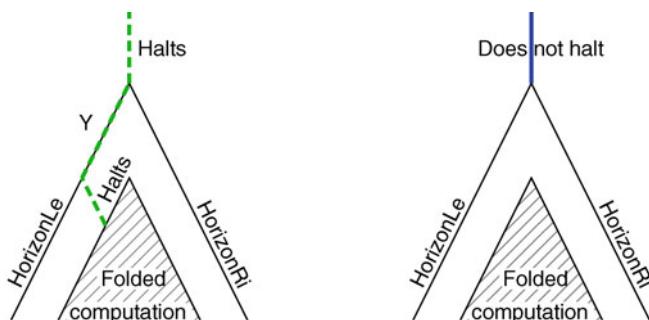


Fig. 29

Framing the folding to collect all signals exiting the folding.



checking its clock, the observer knows that the computation did not halt when it is the case. Altogether, the halting problem (whose undecidability is the cornerstone of Turing computability theory) is decidable!

This model, as well as AGC, clearly computes beyond Turing. In formal term, it can decide Σ_0^1 formulae in the arithmetical hierarchy, which is a recursive/computable predicate prefixed by an existential quantifier. This includes, for example, the consistency of Peano arithmetic and set theory, many conjectures such as the Collatz conjecture and Goldbach's conjecture.

Defining what happens at the singularity point is not easy, especially since the accumulating set can be a line segment, a fractal curve or even a Cantor! (To understand this, take a continuous look at the FSS: what would happen at the bottom of Fig. 21?) For singularity on a single point, a careful continuation has been proposed in Durand-Lose (2009) that allows us to climb the arithmetical hierarchy. There is also another use of isolated singularity as a way to provide limits for analog computation.

5.4 Analog Computation

Unlike collision-based computing and CA, with dimensionless signals in a continuous space, it is possible to encode real numbers as the distance between signals. In the AGC context, this distance is exact—that is, it is a real number in exact precision. As long as signals are parallel, this distance is preserved. This allows one to encode real numbers and to make some computations over them.

Since any real number can be encoded exactly, the model ipso facto falls out of classical computability (because of cardinalities, there is no way to encode all the reals with natural numbers or finite strings). Therefore, an analog model of computation has to be searched for.

With this encoding of real numbers, AGC is equivalent to the linear Blum–Shub–Smale model (BSS) (Blum et al. 1998). In the linear BSS model, variables hold (exact) real numbers and the operations available are addition, multiplication by a constant and branch, according to the sign of a variable. This equivalence is true as long as the BSS machine has an unbounded number of variables (accessed through a context shift like moving a window over an infinite array of variables), there are finite signals and there is no singularity (Durand-Lose 2007).

If singularities are used in a proper way, it becomes possible to multiply two variables. Then the classical BSS model can be implemented in AGC (Durand-Lose 2008b). The simulation is not possible in the other way anymore since, for example, the exact square rooting can also be computed by AGC.

Just as in the discrete case, a proper handling of isolated singularities of any order can be used to decide quantified (over natural but not real numbers) predicates in the BSS model and climb the BSS-arithmetical hierarchy (Durand-Lose 2009).

References

- Adamatzky A (ed) (2002a) Collision-based computing. Springer, London
- Adamatzky A (ed) (2002b) Novel materials for collision-based computing. Springer, Berlin
- Adamatzky A (2004) Collision-based computing in Belousov–Zhabotinsky medium. *Chaos Soliton Fract* 21:1259–1264
- Adamatzky A, De Lacy Costello B (2007) Binary collisions between wave-fragments in sub-excitable Belousov–Zhabotinsky medium. *Chaos Soliton Fract* 34:307–315
- Adamatzky A, Wuensche A (2007) Computing in spiral rule reaction-diffusion hexagonal cellular automaton. *Complex Syst* 16(4):277–298

- Adamatzky A, Wuensche A, De Lacy Costello B (2006) Glider-based computation in reaction-diffusion hexagonal cellular automata. *Chaos Soliton Fract* 27:287–295
- Anastassiou C, Fleischer JW, Carmon T, Segev M, Steiglitz K (2001) Information transfer via cascaded collisions of vector solitons. *Optics Lett* 26:1498–1500
- Atrubin AJ (1965) A one-dimensional real-time iterative multiplier. *IEEE Trans Electron Computers EC-14* (1):394–399
- Banks E (1971) Information and transmission in cellular automata. Ph.D Dissertation, MIT, cited by Toffoli and Margolus (1987)
- Beato V, Engel H (2003) Pulse propagation in a model for the photosensitive Belousov-Zhabotinsky reaction with external noise. In: Schimansky-Geier L, Abbott D, Neiman A, van den Broeck C (eds) *Noise in complex systems and stochastic dynamics*. Proceedings of SPIE, 2003
- Berlekamp ER, Conway JH, Guy RL (1982) *Winning ways for your mathematical plays*, vol 2 Games in particular. Academic, London
- Blum L, Cucker F, Shub M, Smale S (1998) Complexity and real computation. Springer, New York
- Boccara N, Nasser J, Roger M (1991) Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Phys Rev A* 44(2): 866–875
- Cook M (2004) Universality in elementary cellular automata. *Complex Syst* 15:1–40
- Das R, Crutchfield JP, Mitchell M, Hanson JE (1995) Evolving globally synchronized cellular automata. In: Eshelman LJ (ed) *International conference on genetic algorithms '95*. Morgan Kaufmann, San Mateo, CA, pp 336–343
- Delorme M, Mazoyer J (2002) Signals on cellular automata. In: Adamatzky A (ed) *Collision-based computing*. Springer, Berlin, pp 234–275
- Durand-Lose J (1996) Grain sorting in the one dimensional sand pile model. *Complex Syst* 10(3):195–206
- Durand-Lose J (1998) Parallel transient time of one-dimensional sand pile. *Theoret Comp Sci* 205 (1–2):183–193
- Durand-Lose J (2007) Abstract geometrical computation and the linear Blum, Shub and Smale model. In: Cooper S, Löwe B, Sorbi A (eds) *Computation and logic in the real world. 3rd Conference Computability in Europe (CiE '07)*. Springer, no. 4497 in LNCS, pp 238–247
- Durand-Lose J (2008a) Abstract geometrical computation: small Turing universal signal machines. In: Neary T, Seda A, Woods D (eds) *International workshop on the complexity of simple programs*. Cork University Press, Cork, Ireland, December 6–7
- Durand-Lose J (2008b) Abstract geometrical computation with accumulations: beyond the Blum, Shub and Smale model. In: Beckmann A, Dimitracopoulos C, Löwe B (eds) *Logic and theory of algorithms. CiE 2008 (abstracts and extended abstracts of unpublished papers)*. University of Athens, Athens, pp 107–116
- Durand-Lose J (2009) Abstract geometrical computation 3: Black holes for classical and analog computing. *Nat Comput* 8(3):455–472
- Etesi G, Németi I (2002) Non-Turing computations via Malament-Hogarth space-times. *Int J Theor Phys* 41 (2):341–370, gr-qc/0104023
- Field RJ, Noyes RM (1974) Oscillations in chemical systems. iv. limit cycle behavior in a model of a real chemical reaction. *J Chem Phys* 60:1877–1884
- Fischer PC (1965) Generation of primes by a one-dimensional real-time iterative array. *J ACM* 12(3):388–394
- Fredkin EF, Toffoli T (1982) Conservative logic. *Int J Theor Phys* 21(3/4):219–253
- Fredkin EF, Toffoli T (2002) Design principles for achieving high-performance submicron digital technologies. In: Adamatzky A (ed) *Collision-based computing*. Springer, Berlin, pp 27–46
- Hogarth ML (1994) Non-Turing computers and non-Turing computability. In: Hull D, Forbess M, Burian RM (eds) *Biennial meeting of the philosophy of science association*. East Lansing, MI, pp 126–138
- Ilachinski A (2001) *Cellular automata – a discrete universe*. World Scientific, Singapore
- Jakubowski MH, Steiglitz K, Squier RK (1996) When can solitons compute? *Complex Syst* 10(1):1–21
- Jakubowski MH, Steiglitz K, Squier RK (2001) Computing with solitons: a review and prospectus. *Multiple Valued Logic* 6(5–6):439–462
- Kari J (2005) Theory of cellular automata: a survey. *Theoret Comp Sci* 334:3–33
- Krug HJ, Pohlmann L, Kuhnert L (1990) Analysis of the modified complete oregonator (MCO) accounting for oxygen- and photosensitivity of Belousov-Zhabotinsky systems. *J Phys Chem* 94:4862–4866
- Lindgren K, Nordahl MG (1990) Universal computation in simple one-dimensional cellular automata. *Complex Syst* 4:299–318
- Lloyd S, Ng YJ (2004) Black hole computers. *Sci Am* 291 (5):31–39
- Margolus N (1984) Physics-like models of computation. *Phys D* 10:81–95
- Mazoyer J (1996) Computations on one dimensional cellular automata. *Ann Math Artif Intell* 16: 285–309
- Neary T, Woods D (2009) Four fast universal Turing machines. *Fundam Inform* 410(4):443–450
- Ollinger N (2002) The quest for small universal cellular automata. In: ICALP '02, Springer, Heidelberg, no. 2380 in LNCS, pp 318–329

- Rand D, Steiglitz K (2009) Computing with solitons. In: Meyers RA (ed) Encyclopedia of complexity and systems science. Springer, Heidelberg
- Rand D, Steiglitz K, Prucnal P (2005) Signal standardization in collision-based soliton computing. *Int J Unconventional Comput* 1:31–45
- Rendell P (2002) Turing universality of the game of life. In: Adamatzky A (ed) Collision-based computing. Springer, Berlin, pp 513–540
- Rennard JP (2002) Implementation of logical functions in the game of life. In: Adamatzky A (ed) Collision-based computing. Springer, London, pp 491–512
- Richard G, Ollinger N (2008) A particular universal cellular automaton. In: Neary T, Woods D, Seda AK, Murphy N (eds) The complexity of simple programs. National University of Ireland, Cork
- Sarkar P (2000) A brief history of cellular automata. *ACM Comput Surv* 32(1):80–107
- Sendiña-Nadal I, Mihaliuk E, Wang J, Pérez-Muñozuri V, Showalter K (2001) Wave propagation in subexcitable media with periodically modulated excitability. *Phys Rev Lett* 86:1646–1649
- Steiglitz K (2001) Time-gated Manakov spatial solitons are computationally universal. *Phys Rev E* 63:1660–1668
- Toffoli T, Margolus N (1987) Cellular automata machine – a new environment for modeling. MIT Press, Cambridge, MA
- Tyson JJ, Fife PC (1980) Target patterns in a realistic model of the Belousov-Zhabotinsky reaction. *J Chem Phys* 73:2224–2237
- Waksman A (1966) An optimum solution to the firing squad synchronization problem. *Inform Control* 9(1):66–78
- Woods D, Neary T (2006) On the time complexity of 2-tag systems and small universal Turing machines. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06), IEEE Computer Society, Berkeley, CA, pp 439–448
- Wuensche A (2005) Glider dynamics in 3-value hexagonal cellular automata: the beehive rule. *Int J Unconventional Comput* 1:375–398
- Wuensche A, Adamatzky A (2006) On spiral glider-guns in hexagonal cellular automata: activator-inhibitor paradigm. *Int J Modern Phys C* 17(7): 1009–1026
- Yunès JB (2007a) Automates cellulaires; fonctions booléennes. Habilitation à diriger des recherches, Université Paris 7
- Yunès JB (2007b) Simple new algorithms which solve the firing squad synchronization problem: a 7-states 4n-steps solution. In: Durand-Lose J, Margenstern M (eds) Machine, Computations and Universality (MCU '07). Springer, Berlin, no. 4664 in LNCS, pp 316–324

59 Nonclassical Computation — A Dynamical Systems Perspective

Susan Stepney

Department of Computer Science, University of York, UK
susan.stepney@cs.york.ac.uk

1	<i>Introduction</i>	1980
2	<i>Autonomous Dynamical Systems</i>	1981
3	<i>Open Dynamical Systems</i>	2005
4	<i>Constructive Dynamical Systems</i>	2014
5	<i>Discussion and Conclusions</i>	2020

Abstract

In this chapter, computation is investigated from a dynamical systems perspective. A dynamical system is described in terms of its abstract *state space*, the system's current state within its state space, and a rule that determines its motion through its state space. In a classical computational system, that rule is given explicitly by the computer program; in a physical system, that rule is the underlying physical law governing the behavior of the system. Therefore, a dynamical systems approach to computation allows one to take a unified view of computation in classical discrete systems and in systems performing nonclassical computation. In particular, it gives a route to a computational interpretation of physical embodied systems exploiting the natural dynamics of their material substrates.

1 Introduction

In this chapter, we investigate computation from a dynamical systems perspective.

A dynamical system is described in terms of its abstract *state space*, the system's current state within its state space, and a rule that determines its motion through its state space. In a classical computational system, that rule is given explicitly by the computer program; in a physical system, that rule is the underlying physical law governing the behavior of the system. So a dynamical systems approach to computation allows us to take a unified view of computation in classical discrete systems and in systems performing nonclassical computation. In particular, it gives a route to a computational interpretation of physical embodied systems exploiting the natural dynamics of their material substrates.

We start with *autonomous* (closed) dynamical systems: those whose dynamics is not an explicit function of time, in particular, those with no inputs from an external environment. We begin with computationally conventional discrete systems examining their computational abilities from a dynamical systems perspective. The aim here is both to introduce the necessary dynamical systems concepts, and to demonstrate how classical computation can be viewed from this perspective. We then move on to continuous dynamical systems, such as those inherent in the complex dynamics of matter, and show how these too can be interpreted computationally, and see how the material embodiment can give such computation “for free,” without the need to explicitly implement the dynamics.

We next broaden the outlook to *open* (nonautonomous) dynamical systems, where the dynamics is a function of time, in the form of inputs from an external environment, and which may be in a closely coupled feedback loop with that environment.

We finally look at *constructive*, or developmental, dynamical systems, where the structure of the state space is changing during the computation. This includes various growth processes, again investigated from a computational dynamical systems perspective.

These later sections are less developed than for the autonomous cases, as the theory is less mature (or even nonexistent); however these are the more interesting computational domains, as they move us into the arena of considering biological and other natural systems as computational, open, developmental, dynamical systems.

2 Autonomous Dynamical Systems

Consider a dynamical system with N degrees of freedom; it has an abstract state space \mathcal{X}^N . Its state can be defined by N state variables $x_i \in \mathcal{X}$, or, equivalently, by an ND state vector $\mathbf{x} \in \mathcal{X}^N$ (e.g., \mathbf{x} may be a vector of binary bits, or a vector of continuous variables such as position and momentum). The state vector \mathbf{x}_t defines the value of the system state at a given time.

The deterministic dynamics is given by a function $f : \mathcal{X}^N \rightarrow \mathcal{X}^N$, which defines how a state vector \mathbf{x} changes with time, that is, it defines the *trajectory* that the system takes through its state space. So the dynamics associates a vector with each point in the state space, defining how that point evolves under the dynamics. (In conventional use, the meaning of this vector is unfortunately different in the discrete and continuous time cases. In the discrete time case (Sects. 2.1 and 2.2), $\mathbf{x}_{t+1} = f(\mathbf{x}_t)$, and so the vector is the next state; in the continuous time case (Sect. 2.3), $\dot{\mathbf{x}} = f(\mathbf{x})$, and so the vector is the derivative, pointing toward the next state an infinitesimal time later: $\mathbf{x}_{t+dt} = \mathbf{x}_t + f(\mathbf{x}_t) dt$. It would be possible to have a uniform meaning, by redefining the discrete case vector to be the difference in states, with $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t)$ (and an implicit $\Delta t = 1$). However, here we follow the conventional, and inconsistent, use.) If f is not itself an explicit function of time, then the system is *autonomous*.

In general, dynamical systems theory is not concerned with details of individual trajectories, but rather with the qualitative behaviors of sets of trajectories. For example, consider a set of states occupying some initial volume of the state space: As these states evolve under the dynamics, how does the volume change? We are mostly interested here in *dissipative systems*, where the volume contracts to *attractors* (regions of state space that attract trajectories), and we interpret such attractors from a computational perspective. This contracting behavior is a property of systems that dissipate energy or information. (Closed non-dissipative dynamical systems, on the other hand, have no attractor structure.)

2.1 Discrete Space, Discrete Time Dynamical Systems

We start by considering finite discrete spaces (finite number of finite dimensions), with discrete time dynamics $t \in \mathbb{N}$ (where \mathbb{N} is the set of natural numbers).

We take $\mathcal{X} = \mathcal{S}$, some set with finite cardinality $|\mathcal{S}| \in \mathbb{N}$ (typically \mathcal{S} will be the Boolean set \mathcal{B} , but it is not restricted to this). For an N -dimensional system, the state is defined by N state variables $s_i \in \mathcal{S}$, and the state space \mathcal{S}^N comprises $|\mathcal{S}|^N$ distinct discrete states. (When $\mathcal{S} = \mathcal{B}$, these states fall on the vertices of an N -dimensional hypercube.) Let the state vector be $\mathbf{s} \in \mathcal{S}^N$.

The dynamics of a particular system is determined by its particular transition function $f : \mathcal{S}^N \rightarrow \mathcal{S}^N$, with $\mathbf{s}_{t+1} = f(\mathbf{s}_t)$. There are $(|\mathcal{S}|^N)^{|\mathcal{S}|^N}$ such functions f .

Given a particular state $\mathbf{s}_0 \in \mathcal{S}^N$, its trajectory under f is a sequence of states $\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_t, \dots$. Eventually, because the state space is finite, a state that was met before will be met again: there exists a k such that $\mathbf{s}_k = \mathbf{s}_{k+p}$, for some p . Since the dynamics is deterministic, the trajectory will then recur: for all $i \geq k$, $\mathbf{s}_i = \mathbf{s}_{i+p}$. The system has entered an *attractor*, with cycle length or period p . States not on an attractor are called *transient*.

Given a trajectory $\dots, \mathbf{s}_t, \mathbf{s}_{t+1}, \dots$, then \mathbf{s}_t is the *pre-image* of \mathbf{s}_{t+1} in this trajectory, and \mathbf{s}_{t+1} is the *successor* of \mathbf{s}_t . Every state has precisely one successor (because the dynamics is deterministic). It may have zero, one, or more pre-images (trajectories may merge); if it has zero pre-images, it is a *Garden of Eden* state.

The set of all states \mathbf{s}_i whose trajectories lead to the same attractor forms the *basin of attraction* of that attractor. The total state space is partitioned into these basins: Every state is in precisely one basin. Note that there is no necessary correlation between the volume of the basin (the proportion of state space it occupies, and hence the probability that a state chosen at random will be in it) and the length of the attractor that it leads to.

The *microstate* of the system is which particular $\mathbf{s} \in \mathcal{S}$ it is in. The *macrostate* is which particular attractor (or basin of attraction if the microstate is currently a transient state) the system is in.

For a given dynamics f , there is a minimum of one attractor basin (all states are in the same attractor basin, for example, the zero function), and a maximum of $|\mathcal{S}|^N$ (the identity function where every state forms its own single-state attractor basin). There is a minimum transient length of 0 (all states on some attractor cycle, for example, the increment modulo 2^N function, interpreting the binary encoded state \mathcal{B}^N as a number), and a maximum transient length of $|\mathcal{S}|^N - 1$ (e.g., the decrement and halt on zero function). There is a minimum number of Garden of Eden states of 0 (all states on some attractor cycle), and a maximum number of Garden of Eden states of $|\mathcal{S}|^N - 1$ (e.g., the zero function). Nontrivial dissipative computational systems rarely lie at any of these extremes, however. (Note that reversible, non-dissipative systems have no Garden of Eden states, and no merging trajectories.)

2.1.1 Visualizing the Attractor Field

Visualizing the basins of attraction can help in understanding some aspects of their dynamics. For small systems, the most common approach is to lay out the state transition graph to highlight the separate basins, their attractors, and their symmetries (see [Fig. 1](#)). Wolfram (1986b, Fig 9.1) used this approach in early work on cellular automata; Wuensche (Wuensche 2002; Wuensche and Lesser 1992) has developed special purpose layout software, and uses this approach consistently, to highlight aspects of the dynamics.

2.1.2 Computation

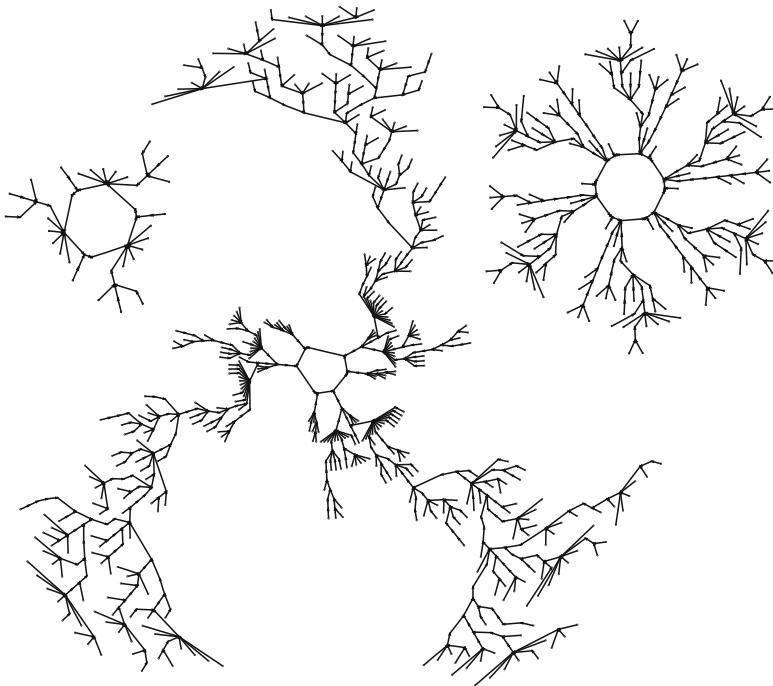
Given a finite system in initial microstate \mathbf{s}_0 (which may be considered to contain an encoding of any input data), the system follows its dynamics f until it reaches the relevant attractor. If this attractor has a cycle length of one, the system then stays in the single attractor state. For longer cycle lengths, the system perpetually repeats the cycle of states.

In Terms of Attractors

The *computation* performed by the system as it follows its dynamics f can be interpreted as the determination of which attractor basin it is in, by progressing to the attractor from its initial state \mathbf{s}_0 .

Fig. 1

Visualization of part of the state transition graph for ECA rule 110 (see [Sect. 2.1.5](#) on Elementary Cellular Automata), on the periodic lattice $N=12$, showing three of the basins of attraction. Each node corresponds to a state s_i ; each edge corresponds to a transition $s_i \rightarrow s_{i+1}$. The leaves of the graphs are Garden of Eden states; the attractor cycles can be seen at the centers of the basins.



The output of the computation may be the microstates, or some suitable projection thereof, of the discovered attractor cycle. (See, e.g., [Sect. 2.1.5](#), Example 1: the density classification task.)

In Terms of Trajectories

Alternatively, the *computation* performed by the system as it follows its dynamics f can be considered to be (some projection of) the microstates it passes through along its trajectory, including both transient and attractor cycle states. (See, e.g., [Sect. 2.1.5](#), Example 2: the Rule 30 PRNG.)

Programming Task

The programming task involves determining a dynamics f that leads to the required trajectories or attractor structure. (See, e.g., [Sect. 2.1.5](#), Example 1: the density classification task.)

For feasible programs, the discovery of an attractor should be performed in polynomial time (implying polynomial length transients and attractor cycles). At the other complexity

extreme, the dynamics should not be defined merely by a $|\mathcal{S}|^N$ -entry lookup table (which would allow all computations to find the attractor in a single step); it should admit a “compressed” description. (See, e.g., [Sect. 2.1.5](#) and [2.1.6](#) which define the global dynamics f in terms of the composition of local dynamics ϕ_i .)

Implementation

Natural physical systems do not tend to directly implement a discrete dynamics, particularly one that has been designed to perform a specific task. However, any such dynamics can be implemented (or simulated) on a classical digital computer. Hence, there are no implementation constraints imposed on the design of the dynamics f .

Inputs and Outputs

The input is encoded into the initial state; the output is decoded from (a projection of) the resulting attractor state(s). It is important when analyzing the complexity of the computation to take into account any “hidden” computation needed to encode the input, or to decode the output. This is particularly important if the computational interpretation is far removed from the dynamics, for example, if there is some kind of virtual machine present.

2.1.3 Virtual Machine Dynamics

In some cases, a dynamics needs to be accompanied by a very carefully chosen initial condition in order to implement the required computation. For example, when cellular automata are used to implement Turing machines (TMs; see [Sect. 2.1.5](#) on Universality), they are given a carefully chosen initial configuration that corresponds to the “program” of the TM, plus the “true” input corresponding to its initial tape. This requirement for an exquisitely tuned initial condition constrains the system to traverse only a very small part of its state space (certain basins of attraction are never explored; some transient trajectories are never taken). What is happening in these cases is that the underlying broader dynamics is being used to implement a “virtual machine” with its own dynamics confined to a small subspace of the underlying system; this subspace and its trajectories correspond to the computation of the virtual machine, and may possibly be implemented more directly. In the continuous case, this more direct implementation is what we want: The natural dynamics of the system, with no need for such highly tuned initial conditions, performs the desired computation.

2.1.4 Infinite-Dimensional State Spaces

When N is (countably) infinite, the dynamics of the system can change qualitatively.

In a finite dissipative system (which contains both transients and cycles), there must be Garden of Eden states, but this is no longer true in an infinite system: Transient behavior on the way to the attractor cycle need not have any starting Garden of Eden states (e.g., the decrement and halt on zero function).

An infinite system need not have an attractor cycle: There is no guarantee that the system will reach a previously seen state (e.g., the increment function). Even if there are attractor cycles, there may be states not in their basins (e.g., the function **if even then add 2 else halt**).

A Turing machine (TM) operates in a state space with a countably infinite (or, more precisely, finite but unbounded) number of dimensions. In dynamical systems terms, halting is reaching one of a number of particular attractor cycles of length 1, which describes the halting states. The output of the TM (the contents of its tape) is simply the microstate of the subspace representing the tape in this halting state. Hence there are potentially many attractors, one for each halting state with different tape contents, corresponding to different initial tape contents (different initial states s_0).

The halting problem means that it is in general undecidable whether a given initial state s_0 is in a halting basin, in some other (“looping”) basin, or not in a basin at all.

2.1.5 Cellular Automata

A finite cellular automaton (CA) comprises N cells laid out in a regular grid or lattice, usually arranged as an n -dimensional torus (n in common examples is typically 1 or 2). Each cell i at time t has a state value $c_{i,t} \in \mathcal{S}$.

Each cell has a *neighborhood* of k cells, comprising itself and certain nearby cells in the grid. This neighborhood is the same for all cells, in that v_i , the neighborhood of c_i , is v_0 , the neighborhood of the origin c_0 , translated by i (☞ Fig. 2a).

The state of cell c_i ’s neighborhood v_i at time t is $\chi_{i,t} \in \mathcal{S}^k$, a k -tuple of cell states that is the projection of the full state onto the neighborhood v_i (☞ Fig. 2b).

The local state transition rule, or update rule, is $\phi : \mathcal{S}^k \rightarrow \mathcal{S}$. (There are $|\mathcal{S}|^{|\mathcal{S}|^k}$ such rules, some of which are related by symmetries. So, since typically $k \ll N$, CA rules capture only a small fraction of all the possible dynamics over an ND space.) These cells form an array of state transition machines. At each timestep, the state of each cell is updated in parallel, $c_{i,t+1} = \phi(\chi_{i,t})$.

The global dynamics f is determined by the local rule ϕ and the shape of the neighborhood. This global behavior from a given initial state is conventionally visualized in the 1D case by drawing the global state at time t as a line of cells (with colors corresponding to the local state), then drawing the state at $t+1$ directly below, and so on (see ☞ Fig. 3). Higher dimensional CAs are conventionally visualized as animations.

☞ Figure 1 shows three basins of attraction of a small CA. There are many “repeated” basins (basins with identical topologies, over different specific states) due to the symmetries in

☞ Fig. 2

CA neighborhood. (a) A CA’s regular neighborhood, v , illustrated in a 2D lattice. The neighborhood of cell i is the image of the neighborhood of cell 0, translated by i . (b) The state of the neighborhood, illustrated in a 1D lattice. χ_i , the state of the neighborhood of cell i , is the projection of the full state s onto the neighborhood v_i .

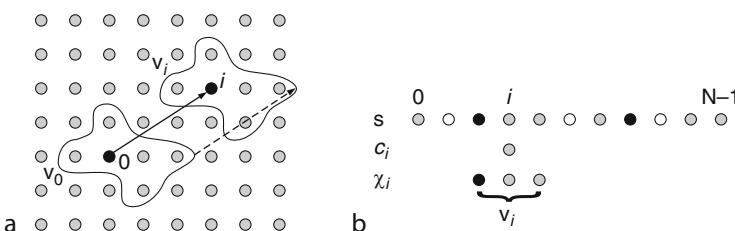
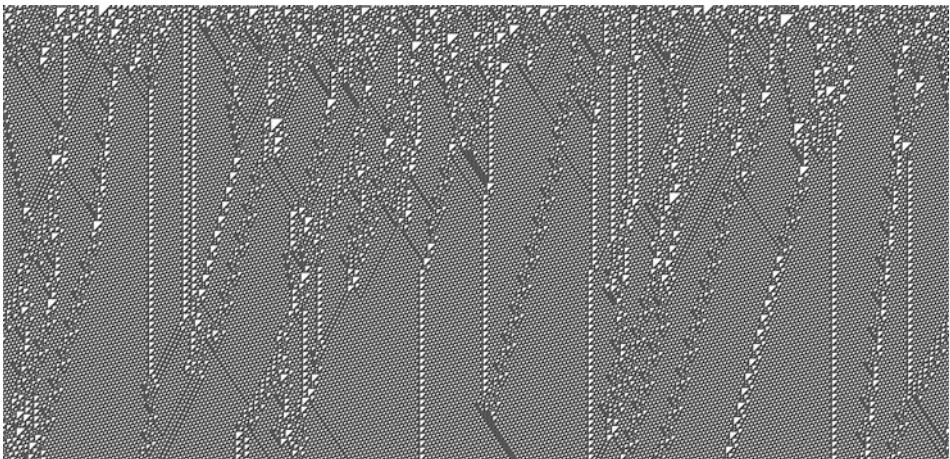


Fig. 3

Visualization of the time evolution of ECA rule 110 (see [Sect. 2.1.5](#) on Elementary Cellular Automata), with $N=971$, with random initial state s_0 , over 468 timesteps. Each horizontal line shows the N -bit representation of the state s_t ; subsequent lines correspond to subsequent timesteps s_t, s_{t+1}, \dots .



a CA rule (Powley and Stepney [2009a, b](#)). Every rule over a finite space with periodic boundary conditions has “shift” symmetries from shifting the arbitrary origin; additionally, some rules also have reflectional and other symmetries.

Elementary Cellular Automata (ECAs)

ECAs are 2-state ($\mathcal{S} = \mathcal{B}$) CAs, with the cells arranged in a 1D lattice, and with a neighborhood size of 3 (comprising the cell and its immediate left and right neighbors). There are $2^{2^3} = 256$ distinct ECA rules, conventionally referred to by a base 10 number $0\dots255$, representing the base 10 interpretation of the rule table bitstring. After reflection and inversion symmetries have been taken into account, there are 88 essentially distinct rules.

Wolfram’s Classification

Wolfram (1984b, a) provides a qualitative characterization of CAs, classifying their long-term evolution into four classes:

1. A unique homogeneous state, independent of the initial state (a single-state attractor cycle); patterns disappear with time
2. A simple periodic pattern of states (short attractor cycles, length $\ll |\mathcal{S}|^N$); patterns become fixed
3. Chaotic aperiodic pattern of states (long attractor cycles, length $O(|\mathcal{S}|^N)$, or no cycles in the infinite case); patterns grow indefinitely
4. Complex localized long-lived structures; patterns grow and contract

Wolfram’s classification scheme has been criticized for a variety of reasons, including the fact that even determination of quiescence (long-term fixed patterns) is undecidable (Culik and Yu [1988](#)). This is a recurring problem for any classification scheme: “CA behavior is so

complex that almost any question about their long-term behavior is undecidable” (Durand et al. 2003). See also Sutner (2005). Wolfram classes are nevertheless widely used in a qualitative manner to distinguish kinds of behaviors.

Class 3 and class 4 CAs demonstrate *sensitive dependence on initial conditions*: The effect of a minimal (one cell state) change to the initial condition propagates across the system, eventually resulting in a completely different dynamics (see ▶ Fig. 4). The 88 essentially distinct ECAs cover all 4 Wolfram classes of behavior.

Universality

Even though CA rules capture only a small fraction of all the possible dynamics over an ND space, there are computationally universal CAs that can emulate a TM. ECA rule 110 (▶ Fig. 3) is universal (Cook 2004), as is Conway’s Game of Life 2D CA (Berlekamp et al. 1982; Rendell 2002). The proof of universality in these cases involves constructing a virtual machine in the CAs, on which is implemented a TM (or equivalent). In other words, the computation is being performed by a carefully engineered initial condition, and a carefully engineered interpretation of the dynamical behavior. So these systems explore only a small fraction of their full state space: the fraction that corresponds to an interpretation of the state of a TM.

Wolfram (1984b, §8) speculates that “class 4 cellular automata are characterized by the capability for universal computation” (in the case of infinite-dimensional CAs).

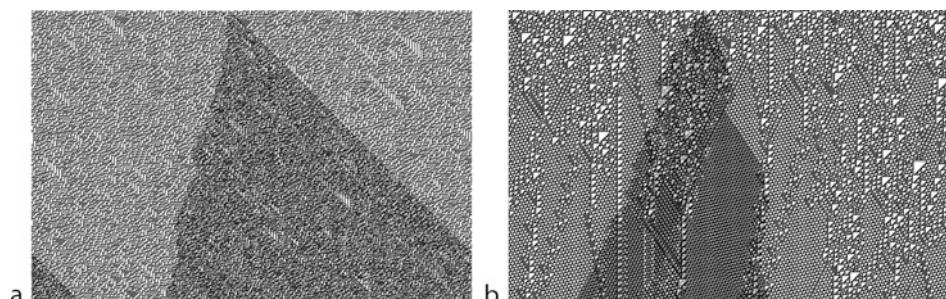
Example CA 1: The Density Classification Task

The requirement is to design a local two-state 1D CA rule ϕ , independent of lattice size N (assume N odd for simplicity), such that the global dynamics f has the following properties: (i) the system has two attractors, each of cycle length 1, one being the all zeros state, one being the all ones state; (ii) any initial state s_0 with more zeros than ones is in the basin of the all zeros state, and vice versa; (iii) the maximum transient length is at worst polynomial in n (i.e., the attractor is discovered in polynomial time). The computation determines which attractor basin the initial state is in, and hence, by inference, whether the initial state has more zeros than ones.

No two-state CA with a bounded neighborhood size can solve this problem exactly on arbitrary lattice size N (Land and Belew 1995). The best rules designed or evolved (e.g., Gács et al. 1978; Mitchell et al. 1996; Wolz and de Oliveira 2008) fail to correctly classify about 20%

◀ Fig. 4

Sensitive dependence on initial conditions. Each plot overlays the evolutions of two initial states differing in only one bit: the growing central dark region is different; the outer regions are the same. $N=971$, random initial state s_0 , over 646 timesteps (a) ECA rule 45; (b) ECA rule 110.



of states, that is, they define a dynamics where these states are in the “wrong” basin of attraction. Bossomaier et al. (2000) investigate the problem specifically in terms of the attractor basins.

Example CA 2: The Rule 30 Pseudorandom Number Generator

Wolfram (1986b) discusses ECA rule 30:

$$\phi_{30}(c_{i-1}, c_i, c_{i+1}) = c_{i-1} \text{ XOR } (c_i \text{ OR } c_{i+1}) \quad (1)$$

Starting from an initial state s_0 with a single cell j “on,” $c_{i,0} = (\text{if } i = j \text{ then } 1 \text{ else } 0)$, (Fig. 5), then the sequence of bits under this single on bit, $\tau = c_{j,0}, c_{j,1}, \dots, c_{j,t}, \dots$ forms a (pseudo) random sequence. Wolfram (1986b) presents evidence that the cycle length of the attractor starting from a state with a single nonzero cell grows exponentially with CA lattice size N .

This sequence τ is a projection (onto the single Boolean state variable c_j) of the trajectory from state s_0 under the dynamics defined by rule 30.

2.1.6 Random Boolean Networks

A random Boolean network (RBN) comprises N nodes. Each node i at time t has a binary valued state, $c_{i,t} \in \mathcal{B}$. Each node has k inputs assigned randomly from k of the N nodes (an input may be from the node itself); the wiring pattern is fixed throughout the lifetime of the network. This wiring defines the cell’s neighborhood, v_i . See Fig. 6.

The state of cell i ’s neighborhood at time t is $\chi_{i,t} \in \mathcal{B}^k$, a k -tuple of cell states that is the projection of the full state onto the neighborhood v_i .

Each node has its own randomly chosen local state transition rule, or update rule, $\phi_i : \mathcal{B}^k \rightarrow \mathcal{B}$. These cells form a network of state transition machines. At each timestep, the state of each cell is updated in parallel, $c_{i,t+1} = \phi_i(\chi_{i,t})$.

Fig. 5

ECA rule 30 (Eq. 1), initial state of a single nonzero cell.

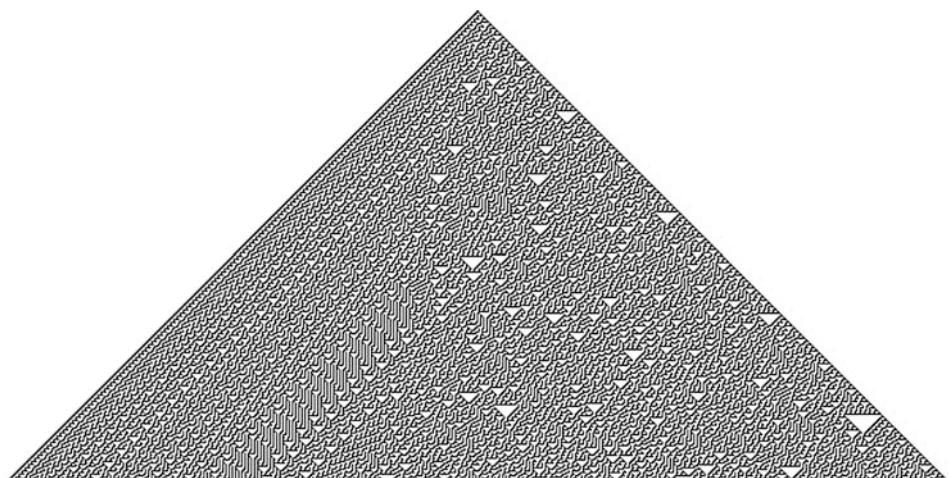
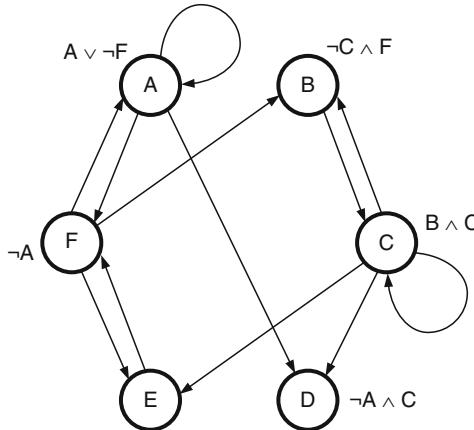


Fig. 6

An example random Boolean network (RBN) with $N=6, k=2$. Each node has $k=2$ inputs; it can have any number of outputs. So the neighborhood function is $v_A=(A,F)$, $v_B=(C,F)$, etc. Each node combines its inputs by a random Boolean function ϕ_i ; that function might ignore one (or both) of the inputs.



The global dynamics f is determined by the local rules ϕ_i and the connectivity pattern of the nodes v_i . In contrast to the regularity of a CA, in an RBN each node has its own random neighborhood connection pattern and its own random update rule. Like CAs, RBNs capture only a small fraction of all the possible dynamics over an ND space.

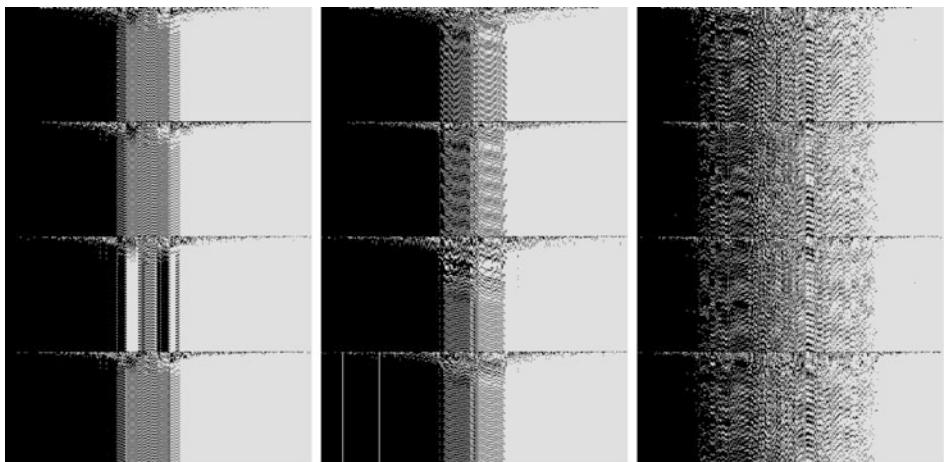
Kauffman (1990, 1993) investigates the properties of RBNs as a function of connectivity k . (The wiring conditions given above are not stated explicitly in these references. However, in the $k=N$ case, Kauffman (1993, p.192) states that “Since each element receives an input from all other elements, there is only one possible wiring diagram.” This implies that multiple connections from a single node are not allowed (otherwise more wiring diagrams would be possible) whereas self connections are allowed (otherwise k would be restricted to a maximum value of $N-1$). Subsequent definitions (e.g., Drossel 2008) explicitly use the same conditions as given here.) Unlike CAs, RBNs tend not to have repeated basins, because of the random nature of the connections, and hence the relative lack of symmetries. Despite all the randomness, however, “such networks can exhibit powerfully ordered dynamics” (Kauffman 1990), particularly when $k=2$ (Fig. 7; Table 1). Drossel (2008) notes that subsequent computer simulation of much larger networks shows that “for larger N the apparent square-root law does not hold any more, but that the increase with system size is faster.”

Kauffman identifies $k=1$ as the “ordered” regime, with a very large number of short period attractors. Large k is the chaotic regime, with very long period attractors (compare Wolfram’s class 3 behavior). $k=2$ occurs at a “phase transition” (Kauffman 1990), separating the ordered and chaotic regimes; it exhibits a moderate number of moderate period attractors.

Kauffman investigates RBNs as simplified models of gene regulatory networks (GRNs). He notes that “cell types are constrained and apparently stable recurrent patterns of gene expression,” and interprets his RBN results as demonstrating that a “cell type corresponds to a state cycle attractor” (Kauffman 1993, p. 467) (in a $k=2$ network).

Fig. 7

Visualization of the time evolution of three typical $k=2$ RBNs, with $N=400$, and initial condition all nodes “off”; after 150 timesteps all nodes are set to “on,” then all nodes are randomized (50% “on,” 50% “off”) every further 150 timesteps, to explore other attractors. They exhibit ordered behavior: short transients and low period attractors. The visualization scheme used here (Stepney 2009) orders the nodes to expose the frozen core (Kauffman 1993, p. 203) of nodes that do not change state on the attractor; this frozen core is well preserved on different attractors.

**Table 1**

Dynamics of random Boolean networks (RBNs) for different k (Adapted from Kauffman (1993, Table 5.1))

k	Attractor cycle length	# Attractors
1	$O(\sqrt{N})$	$O(2^N)$
2	$O(\sqrt{N})$	$O(\sqrt{N})$
>5	$O(2^N)$	$O(N)$

Emergent macrostates of the dynamics, in addition to the attractor cycle length, are the number of nodes whose states change during a cycle, compared to the number that form the static *frozen core*. In the GRN interpretation, the frozen core would correspond to genes whose regulatory state was constant in a particular cell type, and the changing nodes to those genes whose regulatory state was cycling.

2.2 Continuous Space, Discrete Time

We next consider continuous spaces, with $\mathcal{X} = \mathcal{R}$ (where \mathcal{R} is the set of real numbers), with discrete time dynamics $t \in \mathbb{N}$. Let the state vector be $\mathbf{r} \in \mathcal{R}^N$. The dynamics of a particular system is determined by its particular transition function $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$, with $\mathbf{r}_{t+1} = f(\mathbf{r}_t)$.

These systems are called *difference equations* or *iterated maps*. A discrete-time trajectory $\mathbf{r}_t, \mathbf{r}_{t+1}, \mathbf{r}_{t+2}, \dots$ is also called an *orbit*.

The trajectories can display a range of behaviors, corresponding to a range of types of attractor, depending on the system. Trajectories may diverge ($|\mathbf{r}_t| \rightarrow \infty$); they may converge to a fixed point attractor ($\mathbf{r}_t \rightarrow \mathbf{r}^*$, where $f(\mathbf{r}^*) = \mathbf{r}^*$); they may converge to a periodic attractor; they may be chaotic, never repeating but still confined to a particular subregion of the state space. So, continuous space systems have chaotic behavior that differs qualitatively from the “chaotic” behavior of discrete space systems, since the finite discrete systems must eventually repeat and hence be periodic (although with exponentially long periods).

2.2.1 Parameterized Families of Systems

It is often convenient to consider a family of dynamics related by some parameter $p \in \mathcal{P}$, that is, $f(\mathbf{r}, p)$, and investigate how the dynamics of a system vary as a function of this parameter. The trajectories of such parameterized systems can display the whole range of behaviors corresponding to the range of types of attractor, depending on the value of the parameter. A small change to the parameter can make a fixed point move, or become unstable, or periodic, or disappear. It can move a system from period P to period $2P$ (*period doubling*), or from periodic to chaotic behavior. The parameter values where these qualitative changes in the dynamics occur are called *bifurcation points*. As the parameter crosses the bifurcation point, the change in the dynamics can be continuous (smooth), or discontinuous (*catastrophic*).

Many systems exhibit a sequence of period doublings as the parameter changes. Subsequent doublings happen ever more rapidly (requiring ever smaller changes to the parameter), then the system moves into a chaotic regime. This is known as the *period doubling route to chaos*. Appearance of a period doubling cascade in a parameterized system is indication that chaos will ensue if the parameter changes further.

Another typical behavior is the *intermittency route to chaos*. Here the parameter starts in a region with periodic dynamics; as it is changed, the periodic behavior is broken by intermittent bursts of irregular behavior. As the parameter changes further, there are more and more of these bursts, until the behavior becomes completely chaotic. Hence a fully deterministic system may be apparently periodic, interrupted by what look like bursts of noise, where these bursts are simply part of the same overall dynamics of the system, and in need of no external explanation.

2.2.2 Logistic Map

The logistic map, a parameterized family of 1D iterated maps,

$$r_{t+1} = \lambda r_t (1 - r_t) \quad (2)$$

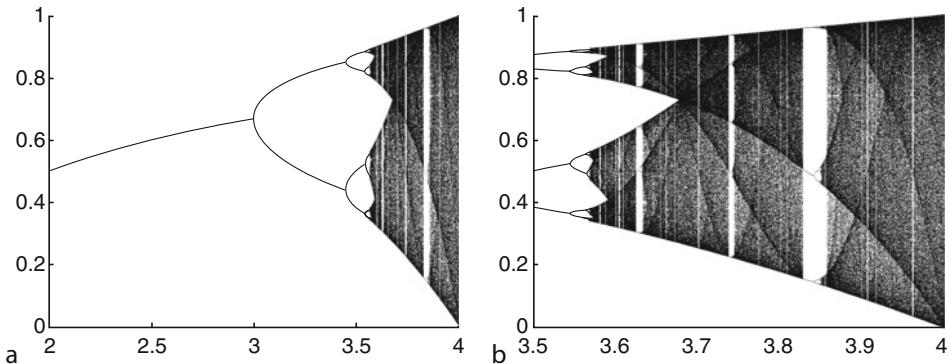
is a well-studied example of such a system (see ▶ Figs. 8 and ▶ 9). It is usually studied for $\lambda \in [0, 4]$, since in this region its dynamics is confined to the unit interval: $r \in [0, 1]$. It does have complex dynamics for other values of λ , but these are not constrained to the unit interval.

Its unintuitively rich, complex properties have been researched in detail, and May (1976) advised:

- ▶ Not only in research, but also in the everyday world of politics and economics, we would all be better off if more people realised that simple nonlinear systems do not necessarily possess simple dynamical properties.

Fig. 8

Orbit diagram of the logistic map, (a) $2 \leq \lambda \leq 4$, showing the period doubling route to chaos; (b) $3.5 \leq \lambda \leq 4$, zooming in on the window of period 3.



The attractor structure, visible in [Fig. 8](#), is summarized in [Table 2](#) as a function of parameter λ . It has period doubling cascades to chaos as λ is increased. The first cascade leads to the onset of chaos at $\lambda = 3.56994\ 56718\dots$. Within this chaotic region, there are windows of periodicity (such as the window of period 3), which then also period-double back to chaos. Given the existence of a period 3 cycle in a map, then every possible period can also be found in that map (Li and Yorke 1975). In the logistic map, there are windows of every period for some $3 < \lambda < 4$. Each of these periodic windows then period-doubles back to chaos as λ increases. The order in which these cascades occur itself has a complex structure, which can be calculated iteratively, in terms of symbolic sequences (Metropolis et al. 1973); the lowest order sequences are shown in [Table 2](#).

Cycles of the same period can nevertheless have different kinds of behaviors: see, for example, [Fig. 10](#).

The logistic map also exhibits the intermittency route to chaos. If the parameter λ falls in a window of periodic behavior, and is then slowly *reduced*, intermittent behavior is seen (e.g., [Fig. 9e](#)), until fully chaotic behavior is reached.

Binary Shift Map

Consider the fully chaotic case, $\lambda=4$. Changing variables, to $x = \frac{1}{\pi} \cos^{-1}(1 - 2r)$, yields the equation for the binary shift (Bernoulli) map:

$$x_{t+1} = 2x_t \bmod 1 \quad (3)$$

If x is expressed in base 2, each iteration of the map results in a left shift of the number (the multiplication by 2), and dropping any resulting leading 1 in front of the binary point (the mod 1). For example, if $x_t = 0.1110001\dots$, then $x_{t+1} = 0.110001\dots$, $x_{t+2} = 0.10001\dots$, etc. If x_0 is rational, its binary expansion is periodic, and hence the iteration will be periodic; if x_0 is irrational, its binary expansion is nonperiodic, and hence the iteration will be nonperiodic. Separate values of x_0 that are the same up to their n th bit will initially have similar iterations, but will eventually diverge, until they are completely different at the n th iteration: This is a manifestation of sensitive dependence on initial conditions.

Fig. 9

Timeseries of the logistic map, 100 iterations, with $r_0=0.1$, for various λ . Top row: (a) $\lambda=2.8$, period 1; (b) $\lambda=3.3$, period 2. Second row: (c) $\lambda=3.5$, period 4; (d) $\lambda=3.74$, period 5. Third row: (e) $\lambda=3.828$, chaos before period 3: period 3 behavior is interleaved with intermittent bursts of chaotic behavior; (f) $\lambda=3.829$, period 3. Bottom row: (g) $\lambda=4$, chaos; (h) $\lambda=4$, with $r_0=0.10001$, demonstrating sensitive dependence on initial conditions.

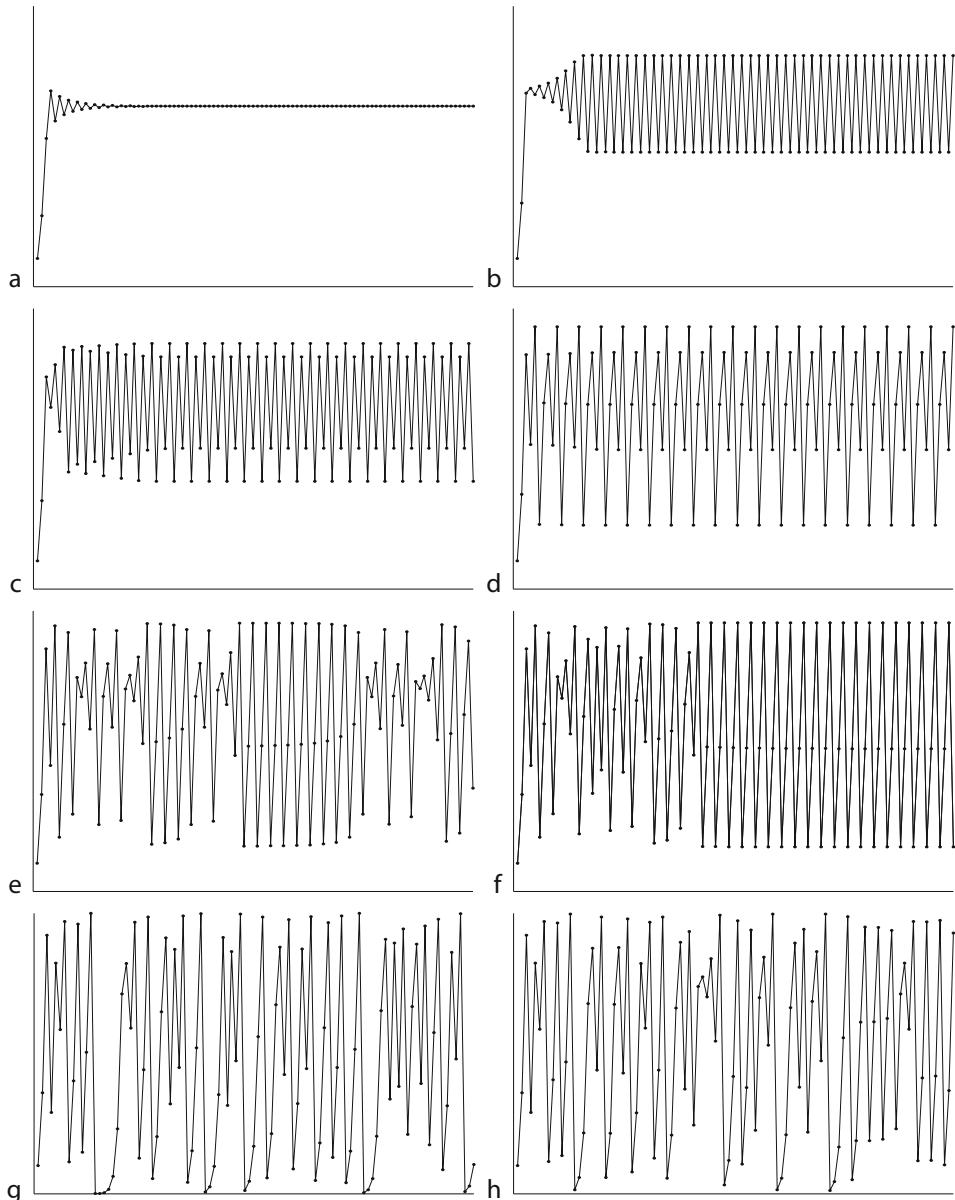


Table 2

Dynamical structure of the logistic map as a function of parameter λ (some λ values taken from Sloane (2008)); including order of occurrence of period doubling cascades, for all initial periods up to 7 (Adapted from Hao and Zheng (1998, Table 2.2)). Higher initial periods cascades are interleaved with these

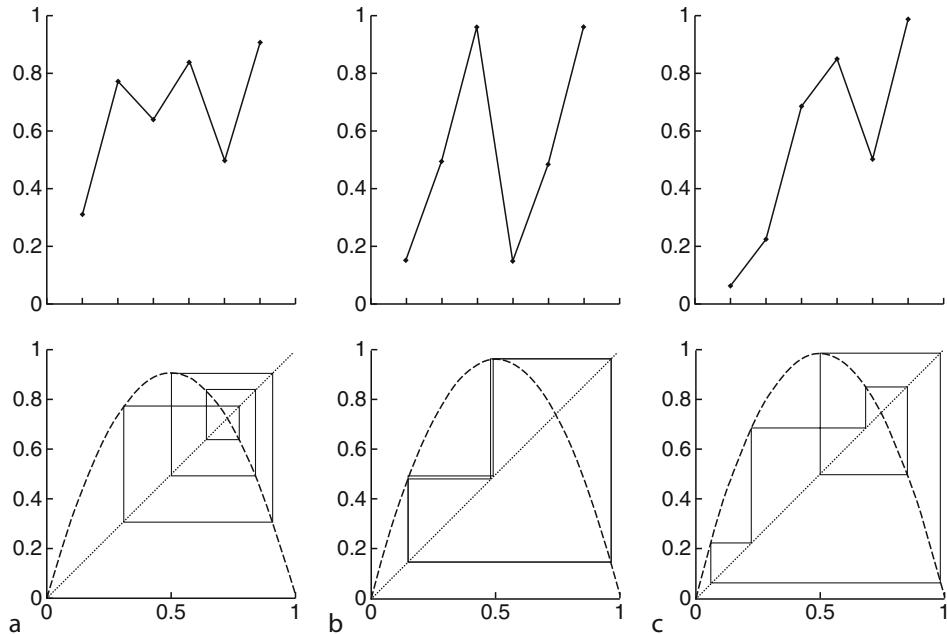
Lambda	Dynamics	Sloane number (Sloane 2008)
(0,1)	Fixed point $r^* = 0$	
(1,3)	Fixed point, function of λ (e.g., $\lambda=2, r^* = 0.5$)	
3	1st period doubling, to period 2	
$1 + \sqrt{6} = 3.449\dots$	2nd period doubling, to period 4	
3.54409 03595...	3rd period doubling, to period 8	A086181
3.56440 72660...	4th period doubling, to period 16	A091517
3.56994 56718...	End of first cascade; first onset of chaos	A098587
3.62655 31616...	Appearance of period 6; period 6 cascade	A118453
3.70164 07641...	1st period 7 cascade	A118746
3.73817 23752...	Appearance of period 5; 1st period 5 cascade	A118452
3.7741...	2nd period 7 cascade	
$1 + 2\sqrt{2} = 3.828427\dots$	Appearance of period 3; sole period 3 cascade	
3.841499...	Period doubling, to period 6	
3.8860...	3rd period 7 cascade	
3.9055...	2nd period 5 cascade	
	4th period 7 cascade	
3.9375...	Period 6 cascade	
	5th period 7 cascade	
3.9601...	Period 4 cascade	
	6th period 7 cascade	
3.9777...	Period 6 cascade	
	7th period 7 cascade	
3.9902...	3rd and last period 5 cascade	
	8th period 7 cascade	
	period 6 cascade	
	9th and last period 7 cascade	
4	Fully chaotic	

Computational Properties

Crutchfield (1994) investigates the *statistical complexity* of the logistic map as a function of λ . The statistical complexity is essentially a measure of the size of a stochastic finite state machine that can predict the statistical properties of a system. He finds that the map has *low* statistical complexity both when the map is periodic and when it is chaotic (essentially random), and has *highest* statistical complexity at the onset of chaos (via the period doubling route, or the intermittent route). At this point, there is a *phase transition* in the complexity of the

Fig. 10

Different classes of behavior of three different period 6 cycles. Top shows the time series of one period; bottom shows the geometry of the behavior in a cobweb diagram. (a) $\lambda = 3.6266$; (b) $\lambda = 3.8418$; (c) $\lambda = 3.93755$.



machine needed to predict the behavior, and these parameter values indicate the highest computational capacity.

Despite these observations, the majority of computational applications of the logistic map exploit its completely chaotic behavior, with λ at or near 4, and use this behavior to implement random number generators (Kanso and Smaoui 2009; Phatak and Rao 1995; Ulam and von Neumann 1947), encryption (Kocarev and Jakimoski 2001; Pareeka et al. 2006), etc.

2.2.3 Coupled Map Lattices

Kaneko (1983, 1984, 1985, 1986) introduces one-dimensional coupled map lattices (CMLs), where an array of N iterated maps are coupled together locally, with the following local dynamics ϕ :

$$r_{i,t+1} = \phi_0(r_{i,t}) + \frac{\varepsilon}{2} (\phi_c(r_{i-1,t}) - 2\phi_c(r_{i,t}) + \phi_c(r_{i+1,t})) \quad (4)$$

where ε is the coupling strength. Each element in the lattice evolves under the dynamics of the local process ϕ_0 , with an additional interaction contribution $\varepsilon\phi_c$ from its neighbors, while passing on a similar amount of its own to its neighbors (under periodic boundary conditions). The local state $r_i \in \mathcal{R}$ and the local dynamics $\phi : \mathcal{R}^3 \rightarrow \mathcal{R}$ define the global state $\mathbf{r} \in \mathcal{R}^N$ and global dynamics $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$. The system is parameterized by the coupling strength ε , as well as by any parameters in ϕ .

An initial value of $r_{i,0} = \kappa$, where all the maps have the same initial value, is trivial, since all maps evolve in lock-step. Kaneko (1985) investigates several cases, one of which is where ϕ_0 is the logistic map with λ in the period 3 window with cycle (r_1^*, r_2^*, r_3^*) , with $r_{i \leq N/2,0} = r_1^*$, $r_{N/2 < i,0} = r_2^*$, or with random $r_{i,0}$, and with $\phi_c = \phi_0$. The dynamics exhibits the period doublings, intermittencies, and chaos of the logistic map, and in addition exhibits spatial patterns and structures similar to 1D CAs. Crutchfield and Kaneko (1987) examine the properties of this class of system in some detail.

Subsequent work generalizes the approach to topologies other than 1D nearest-neighbor (2D, tree-structured, irregular, larger neighborhoods), and allows the coupling to be asymmetric. In particular, Holden et al. (1992) provide a generic formalism, and investigate coupled map lattices in terms of their computational properties. A CML approach to the density classification problem has been evolved (Andersson and Nordahl 1998). Open CMLs are also being exploited computationally (see [Sect. 3.3.1](#)).

2.2.4 Note on Dimensionality and Topology

Papers on CMLs tend to describe them as having “discrete time, discrete space, and continuous state” (Kaneko 1986). Here we describe them as “continuous space,” because here we are talking purely about the *state space*.

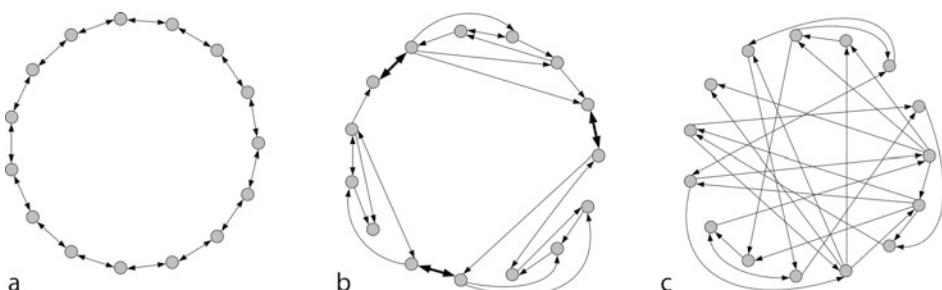
A CML of N maps laid out in 1D line in (physical) space has an (abstract) state space of \mathcal{R}^N . The “dimensionality” of the layout in physical space (a 1D line of maps) is unrelated to the dimensionality of the state space (of ND , because there are N maps); if the same N maps were instead laid out in a 2D grid, or even a 27D grid, say, it would not affect the dimensionality of the state space.

Instead, this “dimensionality” is related to the *topology* of the connections between the maps, and hence the potential information flow between the maps ([Fig. 11](#)).

Compare the information flow visible in a 1D (physical space) CA ([Fig. 4](#)) and that not visible in an RBN with a similar number of cells ([Fig. 7](#)). For this reason, it makes sense to talk of the (physical spatial) dimension of a CA (it has a regular local topology), but not of an RBN (it has an irregular graph topology).

Fig. 11

Dimensionality versus topology. Examples for three $N=5, k=2$ networks with different topologies. (a) Regular $N=15$ CA-like structure arranged in a 1D spatial lattice (periodic boundary conditions); (b) three $N=5, k=2$ RBNs linked in a 1D spatial lattice (that is, random connections with the groups of 5 nodes, lattice links between the groups); (c) general $N=15, k=2$ RBN.



When the physical spatial layout moves from discrete to continuous, the state space moves from being finite (discrete physical space, finite number of cells) or countably infinite (discrete physical space, countably infinite number of cells) to uncountably infinite (continuous physical space). See [Sect. 2.3.4](#).

2.2.5 Numerical Errors and the Shadowing Lemma

Chaotic systems, such as the logistic map with $\lambda=4$, display sensitive dependence on initial conditions: trajectories with nearby initial conditions diverge exponentially (compare [Fig. 9g](#) and [9h](#)).

Such systems are usually studied by numerical simulations, which generate *pseudo-orbits*, because of numerical noise. Additionally, the real physical systems that these maps (and below, differential equations) model are themselves subject to noise during their execution, and during measurement, and so potentially execute pseudo-orbits (or pseudo-trajectories) with respect to the model systems. The question naturally arises: What is the relation of these pseudo-orbits to the model orbits? Are they representative of the modeled dynamics?

Fortunately the answer is (a qualified) “yes.” The *Shadowing Lemma* states that, for certain classes of system, the pseudo-orbit *shadows* (stays close to) some true orbit of the (modeled) system for all time; this true orbit has a slightly different initial condition from the pseudo-orbit. This result has been extended to a wider class of systems, including those studied here, that the pseudo-orbit shadows some true orbit of the system for “a long time” (Hammel et al. 1988).

However, are the true orbits of the model that are shadowed by these pseudo-orbits themselves representative of the underlying dynamics; that is, are they typical true orbits? This is harder to answer, and is clearly false for some particular cases. For example, consider the binary shift map ([Eq. 3](#)). For any limited precision binary arithmetic implementation, *all* pseudo-orbits converge to 0 when the number of iterations (shifts) exceeds the binary numerical precision. But this case appears to be an exception, because most numerical trajectories do not behave like this, and Hayes and Jackson (2005) state:

- ▶ If otherwise reliable-looking pseudo-trajectories *are atypical*, they must be atypical in an extremely subtle way, because researchers have been making apparently reliable, self-consistent, peer-reviewed conclusions based on numerical simulations for decades.

So we continue here in assuming that the simulated pseudo-orbits, and the pseudo-trajectories of actual physical systems, are in general representative of the true dynamics defined by the equations.

2.3 Continuous Space, Continuous Time

We now consider continuous spaces, with $\mathcal{X} = \mathcal{R}$, with continuous time dynamics $t \in \mathcal{R}$. Let the state vector be $\mathbf{r} \in \mathcal{R}^N$. The dynamics of a particular system is determined by its particular transition function $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$, with $\dot{\mathbf{r}} = f(\mathbf{r})$. Hence the system is defined by a set of N coupled first-order ordinary differential equations (ODEs).

We can recast higher order equations into this normal form by adding new variables. For example, consider the 1D equation for damped simple harmonic motion

$$\ddot{r} + \kappa \dot{r} + \omega^2 r = 0 \quad (5)$$

Let $r_1 = r, r_2 = \dot{r}$. Then, rearranging, we can get the 2D normal form version:

$$\dot{r}_1 = r_2 ; \quad \dot{r}_2 = \omega^2 r_1 - \kappa r_2 \quad (6)$$

Note how this normalization takes the single-state variable r , and results in two-state variables r_1 (r , position) and r_2 (\dot{r} , velocity). In a continuous system, and particularly when the state variables are position \mathbf{r} and momentum $m\dot{\mathbf{r}}$, the state space is also called the *phase space*.

Physical systems embody their own specific dynamics. If that dynamics can be controlled and exploited in a computational manner, it can be used to reduce the load on, or even replace, conventional classical digital control in embedded systems (Stepney 2007). Understanding the dynamical behavior of complex material systems from a computational perspective is also a necessary step along the way to understanding biological systems as information processing systems (Stepney 2008).

For a good overview of continuous dynamical systems from an embodied computational perspective, see Beer (1995). Abraham and Shaw (1987) provide an excellent visual description of various concepts such as attractors and bifurcations. For more background, see a textbook such as that by Strogatz (1994).

2.3.1 Kinds of Attractors

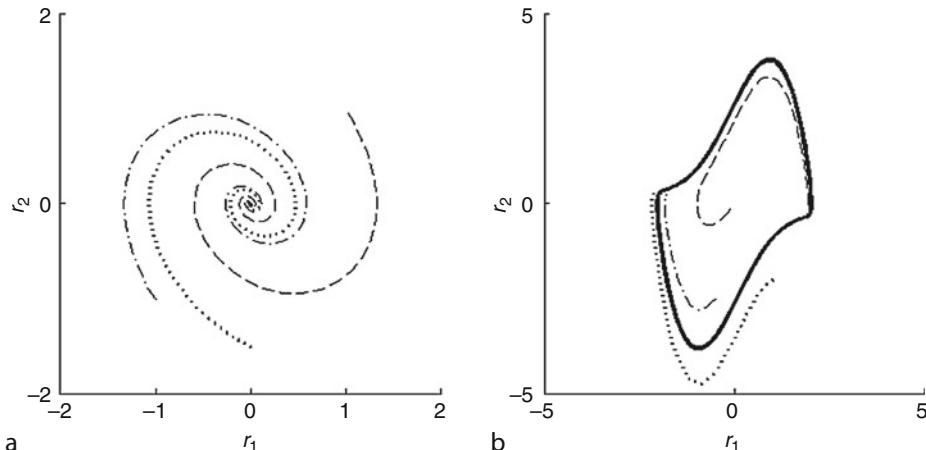
In these continuous time systems, trajectories are continuous paths through a continuous state space. There are four distinct kinds of attractor that can occur.

Point Attractors

A point attractor is a single point in state space that attracts the trajectories in its basin. An example is the equilibrium position and zero velocity that is the unique end state of damped simple harmonic motion (☞ Fig. 12a).

Fig. 12

Attractors and their transient trajectories: (a) point attractor: damped SHM, $\ddot{r} + \lambda \dot{r} + r = 0$, with $\lambda=0.5$; (b) limit cycle attractor: van der Pol oscillator, $\ddot{r} + \lambda(r^2 - r)\dot{r} + r = 0$, with $\lambda=2$.



Limit Cycle Attractors

A limit cycle is a closed loop trajectory that attracts nearby trajectories to it. See, for example, (●) Fig. 12b.

Toroidal Attractors

A toroidal attractor is a 2D surface in state space with periodic boundary conditions: shaped like a torus. Trajectories are confined to the surface of the torus. If the *winding number* (the number of times the trajectory loops around in one dimension while it performs one loop in the other dimension) is rational, the trajectory is periodic, otherwise it is *quasiperiodic*, and eventually covers essentially the entire surface of the torus.

Strange Attractors

A strange attractor attracts trajectories to its region of state space, but within this region, nearby trajectories diverge exponentially: it exhibits sensitive dependence on initial conditions, and thus chaotic behavior. This combination of attraction and divergence requires at least three dimensions in which to occur. The detailed structure of a strange attractor is usually fractal.

Example: Rössler Strange Attractor

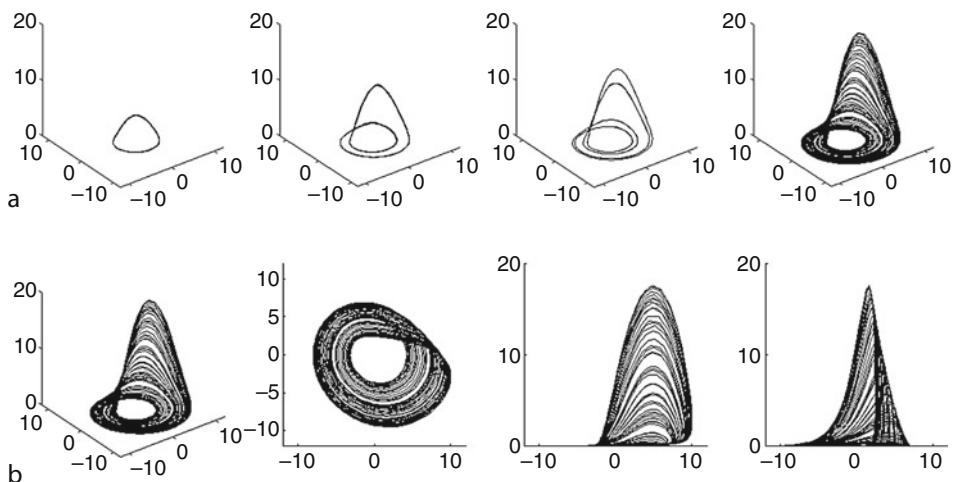
The Rössler system is defined by

$$\begin{aligned}\dot{r}_1 &= -r_2 - r_3 \\ \dot{r}_2 &= r_1 + ar_2 \\ \dot{r}_3 &= b + r_3(r_1 - c)\end{aligned}\tag{7}$$

It is a family of dynamical systems that displays a range of kinds of dynamics, some with strange attractor behavior, some without. It exhibits the period doubling cascade route to chaos (●) Fig. 13a).

■ Fig. 13

Rössler system: (a) period doubling cascade to chaos, with $a=b=0.2$, $c=2.5, 3.5, 4.0, 5.0$; (b) Rössler strange attractor.



The Rössler strange attractor occurs when $a=0.2, b=0.2, c=5$ (☞ Fig. 13b). It is the simplest strange attractor, with only one nonlinear term.

Example: Lorenz Strange Attractor

The Lorenz strange attractor is defined by

$$\begin{aligned}\dot{r}_1 &= 10(r_2 - r_1) \\ \dot{r}_2 &= 28r_1 - r_2 - r_1r_3 \\ \dot{r}_3 &= r_1r_2 - 8r_3/3\end{aligned}\tag{8}$$

(See ☞ Fig. 14.) It is a member of a family of dynamical systems that displays a range of kinds of dynamics, some with strange attractor behavior, some without.

Sensitive dependence on initial conditions is popularly known as *the butterfly effect*. Lorenz suggests (Lorenz 1993, p. 14) that the name may have arisen from the title of a talk he gave in 1972, “Does the Flap of a Butterfly’s Wings in Brazil Set Off a Tornado in Texas?,” coupled with the butterfly-like shape of Lorenz attractor seen from some directions (☞ Fig. 14).

Reconstructing the Attractor

Given a physical continuous dynamical system with a high-dimensional state space (state vector \mathbf{r}_t), one can determine properties of its dynamics, given only scalar discrete time series observations (time series data $r_\tau, r_{2\tau}, \dots, r_{n\tau}, \dots$, where r_t is some scalar projection of the state vector \mathbf{r}_t). In particular, one can distinguish chaos (motion on a strange attractor) from noise.

The process of reconstructing the attractor from this data involves constructing a d -dimensional state vector $\hat{\mathbf{r}}_t$ from a sequence of time-lagged observations:

$$\hat{\mathbf{r}}_{nt} = (r_{nt}, r_{(n+k)\tau}, r_{(n+2k)\tau}, \dots, r_{(n+(d-1)k)\tau})\tag{9}$$

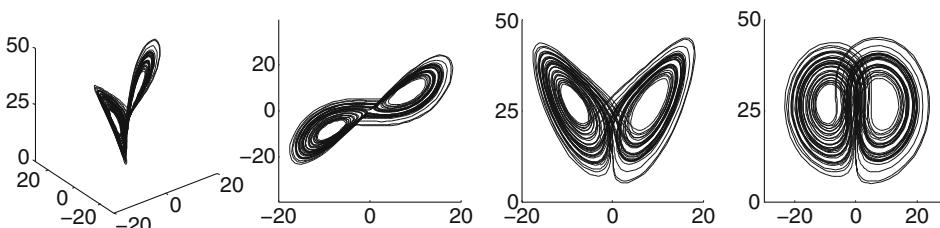
d should be $> 2d_a$, where d_a is the dimension of the system’s attractor; d should also be as small as possible, to avoid fitting noise; k should be large enough that the attractor is sufficiently sampled, but not so large that the correlations are lost. Taken’s *embedding theorem* then relates the invariants of motion, including attractor structure, of $\hat{\mathbf{r}}$ to those of \mathbf{r} . For more on this process, and other techniques for analyzing chaotic systems, see Ott et al. (1994).

Relationship to Discrete State Space Attractors

Wolfram (1984b) draws a rough analogy between his class 1, 2, and 3 CAs (see ☞ Sect. 2.1.5) and point, limit cycle, and strange attractors, respectively. He mentions that the class 4 CAs (the ones conjectured universal) “behave in a more complicated manner.” This might be thought to imply that there is no continuous analog of the discrete class 4 systems, and hence

Fig. 14

Lorenz strange attractor.



no universal computational properties in continuous matter. This is not so, as we see below (Sects. 2.3.4 and 2.3.5).

Kauffman (1990) calls RBNs whose attractor cycle length increases exponentially with n , “chaotic.” He emphasizes that this does not mean that flow on the attractor is divergent (it cannot be, in a discrete deterministic system); the state cycle is the analog of a 1D limit cycle. However, there is an analogy: Exponentially long cycles cover a lot of the state space before repeating (chaotic strange attractors never repeat), and “nearby” states (1 bit different) potentially *do* diverge (even possibly onto another attractor). However, in the discrete system, there is no direct analog of “nearby states diverging exponentially, but staying on the same attractor,” since there is usually no concept of distance between states in discrete dynamical systems, and if there were, successive hops through state space can be of any size: there is no simple “continuity” from which to diverge.

2.3.2 Computation in Terms of Attractors

We can interpret computation as finding which attractor basin the system is in, by following its trajectory to the relevant attractor. The output could be (some projection of) the computed attractor, including a subspace of the state space. Most instances of analog computing fall in this domain.

The programming problem is in finding the relevant dynamics, now restricted to natural (albeit engineered) material system properties, which are not arbitrary. The aim is to minimize the engineering required to implement the desired dynamics, by exploiting the natural dynamics.

Continuous systems computing in this way can exhibit *robustness*. A small perturbation to the system might shift it a small distance from its attractor, but its subsequent trajectory will converge back to the attractor. There can be a degree of continuity in the attractor basins, such that a small perturbation tends to remain in the same basin, unlike the discrete case.

It is not necessary for a dynamical system to have a complex (chaotic, strange) dynamics in order to be interesting or useful. Kelso (1995, p. 53) makes this point eloquently:

- ▶ Some people say that point attractors are boring and nonbiological; others say that the only biological systems that contain point attractors are dead ones. That is sheer nonsense from a theoretic modeling point of view, as it ignores the crucial issue of what fixed points refer to. When I talk about fixed points here it will be in the context of collective variable dynamics of some biological system, not some analogy to mechanical springs or pendula.

That is, the dynamics, including the underlying attractor structure, is part of the specific model, in particular, what state variables are used to capture the real-world system. State variables can capture more sophisticated concepts than simple particle positions and momenta.

2.3.3 Continuous Time Logistic Equation

The logistic growth equation is one of the simplest biologically based nonlinear ODEs. It is a simple model of population growth where there is exponential growth for small populations,

but an upper limit, or *carrying capacity*, that prevents unbounded growth. The equation was suggested by Verhulst in 1836 (see, e.g., Murray 1993, p. 2):

$$\dot{r} = \rho r(1 - r/\kappa) \quad (10)$$

It is rare among nonlinear ODEs in that it has an analytic solution

$$r(t) = \frac{\kappa r_0 e^{\rho t}}{\kappa + r_0(e^{\rho t} - 1)} \quad (11)$$

It has a single point attractor at $r_\infty = \kappa$: whatever the initial population r_0 , it always converges to the carrying capacity κ .

Contrast this smooth behavior with the very different complex periodic and chaotic behavior of its discrete time analog: the logistic map, [Sect. 2.2.2](#). This is a general feature: The discrete time analog of simple ODEs can exhibit similarly complex behavior as the logistic map. This does *not* mean, however, that ODEs themselves are unable to display computationally interesting dynamics.

2.3.4 Infinite Dimensions: PDEs

Consider the reaction–diffusion (RD) equation, which models chemical species reacting (nonlinearly) locally with each other, and diffusing (linearly) through space. The relevant state variables are the concentrations of the reacting diffusing chemicals, and are functions of time, and also of space: $r_i(t, \mathbf{x})$. For a two-chemical species, the RD equation is

$$\begin{aligned} \frac{\partial r_1}{\partial t} &= f_1(r_1, r_2) + k_1 \nabla^2 r_1 \\ \frac{\partial r_2}{\partial t} &= f_2(r_1, r_2) + k_2 \nabla^2 r_2 \end{aligned} \quad (12)$$

Each r_i , since it is a function of continuous space \mathbf{x} , can be thought of as an (uncountably) infinite-dimensional state variable. Rather than having a state vector with a finite number of indices r_i , we can consider an infinite-dimensional state vector indexed by position, $r(\mathbf{x})$. The state variable at each position can itself have multiple components (such as the two chemical concentrations, above), leading to $\mathbf{r}(\mathbf{x})$. The space derivative is used to define the dynamics in terms of a local (infinitesimal) neighborhood.

There is a natural link between partial differential equation (PDE) systems and cellular automata. CAs are one natural way to simulate PDEs in a discrete domain (Adamatzky et al. 2005; Wolfram 1985, prob. 9). Care is needed in this process to ensure that the CA models the correct PDE dynamics, and does not introduce artefacts due to its own discrete dynamics (Weimar and Boon 1994; Wolfram 1986a). Despite this caveat, there are some exact correspondences: *ultradiscretization* (Tokihiro et al. 1996) can be used to derive CA-like rules that preserve the properties of a given continuous system, for a class of integrable PDEs; *inverse ultradiscretization* (Kunishima et al. 2004) transforms a CA into a PDE, preserving its properties. A different approach represents CA configurations using continuous *bump functions* (Omohundro 1984), and derives a PDE that evolves the bumps to follow the given CA rule.

The dynamical theory of these infinite-dimensional spaces is not as well developed as in the finite case. Much of the work concentrates on PDEs whose dynamics can be rigorously reduced to a finite subspace, so that the existing dynamical systems theory is applicable. See, for example, Robinson (2001) and Teman (1997).

2.3.5 Reaction-Diffusion Computers

Reaction-diffusion computers (Adamatzky et al. 2005) use chemical dynamics. The relevant state variables are the concentrations of the reacting diffusing chemicals, which are functions of time and space: $r_i(t, \mathbf{x})$. For a two-chemical species, the relevant RD equation is given by [Eq. 12](#).

Reaction-diffusion systems have a rich set of behaviors, exhibiting spatial-temporal patterns including oscillations and propagating waves. The computation is performed by the interacting wave fronts; the output can be measured from the concentrations of the reagents. RD systems have been used to tackle a wide variety of computation problems (e.g., image processing (Kuhnert et al. 1989), robot navigation (Adamatzky et al. 2004)); here we look at two that demonstrate computation exploiting the natural dynamics, and one that demonstrates the potential for universal computation.

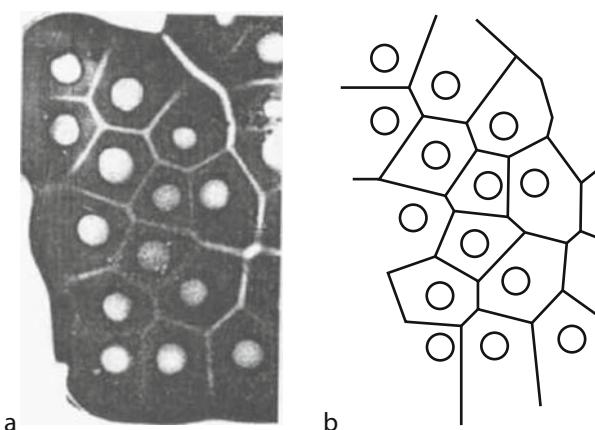
Voronoi Diagrams

An RD computer can solve a 2D Voronoi problem: given a set S of points in the plane, divide the plane into $|S|$ regions R such that every point in a given region $R_i(s_i)$ is closer to s_i than to any other $s_j \in S$.

This problem can be solved directly by a 2D RD computer (Tolmachiev and Adamatzky 1996) ([Fig. 15](#)). One reagent forms a substrate; the second reagent marks the position of the data set of points S . The data-reagent diffuses and reacts with the substrate-reagent, forming waves propagating from each data point, and leaving a colored precipitate. Waves meet at the borders of the Voronoi regions, since their constant speed of propagation implies that they have traveled equal distances from their starting points. When waves meet, they interact and form no precipitate. So the lack of precipitate indicates the computed boundaries of the Voronoi regions.

 Fig. 15

A reaction-diffusion (RD) computer solving a Voronoi problem. (Figure from Tolmachiev and Adamatzky 1996, [Fig. 3](#).)



Shortest Path Searching

A propagating wave technique can be used to find the shortest path through a maze or around obstacles (Steinbock et al. 1995; Agladze et al. 1997; Ito et al. 2004). The maze can be encoded in the chemical substrate, or by using a light mask. A wave is initiated at the start of the maze, and a series of time lapse pictures are taken as the wave propagates at a uniform speed, which provide a series of equidistant locations from the starting point; these are used in a post-processing phase to construct the shortest path (☞ [Fig. 16](#)).

Logic Gates

Propagating waves can be confined to channels (“wires”) and interact at junctions (“gates”) so arranged such that the interactions perform logical operations. See, for example, Motoike and Adamatzky (2004), Sielewiesiuk and Gorecki (2001), and Toth and Showalter (1995). Hence the continuous RD system dynamics can be arranged by careful choice of initial conditions to simulate a digital circuit.

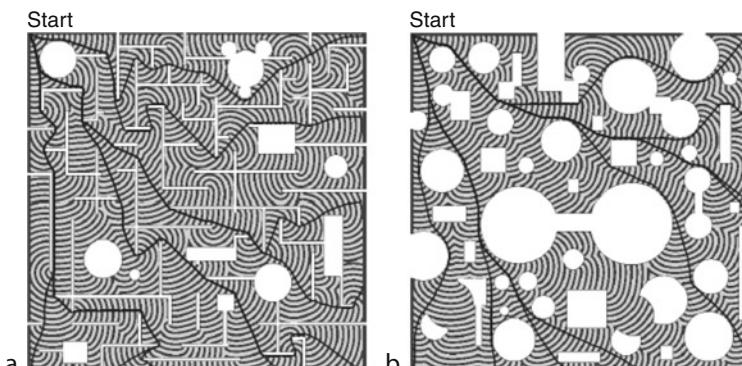
2.3.6 Generic Analog Computers

In general, analog computers gain their efficiency by directly exploiting the physical dynamics of the implementation medium. There is a wide range of problem-specific analog computers, such as the reaction–diffusion computers described above, but there are also general purpose analog computers.

For example, Mills has built implementations of Rubel’s general purpose extended analog computer (Rubel 1993). The computational substrate is simply a conductive sheet (Mills 2008a; Mills et al. 2006), which directly solves differential equations; the system is “programmed” by applying specific potentials and logic functions at particular points in the sheet. Mills has developed a computational metaphor to aid programming by analogy (Mills 2008b), and is currently developing a compiler to enable straightforward solution of given differential equations (Mills 2008, personal communication).

☞ **Fig. 16**

Computer simulation of a shortest path computation in mazes with obstacles. (From Ito et al. 2004.)



2.4 Hybrid Dynamical Systems

So far, all the systems considered have homogeneous dimensions, for example, all \mathcal{S} or all \mathcal{R} . More complicated dynamical systems have heterogeneous dimensions. For example, coupling a classical finite state machine with a continuous system would yield a hybrid system with a dimensionality like $\mathcal{S}^M \times \mathcal{R}^N$.

It is likely that the topology of such a hybrid system would consist of relatively weakly coupled sub-components (► Fig. 11b), which should help in their analysis from a dynamical systems point of view.

2.5 Summary

The classes of autonomous dynamical systems that have been discussed are summarized in ► *Table 3*.

From a computational perspective, one important classification dimension is the implementation: whether the computational dynamics is implemented “naturally,” that is, directly by the physical dynamics of the underlying medium, or whether it is implemented in terms of a virtual machine (VM) itself implemented on that underlying dynamics.

Discrete systems tend to be implemented in terms of VMs. This has the advantage that the computational dynamics is essentially independent of the underlying medium (witness the diversity of systems that implement Boolean logic, for example), and so can be analyzed in isolation. It has the disadvantage of the computational overhead imposed by the VM layer.

One goal of continuous dynamical systems is to provide a computational dynamics closely matched to the physical dynamics, with corresponding gains in efficiency. The downside is that such systems are more likely to be constrained by their physical dynamics, and so are less likely to be Turing-universal computational systems.

3 Open Dynamical Systems

3.1 Openness as Environmental Inputs

The systems described so far are *autonomous* or *closed*. They have an *initial condition* (identifying one state x_0 from the \mathcal{X}^N possible), and then the fixed (non-time-dependent) dynamics

► **Table 3**

Classification of the different kinds of autonomous dynamical systems, in terms of their state space and time evolution

	Discrete: $t \in \mathcal{N}$	Continuous: $t \in \mathcal{R}$
$s \in \mathcal{S}^N$	Finite CAs ; RBNs	
$s \in \mathcal{S}^\infty$	Infinite CAs ; TMs	
$r \in \mathcal{R}$	Iterated maps	
$r \in \mathcal{R}^N$	CMLs	ODEs
$r \in \mathcal{R}^\infty$		PDEs
$x \in \mathcal{S}^M \times \mathcal{R}^N$	Hybrid	

proceeds with no input or interference from the outside world. They move to an attractor, the result of the computation, or they may not discover an attractor, in which case the computation has no result. This is the classical, “ballistic” style of computation exemplified by the Turing machine, or a closed dissipative system relaxing to equilibrium.

Open, or nonautonomous, systems, on the other hand, have dynamics that are governed by parameters that change over time. These parameters are inputs from the environment.

Consider an open dynamical system with N degrees of freedom: its state can be defined by an ND state vector $\mathbf{x}(t) \in \mathcal{X}^N$. The state space is \mathcal{X}^N . Now there is also an input space \mathcal{P} , and an output space \mathcal{Q} . The dynamics f maps the current state and input to the next state and output; $f : \mathcal{X}^N \times \mathcal{P} \rightarrow \mathcal{X}^N \times \mathcal{Q}$.

There is a similarity here to a parameterized *family* of dynamics (⌚ Sect. 2.2.1). But here the parameter p is a function of t , and is considered an *input* to the dynamics, a way of modulating or controlling the dynamics, for example, moving it between periodic and chaotic attractor behaviors.

3.1.1 Timescales

Understanding open systems is significantly more challenging than understanding closed systems, and depends in part on the relationship between the timescale on which the input is changing and the timescale on which the dynamics is acting. Dynamical systems have a “natural” timescale: the time needed to discover the attractor. As Beer (1995) says

- ▶ Because ... the flow is a function of the parameters, in a nonautonomous dynamical system the system state is governed by a flow which is changing in time (perhaps drastically if the parameter values cross bifurcation points in parameter space). Nonautonomous systems are much more difficult to characterize than autonomous ones unless the input has a particularly simple (e.g., periodic) structure. In the nonautonomous case, most of the concepts that we have described above (e.g., attractors, basins of attraction, etc.) apply only on timescales small relative to the timescale of the parameter variations. However, one can sometimes piece together a qualitative understanding of the behaviour of a nonautonomous system from an understanding of its autonomous dynamics at constant inputs and the way in which its input varies in time.

That is, an input changes the dynamics of the system, by changing to a different member of the parameterized family. This new member might have moved attractors, or be on the other side of a bifurcation point with different kinds of attractors. A system immediately after an input will be in the same position in its state space, but the underlying attractor structure of that space may have changed.

So, if the input is changing slowly with respect to the dynamics, the system is able to complete any transient behavior and reach the changed attractor before the input changes the dynamics yet again, even if it has passed through a discontinuous, catastrophic bifurcation point. On these timescales, the system is able to “track” the changing dynamics, and so its behavior can be analyzed piecewise, as a sequence of essentially unchanging systems. Even so, such systems can exhibit *hysteresis*: Restoring a parameter to a previous value may not necessarily restore the system to its corresponding previous state, if this path through parameter space crosses catastrophic bifurcation points.

If the input is changing quickly with respect to the dynamics, then the system is unable to respond to changed dynamics before it has changed again. It will mostly be exhibiting transient behavior.

Most interesting and complex is the case where the input is changing on a timescale similar to that of the dynamics: then the system is influenced by its dynamics, but it may never quite, or only just, reach any attractor before the next change occurs.

The situation can get even more complicated, when the input parameter p is a function of space as well as time, $p(x, t)$. For example, it might be a temperature gradient, or magnetic field gradient, which can also drive the system.

3.1.2 Computation in Terms of Trajectories

Since such open systems need not reach a “halting state” of being on an attractor, the computational perspective is necessarily broader. The computation being performed can be viewed as the *trajectory* the system takes through the changing attractor space: which attractor basins are visited, in which order.

The discussion of timescales implies that, for useful computation, the dynamical timescale of the system should not be significantly slower than the input timescales.

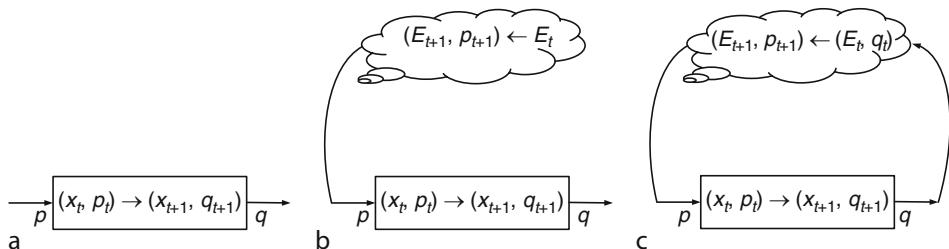
3.1.3 Environmental Constraints

In the simplest open case, it is assumed that the system can be provided with any input, regardless of what it is doing, or has done (Fig. 17a). The input may as well be considered random. This is not particularly interesting, except in the cases where the system can somehow exploit noise, for example, using some kind of informational analog of a ratchet mechanism (e.g., Dasmahapatra et al. (2006)).

This case is formally equivalent to a closed system, as the sequence of arbitrary inputs could conceptually be provided at the start, embedded in an (expanded) state $\mathcal{X}^N \times \text{seq } \mathcal{P}$ as part of the initial condition, along with some pointer to the “current time” value, and the dynamics updated to allow access only to the current value (e.g., see Cooper et al. (2002),

Fig. 17

Inputs. (a) arbitrary input stream; (b) constrained inputs from a structured environment; (c) constrained inputs from an interacting environment in a feedback loop.



where such an approach is taken to embed inputs into the initial state of the model of a formal language). So there is no new computational capability, except as provided by the (potentially, much) larger state space.

Type B: Environmentally Constrained Input Stream

More interesting is the case where the inputs come from an environment that has some rich dynamical structure that the system can couple to and exploit (☞ Fig. 17b). Here, the environment is an autonomous dynamical system, unaffected by any inputs of its own.

Since the environment is autonomous, its sequence of inputs could again conceptually be provided at the start. However, this case is qualitatively different from the previous one. We are now assuming that there is some *structure* in the environment, and hence in the sequence of inputs. This implies that there are regions of the system state space that are never explored, parts of its underlying dynamics that are never exercised. As before when talking of virtual machines (☞ Sect. 2.1.3), the computation is restricted to a subspace: the subspace and its trajectories here correspond to the computation in the context of the structured environment; that is, the inputs provide *information* that the system need not itself compute. And since the environment may be unboundedly large, the sequence of inputs may represent an unboundedly large amount of computation provided to the system.

Type C: Feedback Constrained Input Stream

Most interesting is the case where the environment and system are both open dynamical systems in a rich feedback loop. Then outputs from the system will alter the environment, and affect its subsequent inputs (☞ Fig. 17c). So the actual sequence of inputs cannot even conceptually be provided at the start.

Again, the environment's inputs will be constrained to a region of state space, but here this region is (partly) determined by the system: the environment and systems are coupled, the dynamics of each perturbing the trajectory of the other, in a feedback loop.

Such an environment may well contain other open systems similar to the system being considered. And again, since the environment may be unboundedly large, it may represent an unboundedly large amount of computation provided to the system, but here, the specific computation provided is affected by what the system does, because of the feedback coupling. This opens up the possibility for a system to offload some of its computational burden onto the environment (see ☞ Sect. 3.4.4).

Beer (1995) points out that such a coupled environment and system together form a higher-dimensional autonomous dynamical system, with its own attractors, and, because of its larger state space, that this combined system can “generate a richer range of dynamical behavior than either system could do individually.”

3.2 Open Discrete Space, Discrete Time Dynamical Systems

The autonomous discrete systems have an *initial condition* (identifying one state s_0 from the $|\mathcal{S}|^N$ possible). In the finite case, they always “halt” on an attractor cycle, this halting state (cycle) being the result of the computation; in the infinite case, they either halt on such a cycle, or they may not discover an attractor, in which case the computation has no result.

In the open discrete space, discrete time case, where $\mathcal{X} = \mathcal{S}$, the time evolution of the state, and the output function, are given by $(s_{t+1}, q_{t+1}) = f(s_t, p_t)$.

3.2.1 Modulating the Dynamics

The input can provide a “kick” or perturbation, changing (a few bits of) the state at a particular timestep. This may move the system into a different attractor basin.

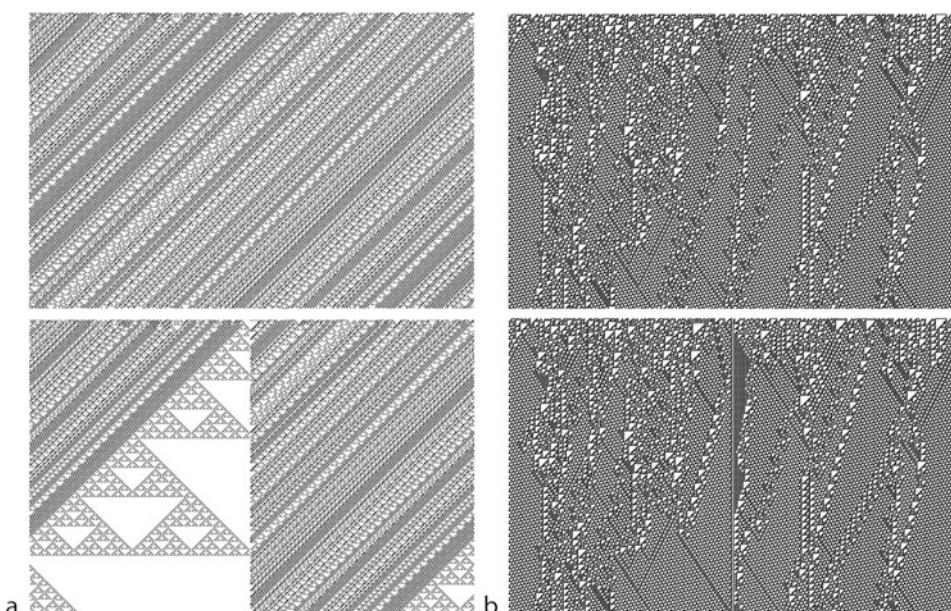
The input might also “clamp” the system into a particular substate (by fixing the value of some bits for many timesteps). This not only perturbs the system at the point where the bits are clamped, but can also change the global dynamics (if the natural dynamics would change the value of the clamped bits, for example), producing new attractors, and changing or removing existing attractors. For example, clamping some bits in a CA can result in “walls” across which information cannot flow, isolating regions, and hence changing the dynamical structure of the system. (See [Fig. 18](#).) Such a simple partitioning is harder to achieve in a more irregularly connected structure such as an RBN.

3.2.2 Perturbing Random Boolean Network State

Kauffman (1990) defines a *minimal perturbation* to the state of an RBN to be flipping the state of a single node at one timestep. Flipping the state of node i at time t is equivalent to changing its update rule at time $t-1$ to be $c_{i,t} = \neg\phi_i(\chi_{i,t-1})$. Such a perturbation leaves the underlying dynamics, and hence the attractor basin structure, the same; it merely moves the current state to a different position in the state space, from where it continues to evolve under the original dynamics: it is a transient perturbation to the state.

■ Fig. 18

Dependence of CA dynamics on a “clamped” bit. The upper plot shows ordinary periodic boundary conditions; the lower plot shows the same initial conditions, but with the central bit “clamped” to 0. (a) ECA rule 26; (b) ECA rule 110.



Kauffman (1990) describes the *stability* of RBN attractors to minimal perturbations: If the system is on an attractor and suffers a minimal perturbation, does it return to the same attractor, or move to a different one? Is the system *homeostatic*? (Homeostasis is the tendency to maintain a constant state, and to restore its state if perturbed.)

Kauffman (1993) describes the *reachability* of other attractors after a minimal perturbation: If the system moves to a different attractor, is it likely to move to any other attractor, or just a subset of them? If the current attractor is considered the analog of “cell type,” how many other types can it *differentiate* into under minimal perturbation?

Kauffman’s results are summarized in [Table 4](#), which picks out the $k=2$ networks as having “interesting” (non-chaotic) dynamics (a small number of attractors, with small cycle lengths) and interesting behavior under minimal perturbation (high stability so a perturbation usually has no effect; low reachability so when a perturbation moves the system to another attractor, it moves it to one of only a small subset of possible attractors).

3.2.3 Perturbing Random Boolean Network Connectivity

Kauffman (1990) also defines a *structural perturbation* to an RBN as being a permanent mutation in the connectivity or in the Boolean function. So a structural perturbation at time t_0 could change the update rule of cell i at all time $t > t_0$ to be $\phi'_i(\chi_{i,t})$ or change the neighborhood of cell i at all time $t > t_0$ to be $\chi'_{i,t}$. Since the dynamics is defined by all the ϕ_i and χ_i , such a perturbation changes the underlying dynamics, and hence the attractor basin structure: It is a permanent perturbation to the dynamics, yielding a new RBN.

Such a perturbation could have several consequences: a state previously on an attractor cycle might become a transient state; a state previously on a cycle might move to a cycle of different length, comprising different states; a state might move from an attractor with a small basin of attraction to one with a large basin; a state might move from a stable (homeostatic) attractor to an unstable attractor; and so on.

Kauffman (1993) relates structural perturbation to the *mutation* of a cell; if there is only a small change to the dynamics, this represents mutation to a “similar” kind of cell.

3.3 Open Continuous Space, Discrete Time Dynamical Systems

In the open continuous space, discrete time case, where $\mathfrak{X} = \mathcal{R}$, the time evolution of the state, and the output function, are given by $(\mathbf{r}_{t+1}, q_{t+1}) = f(\mathbf{r}_t, p_t)$.

Table 4

Dynamics of RBNs for different k (adapted from Kauffman (1993, Table 5.1))

k	Stability	Reachability
1	Low	High
2	High	Low
> 5	Low	High

3.3.1 Open Coupled Map Lattices, and Chaos Computing

Sinha and Ditto (1998, 1999) investigate the computational properties of coupled chaotic logistic maps (ϕ_0 =the logistic map with $\lambda=4$, see Sect. 2.2.3) with open boundaries. The coupling function ϕ_c is a threshold function: $\phi_c=\text{if } r < \theta \text{ then } 0 \text{ else } r - \theta$, and is unidirectional. This unidirectional relaxation propagates along the lattice until all elements have a value below threshold (all the $r_i \leq \theta$), at which point the next timestep iteration of the logistic dynamics occurs (so the timescale of the relaxation is much shorter than the timescale of the logistic dynamics). The boundary of the lattice is open, so the final element can relax below threshold by removing its excess value from the system. Depending on the particular threshold value, and the current state of the system, this process can result in nonlinear avalanches of relaxation along the lattice. Transients are short (typically one dynamical timestep), and the system displays a variety of attractor dynamics, stable to small amounts of noise, and determined by the input threshold parameter.

Sinha and Ditto (1998, 1999) discuss how to use this system to perform computation. A single lattice element with two inputs (either external, or from the coupled output of other lattice elements) and appropriate input value of its threshold can implement a universal NOR logic gate in a single iteration of the logistic dynamics, where the amount of output encodes the result. Hence, networks of such elements can be coupled together to implement more complicated logic circuits. Sinha and Ditto (1999) note that it is the chaotic properties in general, not the logistic map in particular, that give these coupled systems their computational abilities, and suggest a possible implementation based on nonlinear lasers forming a coupled chaotic Lorenz system. They dub their approach *chaos computing*.

Additionally, Sinha and Ditto (1998, 1999) discuss how to implement other functions, such as addition, multiplication, and least common multiple, by suitable choice of numerical encoding, coupling, and input thresholds. These choices program the underlying chaotic dynamics to perform computation directly, rather than emulating a virtual machine of compositions of logic gates (Sinha and Ditto 1999):

- ▶ dynamics can perform computation not just by emulating logic gates or simple arithmetic operations, but by performing more sophisticated operations through self-organization rather than composites of simpler operations.

Despite this observation, their subsequent work (Ditto et al. 2008; Sinha and Ditto 2006) concentrates on using more complicated (but realizable) chaotic dynamics to implement robust logic circuits, that can be readily reconfigured merely by altering thresholds. They emphasize the openness of their approach (Sinha and Ditto 2006):

- ▶ it can yield a gate architecture that can dynamically switch between different gates, without rewiring the circuit. Such configuration changes can be implemented either by a predetermined schedule or by the outcome of computation. Therefore, the flexibility of obtaining different logic operations using varying thresholds on the same physical element may lead to new dynamic architecture concepts

3.4 Open Continuous Space, Continuous Time Dynamical Systems

In the open continuous space, continuous time case, $\mathfrak{X} = \mathfrak{R}$, the time evolution of the state, and the sequence of outputs, are given by $(\dot{\mathbf{r}}, q(t)) = f(\mathbf{r}, p(t))$.

Note again the similarity to a parameterized *family* of dynamics (☞ Sect. 2.2.1), with the (input) parameter p being a function of t . Examples of such input parameters might include the temperature T of the system, or the value of an externally imposed magnetic field \mathbf{B} permeating the system.

3.4.1 Ott, Grebogi, Yorke (OGY) Control Laws

There are unstable periodic orbits in strange attractors. Small perturbations in the control parameter can be used to keep the system in one of these; the required perturbations are calculated using the OGY control laws (Ott et al. 1990). (It is not necessary to know the underlying dynamics to do this; application of the control laws involves calculating the required parameter from observations of the system.) There can be long transient behavior before the system gets “close” to the desired periodic orbit, and noise can result in bursts of chaotic behavior. The approach can also be used to switch between different periodic orbits with different characteristics (with some transient chaotic behavior).

Ott et al. (1990) note that a chaotic system is potentially more flexible, because this approach can be used to hold it in a variety of different periodic orbits, whereas this range of behaviors would require a range of separate systems with non-chaotic dynamics. Sinha and Ditto (☞ Sect. 3.3.1) make similar observations about their coupled nonlinear maps, although there they claim a lower computational burden (the thresholds can be simply calculated, and stored in a lookup table) and shorter (essentially zero) transient behaviors.

3.4.2 Liquid Crystal Systems

Liquid crystals are a form of matter that lies on the boundary between solids and liquids (sometimes called “the fourth phase of matter”). A liquid crystal has both complex dynamics (the molecules can flow and rotate) and complex structure (the molecules are ordered on length scales much bigger than their individual sizes).

Such materials can perform computation. Harding and Miller (Harding and Miller 2004, 2005; Harding et al. 2006) have demonstrated that a liquid crystal chip can be programmed to act as a tone discriminator (a simple arbitrary input system, where the inputs are tones of two different frequencies that are to be discriminated) and as a robot controller (a constrained feedback system, where the inputs from the environment depend on the robot’s position, and the robot’s outputs change its position).

It is currently unclear how the liquid crystal performs its computations: In the referenced cases, the material was programmed using an evolutionary algorithm. It would be interesting to analyze these results from a dynamical systems perspective.

3.4.3 NMR Logic Gates

Nuclear magnetic resonance (NMR) uses radio frequency pulses to manipulate nuclear spins in a magnetic field. Depending on the particular values chosen from a rich potential set of parameters (frequency, phase, duration, delay, and more), pulses can be combined in various ways to produce different outputs. These pulses and the outputs can be interpreted as encoding binary values. Under these interpretations, the system can be used to implement a single

universal logic gate, a circuit of these gates combined in parallel and in sequence that implement other logic gates, and a half-adder circuit (Roselló-Merino et al. 2010).

Under these interpretations, this is a Type A open system: Any arbitrary set of Boolean inputs is permitted, and the system implements the corresponding logic gate computation on these inputs. Under the wider view of the full parameter space, this is a Type B open system, with the inputs being restricted by the environment (the experimenter) to values that can be interpreted as binary bits. Hence, we see how setup corresponds to a logic gate virtual machine (► Sect. 2.1.3) implemented on the underlying dynamics of the nuclear spin system.

3.4.4 Embodiment

Beer (1995) takes a dynamical systems approach to adaptive agents in a changing environment. For example, a robot agent adapts its walking behavior depending on the kind of terrain it is moving across. So an agent receives sensory input from a structured environment (it sees its current surroundings), which affects its internal dynamics (its state changes), which affects its outputs (its leg movements), which in turn affects the environment (at a very minimum, the agent moves to a new location in the environment, and hence sees new surroundings). So “a significant fraction of behavior must be seen as emerging from the ongoing interaction between an agent and its environment.” Beer is talking here of Type C open dynamical systems (► Sect. 3.1.3), coupled to their environment in a feedback loop.

Beer’s aim is to use the language and concepts of dynamical systems theory to develop a theoretical framework for designing adaptive agents. Certain computational tasks, such as planning, can be greatly simplified by exploiting input from the structured environment (e.g., the agent seeing where it is). This relates to Brooks’ design principle: “use the world as its own model” (Brooks 1991).

Beer emphasizes that, because of the coupling with the environment, “an agent’s behavior properly resides only in the dynamics of the coupled system,” and hence cannot be understood or analyzed in isolation. Indeed, an agent is adapted to some environments and not others. Beer analyzes this idea of adaptive fit, and determines that it requires that an agent maintains its trajectory within a certain volume of its state space (which volume may change with time) under perturbations from the environment. This is related to an *autopoietic* (Maturana and Varela 1980) (self-creating and self-maintaining) view of adaptive fitness, and helps define what is needed for *homeostasis*.

This area also links with a whole burgeoning subdiscipline of using dynamical systems theory to model the brain and its cognitive processes. However, that is deemed to be outside the scope of this particular discussion. The interested reader is referred to Kelso (1995).

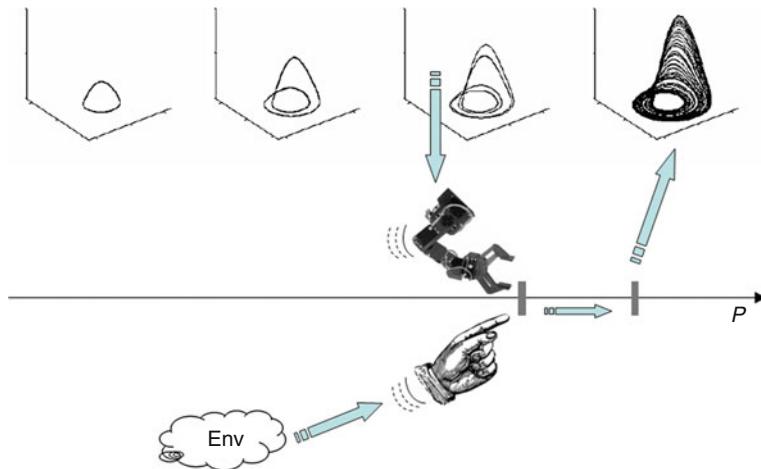
3.5 External Control Versus Internal Dynamics

Here we have considered the parameter $p(t)$ to be an externally provided input, moving a dynamical system between members of its parameterized family. Alternatively, we could have a case where the parameter p was another degree of freedom, or dimension, of the dynamical system, where the dynamics of the system itself affects the value of p (► Fig. 19).

This potentially gives a model of a *self-organizing* system. For example, Melby et al. (2000) describe a logistic map (► Sect. 2.2.2) where the value of the dynamical variable r_t is fed through a low-pass filter to (slowly) affect the parameter λ : The system self-adjusts to values

Fig. 19

Two sources of parameter variation: external control versus internal dynamics.



of λ at the edge of chaos, and does so even when subject to an external force attempting to drive it back into the chaotic region (Melby et al. 2002). The situation is a little more complicated in the presence of noise: The system still self-adjusts to suppress chaos, but a power-law distribution of chaotic outbreaks occurs (Melby et al. 2005).

An alternative view is of a hierarchy of coupled dynamical systems, where the outputs at one level couple to the control parameter(s) at another level. Abraham and Shaw (Abraham 1987; Abraham and Shaw 1987) explore this idea in more detail.

4 Constructive Dynamical Systems

So far, we have considered predetermined fixed state spaces of a given dimensionality. However, this is not the case even for classical computational systems. Their data structures define their abstract state space: every time new memory is allocated in the course of the computation, the state space grows in dimension; memory deallocation shrinks the state space. A dynamical systems perspective on computation needs to consider cases where the state space itself dynamically changes dimensionality, $\mathcal{X}^{N(t)}$.

In such cases, the dynamics *constructs* the state space, and the state space constrains the dynamics. Closed autonomous systems can exhibit such growth, classical computation being one such example. In open systems, this process can be thought of as the flow of information, matter, energy being recruited to construct new dimensions of the system: the system inputs “food” (constructs dimensions), and excretes “waste” (collapses dimensions). This process of modifying its own dimensionality affects the possible dynamics (recall, e.g., that strange attractors require at least a 3D continuous space [\(Sect. 2.3.1\)](#)).

This is an aspect of dynamical systems theory that has received very little attention: the current theory assumes a predefined, fixed state space. Because of the paucity of theory in this area, this section is intended to illustrate the kinds of processes that need to be incorporated in such a theory.

Metadynamics

One approach to incorporating growth might be to model the system with an infinite number of dimensions, confining the dynamics to the finite-dimensional subspace corresponding to the current state space. As new dimensions are needed, the dynamics expands into the preexisting dimensions. However, this approach appears to be a mathematical “trick” that hides the essential properties of the underlying system: precisely what confines the dynamics to a subspace, what causes it to grow into new dimensions, and what determines the topology (the way the information flows between dimensions)? (This approach also has a small technical issue: how to distinguish a system that is not using a particular dimension, from one that is using it, but is currently confined to the zero value of that dimension: how to distinguish “absence” from the “presence of nothing.” This can in turn be solved by a further mathematical trick, of introducing some special value, \perp , to make this distinction. But the tricks are piling up.)

A different approach might be to have these new dimensions start in some small “curled up” form; a dimension could “uncurl” or “unfold” sufficiently to support the current state value along that dimension, and curl back up when no longer needed. Alternatively, the dimensions might be fractal in nature, with the fractal dimension of the state space increasing, gradually “fattening up” the new dimension.

This suggests that the state space itself might have a *metadynamics*. The metadynamics (the high-level dynamics of the state space) can be studied in isolation from the low-level dynamics (trajectories of the system through the state space).

Another form of metadynamics is to allow the topology (see [Sect. 2.2.4](#)) to change dynamically, even if the dimensionality is constant. Such “network dynamics” allows the neighborhood function ([Sect. 2.1.5](#)) to be a function of time, $v(t)$.

Baguelin et al. (Baguelin et al. 2003; Moulay and Baguelin 2005) consider a metadynamical transition rule, which changes the state space (and the corresponding dynamical state transition rule) on a slower timescale than the system’s dynamics: the system spends most of its time evolving under its dynamics, punctuated by relatively few metadynamical state space changes. The overall system dynamics is defined by “concatenating” the various lower-level dynamical behaviors at the metadynamical change points. Baguelin et al. (Baguelin et al. 2003; Moulay and Baguelin 2005) apply this approach to interacting populations of bacteria and phages (the dynamics) that are also evolving (a slower timescale metadynamics).

Whatever approach is used, it needs to be able to cope with different types of dimensions, as the dynamics results in new types of state variables being constructed (e.g., in evolution, new species can be considered to occupy new types of dimensions). If such types cannot be statically predetermined, but computed only by the unfolding dynamics, it is harder to see how to incorporate this into an analysis based on a known, preexisting set of state spaces.

Timescales

Important timescales in a constructive dynamical system are the speed at which the state space grows compared to dynamics on that space. Where growth timescales are much less than the dynamical timescales (a change in the number of dimensions tends to happen after the system has relaxed to an attractor), piecewise approximations may be made in a way similar to slow open systems ([Sect. 3.1.1](#)), as in the metadynamics approach outlined above. But in faster-changing systems (e.g., L-Systems [Sect. 4.1.2](#), where the dimensionality can change every iteration timestep) this is not possible, and new analysis techniques must be sought.

Computation in Terms of Construction

In an autonomous (closed) constructive system, the result of the computation (if the computation halts) could still be considered as the attractor (final macrostate), and additionally include the structure (dimensionality and topology) of the final grown state space.

In an open, non-halting constructive system, the result of the computation could still be considered as the trajectory through a growing state space, and additionally include the structure (dimensionality and topology) of the state space along that trajectory.

4.1 Constructive Discrete Dynamical Systems

4.1.1 Classical Computation

Turing Machine

A Turing machine can be thought of as a growing system: The tape is of finite but unbounded length, and can be considered to grow (lazily) whenever a new tape position is required. It can also be considered to shrink if the last symbol on the tape is erased.

So the question of growth can be linked to undecidability: Whether the dimensionality of the state space stays bounded (if the computation halts or loops), or grows without limit, is formally undecidable.

Object Orientation

Object orientation systems are particularly constructive in this sense: new objects are created, increasing the dimensionality of the state space; when they go out of scope, the state space shrinks again.

Such systems are usually designed on the software engineering principle of ensuring “weak coupling” by small interfaces, which keeps components separate and modular by restricting the information flow. That is, the topology of the dynamics is deliberately restricted.

Contrast the connectivity of RBNs, which have a “small-world” topology, and require a large number of connections to be cut to partition the graph (❷ Fig. 11). This enables rapid information flow, potentially enabling different classes of dynamics. The consequences of these design decisions are rarely addressed from a dynamical systems perspective.

Closed or Open

These classical computational systems can be closed (e.g., the TM model), or open (interactive systems). The open systems are usually analyzed as Type A (❷ Sect. 3.1.3), where the external environment can potentially provide any input.

4.1.2 Rewriting Systems

Imagine trying to “grow” an RBN where, if it got to a certain state, a new node would appear, or an existing node would disappear (with consequent rewiring). The appearance or disappearance of a node would change the dimensionality of the state space (to $N \pm 1$), and also the underlying attractor structure, and hence change the computation being performed.

Similarly, imagine trying to “grow” a 1D CA. One could add a new cell, growing the dimensionality of the state space from \mathcal{S}^N to \mathcal{S}^{N+1} . Where should the cell be added in the line? The CA dynamics is symmetric under renumbering shifts, and so it should be possible to add the cell anywhere with equal ease. But unless it is added at special position $i=N$, subsequent cells have to be renumbered. This illustrates the tyranny of a global coordinate system: “The introduction of numbers as coordinates . . . is an act of violence” (Weyl 1949). Instead we would like a coordinate-free, or purely topological, approach to growth.

L-Systems

Lindenmayer’s L-Systems (Prusinkiewicz and Lindenmayer 1990) are such a coordinate-free approach to growth: They are generative grammars that define how symbols in a string are rewritten (“grow”) depending on their local context (their topology, their neighborhood symbols), not on any global coordinate system.

The simplest D0L-Systems (deterministic, context free) can be described in the notation of this chapter as follows. The state s is a string of elements (symbols) drawn from the finite alphabet \mathcal{S} , so $s \in \mathcal{S}^*$. The local dynamics ϕ (rewriting rules, or productions) is given by $\phi : \mathcal{S} \rightarrow \mathcal{S}^*$. At each timestep, each element in the state string is updated (rewritten) in parallel: $s_{i,t+1} = \phi(s_{i,t})$, and the resulting substrings concatenated to form the new state string. (Note here that indices are used merely to identify the string elements in order to help define the local dynamics. They are not a fundamental part of the string data type, which supports operations such as insertion and deletion, unlike the fundamentally indexed array data type.)

In conventional use, one starts from some initial state (axiom) $s_0 \in \mathcal{S}^+$, and follows the dynamics for several iterations to reveal the “grown” structure. However, the dynamics does not necessarily result in a different number of dimensions (a changed length string). For example, Danks et al. (2007, 2008) use L-Systems to model protein folding: there the rewriting models the change in conformation of the (rendered) string, and does not result in any change in the length of the string itself.

In Parametric L-Systems (Prusinkiewicz and Lindenmayer 1990), each symbol has associated parameters, and so the state space is $(\mathcal{S} \times \mathcal{P})^*$.

In the context-free case, there is no information flow between the dimensions: Each individual dimension grows into new dimensions in a manner based on its own substate alone. Context-sensitive L-Systems (Prusinkiewicz and Lindenmayer 1990) couple the dimensions together, and define a tree-structured topology on the growing space. In standard L-Systems, the topology is explicitly encoded in the state string, by use of reserved symbols to define branching.

L-Systems also include a *rendering* step, where the symbols in the string are interpreted as commands in some language, typically a turtle graphics language, to construct a representation of the string (Prusinkiewicz and Lindenmayer 1990). In standard L-Systems this rendering produces a geometrical structure that follows the tree-structured topology of the state space, typically to produce rendered images of plants. However, other renderings are possible, and can produce non-geometric outputs and outputs with different topologies. For example, Hornby and Pollack (2001) interpret the symbols as instructions for generating topological descriptions of neural networks.

These simplest L-Systems are autonomous dynamical systems, and any computation performed in the rendering step is separate from the dynamics of the growth process.

Parametric L-Systems, where the values of (some) parameters are inputs, are Type A or Type B open dynamical systems (● Sect. 3.1.3). In typical L-System applications, they are used

as Type B, because the input is assumed to be coming from some constrained environmental source (e.g., day length, or other weather conditions).

Environmentally sensitive L-Systems (Prusinkiewicz et al. 1994) are also Type B open dynamical systems (⌚ Sect. 3.1.3) of a different form. The dynamics again takes into account external inputs (e.g., the amount and direction of sunlight, or collision with static obstacles, when modeling plant growth), but here the particular rendering process is coupled into the dynamics, as the input is a function of the specific geometry of the rendered result (rendered leaves shading other leaves, etc.), not just the topology of the abstract state.

Open L-Systems (Měch and Prusinkiewicz 1996) are Type C open dynamical systems (⌚ Sect. 3.1.3). They include a feedback process, so the environment is affected by the L-System (e.g., the environment might be the water supply, and the L-System the growing roots that are affecting the amount of water). Again, this approach couples the particular rendering process into the dynamics.

P-Systems

Păun's P-systems (Păun 2000), and other membrane computing formalisms, are another form of rewriting systems with an underlying tree-structured topology. Here the tree structure models not the branching of a plant, but the nesting of membranes, or containers. Membranes contain symbols, and rules that act on the symbols, and on membranes (creating, connecting, and dissolving them). The computational model successively applies the rules until no more are applicable; the computation has halted, and the symbols in a given membrane are the result of the computation. Much theoretical effort is expended on determining the computational power of different classes of membrane systems: Turing-complete systems exist. So this prototypical system has a rather classical view of computation, but there is a plethora of variants with different computational models.

Topological Approach

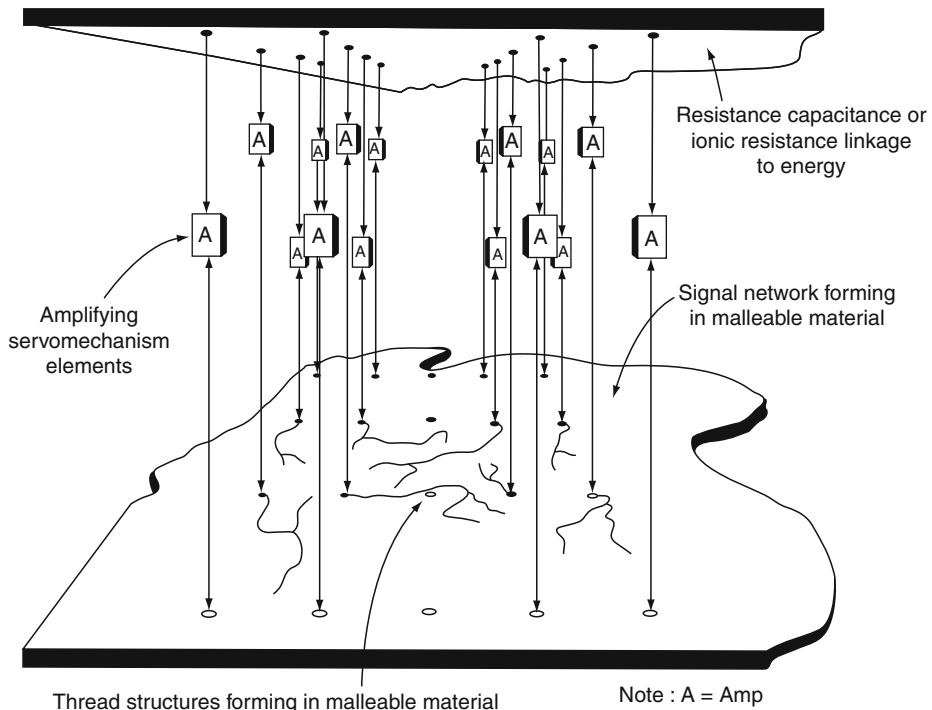
Michel and coworkers (Giavitto and Michel 2002) contrast the “absolute space” coordinate-based philosophy of Newton, with the relative space, coordinate-free philosophy of Leibniz, from a computational perspective. Their approach to (discrete) “dynamical systems with a dynamical structure (DS)²” is topological, focussing on the neighborhood relation and induced subcollections (essentially a set of related elements in a neighborhood). The neighborhood relation can specify structures ranging from the regular spatial neighborhood of a CA to the logical neighborhood of a data structure such as a list, graph, or array (Giavitto et al. 2005). The approach then specifies global dynamics in terms of parallel application of local transformations that rewrite subcollections (Spicher et al. 2004; Giavitto et al. 2005). This approach subsumes dynamic rewriting systems such as L-Systems and P-systems, as well as more static-structured CA-like rewritings (Giavitto et al. 2005).

4.2 Constructive Continuous Dynamical Systems

Gordon Pask was a member of the Cybernetics movement. In the 1950s, he built a system that could “grow” into an “ear” (a sensor capable of detecting sound waves, or magnetic fields) by adaptively laying down suitable conducting filaments (⌚ Fig. 20) in response to environmental inputs and a structured “reward” input.

Fig. 20

Pask's schematic indicating the relationship between the electrode array and the ferrous sulfate medium (adapted from Pask (1960), as cited by Cariani (1993, Fig. 1)).



Cariani (1993) provides an excellent review of this work, and some implication in terms of “organizationally closed” (i.e., able to construct their own input filters), “informationally open” systems. In an open, adaptive, self-constructing system

- ▶ the dimensionality of the signal space can increase over time as new informational channels evolve. Hill-climbing is thus accomplished not only by following gradients upwards, but also by changing the dimensionality of the problem landscape when one can go no further using those dimensions already available.

4.3 Constructive Hybrid Dynamical Systems

Winfree and coworkers (Winfree 2003; Rothemund et al. 2004) are implementing “algorithmic self assembly.” They use carefully designed DNA fragments to implement nanoscale “sticky tiles,” which self-assemble into crystalline structures. The self-assembly can be interpreted as a computation: Which computation is performed depends on the tiles’ design that determines which tiles they will stick to. Their running example is the implementation of a rule 90 ECA (which from a single “on” state grows a Sierpinski gasket). Other tile designs can implement other CAs, or even a representation of the tape of a TM.

The model of this system is a hybrid system. There is an abstract tile assembly model that captures the discrete dynamics of tiles (the particular CA or TM behavior). This is augmented with a kinetic tile assembly model, which captures continuous properties that account for errors in the assembly process.

They speculate that a similar process could be used to construct molecular electronic circuits and other nanoscale devices. That is, design a constructive dynamics to grow a structure that then has an autonomous computational dynamics.

5 Discussion and Conclusions

Classical computational systems can be analyzed in terms of autonomous discrete dynamical systems. The dynamical systems approach can be extended to continuous systems, to give a computational perspective on embodied computational devices. It can be extended again, to include open dynamical systems coupled to a structured dynamical environment, and again to include constructive, or growing systems. The dynamical systems theory is, however, less well developed in these latter cases.

In addition to the straightforward computational perspective, this dynamical systems viewpoint gives insight into important properties of complex computational systems: robustness and emergence.

5.1 Robustness

Homeostasis

Robustness is the ability to maintain function in the presence of perturbations, stresses, or errors. The system exhibits *homeostasis*. Classical computation, on the other hand, is notoriously *fragile*: A single bit change can completely alter the behavior of the system.

From a dynamical systems computational point of view, robustness is the ability to find the attractor even when perturbed from the current trajectory. A system will find the attractor from anywhere within its basin of attraction: If a perturbation nevertheless maintains the basin, the system will be robust to that perturbation.

Here we see the source of classical computation's fragility: extremely small basins of attraction, such that any perturbation is likely to move the system to a different basin, hence to a different attractor, resulting in a different computation.

Compare this situation to the dynamical structure of $k=2$ RBNs (☞ Sect. 2.1.6). There a perturbation is likely to leave the system in the same basin. “Small attractors located inside a volume of states constituting their basins of attraction are the natural image of stable systems exhibiting homeostasis.” (Kauffman 1993, p. 467) So robustness requires large basins of attraction, which implies relatively few macrostates compared to the number of microstates.

Continuous systems have greater potential for robustness. They have a notion of locality: A small perturbation moves the system a small distance in its state space, and is therefore likely to be in the same basin. (Although it is true that as well as fractal strange attractors, there can be fractal-structured basins; in such cases, a small perturbation might end up in one of many other basins.) Alternatively, the small perturbation could be a small change to a parameter. Provided the parameter does not cross a bifurcation point, this will result in only a small change to the dynamics.

Kitano (2004) explicitly casts a discussion of various kinds of biological robustness in terms of dynamical systems attractors.

Precision

Robustness comes at the price of precision, however. Many microstates per macrostate means a loss of precision: we no longer distinguish these microstates. Also, real-valued output from a continuous system cannot be measured with arbitrary precision. Similar to the case when considering the timescales, the precision of the computation and output should be matched to the precision of the environment and inputs.

Homeorhesis

Waddington (1957) notes that homeostasis is too restrictive a term when considering growing systems, since they do not have a “steady” state to which to return: They are embarked on some trajectory through their ever-changing state space. He introduces the term *homeorhesis* (Waddington 1957, p. 32], meaning “constant flow,” in that systems actually maintain a constant trajectory through developmental space. He deprecates the alternative term, *canalization*, because it may (Waddington 1957, p. 43–44]

- ▶ suggest too concrete an image to be suitable as a name for the abstract quality to which it refers; but this seems a less important failing than those involved in the alternative term homeostasis.

Note that this implies that the developmental trajectory is in some sense an attractor of the growth process: A perturbation away from the growth trajectory is attracted back to that *trajectory*, not merely to some final attractor state. In biological terms, this is robust morphodynamics.

5.2 Emergence

The attractors and their basins, and the bifurcation points of parameterized systems, are *emergent properties* of their dynamics (Stepney et al. 2006).

Although the microstates are changing on an attractor cycle, this can lead to a stable macrostate if observed on a timescale longer than the attractor period. In the dynamical systems view, everything is process (motion on an attractor), but when it is viewed on a suitable timescale, it can behave like a thing.

- ▶ An attractor functions as a symbol when it is viewed . . . by a *slow observer*. If the dynamic along the attractor is too fast to be recorded by the slow-reading observer, he then may recognize the attractor only by its averaged attributes . . . , but fail to recognize the trajectory along the attractor as a deterministic system. (Abraham 1987)

The slow observer does not see the intricate dynamics on the attractor, just some averaged behavior, and this dynamics becomes an atomic component in its own right. What is lost is the microstructure; what remains is a stable pattern that becomes an entity in its own right. This connects directly with the concepts of relative timescales, where there is a sufficient difference between a fast timescale (of the underlying dynamics of the system), and a slower timescale (of the inputs perturbing the system (Sect. 3.1.1; of the metadynamic timescales of state space change (Sect. 4; of the slow observer of these processes).

These new high-level emergent entities can have their own state space and (meta)dynamics, particularly when coupled to other dynamical systems such as the environment, or in some

hierarchical structure. So their (higher level) dynamics will exhibit attractors; motion on these attractors will appear as emergent entities to (even slower) observers; and so on.

5.3 The Future

These discussions imply that we need a dynamical systems theory of open, constructive computational systems, where new dimensions of state space, and new types of dimensions, are constructed by the computation as it progresses. This will provide one of the tools we need to enable us to study, understand, design, and reason about robust emergent computational systems, covering the range from classically discrete, to nonclassical embodied systems.

Acknowledgments

My thanks to Ed Powley for providing [Figs. 1](#), [3](#), [4](#), and [18](#), and for drawing attention to the work on exact correspondences between CAs and PDEs in [Sect. 2.3.4](#). My thanks also to Andy Adamatzky, Leo Caves, Ed Clark, Simon Hickinbotham, Mic Lones, Adam Nellis, Ed Powley, and Jon Timmis for comments on a previous draft.

References

- Abraham RH (1987) Dynamics and self-organization. In: Yates FE (ed) *Self-organizing systems: the emergence of order*. Plenum, New York, pp 599–613
- Abraham RH, Shaw CD (1987) Dynamics: a visual introduction. In: Yates FE (ed) *Self-organizing systems: the emergence of order*. Plenum, New York, pp 543–597
- Adamatzky A (ed) (2002) Collision-based computing. Springer, London
- Adamatzky A, Arena P, Basile A, Carmona-Galan R, Costello B, Fortuna L, Frasca M, Rodriguez-Vazquez A (2004) Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors. *IEEE Trans Circuits Syst* 51(5):926–938
- Adamatzky A, Costello BDL, Asai T (2005) Reaction-diffusion computers. Elsevier, Boston, MA
- Agladze K, Magome N, Aliev R, Yamaguchi T, Yoshikawa K (1997) Finding the optimal path with the aid of chemical wave. *Physica D* 106:247–254
- Andersson C, Nordahl MG (1998) Evolving coupled map lattices for computation. In: EuroGP'98, Paris, April 1998. LNCS, vol 1391. Springer, New York, pp 151–162
- Baguelin M, LeFèvre J, Richard JP (2003) A formalism for models with a metadynamically varying structure. In: Proceedings of the European Control Conference, Cambridge, UK
- Beer RD (1995) A dynamical systems perspective on agent-environment interaction. *Artif Intell* 72:173–215
- Berlekamp ER, Conway JH, Guy RK (1982) *Winning ways for your mathematical plays*, vol 2. Academic, New York
- Bossmoai T, Sibley-Punnett L, Cranny T (2000) Basins of attraction and the density classification problem for cellular automata. In: Virtual Worlds 2000, Paris, July 2000. LNAI, vol 1834. Springer, Heidelberg, pp 245–255
- Brooks RA (1991) Intelligence without representation. *Artif Intell* 47:139–159
- Cariani P (1993) To evolve an ear: epistemological implications of Gordon Pask's electrochemical devices. *Syst Res* 10(3):19–33
- Cook M (2004) Universality in elementary cellular automata. *Complex Syst* 15(1):1–40
- Cooper D, Stepney S, Woodcock J (2002) Derivation of Z refinement proof rules: forwards and backwards rules incorporating input/output refinement. Tech. Rep. YCS-2002-347, Department of Computer Science, University of York
- Crutchfield JP (1994) The calculi of emergence. *Physica D* 75:1154
- Crutchfield JP, Kaneko K (1987) Phenomenology of spatio-temporal chaos. In: Bin-Lin H (ed) *Direction in Chaos*, World Scientific, Singapore, pp 272–353
- Culik K II, Yu S (1988) Undecidability of CA classification schemes. *Complex Syst* 2(2):177–190
- Danks G, Stepney S, Caves L (2007) Folding protein-like structures with open L-systems. In: ECAL 2007,

- Lisbon, Portugal, September 2007. LNAI, vol 4648. Springer, Heidelberg, pp 1100–1109
- Danks G, Stepney S, Caves L (2008) Protein folding with stochastic L-systems. In: ALife XI, Winchester, UK, MIT Press, Boston, MA, pp 150–157
- Dasmahapatra S, Werner J, Zauner KP (2006) Noise as a computational resource. *Int J Unconventional Comput* 2(4):305–319
- Ditto WL, Murali K, Sinha S (2008) Chaos computing: ideas and implementations. *Phil Trans R Soc A* 366:653–664
- Drossel B (2008) Random Boolean Networks. In: Schuster HG (ed) Reviews of nonlinear dynamics and complexity, vol 1, Wiley, Weinheim, arXiv:0706.3351v2 [cond-mat.stat-mech]
- Durand B, Formenti E, Varouchas G (2003) On undecidability of equicontinuity classification for cellular automata. In: Morvan M, Rémy E (eds) Discrete models for complex systems, DMCS'03, Lyon, France, June 2003. DMTCS, vol AB, pp 117–128
- Gács P, Kurdyumov GL, Levin LA (1978) One dimensional uniform arrays that wash out finite islands. *Probl Peredachi Inf* 12:92–98
- Giavitto JL, Michel O (2002) Data structure as topological spaces. In: Unconventional models of computation, Kobe, Japan, October 2002. LNCS, vol 2509. Springer, Heidelberg, pp 137–150
- Giavitto JL, Michel O, Cohen J, Spicher A (2005) Computation in space and space in computations. In: UPP 2004, France, September 2004. LNCS, vol 3566. Springer, Berlin, pp 137–152
- Hammel SM, Yorke JA, Grebogi C (1988) Numerical orbits of chaotic processes represent true orbits. *Bull Am Math Soc* 19(2):465–469
- Hao BL, Zheng WM (1998) Applied symbolic dynamics and chaos. World Scientific, Singapore
- Harding SL, Miller JF (2004) A tone discriminator in liquid crystal. In: CEC 2004, Portland, Oregon. IEEE Press, pp 1800–1807
- Harding SL, Miller JF (2005) Evolution in materio: A real-time robot controller in liquid crystal. In: Proc. NASA/DoD Conference on Evolvable Hardware, Washington, DC. IEEE Press, pp 229–238
- Harding SL, Miller JF, Rietman EA (2006) Evolution in materio: exploiting the physics of materials for computation. arXiv:cond-mat/0611462
- Hayes W, Jackson KR (2005) A survey of shadowing methods for numerical solutions of ordinary differential equations. *Appl Numer Math* 53:299–321
- Holden AV, Tucker JV, Zhang H, Poole MJ (1992) Coupled map lattices as computational systems. *Chaos* 2 (3):367–376
- Hornby GS, Pollack JB (2001) Body-brain coevolution using L-systems as a generative encoding. In: GECCO 2001, Morgan Kaufmann, San Francisco, CA, pp 868–875
- Ito K, Aoki T, Higuchi T (2004) Design of an excitable digital reaction-diffusion system for shortest path search. In: ITC-CSCC 2004, Japan
- Kaneko K (1983) Transition from torus to chaos accompanied by frequency lockings with symmetry breaking. *Prog Theor Phys* 69(5):1427–1442
- Kaneko K (1984) Period-doubling of kink-antikink patterns, quasiperiodicity in antiferro-like structures and spatial intermittency in coupled logistic lattice. *Prog Theor Phys* 72(3):480–486
- Kaneko K (1985) Spatiotemporal intermittency in coupled map lattices. *Prog Theor Phys* 74(5):1033–1044
- Kaneko K (1986) Lyapunov analysis and information flow in coupled map lattices. *Physica D* 23:436–447
- Kanso A, Smaoui N (2009) Logistic chaotic maps for binary numbers generations. *Chaos, Solitons & Fractals* 40(5):2557–2568
- Kauffman SA (1990) Requirements for evolvability in complex systems. *Physica D* 42:135–152
- Kauffman SA (1993) The origins of order. Oxford University Press, New York
- Kelso JAS (1995) Dynamic Patterns. MIT Press, Cambridge, MA
- Kitano H (2004) Biological robustness. *Nat Rev Genet* 5:826–837
- Kocarev L, Jakimoski G (2001) Logistic map as a block encryption algorithm. *Phys Lett A* 289 (4–5):199–206
- Kuhnert L, Agladze K, Krinsky V (1989) Image processing using light-sensitive chemical waves. *Nature* 337:244–247
- Kunishima W, Nishiyama A, Tanaka H, Tokihiro T (2004) Differential equations for creating complex cellular automaton patterns. *J Phys Soc Jpn* 73:2033–2036
- Land M, Belew RK (1995) No perfect two-state cellular automata for density classification exists. *Phys Rev Lett* 74(25):5148–5150
- Li TY, Yorke JA (1975) Period three implies chaos. *Am Math Monthly* 82(10):985–992
- Lorenz EN (1993) The essence of chaos. UCL Press, London
- Maturana HR, Varela FJ (1980) Autopoiesis and Cognition. D. Reidel, Boston, MA
- May RM (1976) Simple mathematical models with very complicated dynamics. *Nature* 261:459–467
- Melby P, Kaidel J, Weber N, Hübner A (2000) Adaptation to the edge of chaos in a self-adjusting logistic map. *Phys Rev Lett* 84(26):5991–5993
- Melby P, Weber N, Hübner A (2002) Robustness of adaptation in controlled self-adjusting chaotic systems. *Fluctuation Noise Lett* 2(4):L285–L292
- Melby P, Weber N, Hübner A (2005) Dynamics of self-adjusting systems with noise. *Chaos* 15:033902

- Metropolis N, Stein M, Stein P (1973) On finite limit sets for transformations on the unit interval. *J Comb Theory* 15(1):25–43
- Mills JW (2008a) The architecture of an extended analog computer core. In: UCAS-4, Austin, TX, USA
- Mills JW (2008b) The nature of the extended analog computer. *Physica D* 237(9):1235–1256
- Mills JW, Parker M, Himebaugh B, Shue C, Kopecky B, Weilemann C (2006) “Empty Space” computes: The evolution of an unconventional supercomputer. In: Proceedings of the 3rd ACM computing frontiers conference, New York, pp. 115–126
- Mitchell M, Crutchfield JP, Das R (1996) Evolving cellular automata with genetic algorithms: a review of recent work. In: Goodman ED, Uskov VL, Punch WF (eds) Evolutionary computation and its applications: EvCA'96, Moscow
- Motoike IN, Adamatzky A (2004) Three-valued logic gates in reaction-diffusion excitable media. *Chaos Solitons Fractals* 24:107–114
- Moulay E, Baguelin M (2005) Meta-dynamical adaptive systems and their application to a fractal algorithm and a biological model. *Physica D* 207:79–90
- Murray JD (1993) Mathematical biology, 2nd edn. Springer, New York
- Měch R, Prusinkiewicz P (1996) Visual models of plants interacting with their environment. In: SIGGRAPH '96, ACM, New Orleans, LA pp 397–410
- Omohundro S (1984) Modelling cellular automata with partial differential equations. *Physica D* 10:128–134
- Ott E, Grebogi C, Yorke JA (1990) Controlling chaos. *Phys Rev Lett* 64(11):1196–1199
- Ott E, Sauer T, Yorke JA (eds) (1994) Coping with chaos. Wiley, New York
- Pareekha NK, Patidara V, Sud KK (2006) Image encryption using chaotic logistic map. *Image Vis Comput* 24(9):926–934
- Pask G (1960) The natural history of networks. In: Yovits MC, Cameron S (eds) Self-organizing systems. Pergamon, New York
- Păun G (2000) Computing with membranes. *J Comput Syst Sci* 61(1):108–143
- Phatak SC, Rao SS (1995) Logistic map: a possible random-number generator. *Phys Rev E* 51(4): 3670–3678
- Powley EJ, Stepney S (2009a) Automorphisms of transition graphs for elementary cellular automata. *J Cell Autom* 4(2):125–136
- Powley EJ, Stepney S (2009b) Automorphisms of transition graphs for linear cellular automata. *J Cell Autom* 4(4):293–310
- Prusinkiewicz P, Lindenmayer A (1990) The algorithmic beauty of plants. Springer, New York
- Prusinkiewicz P, James M, Měch R (1994) Synthetic topiary. In: SIGGRAPH '94, ACM, Orlando, FL, pp 351–358
- Rendell P (2002) Turing universality of the game of life. In: Adamatzky A (ed) Collision-based computing. Springer, London, chap 18
- Robinson JC (2001) Infinite-dimensional dynamical systems: an introduction to dissipative parabolic PDEs and the theory of global attractors. Cambridge University Press, Cambridge, MA
- Roselló-Merino M, Bechmann M, Sebald A, Stepney S (2010) Classical computing in nuclear magnetic resonance. *Int J Unconventional Comput* 6 (3–4):163–195
- Rothenmund PWK, Papadakis N, Winfree E (2004) Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol* 2(12):e424
- Rubel LA (1993) The extended analog computer. *Adv Appl Math* 14:39–50
- Sielewiesiuk J, Gorecki J (2001) Logical functions of a cross-junction of excitable chemical media. *J Phys Chem A* 105(35):8189–8195
- Sinha S, Ditto WL (1998) Dynamics based computation. *Phys Rev Lett* 81(10):2156–2159
- Sinha S, Ditto WL (1999) Computing with distributed chaos. *Phys Rev E* 60(1):363–377
- Sinha S, Ditto WL (2006) Exploiting the controlled responses of chaotic elements to design configurable hardware. *Phil Trans R Soc A* 364:2483–2494
- Sloane NJA (2008) The on-line encyclopedia of integer sequences. <http://www.research.att.com/~njas/sequences/> (accessed 10 November 2008)
- Spicher A, Michel O, Giavitto JL (2004) A topological framework for the specification and the simulation of discrete dynamical systems. In: ACRI 2004, Amsterdam, October 2004. LNCS, vol 3305. Springer, Heidelberg, pp 238–247
- Steinbock O, Tóth A, Showalter K (1995) Navigating complex labyrinths: optimal paths from chemical waves. *Science* 267:868–871
- Stepney S (2007) Embodiment. In: Flower D, Timmis J (eds) *In silico immunology*, Springer, New York, chap 12, pp 265–288
- Stepney S (2008) The neglected pillar of material computation. *Physica D* 237(9):1157–1164
- Stepney S (2009) Visualising random Boolean network dynamics. In: GECCO 2009, ACM, New York
- Stepney S, Polack F, Turner H (2006) Engineering emergence. In: ICECCS 2006, IEEE, Stanford, CA pp 89–97
- Strogatz SH (1994) Nonlinear dynamics and chaos. Westview, Boulder, CO
- Sutner K (2005) Universality and cellular automata. In: Machines, computations, and universality 2004, Saint Petersburg, September 2004. LNCS, vol 3354. Springer, Heidelberg, pp 50–59
- Teman R (1997) Infinite-dimensional dynamical systems in mechanics and physics, 2nd edn. Springer, New York

- Tokihiro T, Takahashi D, Matsukidaira J, Satsuma J (1996) From soliton equations to integrable cellular automata through a limiting procedure. *Phys Rev Lett* 76(18):3247–3250
- Tolmachev D, Adamatzky A (1996) Chemical processor for computation of Voronoi diagram. *Adv Mater Opt Electron* 6(4):191–196
- Toth A, Showalter K (1995) Logic gates in excitable media. *J Chem Phys* 103:2058–2066
- Ulam SM, von Neumann J (1947) On combination of stochastic and deterministic processes. *Bull Am Math Soc* 53(11):1120, (abstract 403)
- Waddington CH (1957) The strategy of the genes. Allen and Unwin, London
- Weimar JR, Boon JP (1994) Class of cellular automata for reaction-diffusion systems. *Phys Rev E* 49 (2):1749–1752
- Weyl H (1949) Philosophy of mathematics and natural science. Princeton University Press, Princeton, NJ
- Winfree E (2003) DNA computing by self-assembly. *The Bridge* 33(4):31–38
- Wolfram S (1984a) Computation theory of cellular automata. *Commun Math Phys* 96:15–57
- Wolfram S (1984b) Universality and complexity in cellular automata. *Physica D* 10:1–35
- Wolfram S (1985) Twenty problems in the theory of cellular automata. *Phys Scr T9*:170–183
- Wolfram S (1986a) Cellular automata fluids: basic theory. *J Stat Phys* 45:471–526
- Wolfram S (1986b) Random sequence generation by cellular automata. *Adv Appl Math* 7:123–169
- Wolz D, de Oliveira PPB (2008) Very effective evolutionary techniques for searching cellular automata rule spaces. *J Cell Autom* 3(4):289–312
- Wuensche A (2002) Finding gliders in cellular automata. In: Adamatzky A (ed) Collision-based computing. Springer, London, chap 13
- Wuensche A, Lesser M (1992) The global dynamics of cellular automata. Addison-Wesley, Reading
- Yates FE (ed) (1987) Self-organizing systems: the emergence of order. Plenum, New York

Index

#

- (1+1) EA 817 [26]
- 1-norm ranking support vector machine (SVM) 496 [15]
- 1/5-success rule 684 [22]
- 2-edge colored graph 1247 [37]
- 2-Opt
 - operator 1626 [49]
 - neighborhood 1625 [49]
- 3-CNF 1086 [33]
- 3' end 1075 [33]
- 3-SAT 1086 [33]
- 3-vertex-colorability problems 1155 [34]
- 5' end 1075 [33]

A

- Abelian stabilizer problem 1455 [43]
- absolute value (mapping) 1402 [41]
- abstract geometrical computation (AGC) 1969 [58]
- abstract tile assembly model (aTAM) 1168 [34]
- accumulated progress 745 [25]
- accuracy (approximation) 1927 [57]
- action potential (spike) 343 [10], 535 [17]
- activatable tiles 1339 [39]
- active transport 1222 [36]
- actual solution 995 [31]
- adaptive
 - controller 1696 [50]
 - hardware 1692 [50]
 - judgement 1943 [57]
 - maneuvering logic (AML) 704 [23]
 - search 922 [29]
- adenine (A) 1075 [33]
- adiabatic quantum computing 1502 [44]
- adjacent 1240 [37]
- adjoint 1385 [41]
 - mapping 1387 [41]
- Adleman's experiment 1084 [33]
- advanced encryption standard (AES) 1525 [45]
- advection-diffusion model 319 [9]
- affinity
 - immune systems 1579 [47]
 - maturation 1579 [47]
 - purification 1082 [33]
 - separation 1294 [38]
- agarose gel 1079 [33]
- age
 - individual 676 [22]
 - parameter 676 [22]

agent-based modelling (ABM)

- financial markets 1728 [51]

aggregate fitness 1010 [31]

algorithm(s)

- aiNET (immune network algorithm) 1581 [47]
- ant 1603 [48]
- artificial immune systems 908 [29], 1575 [47]
- bee-inspired 1610 [48]
- bacterial foraging 1609 [48]
- clonal selection (CLONALG) 1579 [47]
- coevolutionary 988 [31]
- continued fractions 1456 [43]
- decryption 1523 [45]
- dendritic cell (DCA) 1584 [47]
- Deutsch-Jozsa 1498 [44]
- encryption 1523 [45]
- estimation of distribution algorithm (EDA)
 - 908 [29]
- evolutionary 625 [20]
- factoring 1381 [41], 1410 [41]
- foraging 1609 [48]
- generative topographic map (GTM) 611 [19]
- genetic 637 [21]
- Gillespie's 1874 [55]
- Grover's (quantum search) 1410 [41], 1448 [42], 1457 [43], 1498 [44]
- growing cell structures (GCS) 604 [19]
- growing grid (GG) 604 [19]
- immune 1578 [47]
- immune-inspired 1576 [47]
- immune network 1581 [47]
- linear programming (LP) 497 [15]
- LUES (BQP) 1551 [46]
- memetic 905 [29]
- Metropolis 1461 [43]
- natural gradient 443 [13]
- negative selection 1582 [47]
- nondominated sorting genetic (NSGA-II) 887 [28], 1766 [52]
- quantum computing 1451 [43]
- self-organizing maps (SOM) 589 [19]
- SHA-1 (secure hashing) 1525 [45]
- Shor's 1455 [43], 1522 [45]
- swarm intelligence 1604 [48]

algorithmic systems biology 1835 [54]

- modeling biological systems 1839 [54]
- underlying metaphor 1838 [54]
- workflow 1858 [54]

- alkaline 1507 [44]
- alkaline-earth 1505 [44]
- α -function 543 [17]
- alpha-male 1053 [32]
- alphabet, trajectory 1097 [33]
- Amari–Hopfield neural network 554 [17]
- Ambient calculus 1877 [55]
- amino acid 1077 [33]
- AMPA/kainate synaptic receptor 544 [17]
- amplitude 1391 [41]
 - amplification 1457 [43]
 - estimation 1458 [43]
- analog circuits
 - evolvable hardware 1676 [50]
- analog computing 2004 [59]
- analysis of coevolution 1011 [31]
- analytical science
 - natural computing applications 1758 [52]
- ancilla bits 1411 [41]
- AND
 - circuit 1207 [36]
 - gate 1207 [36]
 - OR tree 1475 [43]
- annealing 1077 [33]
- anomaly detection 1586 [47]
- ant
 - algorithms 1603 [48]
 - colony optimization (ACO) 908 [29], 1613 [48]
 - system (AS) 1613 [48]
- antibodies 1577 [47]
- anyonic quantum computing 1502 [44]
- application-specific integrated circuit (ASIC)
 - 1661 [50]
- approximate
 - eigenvalue sampling 1549 [46]
 - reasoning 1933 [57]
- approximation space (AS) 1927 [57]
- aptamer 1763 [52]
- antenna design 1679 [50], 1786 [52]
- archives 1005 [31]
- art
 - evolutionary 1781 [52]
- artificial
 - cell 1227 [36]
 - chemistry 1881 [55]
 - economies 1826 [53]
 - receptor 1227 [36]
 - recognition ball (ARB) 1581 [47]
 - societies 1826 [53]
- artificial ant 718 [24]
- artificial immune systems (AIS) 908 [29], 1575 [47]
 - algorithm 1579 [47]
 - applications, anomaly detection 1586 [47]
 - applications, network intrusion detection 1583 [47]
 - applications, general 1589 [47]
 - engineering 1578 [47]
- artificial intelligence (AI)
 - strong and weak versions 1807 [53]
- artificial life 1805 [53]
 - art 1825 [53]
 - biology 1822 [53]
 - cellular automata 1817 [53]
 - computer games 1825 [53]
 - constructive systems 1820 [53]
 - dynamical systems 1819 [53]
 - emergence 1809 [53]
 - engineering 1823 [53]
 - environmental science 1823 [53]
 - graphics 1824 [53]
 - hierarchy 1809 [53]
 - history 1806 [53]
 - Langton’s definition 1806 [53]
 - medicine 1823 [53]
 - networks (graphs) 1821 [53]
 - rewriting systems 1818 [53]
 - virtual worlds 1824 [53]
 - von Neumann 1806 [53]
- artificial neural network (ANN) 631 [20]
 - checkers (game) 1746 [52]
- asexual reproduction 628 [20]
- associative memory 554 [17]
- associative molecular memory
 - 1282 [38]
- attainment function method 893 [28]
- autocatalytic sets 1811 [53]
- automatically defined functions (ADFs)
 - 718 [24]
- autonomous
 - biomolecular computation 1327 [39]
 - computation 1205 [36]
 - DNA walkers 1347 [39]
 - molecular cascade 1345 [39]
- autonomy 1213 [36]
- autopoiesis 1810 [53]
- autopoietic systems 1810 [53]
- autoregressive
 - integrated moving average (ARIMA) model 467 [14]
 - moving average with exogenous inputs (ARMAX) 403 [12]
- Avida (artificial life) 1815 [53]
- axial progress rate 776 [25]
- axoaxonic synapse 540 [17]
- axodendritic synapse 540 [17]
- axon 540 [17]
- axosomatic synapse 540 [17]

B

- B cells 1577 [47]
- backbone 1075 [33]
- backpropagation 337 [10], 359 [10]
 - decorrelation 365 [10]

- bacterial
 - chemotaxis 1609 [48]
 - computing 1203 [36]
 - foraging algorithm 1609 [48]
- Baldwin effect 906 [29], 946 [30]
- ballistic computer 1950 [58]
- base pairs (bp) 1076 [33]
- basin of attraction 1040 [32]
 - dynamical systems 1982 [59]
- Batik pattern (interactive evolution design) 1781 [52]
- B-boundary region (rough-fuzzy computing) 1926 [57]
- BCCS (basic CCS) 1867 [55]
- beam search 917 [29]
- bees 1610 [48]
- B-elementary (rough-fuzzy computing) 1925 [57]
- Bell's inequalities 1414 [42], 1532 [45]
 - Alice 1420 [42]
 - Bob 1420 [42]
 - Cecil 1425 [42]
- Bell's theorem 1420 [42]
- Belousov-Zhabotinsky (BZ)
 - processor 1902 [56]
 - processor, light-sensitive subexcitable 1903 [56]
 - reaction 303 [9], 1900 [56], 1960 [58]
- Bernoulli measure 12 [1], 264 [8]
- Berry phase 1502 [44]
- best worst case 1001 [31]
- betabinders 1841 [54], 1880 [55]
- BetaWB (tools based on BlenX language) 1856 [54]
- Bhatnagar-Gross-Krook (BGK) model 315 [9]
- billiard ball(s)
 - collision-based computing devices 1952 [58]
 - model 248 [7]
- binary
 - shift map 1992 [59]
 - support vector machine (SVM) 480 [15]
- binding site 577 [18]
- B-indiscernible (rough-fuzzy computing) 1925 [57]
- BioAmbient calculus 1877 [55]
- bioinformatics 565 [18]
- bioinspired hardware 1698 [50]
- biomolecular devices 1319 [39]
- biomolecules 1129 [34]
- bionics 702 [23]
- bio-operation 1075 [33]
- Bio-PEPA 1873 [55]
- bit commitment 1527 [45]
 - quantum 1537 [45]
- black-box optimization 821 [26]
- black-hole model (collision-based computing) 1975 [58]
- Black-Scholes model 1726 [51]
- BlenX (modeling language, systems biology) 1848 [54], 1891 [55]
- blind source separation (BSS) 436 [13]
- bloat 722 [24]
- Bloch sphere 1396 [41]
- block cipher 1525 [45]
- Blondie24 (checkers playing program) 1744 [52]
- Blum-Shub-Smale (BSS) model 1976 [58]
- blunt end 1078 [33]
- body force 326 [9]
- Boltzmann machine 351 [10]
- bond-free property 1098 [33]
- Boolean circuit
 - crossing 208 [6]
 - delay 207 [6]
 - fan-out 208 [6]
 - gate 208 [6]
 - reasoning 1931 [57]
 - simulation, cellular automata 207 [6]
 - turn 207 [6]
 - wire 207 [6]
- Borel measure 43 [2]
- Bose-Einstein condensate 1506 [44]
- bottom DNA strand 1105 [33]
- bounce back rule 325 [9]
- boundary conditions 323 [9]
 - adiabatic 292 [9]
 - fixed 292 [9]
 - periodic 292 [9]
 - reflecting 293 [9]
- bounded-error
 - probabilistic polynomial-time (BPP) 1547 [46]
 - quantum polynomial-time (BQP) 1540 [45], 1548 [46]
- bounded support vector 494 [15]
- bracket Kauffman polynomial 1559 [46]
- braid 1559 [46]
 - group 1559 [46]
- branch and bound 917 [29]
- branch migration 1165 [34]
- brane
 - calculi 1368 [40], 1878 [55]
 - calculus, simplified (sBrane) 1878 [55]
- BRIAN (spiking neuron simulator) 369 [10]
- bridge molecule 1160 [34]
- brood sorting 1604 [48]
- Broyden-Fletcher-Goldfarb-Shanno method 921 [29]
- bulge loop 1142 [34]

C

- calculus/calculi
 - Ambient 1877 [55]
 - BioAmbient 1877 [55]
 - biology 1841 [54]
 - brane 1878 [55]
 - of communicating systems (CCS) 1867 [55]
 - pi- 1874 [55]
 - stochastic pi- 1884 [55]
- calyx of Held 1893 [55]

- candidate
 - node 520 [16]
 - solution 998 [31]
- cargo
 - loading 1222 [36]
 - unloading 1222 [36]
- Cartesian genetic programming (CGP) 1669 [50]
- cavity QED (quantum electrodynamical) 1503 [44]
- Cayley graph 172 [5]
- cell
 - assembly 356 [10]
 - differentiation 301 [9]
- cellular automata (CA)
 - algorithmic tools 77 [3]
 - almost equicontinuous 27 [2]
 - artificial life 1817 [53]
 - attractor 28 [2]
 - bacterial computing 1219 [36]
 - balance theorem 10 [1], 65 [2]
 - bijective 6 [1]
 - block rule 236 [7]
 - blocking word 21 [1], 31 [2]
 - Boolean circuits simulation 207 [6]
 - chaotic 22 [1], 27 [2]
 - classification 299 [9]
 - closing 33 [2], 66 [2]
 - collision-based computing 1953 [58]
 - communication, one-way 7 [1], 125 [4], 222 [6]
 - communication, two-way 125 [4]
 - computations on 159 [5]
 - configurations 28 [2], 65 [2], 79 [3]
 - configurations, asymptotic 13 [1], 269 [8]
 - configurations, basic initial 81 [3]
 - configurations, Bernoulli measure 12 [1], 264 [8]
 - configurations, Besicovitch pseudometric 47 [2]
 - configurations, Borel measure 43 [2]
 - configurations, boundary 269 [8]
 - configurations, Cantor topology 9 [1], 28 [2], 268 [8]
 - configurations, compact metric 9 [1]
 - configurations, cylinder 9 [1], 31 [2]
 - configurations, equicontinuity point 21 [1]
 - configurations, finite pattern 10 [1], 268 [8]
 - configurations, finite unbounded 8 [1]
 - configurations, Garden-of-Eden 10 [1], 234 [7]
 - configurations, generic space 47 [2]
 - configurations, Gibbs measure 264 [8]
 - configurations, ground 284 [8]
 - configurations, observation window 20 [1], 270 [8]
 - configurations, orphan 11 [1], 20 [1]
 - configurations, pattern 10 [1], 65 [2], 268 [8]
 - configurations, periodic 8 [1]
 - configurations, product topology 9 [1], 268 [8]
 - consequences 174 [5]
 - conservation laws 259 [8]
 - de Bruijn diagram 16 [1], 276 [8]
 - definition 5 [1], 79 [3], 199 [6], 233 [7], 288 [9]
 - dependency graph 128 [4], 161 [5], 172 [5]
 - division 113 [3]
 - dynamical systems 25 [2]
 - elementary 8 [1], 1986 [59]
 - equicontinuous 21 [1], 27 [2]
 - firing squad 110 [3]
 - folding space-time 116 [3]
 - freezing 111 [3]
 - Game of Life 208 [6], 293 [9]
 - gliders 1951 [58]
 - influences 174 [5]
 - injective 6 [1], 32 [2], 234 [7]
 - input mode, parallel 128 [4], 162 [5]
 - input mode, sequential 128 [4], 162 [5]
 - invariant measure 12 [1], 44 [2], 264 [8]
 - inverse 6 [1], 14 [1], 234 [7]
 - invertible 234 [7]
 - language recognition 126 [4]
 - lattice Boltzmann modeling 287 [9]
 - limit set 28 [2]
 - measure attractor 46 [2]
 - mixing 22 [1], 27 [2]
 - multiplication 104 [3]
 - neighborhood 5 [1], 268 [8], 290 [9]
 - neighborhood, Margolus 236 [7]
 - neighborhood, Moore 6 [1]
 - neighborhood, radius-1/2 7 [1]
 - neighborhood, von Neumann 6 [1]
 - nilpotent 6 [1], 20 [1]
 - number-conserving 284 [8]
 - one-dimensional 1962 [58]
 - one-way totalistic lookup 222 [6]
 - open 32 [2]
 - parallel Turing machines table lookup 221 [6]
 - particle weight functions 48 [2]
 - partitioned 237 [7]
 - periodic 20 [1]
 - permutive 33 [2], 66 [2]
 - positively expansive 21 [1], 27 [2], 284 [8]
 - preinjective 13 [1]
 - reversible 6 [1], 10 [1], 14 [1], 232 [7]
 - rules 290 [9]
 - rules, Banks universal CA 213 [6]
 - rules, Copper 191 [6]
 - rules, forest fire 304 [9]
 - rules, four states universal CA 223 [6]
 - rules, Frisch–Hasslacher–Pomeau (FHP) 310 [9]
 - rules, Gacs–Kurdyumov–Levin 54 [2]
 - rules, Game of Life 208 [6], 293 [9]
 - rules, Hardy–Pomeau–de Pazzis (HPP) 307 [9]
 - rules, Just Gliders 261 [8]
 - rules, Langton ant 293 [9]
 - rules, Lindgren and Nordahl universal CA 215 [6]
 - rules, majority CA 12 [1]
 - rules, parity rule 290 [9]

- rules, product rule 31 [2]
- rules, random walk 57 [2]
- rules, rule 110 8 [1], 14 [1], 216 [6]
- rules, sand pile 266 [8]
- rules, Smith III universal CA 215 [6]
- rules, traffic 261 [8], 279 [8]
- rules, twisted majority 300 [9]
- second-order 240 [7]
- sensitive to initial conditions 21 [1], 27 [2]
- signal(s) 80 [3], 168 [5], 222 [6], 1965 [58]
- slicing construction 66 [2]
- space-time diagram 6 [1], 161 [5], 1970 [58]
- spiral rule 1953 [58]
- subshift attractor 34 [2]
- surjective 6 [1], 32 [2], 234 [7]
- time complexity, linear time 117 [3], 129 [4]
- time complexity, real-time 104 [3], 129 [4], 163 [5]
- transitive 22 [1], 27 [2]
- trellis 7 [1], 161 [5], 172 [5]
- undecidability results 20 [1]
- universality 189 [6]
- universality, computational 190 [6], 240 [7], 300 [9]
- universality, intrinsic 190 [6], 252 [7]
- universality, reversible intrinsic 226 [6], 252 [7]
- universality, Turing 190 [6], 240 [7]
- cemetery formation (ants) 1605 [48]
- center-of-gravity (COG) defuzzification 402 [12]
- central pattern generator (CPG) 549 [17]
 - pyloric 549 [17]
 - swimming 549 [17]
- CF (language) 1132 [34]
- chain terminator in PCR 1084 [33]
- checkers (game)
 - Blondie24 program 1746 [52]
 - Chinook program 1749 [52]
- chemical
 - master equation (CME) 1882 [55]
 - processors 1902 [56]
 - synapse 541 [17]
- chemoreceptor 551 [17]
- chemotaxis 1609 [48]
- Chernoff bounds 830 [26]
- chosen
 - ciphertext attack 1526 [45]
 - message attack, adaptive 1527 [45]
 - plaintext attack 1526 [45]
- chromatic polynomial 1584 [46]
- chromatogram 1761 [52]
- chromosome 1076 [33]
- Church–Turing thesis 1382 [41], 1407 [41]
- ciliate 1092 [33], 1233 [37]
 - conjugation 1236 [37]
 - gene assembly 1237 [37]
- ciphertext 1523 [45]
- circuit QED (quantum electrodynamical) 1516 [44]
- circular splicing systems 1148 [34]
- classification
 - immune systems 1578 [47]
 - tasks 938 [30]
- classifier system 1771 [52]
- clause module 1087 [33]
- Clauzer–Horne–Shimony–Holt (CHSH) inequality 1431 [42]
- Clione 551 [17]
- clonal selection
 - algorithm (CLONALG) 1579 [47]
 - theory 1579 [47]
- closed-loop multiobjective optimization 1758 [52]
 - Belousov–Zhabotinsky (BZ) processor 1906 [56]
- cluster state quantum computing 1502 [44]
- clustering
 - ant-based algorithms 1604 [48]
 - swarm intelligence algorithms 1604 [48]
- coding DNA 1076 [33]
- codon 1077 [33]
 - table 1313 [38]
- coevolution 987 [31]
- coevolutionary
 - algorithm 988 [31]
 - evaluation 1005 [31]
 - evolutionary algorithm 633 [20]
 - learning 947 [30]
 - principles 987 [31]
 - progress 1011 [31]
- collaboration 999 [31]
- collision 308 [9], 326 [9]
 - based computing 1900 [56], 1949 [58]
 - free path 1906 [56]
 - term 312 [9]
- combat maneuvers, classifier system 1744 [52]
- combinatorics 1771 [52]
- comma-free words 1312 [38]
- communication 1010 [31]
 - paradigm 1205 [36]
- comparability 882 [28]
- competition models 301 [9]
- competitive
 - coevolution 990 [31]
 - domain 994 [31]
 - Hebbian learning (CHL) 601 [19]
- complementary alphabet 1133 [34]
- completely positive 1403 [41]
 - map 1461 [43]
- complex systems 293 [9]
- complexity
 - parallel 1258 [37]
- component 996 [31]
 - cyclic 1248 [37]
 - linear 1248 [37]

- compositional
 - coevolution 999 [31]
 - problem 997 [31]
 - compound systems (quantum information processing) 1397 [41]
 - computation
 - molecule 1160 [34]
 - universality 190 [6], 240 [7], 300 [9]
 - computational
 - basis 1395 [41]
 - complexity, evolutionary algorithm 815 [26]
 - geometry, reaction-diffusion application 1904 [56]
 - models, biosystems 1867 [55]
 - computing
 - by carving 1156 [34]
 - developmental 1711 [51]
 - evolutionary 1710 [51]
 - grammatical 1711 [51]
 - immunocomputing 1710 [51]
 - neurocomputing 1709 [51]
 - physical 1711 [51]
 - social 1710 [51]
 - with words 1928 [57]
 - concept drift 1923 [57]
 - confluent 1252 [37]
 - conformational transition 1308 [38]
 - connection topology 550 [17]
 - conservation laws 259 [8], 271 [8], 313 [9]
 - local 279 [8]
 - mass 315 [9]
 - momentum 315 [9]
 - conservative logic 1951 [58]
 - constraint
 - function 874 [28]
 - handling 890 [28]
 - -integrating preference relation 890 [28]
 - satisfaction 632 [20]
 - constructive systems, artificial life 1820 [53]
 - contamination models 302 [9]
 - continued fractions algorithm 1456 [43]
 - continuity equation 279 [8], 316 [9]
 - continuous
 - evolutionary programming 704 [23]
 - time Markov chain (CMTC) 1873 [55]
 - Contourlet transform 386 [11]
 - controlled
 - NOT (CNOT) gate 1499 [44]
 - rotation quantum logic operation (CROT) 1500 [44]
 - convergence
 - analysis 1022 [31]
 - properties 741 [25]
 - convex set 1389 [41]
 - Cooper pair 1513 [44]
 - cooperation models 300 [9]
 - cooperative
 - coevolution 990 [31]
 - domain 994 [31]
 - co-optimization problem 995 [31]
 - coordinate system 1027 [31]
 - coordination 1011 [31]
 - Coreworld (artificial life) 1814 [53]
 - CoSBi Lab (algorithmic systems biology) 1855 [54]
 - co-search problem 995 [31]
 - Coulomb interaction 1507 [44]
 - counter machine 275 [8]
 - coupled map lattices 1995 [59], 2011 [59]
 - covalent bonds 1075 [33]
 - covariance
 - learning rule 556 [17]
 - matrix 680 [22]
 - matrix adaptation (CMA) 694 [22]
 - matrix adaptation evolution strategy (CMA-ES) 1052 [32]
 - crisp
 - function 1936 [57]
 - set 1926 [57]
 - cross tile 1329 [39]
 - crossover 639 [21], 712 [24], 858 [27]
 - 1-point 677 [22]
 - uniform 677 [22]
 - crowding 1048 [32]
 - deterministic 1048 [32]
 - cryptography, quantum 1521 [45]
 - CS (language) 1132 [34]
 - curse-of-dimensionality problem 487 [15]
 - Curtis–Hedlund–Lyndon theorem 9 [1], 65 [2]
 - cycle tag systems 1966 [58]
 - cyclic
 - component 1248 [37]
 - tag system 244 [7]
 - cytokines 1577 [47]
 - Cyto-Sim (simulator, biochemical processes) 1857 [54]
 - cytosine (C) 1075 [33]
 - cytoskeletal filament 2290 [36]
 - cytosolic calcium 546 [17]
- D**
- D2Q9 lattice 314 [9]
 - D3Q19 lattice 315 [9]
 - DAMNED (spiking neuron simulator) 369 [10]
 - danger theory 1584 [47]
 - Darwinian evolution 945 [30]
 - data
 - mining 951 [30]
 - splitting 467 [14]
 - data encryption standard (DES) 1525 [45]
 - d-Detect 1134 [34]
 - DdQq lattice 313 [9]

- decision
 - making 553 [17]
 - space 874 [28]
 - trees 631 [20]
 - vector 874 [28]
- decoherence-free subspace 1501 [44]
- decomposable 1398 [41]
 - state 1399 [41]
- decryption algorithm 1523 [45]
- demixing match 436 [13]
- denaturation 1077 [33]
- dendritic cell algorithm (DCA) 1584 [47]
- dendroaxonic synapse 541 [17]
- dendrodendritic synapse 541 [17]
- dendrosomatic synapse 541 [17]
- density
 - distributions 311 [9]
 - matrix 1388 [41]
 - task 300 [9]
- deoxyribonucleoside triphosphate (dNTP) 1083 [33]
- deoxyribose 1076 [33]
- deoxyribozyme (DNAzyme) 1156 [34], 1324 [39]
- derandomized evolution strategies 1052 [32]
- descriptor 1245 [37]
 - assembled 1262 [37]
 - associated 1263 [37]
 - circular 1263 [37]
 - internal eliminated sequences (IES) 1262 [37]
 - macronuclear destined segments (MDS) 1262 [37]
 - micronuclear 1263 [37]
 - micronuclear MDS-IES 1262 [37]
 - linear 1263 [37]
 - pointer, MDS-IES 1263 [37]
- design
 - antenna 1679 [50], 1786 [52]
 - evolutionary 1668 [50]
 - gear train 1766 [52]
 - innovation 1764 [52]
 - welded beam 1768 [52]
- design of experiments (DoE) 920 [29]
- desire edge 1247 [37]
- detection
 - problem 1463 [43]
 - protocol 1343 [39]
- deterministic
 - evolution equations 747 [25]
 - finite automaton (DFA) 1142 [34]
 - Turing machine (DTM) 1132 [34]
- Deutsch–Jozsa algorithm 1498 [44]
- developmental
 - biology 1217 [36]
 - computing 1657 [50]
 - encoding 1668 [50]
 - genetic programming (GP) 1677 [50]
- diagnostic molecular automaton (DMA) 1160 [34]
- dideoxyribonucleoside triphosphate (ddNTP) 1082 [33]
- difference equations 747 [25], 1991 [59]
- diffusion model 318 [9]
- digital filter, optimization via evolvable hardware 1672 [50]
- dipole–dipole interaction 1506 [44]
- Dirac notation 1385 [41]
- direct sum 1389 [41]
- directed Hamiltonian path problem (DHPP)
 - 1147 [34]
 - DHPP Pot 1147 [34]
- discernibility 1931 [57]
- discrete dynamical system 26 [2]
 - conjugacy 27 [2]
 - factor 27 [2]
 - homomorphism 26 [2]
 - subsystem 27 [2]
- discrete logarithm 1382 [41], 1455 [43], 1546 [46]
- discrete time evolution 1403 [41]
 - closed 1404 [41]
- discrete time map 539 [17]
- discriminant matrix 1462 [43]
- displacement whiplash PCR 1092 [34]
- dissipation law 272 [8]
- distribution
 - binomial 678 [22]
 - entropy 678 [22]
 - geometric 680 [22]
 - maximum entropy 678 [22]
 - multivariate normal 689 [22]
 - uniform discrete 678 [22]
- diversity loss
 - evolutionary strategies 1043 [32]
 - genetic algorithms 1042 [32]
- DiVincenzo criteria 1495 [44]
- DNA (deoxyribonucleic acid) 1075 [33]
 - blunt end 1078 [33]
 - chip 1763 [52]
 - code design 1155 [34]
 - computing 1073 [33], 1185 [35]
 - computing, first experiment 1084 [33]
 - computing, grammatical model 1148 [34]
 - cutting 1080 [33]
 - device for medical diagnosis 1350 [39]
 - double helix 1076 [33], 1323 [39]
 - double strand (dsDNA) 1076 [33]
 - encoding information 1076 [33]
 - hairpin 1092 [33]
 - Holliday junction 1328 [39]
 - involution 1091 [33]
 - lattice 1328 [39]
 - library 1088 [33]
 - ligase 1078 [33]
 - melting 1079 [33]
 - memory 1077 [33], 1281 [38]
 - molecule 1075 [33]
 - nanobot 1349 [39]

- nanostructure 1320 [39]
 - negatively charged 1081 [33]
 - orientation 1076 [33]
 - origami 1336 [39]
 - overhang 1078 [33]
 - -Pascal framework 1148 [34]
 - polymerase enzyme 1080 [33]
 - probe 1087 [33]
 - RAM 1092 [33]
 - replication 1083 [33]
 - separation by length 1081 [33]
 - separation by size 1078 [33]
 - single strand (ssDNA) 1075 [33]
 - stem-loop 1327 [39]
 - sticky end 1078 [33], 1328 [39]
 - strand orientation 1075 [33]
 - sugar-phosphate backbone 1076 [33]
 - synthesis 1077 [33]
 - tile 1326 [39]
 - trajectory 1097 [33]
 - walker 1346 [39]
 - watermark 1313 [38]
 - DNAzyme (deoxyribozyme) 1156 [34], 1324 [39]
 - domain role 993 [31]
 - dominance
 - depth 887 [28]
 - rank 887 [28]
 - double
 - crossover molecule 1169 [34]
 - occurrence string 1245 [37]
 - double rule
 - graphs 1249 [37]
 - strings 1248 [37]
 - drift 833 [26]
 - DSPACE($f(n)$) (computational complexity theory) 1132 [34]
 - dual problem 483 [15]
 - DX tile 1329 [39]
 - dynamic
 - analysis 1024 [31]
 - environments 634 [20]
 - fitness sharing 1046 [32]
 - molecular alignment 1060 [32]
 - partial reconfiguration 1662 [50]
 - peak identification (DPI) 1047 [32]
 - synapse 545 [17]
 - dynamical
 - invariants 551 [17]
 - optimization 807 [25]
 - dynamical systems 741 [25], 1979 [59]
 - artificial life 1819 [53]
 - autonomous 1981 [59]
 - constructive 2014 [59]
 - constructive continuous 2018 [59]
 - constructive discrete 2016 [59]
 - constructive hybrid 2019 [59]
 - continuous space, continuous time 1997 [59], 2011 [59]
 - continuous space, discrete time 1990 [59], 2010 [59]
 - developmental 1980 [59]
 - discrete space, discrete time 1981 [59], 2008 [59]
 - hybrid 2005 [59]
 - nonautonomous 1980 [59]
 - open 2005 [59]
 - transition function 1981 [59]
- E**
- Earnshaw theorem 1504 [44]
 - eater-glider machine 1958 [58]
 - echo state networks 362 [10]
 - EcoRI 1144 [34]
 - edge 1240 [37]
 - desire 1247 [37]
 - merge 1251 [37]
 - reality 1247 [37]
 - eigenvalue estimation 1455 [43]
 - Einstein–Podolsky–Rosen (EPR) 1418 [42], 1540 [45]
 - Einstein summation convention 316 [9]
 - elasticity model 322 [9]
 - electrical synapse 541 [17]
 - electron-nuclear spin double resonance (ENDOR) 1510 [44]
 - electron spin-resonance (ESR) 1510 [44]
 - element distinctness
 - function 352 [10]
 - problem 1466 [43]
 - elementary
 - cellular automata 1986 [59]
 - formal system (EFS) 1139 [34]
 - elliptic curves 1455 [43]
 - emergence
 - artificial life 1809 [53]
 - dynamical systems 2021 [59]
 - emergent geometric organization 1026 [31]
 - empirical black-box analysis 1022 [31]
 - encryption algorithm 1523 [45]
 - end 1240 [37]
 - endonuclease digestion 1080 [33]
 - energy 260 [8], 269 [8]
 - average, per cell 270 [8]
 - conservation 271 [8]
 - difference 269 [8]
 - dissipation 272 [8]
 - expected, per cell 270 [8]
 - flow 279 [8]
 - interaction potential 269 [8]
 - interaction range 269 [8]
 - ensemble
 - modeling 472 [14]
 - quantum computing 1508 [44]
 - ensembles, learning 950 [30]

entangled state 1399 [41]
 entanglement 1414 [42]
 environmental selection 889 [28]
 enzyme-free logic gate 1165 [34]
 epsilon
 – constraint method 881 [28]
 – dominance 875 [28]
 – indicator 883 [28]
 equality machine 1136 [34]
 equilibrium 264 [8], 315 [9]
 equilibrium solutions 747 [25]
 erasing 1308 [38]
 ergodic theorem 271 [8]
 error correction 1500 [44]
 error-resilient tiling 1339 [39]
Escherichia coli (*E. coli*) 1224 [36]
 estimation of distribution algorithm (EDA)
 908 [29]
 eukaryote 1235 [37]
 even parity 718 [24]
 evolution
 – criterion 769 [25]
 – equations 747 [25]
 evolutionary
 – art 1781 [52]
 – computation (EC) 626 [20]
 – design 1668 [50]
 – experimentation 1758 [52]
 – game theory 1019 [31]
 – multiobjective optimization (EMO)
 871 [28]
 – path 745 [25]
 – programming (EP) 626 [20], 699 [23]
 evolutionary algorithm (EA) 626 [20], 675 [22]
 – black-box complexity 823 [26]
 – computational complexity 815 [26]
 – convergence results 855 [27]
 – generalized 625 [20]
 – niching 1035 [32]
 – no free lunch (NFL) 822 [26]
 evolutionary strategies (ES) 361 [10], 626 [20], 675 [22],
 704 [23], 742 [25]
 – (1+1) 676 [22]
 – CMA 694 [22]
 – CMA-EGS 695 [22]
 – CMSA 696 [22]
 – CORR 689 [22]
 – CORR-ES matrix, elementary rotation
 691 [22]
 – EGS 688 [22]
 – GSA 692 [22]
 – LS-CME 692 [22]
 – operational reach 681 [22]
 evolvable hardware (EHW) 704 [23], 1657 [50]
 evolved neural networks 705 [23]
 evolving neural networks 954 [30]

excitable
 – chemical coupled with manipulator 1912 [56]
 – chemical medium 1901 [56]
 – media 265 [8], 1916 [56], 1960 [58]
 excitatory synapse 544 [17]
 exciton 1513 [44]
 exclusion principle 308 [9]
 executable objects 632 [20]
 existential unforgeability 1527 [45]
 exon 1076 [33]
 expectation vector 680 [22]
 expected value 1393 [41]
 extended H system 1192 [35]
 extraction by pattern 1082 [33]
 extreme learning machine (ELM) 509 [16]
 extrinsic evolution 1658 [50]
F
 factoring 1509 [44], 1546 [46]
 – algorithm 1382 [41], 1410 [41]
 far-off-resonant traps (FORTs) 1506 [44]
 fault-tolerant computation 1500 [44]
 feasible set 675 [22], 874 [28]
 feature
 – construction 948 [30]
 – selection 496 [15], 947 [30]
 – space 480 [15]
 feeding central pattern generator (CPG) 549 [17]
 FHP (Frisch–Hasslacher–Pomeau) model 310 [9]
 field-programmable
 – analog array (FPAA) 1663 [50]
 – gate array (FPGA) 1661 [50]
 – transistor array (FPTA) 1662 [50]
 financial applications
 – agent-based modelling 1728 [51]
 – algorithmic trading 1713 [51]
 – arbitrage 1716 [51]
 – artificial immune system methods 1710 [51]
 – asset allocation 1719 [51]
 – credit risk assessment 1724 [51]
 – derivatives modeling 1725 [51]
 – evolutionary computation 1710 [51]
 – forecasting 1711 [51]
 – grammatical evolution 1711 [51]
 – index tracking 1722 [51]
 – investment analysis 1714 [51]
 – model induction 1708 [51]
 – neural networks 1709 [51]
 – portfolio management 1719 [51]
 – risk management 1723 [51]
 – technical analysis 1750 [52]
 – technical indicators 1714 [51]
 – trade execution 1717 [51]
 – trading rules 1749 [52]
 – volume weighted average price (VWAP) 1718 [51]
 Fine and Wilf theorem 1121 [33]

- finite
 - automaton 1197 [35], 1406 [41]
 - population model, Markov model 658 [21]
 - state machines (FSM) 196 [6], 700 [23]
- firing squad synchronization problem 110 [3]
- firmware 1660 [50]
- first occurrence 1250 [37]
- Fisher's constructability 100 [3]
 - marking times 100 [3]
- fitness 627 [20], 711 [24]
 - bias 629 [20]
 - -biased selection 629 [20]
 - function 675 [22]
 - landscape 629 [20], 724 [24]
 - mapping 919 [29]
 - optimization 631 [20]
 - -proportional selection 627 [20]
 - rank 629 [20]
 - sharing 1045 [32]
- FitzHugh–Nagumo model 537 [17]
- flocking 1610 [48]
- flow 260 [8], 279 [8]
 - polynomial 1584 [46]
- fluorescence resonance energy transfer (FRET) 1301 [38]
- flux qubits 1515 [44]
- FokI 1156 [34]
- foraging
 - algorithms 1609 [48]
 - bacteria 1609 [48]
 - honeybees 1610 [48]
- force 326 [9]
- forest fire rule 304 [9]
- formal systems 1863 [55]
- formula evaluation 1550 [46]
- Fourier transform 1495 [44]
- Fredkin gate 248 [7], 1500 [44]
- free diffusion 1222 [36]
- free lunch, coevolution 1018 [31]
- fuzzy dynamic model 404 [12]
- fuzzy logic 969 [30], 1921 [57]
 - applications 1933 [57]
 - connectives 1937 [57]
 - inference rules 1939 [57]
 - propositions 1937 [57]
 - quantifiers 1938 [57]
 - reasoning 1938 [57]
- fuzzy sets 1933 [57]
 - composition 1936 [57]
 - cutworthy properties 1936 [57]
 - fuzzy–rough sets 1941 [57]
 - intersection and union 1935 [57]
 - projection 1936 [57]
- G**
- gap junction 1221 [36]
- game 994 [31]
- Game of Life 208 [6], 293 [9]
 - crash 209 [6]
 - crossing 209 [6]
 - duplicator 209 [6]
 - eater 209 [6]
 - glider 209 [6]
 - gun 209 [6]
- gamma-aminobutyric acid (GABA)
 - GABA_A, fast inhibitory synaptic receptor 544 [17]
 - GABA_B, slow inhibitory synaptic receptor 544 [17]
- gap junction 541 [17]
- Garden-of-Eden theorem 13 [1], 65 [2], 234 [7]
- gear train design 1766 [52]
- Geffert normal form 1148 [34]
- gel electrophoresis 1081 [33], 1212 [36]
- gemini-peptide lipid 1225 [36]
- gene 1076 [33]
- gene assembly in ciliates 1233 [37]
- gene flow 1038 [32]
- gene–meme coevolution 923 [29]
- general regular Post (GRP) system 1138 [34]
- generalization 939 [30]
- generalized
 - evolutionary algorithm 625 [20]
 - Gaussian probability distribution 445 [13]
 - mass action (GMA) laws 1881 [55]
 - predictive controller (GPC) 418 [12]
 - progress coefficients 771 [25]
- generative topographic map (GTM) algorithm 611 [19]
- GENESIS (spiking neuron simulator) 369 [10]
- genetic
 - circuit 1205 [36]
 - code 1076 [33]
 - divergence 1039 [32]
 - drift 1038 [32]
 - fuzzy systems 969 [30]
 - memory 1218 [36]
 - repair (GR) hypothesis, dominant recombination 773 [25]
- genetic algorithms (GA) 626 [20], 637 [21], 704 [23]
 - limitations 662 [21]
 - niche 963 [30]
 - resource scheduling 665 [21]
 - selective pressure 642 [21]
- genetic programming (GP) 638 [21] 709 [24], 948 [30]
 - Cartesian (CGP) 1669 [50]
 - developmental 1677 [50]
 - Markov chain model 721 [24]
 - open issues 729 [24]
 - real-world applications 719 [24]
 - theory 720 [24]
- genetics-based machine learning (GBML) 937 [30], 1740 [52]
- genome 1076 [33]
- genomic computer 1217 [36]

- genotype 704 [23]
 - environment interaction (G-by-E) 1039 [32]
 - mapping 918 [29]
 genotypic representations 628 [20]
 geometric quantum computing 1502 [44]
 geometrical computation 1969 [58]
 GFP protein 1207 [36]
 Gibbs
 - equilibrium probability distribution 555 [17]
 - measure 264 [8]
 Gibbsian specification 264 [8]
 Gillespie's algorithm 1874 [55]
 gliders
 - cellular automata 1951 [58]
 - collision-based computing 1954 [58]
 - guns (reaction-diffusion processor) 1917 [56]
 global vs. local search balance 910 [29]
 glued tree(s) 1472 [43]
 - problem 1547 [46]
 gradient
 - evolutionary 688 [22]
 - stochastic 688 [22]
 graduated granulation 1922 [57]
 Gram–Schmidt orthogonalization 921 [29]
 grammatical
 - computing 1711 [51]
 - evolution 1711 [51]
 - model (DNA computing) 1148 [34]
 granular computing 1922 [57]
 granule 1925 [57]
 graph 1239 [37]
 - 2-edge colored 1247 [37]
 - 3-coloring 1546 [46]
 - abstract reduction graph 1251 [37]
 - automorphism problem 1457 [43]
 - coloring 1636 [49]
 - double rule (gdr) 1249 [37]
 - empty 1240 [37]
 - negative rule (gnr) 1249 [37]
 - overlap 1246 [37]
 - pointer-component 1252 [37]
 - positive rule (gpr) 1249 [37]
 - reduction 1250 [37]
 - reduction graph 1246 [37]
 - signed 1240 [37]
 greedy randomized adaptive search procedure (GRASP) 1627 [49]
 Greenberg–Hastings model 265 [8], 303 [9]
 Greenberger–Horne–Zeilinger (GHZ) 1425 [42]
 grid 172 [5]
 - composition 184 [5]
 - computation 171 [5]
 - definition 175 [5]
 - rational 173 [5]
 - recursion 186 [5]
 - simulation 177 [5]
 Grover's algorithm (quantum search) 1410 [41], 1448 [42], 1457 [43], 1498 [44]
 growing
 - cell structures (GCS) algorithm 604 [19]
 - grid (GG) algorithm 604 [19]
 - neural gas (GNG) 603 [19]
 guanine (G) 1075 [33]
- ## H
- H form EFS 1139 [34]
- H measure 1100 [33]
- H scheme 1188 [35]
- H systems 1148 [34]
- Hadamard
 - gate 1453 [43], 1500 [44]
 - test 1454 [43]
 - –Walsh gate 1411
- hairpin 1243 [37]
 - completion 1095 [33], 1151 [34]
 - DNA 1308 [38]
 - DNA memory 1308 [38]
 - frame 1095 [33]
 - –free word 1095 [33]
 - languages 1149 [34]
 - reduction 1095 [33], 1151 [34]
 - scattered 1095 [33]
 - structure 1149 [34]
- Hall effect 1503 [44]
- Hall's marriage theorem 282 [8]
- Hamiltonian 1383 [41], 1405 [41], 1461 [43], 1503 [44]
 - mechanics 1383 [41]
 - path problem (HPP) 1084 [33], 1147 [34], 1277 [37], 1326 [39], 1374 [40]
- hammerhead ribozyme 1164 [34]
- Hamming distance 1100 [33]
- hardware
 - adaptive 1692 [50]
 - bioinspired 1698 [50]
 - evolvable (EHW) 1657 [50]
 - self-repairing 1692 [50]
- hash function 1523 [45]
- Hebbian learning 357 [10]
 - competitive (CHL) 601 [19]
 - prescription 554 [17]
- Heisenberg uncertainty
 - principle 1419 [42], 1533 [45]
 - relation 1393 [41]
- helical DNA 1076 [33]
- heritability 1039 [32]
- Hermitian inner product 1384 [41]
- heteroclinic trajectory 553 [17]
- heteroscedastic kernels 616 [19]
- heuristic programming 702 [23]
- hidden
 - lattice problem 1453 [43]
 - subgroup problem 1453 [43], 1547 [46]

- hierarchical molecular memory 1296 [38]
 hierarchy, artificial life 1809 [53]
 Hilbert space 1382 [41]
 hitting time theorem 1465 [43]
 Hodgkin–Huxley model 343 [10], 536 [17]
 holonomic quantum computing 1502 [44]
 homeostasis 1590 [47]
 HOMFLYPT (polynomial) 1568 [46]
 honest-verifier quantum statistical zero-knowledge
 (HVQSZK) 1539 [45]
 honeybees
 – bee-inspired algorithms 1610 [48]
 – foraging 1610 [48]
 – waggle dance 1610 [48]
 Hopfield network 351 [10], 358 [10]
 HPP (Hardy–Pomeau–de Pazzis) rule 307 [9]
 HT system 1153 [34]
 hybridization 1077 [33], 1308 [38], 1323 [39]
 – reaction 1324 [39]
 – undesired 1090 [33]
 hybridization (search) 892 [28], 911 [29]
 hydrogen 1509 [44]
 – bond 1076 [33]
 hyperbolic equilibrium solution 752 [25]
 hyperellipsoid 689 [22]
 hyperheuristics 923 [29], 1774 [52]
 – bin packing 1778 [52]
 – satisfiability (SAT) 1779 [52]
 hyperplane 480 [15]
 hypervolume indicator 883 [28]
- I**
- ideal
 – point 878 [28]
 – team 1000 [31]
 image
 – compression (evolvable hardware) 1695 [50]
 – filter, evolution 1682 [50]
 – quality assessment (IQA) 378 [11]
 immune
 – algorithm 1578 [47]
 – -inspired algorithms 1576 [47]
 – network, algorithm 1581 [47]
 – network, theory 1579 [47]
 – response 1577 [47]
 – system 1577 [47]
 – system, adaptive 1577 [47]
 – system, artificial (AIS) 1575 [47]
 – system, human 1577 [47]
 – system, innate 1577 [47]
 – system, simulation 1576 [47]
 immunology 1576 [47]
 incomparability 882 [28]
 incremental evolution (evolvable hardware) 1668 [50]
 independent component analysis (ICA) 435 [13]
 indiscernibility 1925 [57]
- individual 675 [22]
 induced subgraph 1240 [37]
 inducer molecule 1207 [36]
 induction 938 [30]
 infinite population model 656 [21], 1019 [31]
 inflammatory response 1577 [47]
 Infomax 441 [13]
 information
 – medium 1204 [36]
 – molecule 1206 [36]
 – rate 1228 [36]
 informative dimensions 1026 [31]
 inhibitory synapse 544 [17]
 initial distribution 850 [27]
 inner product 1384 [41]
 innovation 1764 [52]
 – gear train design 1766 [52]
 – welded beam design 1768 [52]
 input probability density 595 [19]
 insertion–deletion (ins–del) 1148 [34], 1185 [35],
 1196 [35]
 instrument optimization 1761 [52]
 integer factorization 1455 [43]
 integrate-and-fire
 – model 538 [17]
 – neuron 346 [10]
 intelligent crossover 921 [29]
 interacting stochastic automata 1883 [55]
 interaction 993 [31], 1005 [31]
 – potential 269 [8]
 – technology 1945 [57]
 interactive
 – domain 990 [31]
 – evolution design 1781 [52]
 interference (quantum information processing)
 1392 [41]
 internal eliminated sequences (IES) 1236 [37]
 internal representation 632 [20]
 interspike interval 551 [17]
 intertwined spirals 718 [24]
 interval 1245 [37]
 intraburst firing pattern 550 [17]
 intramolecular model, simple 1254 [37]
 intramolecular operation
 – dlad 1243 [37]
 – dlad, simple 1256 [37]
 – hi 1243 [37]
 – hi, simple 1256 [37]
 – ld 1243 [37]
 – ld, boundary 1244 [37]
 – ld, simple 1244 [37] 1256 [37]
 intrinsic
 – evolution 1681 [50]
 – universality 190 [6], 252 [7]
 intron 1076 [33]
 invariants 1261 [37]

- inversion 1239 [37]
 involution 1132 [34]
 Ising model 262 [8]
 isotropy 314 [9]
 iterated maps 747 [25], 1991 [59]
 iterated PA-matching operation 1155 [34]
 Izhikevich neuron 345 [10]
- J**
- Java containers, evolutionary algorithms 1791 [52]
 Jaynes–Cummings model 1503 [44]
 job-shop schedules 632 [20]
 Johnson graph 1466 [43]
 joint systems 1397 [41]
 Jones polynomial 1559 [46]
 Josephson
 - junction 1514 [44]
 - tunnelling 1514 [44]
 judgement technology 1944 [57]
- K**
- κ -calculus 1845 [54]
 Kauffman diagram 1563 [46]
 Kendall's τ 500 [15]
 kernel
 - function 487 [15]
 - trick 487 [15]
 KInfer (parameter estimation software) 1856 [54]
 knapsack problem 872 [28]
 knowledge
 - inference 1853 [54]
 - technology 1944 [57]
 Kochen–Specker theorem 1414 [42]
 Kohonen model 586 [19]
 Kraus representation 1403 [41]
 Kronecker product 1385 [41]
 Kuhn–Tucker conditions 484 [15]
- L**
- labelled transition system (LTS) 1867 [55]
 Lagrange
 - function 491 [15]
 - multiplier 491 [15]
 Lagrangian relaxation 917 [29]
 Lamarckian evolution 906 [29], 945 [30]
 Lampert signature 1536 [45]
 Landau–Zener transition 1514 [44]
 lane, in gel 1081 [33]
 Langton ant 293 [9]
 language (DNA computing)
 - bond-free 1098 [33]
 - solid 1096 [33]
 - strictly θ 1096 [33]
 - θ -compliant 1096 [33]
 - θ -free 1096 [33]
 – θ - k -code 1096 [33]
 – θ -non-overlapping 1096 [33]
 – θ -overhang-free 1096 [33]
 – θ -sticky-free 1096 [33]
 Laplacian 1470 [43]
 Larmor frequency 1508 [44]
 Latin hypercubes sampling 921 [29]
 lattice
 - BGK (LBGK) model 315 [9]
 - Boltzmann (LB) execution loop 324 [9]
 - Boltzmann (LB) models 287 [9], 311 [9]
 - D2Q9 314 [9]
 - D3Q19 315 [9]
 - DdQq 313 [9]
 - gas automata (LGA) 307 [9]
 - hexagonal 312 [9]
 - square 312 [9]
 - topologies 321 [9]
 leaky integrate-and-fire neuron 344 [10]
 learner (coevolution) 998 [31]
 learning
 - genetics-based machine (GBML) 937 [30]
 - parameter 744 [25]
 learning classifier systems (LCS) 908 [29], 947 [30]
 - aircraft maneuvers 1741 [52]
 - XCS 1778 [52]
 legal string 1245 [37]
 Leggett–Garg inequalities [42] 1444
 length-encoding technique 1204 [36]
 LETOR (learning to rank, benchmark dataset) 503 [15]
 level sets 1040 [32]
 Levenshtein distance 1101 [33]
 l-histidine 1509 [44]
 ligase enzyme 1078 [33], 1324 [39]
 ligation 1078 [33]
 limit distribution 856 [27]
 LIN (language) 1132 [34]
 Lindenmayer systems (L-systems) 2017 [59]
 linear
 - classifier 480 [15]
 - component 1248 [37]
 - generative model 447 [13]
 - mapping 1386 [41]
 - programming (LP) 497 [15]
 - ranking function 493 [15]
 liposomal
 - assembly 1225 [36]
 - dissociation 1225 [36]
 liquid
 - crystal systems 2012 [59]
 - state machines 362 [10]
 little dynamics 557 [17]
 loading
 - parameter 557 [17]
 - problem 354 [10]

- local
 - equilibrium 319 [9]
 - field 555 [17]
 - measurement 1417 [42]
 - optima 916 [29]
 - quality change 758 [25]
 - realism 1422 [42]
 - search 818 [26]
 - search pseudocode 1625 [49]
 - local Hamiltonian (LH) 1549 [46]
 - eigenvalue distribution (LHED) 1555 [46]
 - eigenvalue sampling (LHES) 1550 [46]
 - minimum eigenvalue (LHME) 1549 [46]
 - local unitary
 - average eigenvalue 1568 [46]
 - unitary phase sampling (LUPS) 1550 [46]
 - logarithmic adaptation response 760 [25]
 - logic, reaction–diffusion application 1904 [56]
 - logistic map, dynamical system 1991 [59]
 - logistics 1771 [52]
 - long-term
 - depression (LTD) 349 [10]
 - potentiation (LTP) 349 [10]
 - loop 143 [37]
 - Lorenz attractor 2000 [59]
 - loss function 498 [15]
 - Lotka–Volterra dynamics 552 [17]
 - LUES (BQP algorithm) 1551 [46]
 - Lyapunov function 284 [8], 854 [27]
 - lymphocytes 1577 [47]
 - Lyndon–Schützenberger equation 1122 [33]
- M**
- machine learning 632 [20], 937 [30]
 - coevolutionary 947 [30]
 - genetic cooperative-competitive learning 945 [30]
 - genetics-based (GBML) 937 [30], 1740 [52]
 - iterative rule learning 944 [30]
 - metalearning 941 [30]
 - Michigan evolution 941 [30]
 - online evolutionary 946 [30]
 - Pittsburgh evolution 941 [30]
 - theory 968 [30]
 - macronuclear destined segments (MDS) 1236 [37]
 - magnetic beads 1082 [33], 1294 [38]
 - margin 480 [15]
 - Markov chain 721 [24]
 - limit distribution 856 [27]
 - limit theorems 856 [27]
 - model, finite population 658 [21]
 - optimal state 856 [27]
 - optimal state set 856 [27]
 - stationary distribution 856 [27]
 - trace 1564 [46]
 - mass conservation 315 [9]
 - master equation 555 [17]
- mating restriction 1051 [32]
 - matrix
 - cogredient 855 [27]
 - column allowable 855 [27]
 - decomposition, Cholesky 696 [22]
 - decomposition, spectral 694 [22]
 - diagonal-positive 855 [27]
 - ergodic 1463 [43]
 - Hessian 689 [22]
 - irreducible 855 [27]
 - non-negative 855 [27]
 - orthogonal 681 [22]
 - permutation 855 [27]
 - positive 855 [27]
 - positive definite (p.d.) 681 [22]
 - positive semidefinite (p.s.d.) 681 [22]
 - primitive 855 [27]
 - reducible 855 [27]
 - regular 681 [22]
 - singular 681 [22]
 - stable 855 [27]
 - stochastic 855 [27], 1461 [43]
 - symmetric 681 [22]
 - maximization, expected utility 1001 [31]
 - maximum
 - conductance 536 [17]
 - peak ratio (MPR) 1058 [32]
 - storage capacity 558 [17]
 - McCulloch–Pitts model 539 [17]
 - mean
 - average precision (MAP) 500 [15]
 - opinion score (MOS) 379 [11]
 - measure
 - -many quantum automaton 1407 [41]
 - -once quantum automaton 1407 [41]
 - medial graph 1567 [46]
 - melting 1077 [33]
 - temperature 1078 [33]
 - membrane
 - computing 1355 [40]
 - P system 1368 [40]
 - potential 339 [10]
 - structure 1356 [40]
 - systems 1876 [55]
 - meme 907 [29]
 - memetic algorithm 905 [29]
 - approximate fitness function 920 [29]
 - approximate methods hybridization 916 [29]
 - continuous optimization 921 [29]
 - diversity preservation 919 [29]
 - exact methods hybridization 916 [29]
 - multimeme methods 921 [29]
 - pattern 908 [29]
 - pattern language 926 [29]
 - refinement strategy pattern 914 [29]
 - scheduler 914 [29]

- self-generating methods 923 [29]
 - surrogate objective function 919 [29]
 - template pattern 911 [29]
 - memetics 945 [30]
 - memory
 - DNA 1281 [38]
 - mechanisms 1005 [31]
 - SRAM 1661 [50]
 - Mercer's theorem 487 [15]
 - merge edges 1251 [37]
 - message authentication (quantum cryptography) 1524 [45]
 - metaheuristic 628 [20], 675 [22]
 - evolutionary algorithm (EA) 675 [22]
 - evolutionary strategy (ES) 675 [22]
 - instantiation 676 [22]
 - parameterization 676 [22]
 - metamodels 920 [29]
 - metric
 - full-reference (FR) 379 [11]
 - no-reference (NR) 379 [11]
 - reduced-reference (RR) 379 [11]
 - Metropolis algorithm 1461 [43], 1628 [49]
 - microarray 1763 [52]
 - microdynamics 309 [9]
 - microelectromechanical systems (MEMS) 1681 [50]
 - micro-rules (social insects) 1606 [48]
 - microstate (dynamical systems) 1982 [59]
 - mildly context-sensitive languages 1152 [34]
 - minimal
 - element (order theory) 877 [28]
 - interpretation (quantum mechanics) 1392 [41]
 - set (order theory) 878 [28]
 - mirror image 1131 [34]
 - mixed state 1389 [41]
 - mixing 1009 [31]
 - time 1473 [43]
 - mobile localizations 1952 [58]
 - model
 - analysis 1019 [31]
 - calibration 1853 [54]
 - elementary 1255 [37]
 - intermolecular 1240 [37]
 - intramolecular 1242 [37]
 - parallel 1255 [37]
 - simple 1255 [37]
 - specification (systems biology) 1841 [54]
 - modeling
 - languages (systems biology) 1841 [54]
 - Rosen 1837 [54]
 - molecular
 - addressing 1310 [38]
 - beacon 1301 [38]
 - biology 1074 [33]
 - communication 1219 [36]
 - computing 1129 [34]
 - container 1220 [36]
 - graphs 1155 [34]
 - machine 1133 [34]
 - memory 1282 [38], 1305 [38]
 - propagation system 1220 [36]
 - momentum conservation 315 [9]
 - monomer 1075 [33]
 - monotonicity 1012 [31]
 - Monte Carlo methods 1461 [43]
 - morphogenesis 634 [20]
 - Morris–Lecar model 538 [17]
 - motor protein 1222 [36]
 - Mott insulator 1506 [44]
 - multiclass classification 480 [15]
 - multicomponent flows 328 [9]
 - multiobjective optimization 633 [20], 874 [28], 1003 [31], 1760 [52]
 - closed-loop 1758 [52]
 - multiobjectivization 898 [28]
 - nondominated sorting genetic algorithm (NSGA-II) 887 [28], 1766 [52]
 - multiperiod (multistep) forecasting 471 [14]
 - multiple
 - constant multiplier (MCM) 1674 [50]
 - criteria decision making (MCDM) 873 [28]
 - multiplexer 718 [24]
 - multiplexing 1528 [45]
 - multiplication 160 [5], 184 [5]
 - real-time 163 [5]
 - multipopulation EA 633 [20]
 - multiset 1358 [40]
 - rewriting rules 1359 [40]
 - multiscale geometric analysis (MGA) 383 [11]
 - multislicing models 1139 [34]
 - mutant eukaryotic cell 1225 [36]
 - mutation 712 [24], 858 [27]
 - bit-flipping 677 [22]
 - integer 679 [22]
 - strength 742 [25]
- N**
- nadir point 878 [28]
 - NAND tree 1475 [43]
 - Nash equilibrium 1002 [31]
 - natural gradient algorithm 443 [13]
 - Navier–Stokes equations 316 [9], 326 [9]
 - negative
 - entropy (negentropy) 442 [13]
 - letter (pointer) 1245 [37]
 - rule, graphs 1249 [37]
 - rule, strings 1248 [37]
 - selection algorithm 1582 [47]
 - vertex 1240 [37]
 - neighborhood 1240 [37], 1625 [49]
 - Nelder–Mead simplex 921 [29]
 - nested primer molecular memory (NPMM) 1305 [38]

- network intrusion detection 1583 [47]
 - networks (graphs, artificial life) 1821 [53]
 - neural
 - computation 335 [10]
 - gas (NG) 601 [19]
 - signature 551 [17]
 - neural networks 336 [10], 565 [18]
 - architectures 569 [18]
 - bioinformatics 565 [18]
 - construction 511 [16]
 - evolving 953 [30]
 - modeling biological 533 [17]
 - radial basis function (RBF) 572 [18]
 - recurrent (RNN) 571 [18]
 - single-hidden-layer feedforward (SLFN) 507 [16]
 - time-series forecasting 461 [14]
 - topology 560 [17]
 - neurofuzzy network 401 [12], 407 [12]
 - neuromodulator 550 [17]
 - neuron
 - integrate-and-fire 346 [10]
 - Izhikevich 345 [10]
 - leaky integrate-and-fire 344 [10]
 - quadratic integrate-and-fire 345 [10]
 - spiking 335 [10]
 - NEURON (spiking neuron simulator) 369 [10]
 - neurotransmitter 541 [17]
 - acetylcholine 544 [17]
 - aspartate 544 [17]
 - dopamine 544 [17]
 - gamma-aminobutyric acid (GABA) 544 [17]
 - glutamate 544 [17]
 - glycine 544 [17]
 - vesicle 541 [17]
 - Newtonian
 - equation 1383 [41]
 - mechanics 1382 [41]
 - niche 1036 [32]
 - capacity 1047 [32]
 - hosting capacity 1038 [32], 1053 [32]
 - niching 1035 [32]
 - clearing 1047 [32]
 - clustering 1048 [32]
 - conceptual designs 1036 [32]
 - crowding 1048 [32]
 - ES techniques 1051 [32]
 - experimental methodology 1055 [32]
 - GA techniques 1045 [32]
 - island model 1050 [32]
 - Mahalanobis distance 1063 [32]
 - mating schemes 1051 [32]
 - mission statement 1045 [32]
 - niche-radius 1046 [32], 1060 [32]
 - sequential 1049 [32]
 - success rate 1058 [32]
 - Nissen model of expectation 1649 [49]
 - nitrogen vacancies in diamond 1511 [44]
 - n-mer 1075 [33]
 - N-methyl-D-aspartate (NMDA) synaptic receptor 544 [17]
 - no free lunch (NFL) theory 725 [24]
 - Noether's theorem 284 [8]
 - noise strength 806 [25]
 - nonclassical computation 1979 [59]
 - noncoding DNA 1076 [33]
 - non-contextuality 1441 [42]
 - nondeterministic finite automaton (NFA) 1142 [34]
 - NFA Pot 1145 [34]
 - nondominated
 - front 1003 [31]
 - sorting 887 [28]
 - sorting genetic algorithm (NSGA-II) 887 [28], 1766 [52]
 - nonhyperbolic equilibrium solution 752 [25]
 - noninteractive quantum perfect zero-knowledge (NIQPZK) 1540 [45]
 - nonlinear
 - autoregressive moving average with exogenous inputs (NARMAX) 403 [12]
 - process modelling and control 401 [12]
 - nonlocal box 1440 [42]
 - non-malleability 1526 [45]
 - normal matrix 1389 [41]
 - normalized discounted cumulative gain (NDCG) 500 [15]
 - NP (nondeterministic polynomial-time, computational complexity theory) 1132 [34], 1525 [45]
 - complete 1546 [46]
 - NSPACE(f(n)) (computational complexity theory) 1132 [34]
 - nuclear magnetic resonance (NMR) 1508 [44]
 - logic gates 2012 [59]
 - nuclear spin 1507 [44]
 - nucleoside 1075 [33]
 - nucleotide (nt) 1075 [33]
 - nucleus
 - macro 1235 [37]
 - micro 1235 [37]
 - micro, scrambled gene 1275 [37]
- ## O
- objective
 - conflict 896 [28]
 - function 675 [22], 874 [28]
 - space 874 [28]
 - vector 874 [28]
 - oblivious transfer 1528 [45]
 - quantum 1537 [45]
 - observable 1392 [41]
 - local 270 [8]
 - observable sticker system 1113 [33]
 - simple regular 1113 [33]

- occurrence, letter 1239 [37]
 offspring 675 [22]
 oligonucleotide 1075 [33]
 - replacement 1165 [34]
 one-time pad 1524 [45]
 one-way
 - function 1525 [45]
 - permutation 1527 [45]
 - permutation, quantum 1529 [45]
 - quantum computing 1502 [44]
 one-wayness 1526 [45]
 open
 - dynamical systems 2005 [59]
 - -ended coevolution 1028 [31]
 - issues (coevolution) 1025 [31]
 operational transconductance amplifier (OTA) 1665 [50]
 operator disruption 1042 [32]
 optical
 - lattices 1506 [44]
 - qubits 1516 [44]
 optimization
 - tasks 938 [30]
 - time 819 [26]
 optimizer tracking 807 [25]
 orbit (dynamical system) 1991 [59]
 organic diversity 1038 [32]
 orientation 1075 [33]
 orthogonal complement 1389 [41]
 Ott–Grebogi–Yorke (OGY) control laws 2012 [59]
 overhang 1078 [33]
 overlap 1245 [37]
 - function 557 [17]
 - graph 1246 [37]
- P**
- P (polynomial-time, computational complexity theory)
 1132 [34], 1525 [45], 1546 [46]
- P systems 1355 [40], 2018 [59]
 - active membranes 1365 [40]
 - cell-like 1361 [40]
 - (mem)brane 1368 [40]
 - neural-like 1370 [40]
 - population 1370 [40]
 - spiking neural 1371 [40]
 - string objects 1367 [40]
 - sympport/antiport 1363 [40]
 - tissue-like 1369 [40]
- pad mismatch 1341 [39]
 pairwise
 - coupling method 480 [15]
 - difference 494 [15]
- palladium processor 1902 [56]
 parabolic ridge 762 [25]
 parallel
 - associative (PA) matching 1153 [34]
 - associative memory (PAM) model 1148 [34]
- evolutionary algorithm (EA) 633 [20]
 – random-access model (PRAM) 1148 [34]
 – writing and erasing 1309 [38]
- parallelism 630 [20]
 parameter optimization 626 [20]
 parents 675 [22]
 - selection 629 [20]
- Pareto
 - coevolution 998 [31]
 - covering 1003 [31]
 - dominance 875 [28]
 - optimal 634 [20]
 - optimal front 877 [28]
 - optimal minimal equivalent set 1003 [31]
 - optimal set 877 [28], 1003 [31]
 - optimality 877 [28]
 - set approximation 882 [28]
- partial
 - differential equations (dynamical systems) 2002 [59]
 - trace 1400 [41]
- particle 260 [8], 281 [8]
 - distribution 50 [2]
 - flow 281 [8]
 - swarm optimization (PSO) 908 [29], 1616 [48]
 - weight function 48 [2]
- partition function 1477 [43]
 pathogen 1577 [47]
 pattern 907 [29]
 - language 907 [29]
 - recognition (immune systems) 1578 [47]
- Paul trap 1504 [44]
 Pauli spin matrix 1395 [41]
 Peano curve 593 [19]
 pebble redistribution game 281 [8]
 Pell's equation 1547 [46]
 Penning trap 1504 [44]
 perceptron 336 [10], 351 [10]
 perfect
 - secrecy 1524 [45]
 - zero-knowledge (PZK) 1540 [45]
- performance
 - coevolutionary algorithms 1013 [31]
 - evolutionary multiobjective optimization (EMO) 892 [28]
 - image quality assessment (IQA) 391 [11]
 - single-hidden-layer feedforward neural network (SLFN) 529 [16]
- performance evaluation process algebra (PEPA) 1873 [55]
 - Bio-PEPA (biochemical aspects) 1873 [55]
- PH (polynomial hierarchy) 1569 [46]
 phagocytes 1577 [47]
 phase 345 [10]
 - estimation (PE) 1550 [46]
 - gap 1463 [43]
 - sampling 1549 [46]

- phenotype 702 [23]
 - mapping 918 [29]
- phenotypic representations 628 [20]
- pheromone trails 1602 [48]
- phrase-structure grammar 1131 [34]
- π -/pi-calculus 1874 [55]
 - stochastic 1841 [54]
 - systems biology 1852 [54]
- plaintext 1523 [45]
- plasticity 1039 [32]
- plat closure 1560 [46]
- pluggable set 174 [5]
- pluripotential solver cell 929 [29]
- p-median problem 1626 [49]
- Poincaré sphere 1396 [41]
- pointer 1236 [37]
 - component graph 1252 [37]
 - repeat, direct 1243 [37]
 - repeat, inverted 1243 [37]
 - role of 1265 [37]
- polar
 - decomposition 1402 [41]
 - representation 1387 [41]
- polyacrylamide gel 1079 [33]
- polyautomata 1887 [55]
- polychronization 356 [10]
- polymer 1075 [33]
- polymerase
 - chain reaction (PCR) 1083 [33], 1296 [38], 1322 [39]
 - enzyme 1080 [33], 1324 [39]
- polynucleotide 1075 [33]
- population 675 [22]
 - diversity 917 [29], 1041 [32]
 - initialization 711 [24]
 - size 630 [20]
- positive
 - letter (pointer) 1245 [37]
 - mapping 1387 [41], 1402 [41]
 - rule, graphs 1249 [37]
 - rule, strings 1248 [37]
 - vertex 1240 [37]
- positive semidefinite (p.s.d.) 1387 [41]
- post-stall technology (PST), aircraft maneuvers 1740 [52]
- Post system 1138 [34]
- postsynaptic
 - potential (PSP) 339 [10]
 - receptor 544 [17]
- potential solution 995 [31]
- preference
 - articulation 891 [28]
 - relation 875 [28]
- preselection schemes 1048 [32]
- pressure 316 [9]
 - boundary condition 325 [9]
- primal problem 494 [15]
- prime numbers sieve 168 [5]
- probability distribution 1550 [46]
- probe 1087 [33]
- problem difficulty 724 [24]
- process calculi 1864 [55]
 - comparison, ordinary differential equations 1881 [55]
 - general 1866 [55]
 - example, ATP/ADP 1870 [55]
 - example, glycolysis 1870 [55]
 - model checking 1888 [55]
 - modelling biological systems 1867 [55]
 - parallel composition 1868 [55]
 - quantitative information 1872 [55]
 - systems biology 1868 [55]
 - universality 1888 [55]
- production systems 959 [30]
- programmable biomolecular computation 1320 [39]
- progress rate 758 [25]
- projection
 - gene assembly, mathematics 1239 [37]
 - quantum information processing, mathematics 1389 [41]
 - valued measure (quantum information processing, mathematics) 1393 [41]
- prokaryote 1235 [37]
- promise problem 1548 [46]
 - PromiseBQP (polynomial time, quantum computer) 1548 [46]
 - PromiseQMA (quantum Merlin–Arthur) 1550 [46]
- propagation 308 [9], 324 [9]
 - environment 1220 [36]
- protein(s) 567 [18], 1076 [33]
 - bioinformatics 569 [18]
 - structure 568 [18]
 - synthesis mechanism 1205 [36]
 - synthesis system 1213 [36]
- protocol
 - B92 1531 [45]
 - BB84 1531 [45]
 - BBM92 1532 [45]
 - E91 1532 [45]
- protozoa 1235 [37]
 - ciliated 1235 [37]
- pseudoknot 1095 [33]
- pseudopalindrome 1078 [33]
- PSPACE (computational complexity theory) 1528 [45], 1539 [45]
- public-key
 - cryptography 1455 [43]
 - encryption 1522 [45]
 - encryption, quantum 1532 [45]
- pure state 1385 [41], 1388 [41], 1396 [41]
- purine 1075 [33]
- pyloric central pattern generator (CPG) 549 [17]

PyNN (spiking neuron simulator) 369 [10]
pyrimidine 1075 [33]

Q

quadratic integrate-and-fire neuron 345 [10]
quadratically signed weight enumerator 1568 [46]
quality
– gain 758 [25]
– indicator 882 [28]
quantitative variation 1039 [32]
quantum
– automaton, measure-many 1407 [41]
– automaton, measure-once 1407 [41]
– bit (qubit) 1394 [41]
– circuit 1409 [41]
– commitment 1537 [45]
– computing 1381 [41]
– control 1060 [32]
– cryptography 1521 [45]
– digital signature 1536 [45]
– dot 1512 [44]
– electrodynamical (QED) 1503 [44], 1516 [44]
– finite automaton 1407 [41]
– Fourier transform (QFT) 1497 [44], 1453 [43],
 1557 [46]
– information processing 1381 [41]
– key distribution 1530 [45]
– mechanics 1413 [42]
– non-demolishing (QND) 1503 [44]
– parallelism 1497 [44]
– public-key encryption 1532 [45]
– random access codes 1445 [42]
– reduction of communication complexity 1433 [42]
– state 1414 [42]
– statistical zero-knowledge (QSZK) 1540 [45]
– system, closed 1404 [41]
– system, open 1404 [41]
– teleportation 1505 [44]
– Turing machine 1407 [41]
– walk searching 1550 [46]
– walks 1461 [43]
– zero-knowledge (QZK) 1539 [45]

quantum computing 1381 [41]
– adiabatic 1502 [44]
– algorithms 1451 [43]
– all-optical 1516 [44]
– anyonic 1502 [44]
– cluster state 1502 [44]
– ensemble 1508 [44]
– geometric 1502 [44]
– holonomic 1502 [44]
– large-scale 1493 [44]
– one-way 1502 [44]

quantum information processing 1381 [41]
– compound systems 1397 [41]
– Hilbert space formalism 1382 [41]

– mathematics 1381 [41]
– quantum states 1392 [41]
– spectral representation 1389 [41]
– subsystem states 1398 [41]

quantum protocol scheme

- GC01 1536 [45]
- KKNY05 1534 [45]
- OTU00 1533 [45]

quasiparticle tunnelling 1514 [44]

qubits

- charge 1514 [44]
- flux 1515 [44]
- phase 1516 [44]
- superconducting 1513 [44]

Quine programs 1816 [53]

quorum sensing 1218 [36]

R

R indicator family 884 [28]
Rabi oscillation 1503 [44], 1514 [44]

radial

- basis function (RBF) neural network 351 [10],
 572 [18]
- progress rate 776 [25]

Raman setup 1505 [44]

Ramsay interferometry 1515 [44]

random Boolean networks (RBN) 1989 [59]
– models of genetic regulatory networks 1988 [59]

random descent 1625 [49]

random walk

- classical 1461 [43]
- continuous time quantum 1547 [46]
- nodes 510 [16]
- quantum 1461 [43]
- quantum continuous time 1461 [43]
- quantum discrete time 1466 [43]

rank

- order coding 341 [10]
- -proportional selection 629 [20]

ranking support vector machine (SVM) 495 [15]

rate coding 340 [10]

rational relation 1101 [33]

RE (language) 1132 [34]

reaction-diffusion

- application, computational geometry 1904 [56]
- computing 1898 [56], 2003 [59]
- logic application 1904 [56]
- model 318 [9]
- processors 1901 [56]
- processors, computationally universal 1914 [56]
- processors, coupled with actuators 1911 [56]
- processors, coupled with manipulators 1911 [56]
- robotics application 1904 [56]

ready-releasable pool 541 [17]

real-time 163 [5]

reality edge 1247 [37]

- recombination 858 [27]
 - compound 677 [22]
 - composite 677 [22]
 - discrete 680 [22]
 - dominant 680 [22]
 - intermediary 680 [22]
 - reconfigurable
 - devices 1658 [50]
 - technology 1659 [50]
 - recurrent
 - neural network (RNN) 571 [18]
 - set 856 [27]
 - state 856 [27]
 - recursion 186 [5]
 - Redi (reaction-diffusion simulator) 1857 [54]
 - reduction
 - graph 1246 [37]
 - graph, abstract 1251 [37]
 - parallel 1258 [37]
 - simple 1256 [37]
 - simple, parallel 1256 [37]
 - simple, successful 1256 [37]
 - simple, unsuccessful 1256 [37]
 - strings 1249 [37]
 - refinement 881 [28]
 - refractory period 556 [17]
 - REG (language) 1132 [34]
 - regression tasks 949 [30]
 - relation graph 876 [28]
 - relative solvent accessibility (RSA) 578 [18]
 - relaxation time 315 [9]
 - reliability polynomial 1558 [46]
 - renaturation 1077 [33]
 - reporter gene 1213 [36]
 - representation(s) 710 [24]
 - binary 645 [21]
 - Gray code 646 [21]
 - real-valued 645 [21]
 - of problem 676 [22]
 - representer theorem 498 [15]
 - reproductive operators 628 [20], 634 [20]
 - research
 - paradigm 907 [29]
 - program 907 [29]
 - reservoir computing 361 [10]
 - Reservoir Computing Toolbox 369 [10]
 - restriction
 - endonuclease 1078 [33]
 - enzyme 1078 [33], 1324 [39]
 - site 1078 [33]
 - reversibility
 - cellular automata 4 [1], 232 [7], 264 [8]
 - quantum information processing 1409 [41]
 - reversible
 - intrinsic universality 226 [6], 252 [7]
 - logic element 247 [7]
 - rewriting systems 2016 [59]
 - artificial life 1818 [53]
 - Reynolds' rules 1611 [48]
 - ribonucleic acid (RNA) 1076 [33]
 - polymerase 1207 [36]
 - ribose 1076 [33]
 - rigid motion 1401 [41]
 - robotics
 - Belousov-Zhabotinsky (BZ) processor 1909 [56]
 - reaction-diffusion application 1904 [56]
 - robustness 1020 [31]
 - Rössler attractor 1999 [59]
 - rotary element 252 [7]
 - rotate-and-simulate method 1138 [34], 1190 [35]
 - rotation 1401 [41]
 - rough
 - inclusion relation 1927 [57]
 - membership function 1928 [57]
 - mereology 1928 [57]
 - sets 1924 [57]
 - rough-fuzzy
 - computing 1921 [57]
 - hybridization 1940 [57]
 - RSA (Rivest-Shamir-Adleman) cryptosystem 1522 [45]
 - runtime analysis 1021 [31]
 - Rydberg state 1505 [44]
- S**
- S. nova, actin I 1251 [37]
 - salting route optimization 1772 [52]
 - sand pile model 266 [8]
 - satisfiability (SAT) 1177 [34]
 - bin packing 1779 [52]
 - problem 1546 [46]
 - superheuristics 1779 [52]
 - sBrane (simplified brane calculus) 1878 [55]
 - scaffold strand 1336 [39]
 - scaffolding 1322 [39]
 - scalarizing function 880 [28]
 - schema
 - theorem 653 [21]
 - theories 721 [24]
 - Schmidt decomposition 1415 [42]
 - Schrödinger's equation 1385 [41], 1405 [41], 1506 [44]
 - search
 - algorithm design 884 [28]
 - space 874 [28]
 - space structure 1026 [31]
 - stagnation 916 [29]
 - secondary structure 1090 [33]
 - selection 639 [21], 711 [24], 849 [27]
 - fitness proportional selection 641 [21]
 - pressure 1042 [32]
 - noise 1042 [32]
 - tournament 640 [21]
 - tournament, unbiased 641 [21]

- selective
 - pressure (genetic algorithms) 642 [21]
 - writing, erasing (DNA memory) 1309 [38]
- self-adaptation 704 [23], 865 [27]
 - response (SAR) 759 [25]
- self-adjoint 1387 [41], 1416 [42]
- self-assembly 1320 [39]
 - computation 1142 [34]
- self-generating strategies 923 [29]
- self-organizing maps (SOM) 585 [19]
 - algorithm 589 [19]
- self-reconfigurable analog array (SRAA) 1693 [50]
- self-repairing hardware 1692 [50]
- semantic
 - complexity class 1548 [46]
 - security 1526 [45]
- sequence alignment 1314 [38]
- sequential minimal optimization (SMO) 497 [15]
- set
 - -oriented problem transformation 882 [28]
 - preference relation 882 [28]
 - problem 882 [28]
- sets, equivalent 1262 [37]
- sexual reproduction 628 [20]
- SHA-1 (secure hashing algorithm) 1525 [45]
- shadowing lemma 1997 [59]
- Shannon's theorem 1524 [45]
- shape space 1579 [47]
- sharp ridge 762 [25]
- Shor's algorithm 1455 [43], 1522 [45]
- short-term
 - depression (STD) 349 [10]
 - potentiation (STP) 349 [10]
- shuffle, on trajectory 1098 [33]
- signal(s) 168 [5], 222 [6]
 - CA-constructible 81 [3]
 - cellular automata 1965 [58]
 - computational systems 1969 [58]
 - creation 175 [5]
 - definition 80 [3], 175 [5]
 - emission 81 [3]
 - exponential slope 97 [3]
 - infinite families 96 [3]
 - interaction 81 [3], 175 [5]
 - parabolic slope 97 [3]
 - slope 81 [3]
 - slope limitation 106 [3]
 - speed 81 [3]
 - transduction system 1227 [36]
 - translation 81 [3]
- signed
 - graph 1240 [37]
 - string 1239 [37]
- silicon nuclear magnetic resonance (NMR) 1510 [44]
- similarity relation 1101 [33]
- simple complement 1081 [33]
- simulated annealing (SA)
 - acceptance probability 1628 [49]
 - applications 1648 [49]
 - cooling schedule 1632 [49]
 - introduction 1624 [49]
 - overview 1628 [49]
 - pseudocode 1629 [49]
 - starting temperature 1632 [49]
- simultaneous maximization of all outcomes 1001 [31]
- single-hidden-layer feedforward neural network (SLFN) 507 [16]
- single relaxation time 315 [9]
- singlet state 1415 [42]
- singular value 1402 [41]
- skeletonization (reaction-diffusion computing) 1904 [56]
- slack variable 484 [15]
- Snazer (visualization tool) 1857 [54]
- soap film (computation system) 1899 [56]
- social insects 1601 [48]
- soft margin
 - parameter 494 [15]
 - support vector machine (SVM) 496 [15]
- software
 - design (evolutionary algorithms) 1790 [52]
 - development (memetic algorithms) 928 [29]
 - testing (evolutionary algorithms) 1792 [52]
- solid support 1082 [33]
- solitons (collision-based computing) 1952 [58]
- Solovay-Kitaev theorem 1411 [41]
- solution
 - concept 995 [31], 999 [31]
 - -oriented problem transformation 880 [28]
- source vertex 1247 [37]
- space-time diagrams (cellular automata) 1970 [58]
- span-program evaluation 1550 [46]
- sparse
 - Hamiltonian 1552 [46]
 - matrix powered entry (SMPE) 1556 [46]
- spatial
 - addressing 1310 [38]
 - topology 633 [20]
- speciation 1039 [32]
 - secondary contact 1039 [32], 1051 [32]
- species 1038 [32]
- spectral representation (quantum information processing) 1389 [41]
- speed of sound 317 [9], 321 [9]
- spike (action potential) 343 [10], 535 [17]
 - density code 340 [10]
 - raster 341 [10]
 - response model 346 [10]
 - -timing dependent plasticity (STDP) 349 [10]
 - trains 341 [10]
- SpikeNET software 369 [10]

- SpikeProp (supervised learning) 359 [10]
- spiking-bursting regime 538 [17]
- spiking neuron
 - convergence conjecture 353 [10]
 - model 343 [10]
- spiking neuron network (SNN) 335 [10]
 - supervised learning 359 [10]
 - unsupervised learning 358 [10]
 - winner-takes-all 358 [10]
- spiking neuron simulator
 - BRIAN 369 [10]
 - DAMNED 369 [10]
 - GENESIS 369 [10]
 - NEURON 369 [10]
 - PyNN 369 [10]
- spiral rule cellular automaton 1953 [58]
- splicing
 - operation 1188 [35]
 - rule 1188 [35]
 - rule, tree structures 1152 [34]
 - scheme, tree structures 1153 [34]
- SRAM memory 1661 [50]
- stabilizer code 1501 [44]
- Stark shift 1507 [44], 1510 [44]
- start codon 1077 [33]
- state 1388 [41]
 - space 848 [27], 1385 [41]
 - space, Davis–Vose model 850 [27]
 - space, generic model 850 [27]
 - transformation, continuous 1405 [41]
 - transformation, unitary 1405 [41]
 - vector 1383 [41], 1390 [41]
- static synapses 544 [17]
- stationary
 - distribution 856 [27]
 - localizations 1952 [58]
- statistical
 - mechanics 262 [8]
 - zero-knowledge (SZK) 1539 [45]
- statocyst 551 [17]
 - receptor neuron 551 [17]
- statolith 551 [17]
- steady state 717 [24]
 - evolutionary algorithm 947 [30]
- steepest descent 1625 [49]
- step-size 688 [22], 742 [25]
- step-size control
 - cumulative 687 [22]
 - mutative 684 [22]
 - mutative, correlated momentum 688 [22]
 - mutative, derandomized scaling 686 [22]
 - mutative, with momentum 685 [22]
 - success frequency 683 [22]
- sticker
 - language 1107 [33]
 - system 1107 [33]
- sticking operation 1106 [33]
- sticky end 1078 [33]
- stigmergy 1601 [48]
- Stinespring representation 1405 [41]
- stochastic
 - π -/pi-calculus 1841 [54], 1884 [55]
 - tunneling (STUN) 1646 [49]
- stochastic convergence 847 [27]
 - complete 851 [27]
 - evolutionary algorithm, complete 855 [27]
 - evolutionary algorithm, in mean 855 [27]
 - evolutionary algorithm, in probability 855 [27]
 - evolutionary algorithm, with probability 1 855 [27]
 - in mean 851 [27]
 - in probability 851 [27]
 - mixture 1389 [41]
 - with probability 1 851 [27]
- stochastic process
 - discrete time 848 [27]
 - index set T 848 [27]
 - Markov chain 849 [27]
 - Markov process, discrete time 849 [27]
 - Markov process, time-homogeneous 849 [27]
 - Markov process, time-inhomogeneous 849 [27]
 - state space 848 [27]
- stop codon 1077 [33]
- strain rate tensor 317 [9]
- strand displacement 1164 [34], 1324 [39]
- strange attractors
 - Lorenz 2000 [59]
 - Rössler 1999 [59]
- strategy (evolutionary strategies)
 - comma 676 [22]
 - parameter 676 [22]
 - plus 676 [22]
- strategy pattern 914 [29]
- streaming 308 [9]
- streptavidin-biotin 1205 [36]
- string
 - conjugate 1239 [37]
 - double occurrence 1245 [37]
 - double rule (sdr) 1248 [37]
 - double rule (sdr), dual 1253 [37]
 - legal 1245 [37]
 - negative rule (snr) 1248 [37]
 - positive rule (spr) 1248 [37]
 - positive rule (spr), dual 1253 [37]
 - reduction 1249 [37]
- strings
 - equivalent 1239 [37], 1262 [37]
- SU(2) (group theory) 1507 [44]
- subexcitable chemical media 1900 [56]
- subshift 28 [2]
 - finite type 29 [2]
 - language of 28 [2]

- signal 38 [2]
 - sofic 29 [2]
 - subsystem states (quantum information processing) 1398 [41]
 - successful reduction
 - graphs 1250 [37]
 - strings 1249 [37]
 - sugar-phosphate 1075 [33]
 - superconducting
 - quantum interference device (SQUID) 1514 [44]
 - qubits 1513 [44]
 - superdense coding 1433 [42]
 - superheuristics 1774 [52]
 - bin packing 1778 [52]
 - satisfiability (SAT) 1779 [52]
 - supermartingale 853 [27]
 - superoperator 1403 [41]
 - superposition 1391 [41]
 - support vector 482 [15]
 - regression (SVR) function 491 [15]
 - support vector machine (SVM) 479 [15]
 - 1-norm ranking 496 [15]
 - binary 480 [15]
 - classification 480 [15]
 - ranking 493 [15]
 - regression 491 [15]
 - soft margin 496 [15]
 - standard ranking 496 [15]
 - SVM-light 500 [15]
 - surrogate mutations 772 [25]
 - SWAP (quantum computing)
 - gate 1500 [44]
 - iSWAP 1499 [44]
 - $\sqrt{2}$ SWAP 1500 [44]
 - swarm
 - intelligence (SI) 1599 [48]
 - robotics 1608 [48]
 - swimming central pattern generators (CPG) 549 [17]
 - switching dynamics 553 [17]
 - symbolic
 - dynamical systems 28 [2]
 - regression 718 [24]
 - symmetric-key cryptosystem 1523 [45]
 - synapses 540 [17]
 - axoaxonic 540 [17]
 - axodendritic 540 [17]
 - axosomatic 540 [17]
 - chemical 541 [17]
 - dendroaxonic 541 [17]
 - dendrodendritic 541 [17]
 - dendrosomatic 541 [17]
 - dynamic 545 [17]
 - electrical 541 [17]
 - excitatory 544 [17]
 - inhibitory 544 [17]
 - static 544 [17]
 - synaptic
 - conductance 542 [17]
 - current 542 [17]
 - delay 556 [17]
 - depression 545 [17]
 - facilitation 545 [17]
 - plasticity 349 [10]
 - reversal potential 542 [17]
 - synchrony 356 [10]
 - synfire chains 356 [10]
 - synthetic
 - biology 1180 [34], 1205 [36], 1314 [38], 1808 [53], 1822 [53]
 - multicellular system 1224 [36]
 - systematic evolution of ligands by exponential enrichment (SELEX) 1763 [52]
 - systems biology 1835 [54], 1864 [55]
 - modeling languages 1841 [54]
 - scalability 1839 [54]
- T**
- tag system 214 [6]
 - cyclic 244 [7]
 - take-over time 1043 [32]
 - target vertex 1247 [37]
 - T-cell receptors (TCR) 1577 [47]
 - Tchebycheff method 880 [28]
 - technical analysis, finance 1750 [52]
 - technology
 - interaction 1945 [57]
 - judgement 1944 [57]
 - knowledge 1944 [57]
 - temperature control 1308 [38]
 - Temperley–Lieb algebra 1563 [46]
 - template-guided recombination
 - DNA 1266 [37]
 - RNA 1268 [37]
 - temporal
 - Bell inequalities 1444 [42]
 - coding 341 [10]
 - tensor
 - networks 1477 [43]
 - product 1385 [41]
 - termite nests 1601 [48]
 - test 996 [31]
 - -based problem 997 [31]
 - -bed analysis 1013 [31]
 - role 996 [31]
 - tube computation 1137 [34]
 - theory 928 [29]
 - domination analysis 928 [29]
 - polynomial local search 928 [29]
 - thermal model 328 [9]
 - thermodynamic limit 557 [17]
 - thermodynamics, statistical 1624 [49]

- Theta
 - learning 360 [10]
 - neuron 345 [10]
- thresholding 336 [10]
- thymine (T) 1075 [33]
- Tierra 1825 [53]
- tiles, tiling 17 [1], 275 [8]
 - aperiodic set 18 [1]
 - blank 275 [8]
 - finite 275 [8]
 - problem 19 [1]
 - Wang 17 [1], 275 [8]
- time
 - *series forecasting (neural networks)* 461 [14]
 - step 163 [5]
- Toffoli gate 1411 [41], 1495 [44]
- top DNA strand 1105 [33]
- topographic map
 - formation 586 [19]
 - generative (GTM) 611 [19]
 - growing 600 [19]
 - kernel 609 [19]
 - recurrent 605 [19]
- topographical organization 586 [19]
- topology
 - Besicovitch 47 [2]
 - Cantor 9 [1], 28 [2], 268 [8]
 - vague 268 [8]
 - weak* 268 [8]
- tournament selection 629 [20]
- trace 1388 [41]
 - closure 1560 [46]
- tracing over 1399 [41]
- traffic models 305 [9]
- trajectory 1097 [33]
- transcription 1076 [33]
- transient state 856 [27]
- transition
 - function 1406 [41]
 - molecules 1158 [34]
 - probability 848 [27]
- transition matrix 850 [27]
 - normal forms 856 [27]
 - product decomposition 858 [27]
- translation 1076 [33]
- trapped
 - electrons 1506 [44]
 - ions 1504 [44]
 - neutral atoms 1506 [44]
- traveling salesperson problem (TSP) 610 [19], 628 [20], 660 [21], 878 [28], 916 [29], 1147 [34], 1614 [48], 1626 [49], 1635 [49]
- trellis 7 [1], 161 [5], 172 [5]
- triangular
 - conorm (t-conorm) 1935 [57]
 - norm (t-norm) 1935 [57]
- truncation selection 629 [20]
- Turing machine 214 [6], 240 [7], 1407 [41], 1966 [58]
 - deterministic (DTM) 1132 [34]
 - reversible 240 [7]
- Turing universality 190 [6], 240 [7]
- Tutte
 - dichromatic polynomial 1558 [46]
 - polynomial 1477 [43], 1567 [46]
- two-head finite automaton 1115 [33]
- twisted majority rule 300 [9]
- TX tile 1329 [39]
- U**
- unbounded support vector 494 [15]
- underlying objectives 1026 [31]
- uniform computation 1410 [41]
- unitary 1401 [41]
- universal
 - H system 1194 [35]
 - processors 1914 [56]
 - set of gates 1411 [41]
- universality 190 [6], 300 [9]
- up-and-down states 557 [17]
- uracil (U) 1076 [33]
- V**
- vague
 - complex concepts 1943 [57]
 - concepts 1922 [57]
- vagueness 1922 [57]
- Vapnik–Chervonenkis (VC) dimension 353 [10]
- variation operator design guidelines 677 [22]
 - control 677 [22]
 - reachability 677 [22]
 - unbiasedness 677 [22]
- vector state 1390 [41]
- vertex 1240 [37]
 - positive/negative 1240 [37]
 - source 1247 [37]
 - target 1247 [37]
- vesicular
 - decapsulation 1220 [36]
 - encapsulation 1220 [36]
- virtual
 - cell experiment (VICE) 1874 [55]
 - reconfigurable circuit (VRC) 1682 [50]
- viscosity 317 [9]
- visible
 - differences predictor (VDP) 379 [11]
 - discrimination model (VDM) 379 [11]
- visual information fidelity (VIF) method 380 [11]
- volume-weighted average price (VWAP) 1718 [51]
- Voronoi diagrams, reaction-diffusion systems 1904 [56]

W

Watson–Crick automaton 1114 [33]

- 1-limited 1115 [33]
- all-final 1115 [33]
- deterministic 1116 [33]
- simple 1115 [33]
- stateless 1115 [33]
- strongly deterministic 1116 [33]
- weakly deterministic 1116 [33]

Watson–Crick complementarity 1076 [33]

wave propagation 320 [9]

wavelet

- -domain natural image statistic metric (WNISM) 381 [11]
- transform 385 [11]

weak

- Pareto dominance 875 [28]
- refinement 881 [28]

WEBSOM (self-organizing maps for Internet exploration) 597 [19]

weighted sum method 880 [28]

welded beam design 1768 [52]

wet implementations (molecular computing) 1129 [34]

whiplash PCR 1322 [39]

Whitney rank polynomial 1558 [46]

Wigner crystal 1505 [44]

winner-take-all mode 553 [17]

winnerless competition 553 [17]

Wisdom Technology (WisTech) 1942 [57]

wobbled codon 1313 [38]

Wolfram rules 298 [9]

words (DNA computing) 1118 [33]

writhe 1559 [46]

writing 1308 [38]

X

XCS learning classifier system 1778 [52]

Y

Yao’s minimax principle 823 [26]

Z

Zeeman

- state 1505 [44]
- substate 1505 [44], 1511 [44]
- zero-knowledge 1528 [45], 1539 [45]
- protocol 1546 [46]

