Chapter 1

Evolution – A Universal Principle

In this chapter we give an overview of evolution in nature, and its algorithmic models in computers being used to mostly solve non–biological problems. We present the main principles of natural (or wild) evolution with its prerequisites, and point out how these principles can be transferred to information processing machines, by identifying the class of problems suited to evolutionary methods. The properties of these problem instances, and the common features of evolutionary algorithms are described. Hereafter, the various branches of evolutionary algorithms are discussed, specifically paying attention to genetic algorithms, as this thesis is devoted to biologically motivated extensions to this paradigm. An overview of main stages of genetic algorithms, basic implementations, and analytical tools for theoretical investigations is given.

1.1 Natural Evolution

Though, the title suggests the universality of the principle of evolution, we will use the terms natural and artificial evolution, or *Evolutionary Computation*, in order to indicate the current context. Artificial evolution, a branch of computer science, deals with the algorithmic solutions of optimization problems by mimicking natural evolution, whose principles have been firstly laid out by Charles Darwin in his pioneering work *The Origin of Species* [Dar72]. According to Darwin's theory of evolution, the driving force of biological development towards highly adapted individuals is natural selection which is commonly summarized using the term *Survival of the Fittest*. Selection is exerted due to limited resources (food, mates, living space) and gives indi-

viduals of higher fitness a higher chance for survival and reproduction.

But natural selection on its own would rapidly lead to populations of nearly identical individuals without any further increase in adaptation to the environment. Thus, nature "invented" a mechanism which allows for changes in the *Phenotype* of an individual. The term phenotype subsumes the appearance of the individual and its interactions with the environment. The phenotypic variance is caused by *Mutations* in the *Genotype* which is the general blueprint for the development of a specific phenotype. Mutations (more bluntly errors) in genotypes allow for the development of phenotypes with some new and different traits. These traits are then (de)selected according to their adaptation to the environment. Individuals with traits being superior to current traits in the population will be able to collect a higher share of resources, and will crowd out individuals with less fit traits. It should be pointed out that the term *Fitness* reflects much more the degree of adaptation of an individual to the environment (this includes the ability to quickly adapt to environmental changes [Dar72]) than its physical strength. The latter is in most cases only part of an individuals's fitness. Mutations can be introduced by a variety of means ranging from nuclear radiation to biochemical information processing errors. Mutation is inevitably generated by systems far from equilibrium, i.e. different states of energy force state transitions.

Another important feature of most biological systems is the uni–directional information flow from genotype to phenotype which is also known as the *Central Dogma* of molecular genetics [Bäc96, Fut90]. This is in contradiction to *Lamarckian Evolution*, a theory suggested by Jean–Baptiste Lamarck at the beginning of the 19th century, which postulates the *Inheritance of Acquired Traits*. According to Lamarck's ideas newly acquired traits can be directly passed to the next generation, but this would demand a mechanism which transfers the information about these superior traits back to the genotype. With the exception of some viruses, where observations indicating a form of Lamarckian evolution have been made, no backward flow of information from phenotype to genotype could be detected in biological systems [Fut90]. It is believed that the necessary biological requirements for non–darwinistic evolution are too complex, as a mechanism would be needed to transfer the information of phenotypical traits to the genotype. However, it cannot be excluded that natural evolution will develop such mechanisms, and they have already been implemented in artificial evolutionary systems, e.g. robot trajectory generation [Dav90b, Dav91].

One of the most fascinating questions is the origin of evolution, i.e. the emergence of self–reproducing units. The molecular biologist Manfred Eigen developed a theory which explains the very first steps of evolution by simple

molecules being able to self–replicate leading to the first biological cells, i.e. *Molecular Darwinism* [Eig71]. He identified the necessary conditions for Darwinian selection [Bäc96] being

- *Metabolism*: Any individual species must be built up from energy-rich matter which is transformed into energetically lower states, i.e., the system must be open and far from equilibrium.
- *Self–Reproduction*: Only by means of self–reproduction concurrent behavior and selection can emerge.
- *Mutation*: New information can only be generated by a process of self–replication errors.

These conditions are not only the prerequisites for the evolution of macromolecules, but are necessary for any kind of evolutionary system, and evolution is more and more thought of as a universal principle not only limited to biological systems [Kel94]. In fact, if we view a computer program as a process consuming electrical energy and dissipating thermal energy (metabolism), if the program can replicate itself (or parts of it), and errors are generated during self–replication, then the necessary conditions for Darwinian evolution are present. Thus, natural and artificial evolution (in computers) are based on the same fundamental principles, and there is hope that evolutionary computation can create "intelligence" comparable or even superior to the state of the art in biological systems on earth, i.e. humans.

1.2 Evolutionary Computation

1.2.1 Global optimization

Mainly, artificial evolution is employed to solve *Global Optimization Problems* of exponential complexity (NP–complete or NP–hard problems). As all optimization algorithms, they have to face the problems of *Local Optima*, i.e. "false" optima being inferior to the global optimum, where the evolutionary search can be trapped. A variety of algorithms (e.g., *Simulated Annealing* mimicking the process of cooling of crystals [vLA89]) have been proposed to circumvent the local optima problem, however, theoretical results from optimization theory and theory of algorithms imply that these efforts are bound to fail in general.

Törn and Žilinskas state that in global optimization no general criterion for identification of the global optimum exists [TŽ90, Bäc96]. In other words,

if the global optimum is unknown (and this is the reason we use algorithms to find them), we often cannot even determine, whether the optimum found by an algorithm is a local or a global one. Wolpert and Macready state in their *No Free Lunch Theorem* that

> ... all algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions [WM96].

Thus, even if we find methods for a specific algorithm which help to avoid local optima in one cost function, performance of that algorithm will be degraded with another problem or class of problems.

To our understanding the NFL–Theorem does not exclude the possibility to construct a meta–algorithm being capable of selecting well–performing algorithms for specific problems. According to Edelman's Darwinistic brain theory *Neural Darwinism* [Ede87] this may be the way our brain adapts to the whole variety of tasks it is supposed to master. Evolutionary old brain structures act as critics of neural activity and its outcome. They change the strength of connections between neurons according to the degree of task completion, i.e. spotting an object and grasping it, and assign groups of neurons to specific tasks during ontogeny, i.e. the course of development of an individual organism. Thus, using a different terminology the critic could be viewed as a meta–algorithm selecting (and in the case of neural darwinism even constructing) algorithms for a specific problem (task).

To conclude these thoughts, it should be noted that methods of artificial evolution are often used to find solutions in extremely complex problem spaces, where it is usually sufficient to discover a "good" solution in an appropriate time. Several properties of problem spaces suited for application of evolutionary algorithms can be identified [MTH89] (in this paper the alternative term (problem) surface is used).

- Infinite large surfaces.
- Indifferentiable surfaces.
- Complex and noisy surfaces, i.e. the mapping from genotype to phenotype is ambiguous.
- Deceptive surfaces, i.e. suboptimal solutions guide the search towards local optima.
- Multimodal surfaces, i.e. a number of local optima.

Most real world problems have one or more of these properties, thus, evolutionary algorithms can be utilized in a wide variety of problem domains, where practitioners are mostly content with solutions being superior to existent ones.

1.2.2 Basic components of an evolutionary algorithm

Let us now identify the basic components of any *Evolutionary Algorithm*, namely

- Genotype Encoding of Individual
- Population of Individuals
- Fitness Function
- Selection Method
- Genetic Operators (Mutation)

First of all, the problem and the parameters to be optimized have to be encoded in a genotype often called *Chromosome* in the field of evolutionary algorithms, though, these terms are not correctly transferred from biology. The biological genotype comprises a number of chromosomes and stands for the complete genetic information, while these two terms are used interchangeably in the field of evolutionary computation. The encoding of the parameters of a problem is one of the most crucial steps in the design of an evolutionary algorithm, as it can severely affect its performance, e.g. the ordering of parameters on the chromosome can be very important. Depending on the specific evolutionary algorithm, the parameters are encoded as real values, integers, booleans, or rules. A single encoded parameter is often referred to as *Gene*, but some researchers have also used that term for parts of a parameter. The specific value a gene can take on is often called *Allele*. The number and size (resolution) of parameters affects the length of the chromosome, whose size is fixed with most instances of evolutionary algorithms (with the prominent exception of *Genetic Programming* [Koz92]). Technically, the computer chromosome is an array (concatenation) of real, integer, byte, or bit values, or a tree of atomic program instructions (genetic programming).

A number of chromosomes form a population of individuals representing different solutions to the problem. The term individual corresponds to the phenotypical expression of the genotype, i.e. the decoded chromosome. This

process is often referred to as *Genotype/Phenotype Mapping*. The variety of solutions in a population is one of the unique features of evolutionary algorithms, as the "knowledge" required for the (sub)optimal solution is spread over a group of individuals and can be combined to create an offspring representing a better solution.

The quality of the solution is determined by the fitness function evaluating the performance of each individual in a population. This evaluation is often done in two steps by using an objective function value (the direct evaluation of parameters corresponding to an individual) and scaling this value to the actual fitness function value. Essentially, this *Fitness Scaling* is performed so as to overcome numerical problems with certain selection methods, but this also biases the evaluation of the quality of a solution [lMT93]. The fitness function represents the environment (or more precisely the feedback of the environment), and is, therefore, crucial to the performance of an evolutionary algorithm. The definition of a fitness function might seem a simple task for a real valued optimization problem, but can be very difficult with less obvious evaluations of the quality of a solution. Usually, the fitness function represents a static environment, as the function never changes, but this does not correspond to biological environments which are typically dynamic. In fact, the environment also adapts to the individual(s), a process called *Coevolution* [Kel94], researchers are about to integrate in evolutionary algorithms. An interesting example of artificial coevolution is the evolution of game–playing *Artificial Neural Networks* (ANNs), where the fitness of an individual ANN is simply given by winning or losing against another ANN [MM93].

The selection method simulates the competition for resources (survival of the fittest) and determines which individuals are to be placed in the *Mating Pool*, i.e. may generate offspring. Thus, selection marks the transition between *Generations*, where a new population (offspring) replaces the old population (parents) which is called generational scheme with a generation gap $G = 1.0$. Variants with $0.0 < G < 1.0$ have been proposed [DeJ75], where only parts of a population are replaced by offspring, or the next generation is selected from parents and offspring ($(\mu + \lambda)$ –scheme in *Evolution Strategies*, Section 1.2.3). A variety of selection schemes has been devised and we will mention the most important methods for GAs in Section 1.3. The common goal of all selection schemes is to guide the evolutionary search towards populations of increasing fitness, at the same time maintaining genetic diversity to avoid *Premature Convergence* of a population, i.e. the individuals of a population are very similar or even identical and get stuck in a local optimum [BBM93].

Finally, the genetic operators introduce random changes to the population which either results in *Exploration* of the search space, i.e. evaluation of new search space regions, or *Exploitation*, i.e. a fine–grained search in "good" regions already discovered. Exploration can be mainly credited to mutation introducing random changes in parameters (genes), while exploitation can be accounted to recombination (the exchange of genetic material of two or more parents) of existing solutions (chromosomes). Unfortunately, these dependencies are often exchanged, as the specific interactions also depend on the encoding and selection schemes. However, mutation and recombination are the two dominant genetic operators in evolutionary algorithms with a great variety of specific implementations and different priorities.

1.2.3 Instances of evolutionary algorithms

During the last 35 years several distinct branches of evolutionary algorithms have emerged, and we will briefly discuss similarities and differences of these approaches in this section. According to [Bäc96] the main categories are

- Evolution Strategies
- Evolutionary Programming
- Genetic Algorithms

1.2.4 Evolution strategies

Evolution Strategies (ES) have been developed by Bienert, Rechenberg, and Schwefel in Germany in the 1960s. First experiments with these algorithms have been performed with hydrodynamical problems, e.g., shape optimization of a bent pipe. One of he first ES–variants was a $(1+1)$–ES, thus, one parent solution generates one offspring and the best of these two solutions is selected. This two–membered ES was later extended to multi–membered ES, namely $(\mu+\lambda)$–ES and (μ,λ)–ES, where μ is the number of parent individuals and λ the number of generated offspring. The (μ,λ)–ES, where a new generation is created by selecting the best μ individuals out of λ offspring (implying $\lambda > \mu$) proved to be superior and is the currently prevailing selection scheme in ES.

In ES the individuals are represented as object parameter vectors $\vec{x} \in \mathbf{R}^n$ and the objective function (being identical with the fitness function) is given by $f : \mathbf{R}^n \to \mathbf{R}$. In addition to the n object parameters of $\vec{x}$ one to n standard deviations σ_i are incorporated in $\vec{x}$. These *Strategy Parameters* determine the degree of mutation for all object parameters (one strategy parameter)

or for every single object parameter (n strategy parameters). The actual real values added to the object parameters (mutation) are drawn from an n–dimensional normal distribution with standard deviations σ_i. Moreover, the σ_i can be linearly correlated by rotation angles α_{ij} which enable the rotation of mutation ellipsoids (surfaces of equal probability density to place an offspring).

The strategy parameters are subjected to mutation as well, thus, the mutation of object parameters is adaptive. This capability is termed *Self–Adaptation* [Sch95] and is the main characteristic of ES. Various forms of recombination have been proposed for ES, which proved especially useful for the adaptation of strategy parameters. Schwefel suggests discrete recombination of object variables (by using either object parameter $x_{S,i}$, or $x_{T,i}$ of two parents S and T), and intermediate recombination for strategy parameters (by "averaging" parameters $\frac{x_{S,i}+x_{T,i}}{2}$) [Bäc96].

1.2.5 Evolutionary programming

Evolutionary Programming (EP) was proposed by L. J. Fogel et al. [FOW66] in the 1960s and has been further developed by his son D. B. Fogel [Fog91]. It is in many aspects closely related to evolution strategies, but researchers of both camps worked independently of each other until contacts have been established in the last years.

Evolutionary programming is mainly employed for continuous parameter optimization and individuals are represented as combinations of object parameters and strategy parameters (standard deviations) $\vec{a} = (\vec{x}, \vec{\nu})$ with $x_i \in \mathbf{R}^n$ and $\nu_i \in \mathbf{R}^n_+$, where the ν_i represent standard deviations for the n object parameters only used with *Meta*–EP. Generally, the fitness of individuals is evaluated by imposing a random alteration κ on the objective function value. Thus, an additional mutation is introduced, when mapping the genotype to the phenotype.

The mutation introduced on the genotype level is also different from other approaches, as it uses the fitness value of an individual to determine the mutation to be applied. With standard EP (without strategy parameters) an object parameter x_i is mutated as follows

$$x_i' = x_i + N_i(0,1)\sqrt{\beta_i f(\vec{x}) + \gamma_i}, \tag{1.1}$$

where $N_i(0,1)$ is the normal distribution with mean 0 and standard deviation 1 (the i symbolizes the repeated sampling of the distribution for each object parameter) and the constants β_i and γ_i have to be either determined experimentally for a specific problem, or are often set to $\beta_i = 1$ and $\gamma_i = 0$.

As this mutation operator can lead to problems, e.g. for optima with fitness function values not equal zero the object parameter values tend to fluctuate, and convergence to the optimum is impossible, meta–EP introduces standard deviations for each object parameter and mutates these standard deviations, as well. Hence, meta–EP and evolution strategies are quite similar with the exception of the order of mutation of object and strategy parameters (in ES the strategy parameters are mutated firstly, with meta–EP the object parameters), and the probability distributions used for alterations of strategy parameters (in ES log–normal distribution, with meta–EP normal distribution) [Bäc96].

EP does not use recombination operators which can be seen as the main characteristic of evolutionary programming. The EP selection method utilizes the combination of μ parent individuals and μ offspring ($(\mu + \mu)$–selection). For every of the 2μ individuals a tournament with q randomly chosen individuals is performed by comparing the fitness of the current individual with the random subset. This procedure results in a score $w_k \in \{0, \ldots, q\}$ for each individual which is based on the number of times a specific individual is superior to the q randomly chosen individuals. The 2μ individuals are then ranked based on their score and the μ best scoring individuals are selected to form the next generation. This selection scheme can be viewed as a variant of *Tournament Selection* (with tournament size $q+1$) being one of the main selection schemes used with GAs (Section 1.3). Also, it can be proved that for $\lim_{q\to\infty}$ the discussed EP selection method is identical with probability 1 to a $(\mu + \mu)$–selection in ES [Bäc96].

1.3 Genetic Algorithms

Quite naturally, the first steps towards an algorithmic simulation of evolution and the underlying genetics have been made by biologists [Fra57] so as to model natural populations and study theories of evolutionary genetics. A well–known theorem in this field is Fisher's *Fundamental Theorem of Natural Selection* which states that

> the rate of increase of fitness of any organism at any time is equal to its genetic variance at that time.

However, this theorem has been much debated in biology and one of the arguments against it is that above theorem assumes a constant fitness of an organism independent of other species [Smi89]. This is not the case in nature, as a variety of species often compete for the same resources. Interestingly, the scenario of a constant (static) fitness (function) is the norm with evolutionary

algorithms, and we are not sure, if anybody has yet tried to transfer Fisher's theorem to the evolution of artificial individuals.

The first to apply basic principles of evolution and genetics for the solution of non–biological problems was the psychologist John Holland, who laid the foundations towards a theory of adaptive systems communicating with their environment [Hol75]. This work and the book by David E. Goldberg [Gol89], who was the first to apply GAs to a technical problem (optimization of gas pipeline operation [Gol83]), can be seen as the fundaments of GA literature, as they are quoted in each piece of work dealing with GAs.

We start out with the basic functional units of a genetic algorithm by using a pseudo–code representation (Table 1.1), and will discuss the single GA stages in more detail below.

A Basic Genetic Algorithm

```
BEGIN
generate initial population
compute fitness of each individual

WHILE NOT terminationCriterion DO
  FOR populationSize
    select individuals to form the mating pool
    crossover each two individuals to give two offspring
    apply mutation to offspring
    insert offspring in new generation
    compute fitness of each (new) individual
  ENDFOR
ENDWHILE

END
```

Table 1.1: Pseudo code for a basic GA.

1.3.1 Genotype representation

GAs operate on chromosomes built by the most basic units of information in digital computers, i.e. zeros and ones. Mostly, genotypes are of a fixed length l with $b_i \in \{0, 1\}$ and $i \in \{1, \ldots, l\}$. Often, the elements of the bit string b_i are termed as genes, but to establish a more correct analogy to natural genetics (Section 2) we will use the term *Base* for a single bit throughout this

thesis. According to this definition, a set of adjacent bases (a subset of the bases of the chromosome) is a *Gene*. The position of genes representing the parameters to be optimized is defined by the human user. This is already a very critical decision, as Goldberg's *Building Block Hypothesis* states that a GA works by identifying small building blocks on the chromosome which contribute (positively) to the individual's fitness, and by grouping these building blocks to larger blocks of increased fitness guiding the search towards the optimum. This task is clearly dependent on the ordering of parameters on the chromosome, and on their interactions, again, labeled with a natural genetic term, namely *Epistasis*. This term reflects the non–linear interaction of parameters (genes), i.e. a gene is only expressed in combination with another specific gene.

More technically, certain encoded parameters are non–linearly dependent [Dav90a]. According to the building block hypothesis, these parameters should be positioned close to each other in order to ease the GA search for the building block comprising these epistatic parameters. However, for most real world problems it is very difficult, if not impossible, to determine non–linear dependencies (and their strength of interaction), thus, it would be convenient, if the GA could group these parameters on its own. Holland proposed an *Inversion* operator in his very first GAs [Hol75], where the base sequence of random parts of the chromosome is reversed (this operation demands additional labels for the bases to identify their gene membership), hence, the genes are shuffled in the course of evolution. However, inversion is not implemented with most current GA applications indicating that it was deselected in GA evolution.

Instead of interpreting a gene as a binary coded bit string, researchers investigated the use of *Gray Codes* and their superiority to the former has been shown for specific problems [CS88]. The reason for investigating Gray codes is the unequal weighting of bases with the standard binary coding, i.e. the mutation of the most significant base of a gene causes much more changes in parameter value than mutation of the least significant base. With Gray codes subsequent integers are encoded with a *Hamming Distance* $h_d = 1$, thus, a mutation of a single base of a gene results necessarily (not sufficiently) in a small change of value. Recent work suggested the choice of *E–Code* which proved also superior to Gray code for a suite of test problems [EC96]. Obviously, the integer value produced by these codes can be mapped to an interval in the real numbers, thus, enabling the encoding of real values [Mic91].

In his dissertation, Field presents a theoretical framework for *Multary Representation* [Fie96] where he unifies binary and nonbinary representations, i.e. an alphabet with a cardinality $\mid \alpha \mid \geq 2$. He claims that representations

should be selected according to the specific problem, and argues that binary representation might not be the best choice for any problem. At this point it should be noted that nature operates with an alphabet of four letters (Section 2).

Summarizing it can be said that the issue of representation is an open field in GA research and we will present some mechanisms to evolve location and representation of genes based on biological analogies in this thesis.

1.3.2 Fitness evaluation

One of the most important issues in design of a GA is the proper choice of the fitness function, as it gives the algorithm feedback about preferred regions of the search space. If the fitness function does not reflect the underlying problem, the whole process will not perform satisfactorily. Besides encoding, the fitness function is the only part of a GA, where problem domain specific knowledge has to be utilized. Moreover, in this thesis we will reduce the amount of problem knowledge for encoding to a rough estimation of the chromosome length, and the encoding is evolved implicitly according to the fitness function.

In combination with specific selection schemes a scaling of fitness values is needed, e.g. mapping the fitness function values $f(x) \in \mathbf{R}$ to a scaled fitness $f_s(x) \in \mathbf{R}_+$. However, the fitness scaling also biases the selection scheme, as the scaled values do not exactly reflect the fitness of the individual [Bak87].

1.3.3 Mutation

The standard mutation operator in GAs is simply implemented by changing base values (base flip, Figure 1.1) with a mutation rate (probability) p_m which is commonly suggested in the range from $0.001 - 0.01$ throughout the GA literature. If the value (sampled anew for each base) from a *Pseudo Random Number Generator* (PRNG) with uniform distribution (Section 4.1) indicates a mutation of a base, the base is simply set to the complementary value (if operating on binary representations, otherwise the base value is changed to a different letter of the used alphabet). Originally, mutation was introduced by Holland as a background operator [Hol75, Bäc96], i.e. a second–string genetic operator, and should reintroduce alleles that got lost at certain locations to the population [Gol89]. However, at late stages of a GA the genotype of individuals might become very similar resulting in inefficiency of recombination (Section 1.3.4), and mutation is the only chance to discover new regions of the search space, if the evolutionary search got stuck in a local optimum. Later in this thesis, we will present a recombination scheme

evolving due to mutation and selection which leaves mutation as the single operator being able to discover new building blocks.

1.3.4 Crossover

Recombination in GAs is mostly termed *Crossover* and is thought to be the most important search operator. The basic idea is that by exchanging parts of the genotype of two parents good building blocks can be combined on the chromosome of an offspring, hereby, increasing its fitness towards the optimum. Basically, this is the essence of population based search, where it is assumed that the problem knowledge is spread over all individuals and can be exploited to find the optimum by combining all "pieces" of knowledge. The crossover operator is applied with probability p_c (crossover rate) ranging from $0.6 - 0.9$ with most GA implementations.

Holland introduced the basic *One–Point Crossover* (Figure 1.1) which is applied by randomly choosing a *Crossover Site* at a position $x_1 \in \{1, \ldots, l-1\}$, where x_1 denotes the "gap" (site) between bases. The bases to the right (we will later call this direction *downstream*) of the crossover site are then exchanged between two parents to generate two offspring, both containing genetic material of the two parents. However, this crossover operator is affected by *Positional Bias*, as the exchange probability of bases increases with base index. As a consequence, the last base on the chromosome receives a probability of 1 to be exchanged. In order to combat the positional bias *k–Point Crossover* has been introduced which employs k crossover sites $x_1 \ldots x_k$.

With this operator chromosome pieces between odd site indices to the left and subsequent even site indices to the right are swapped, while the remaining pieces keep their position. If k is odd, the site $x_{k+1} = l$ is added to the set of crossover sites. An odd number of crossover sites results in an asymmetrical exchange probability [Bäc96], thus, an even number of crossover sites should be implemented. 2–point crossover (Figure 1.1) is a very common crossover operator used with current GA implementations, and we will use it as the default conventional crossover operator with the ptGA proposed in this thesis (Appendix C).

A random number of crossover points is induced by *Uniform Crossover* [Sys89], where a crossover mask of length l is generated randomly for each crossover operation. The bits in the crossover mask indicate, if the base corresponding to the mask's bit position is to be exchanged. This operator subsumes all possible k–point crossovers with $k \in \{1, \ldots, l-1\}$ and proved superior to 1–point crossover on a set of test functions [Sys89].

Many problem–specific crossover operators have been proposed so as to keep the set of feasible chromosomes closed under crossover, i.e. no chromo-

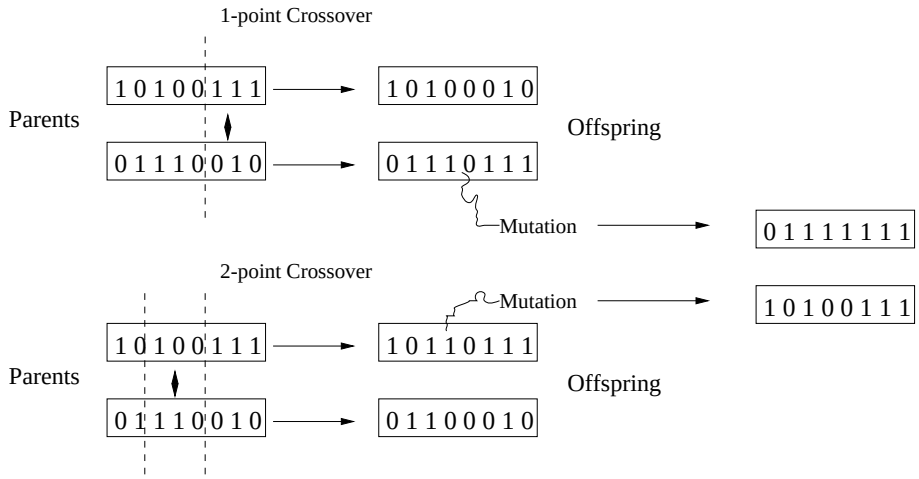

Figure 1.1: Basic genetic operators in GAs: 1–point, 2–point crossover, and mutation.

somes representing solutions not in the search space can be generated. An example for such an operator is *Partially Matched Crossover* (PMX) [Gol85] which recombines the order of genes, e.g. tours in a *Traveling Salesman Problem*, without violating problem constraints, i.e. generation of invalid tours. An interesting operator in the context of our thesis is *Punctuated Crossover* [SM87], where the chromosome length is doubled by appending l bases (punctuation string) which serve as a crossover mask. A base value $b_v = 1$ in the punctuation string indicates a base exchange at the corresponding site in the parameter coding part of the chromosome. Thus, the number and position of crossover sites is evolved, as the punctuation string is equally subjected to mutation and crossover. While this method employs additional bases to code for the crossover mask, the emerging ptGA crossover operator which will be proposed in this thesis is a by–product of the parameter encoding and its evolution and is a direct consequence of the gene structure without using any additional bases. Before we proceed with the presentation of the next (and last) essential mechanism in GAs, i.e. the selection method, we will briefly focus on mathematical tools to analyze GAs and their behavior. This is a necessary prerequisite for a deeper understanding of selection.

1.3.5 Analytical tools

Generally, GA analysis concentrates on specific parts of the algorithm, and overall GA behavior is far from being completely understood. To make things worse, the ptGA introduced in this thesis increases complexity even further, but for now we will restrict the discussion to conventional GAs, and start

with some definitions.

A *Schema* is a formal description of specific parts of the structure of a chromosome. Using binary coded strings we introduce a third symbol '$*$' which represents a "don't care" position on the genotype. $1 * * * * * * *$ and $*01 * * * *1$ are examples of schemata of length $L = 8$. The order σ of a schema is the number of defined base positions in the string, or more formally

$$\sigma =\mid \{i \mid b_i \in \{0, 1\} \mid \tag{1.2}$$

which yields $\sigma = 1$ and $\sigma = 3$ in our examples, respectively.

The *Defining Length* δ of a schema is the distance between the rightmost defined position $b_{i,r}$ and the leftmost defined position $b_{i,l}$ which gives

$$\delta = max\{i \mid b_i \in \{0, 1\}\} - min\{i \mid h_i \in \{0, 1\}\}. \tag{1.3}$$

Again, looking at above examples we arrive at $\delta = 0 - 0 = 0$, and $\delta = 7 - 1 = 6$, respectively.

Schemata can also be interpreted as L–dimensional *Hyperplanes* in the search space, where the order of the hyperplane is the order of the schema. Points in the search space are located at the corners of a hypercube and are members of $2^L - 1$ hyperplanes. The hyperplanes partition the search space and due to the population based search in GAs many hyperplanes are sampled, when a population of strings is evaluated [Whi93]. This leads to the notion of *Implicit Parallelism* [Hol75] describing the fact that many hyperplanes (schemata) are sampled simultaneously. The number of parallel schemata evaluations per generation can be estimated with $\mathcal{O}(n^3)$. Thus, GAs process a much larger quantity of schemata than n, i.e. the number of individuals in the population. A more general quantitative assessment of the degree of implicit parallelism is given in [BD93].

The *Schema Theorem* [Hol75] provides a lower bound on the change in the sampling rate for a single schema (hyperplane) from generation t to generation $t + 1$.

The *Average Schema Fitness* is given by

$$\overline{f}(H^t) = \frac{1}{m(H^t)} \sum_{x_i \in H^t} f(x_i) \tag{1.4}$$

where $m(H^t)$ is the number of occurrences of a specific schema (hyperplane) in generation t and $f(x_i)$ is the fitness of individual x_i.

The *Average Fitness* is given by

$$\overline{f}^t = \frac{\sum_{i=1}^{n} f(x_i)}{n} \tag{1.5}$$

with n being the population size.

The selection probability $p_s(x_i)$ with *Proportional Selection* is given by

$$p_s(x_i) = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)}, \tag{1.6}$$

thus, each individual is selected for crossover with a specific probability $p_s(x_i)$.

Using these three equations we derive

$$\frac{\overline{f}(H^t)}{\overline{f}^t} = \frac{n \sum_{x_i \in H^t} p_s(x_i)}{m(H^t)} = \frac{m(H^{t+1})}{m(H^t)} \tag{1.7}$$

If the average schema fitness is greater than the average fitness, the ratio $\frac{\overline{f}(H^t)}{\overline{f}^t} > 1$. This results in an exponential growth of the specific schema, since $m(H^{t+1}) = m(H^t)(1+c)$ with $c > 0$, and gives $m(H^t) = m(H^0)(1+c)^t$. Similar observations can be made for a schema of lower than average fitness which lead to exponential decrease of the number of the specific schema. However, some flaws can be identified with the above derivations. Equation 1.7 only describes the transition from generation t to $t+1$, thus, the above generalization to generation $0-t$ is not exact, as the average schema fitness of H^t changes in relation to other schemata in the new generation. Moreover, the selection process is biased with finite population sizes, as the proportion of offspring in the new generation does not exactly reflect the selection probability of parents (sampling error) [Whi93]. Hence, the schema theorem should be viewed more as a qualitative than a quantitative description of schemata dynamics stating that schemata of high fitness receive an increasing number of trials in the course of evolution.

Problems for the GA can arise, if combinations of high fit schemata of low order lead to low fit schemata of higher order, i.e. to non-optimal solutions. These problems are called GA–*deceptive*, but the exact definition of what constitutes a deceptive problem is difficult, as presumably deceptive problems could be easily solved by a GA and vice versa. More detailed definitions and discussions of deception can be obtained from [Whi91].

So far we have only considered proportionate selection and its influence on schema growth, but the main genetic operators crossover and mutation can destroy useful schemata. As Equation 1.7 is only concerned with surviving schemata, we need to calculate the survival rates of schemata with the following (simplified) probabilities

$$1 - p_c \frac{\delta(H^t)}{l-1} \tag{1.8}$$

being the *Schema Survival Probability* under 1–point crossover with p_c as the crossover rate and

$$(1 - p_m)^{\sigma(H^t)} \tag{1.9}$$

being the schema survival probability under mutation with p_m as the mutation rate. Finally, this gives the schema theorem or *The Fundamental Theorem of GAs* [Gol89]

$$m(H^{t+1}) \geq m(H^t) \frac{f(H^t)}{\overline{f}^t} (1 - p_c \frac{\delta(H^t)}{l-1}) (1 - p_m)^{\sigma(H^t)} \tag{1.10}$$

Further refinement of this inequality can be achieved by calculating losses and gains more precisely [BG87]. E. g. the above theorem (Equation 1.10) is overly conservative with respect to crossover. Crossing over two chromosomes which both comprise the same specific schema does not cause disruption.

Another analytical tool for studying GAs is the *Walsh Transform* which can be used to analyze an analytically given fitness function, mainly, for theoretical purposes. The given function on a discrete, finite range of values is separated into *Walsh Functions* with coefficients

$$\omega_j = \frac{1}{2^L} \sum_{x=0}^{2^L-1} F(x) \psi_j(x) \tag{1.11}$$

$$\psi_j(x) = \begin{cases} 1 & \text{if x} \wedge \text{j even parity} \\ -1 & \text{otherwise} \end{cases} \tag{1.12}$$

The index j can be interpreted as the order of the hyperplane partition of a given fitness function in the slightly different *Hyperplane Transform* [Hol88].

The back transform

$$F(x) = \sum_{j=0}^{2^L-1} \omega_j \psi_j(x) \tag{1.13}$$

is called the *Walsh Polynomial* representing $F(x)$. There is a close connection between the Walsh transform and schemata which led to the method of *Walsh–Schema Transform* proposed by Bethke [Bet80]. The Walsh–schema transform expresses the fitness of a schema H as a sum of progressively higher–order Walsh coefficients ω_j. In order to calculate the fitness of a

schema $f(H)$ only "subsuming coefficients", i.e. coefficients representing the schema, need to be summed. The Walsh transform was also used to analyze deceptive functions [Tan89, Gol91].

Finite *Markov Chains* have been used to analytically investigate genetic operators selection and mutation for idealized individuals (single base chromosomes) [GS87a, Hor93]. An analysis for a finite population, where all possible populations of size n with individuals of chromosome length l are represented by their Markov chain states and transition probabilities is given in [NV92]. However, the enormous number of states for realistic GA parameters does not allow numerical evaluations of this model.

In his dissertation Manela suggests abstract *Group Theory* as a general framework for GA analysis [Man93], and discusses some ideas to evolve GA parameters by means of *Modifier Genes* which are appended to the chromosome, and control GA parameter values. Basically, this is a generalization of the mechanisms of punctuated crossover [SM87] as described above, and would allow self–adaptation of GA parameters. A *Meta–GA* has been used for the evolution of GA parameters in [Gre86] which showed that GA parameters. e.g. mutation and crossover rate, population size, are robust in a wide range of values, i.e. do not significantly alter GA performance.

1.3.6 Selection

The selection scheme is at the core of the operation of a GA, as it models the survival of individuals, or more precisely, the competition for mates, as the selected individuals are placed in the mating pool to be recombined by crossover. The selection method should balance exploration and exploitation of the search space which is dependent on the *Selection Pressure*. A method selecting only the best individuals of a population exerts a high selection pressure (exploitation), but this will rapidly lead to a prematurely converged population. On the other hand, little or no selection pressure, e.g., selecting individuals randomly, will cause a high degree of exploration. However, GA search is no longer guided towards regions of higher fitness, and is degraded to a form of *Random Search*.

As the schema theorem (Equation 1.10) has been derived assuming proportional selection (Equation 1.6), where the probability of selection is based on the ratio of individual fitness to the population's average fitness, it has long been the dominant selection method and is still used in many current GA implementations. Specific implementations of proportionate selection are *Roulette Wheel* selection and variants hereof, e.g. *Remainder Stochastic Sampling*, *Remainder Stochastic Independent Sampling*, and *Stochastic Universal Sampling* (SUS) [Bak87]. Basically, these algorithms use the imag-

ination of a roulette wheel, whose slots of unequal sizes represent specific selection probabilities. Successive random spin(s) (or one with SUS) of the wheel selects individuals corresponding to the slot the ball has fallen into.

Considering two population scenarios reveals the problems associated with proportionate selection. If one individual is of much higher fitness compared to all others, it will essentially take over the population, as its selection probability $p_s \approx 1$. On the other hand, if all individuals have very similar fitness values, the selection pressure becomes very small, and *Genetic Drift*, i.e. sampling errors due to the finite population size will favor specific individuals randomly, will occur. Therefore, a variety of scaling methods have been suggested to overcome these problems, e.g. *Sigma Scaling* [For85] which is defined by

$$f_s(x_i) = \frac{f(x_i) - (\overline{f} - f_\sigma)}{f_\sigma}, \tag{1.14}$$

where $f_s(x_i)$ is the scaled fitness of individual x_i, and f_σ the standard deviation of unscaled fitness values of the population. However, some scaled fitness values might become negative, and even more problematic is the biasing of observed fitness. Altogether, the usefulness of sigma scaling seems unclear both from a theoretical and a practical point of view [Bäc96]. All these problems stem from the dependence of selection probability on the numerical fitness values which often are more qualitative than quantitative indicators for the fitness of individuals. Thus, a ranking of individuals based on their fitness is a logical consequence. Recently, ranking methods have gained increasing popularity and are thought to be superior to fitness scaling methods [lMT93, BT95].

Fitness Ranking ranks the individuals of a population according to their fitness and selection probabilities are either assigned linearly [Bak85] or exponentially [Dav89]. Exponential ranking can be tuned to assign higher selection probabilities to less fit individuals so as to preserve genetic diversity in a population.

Tournament Selection [Bri81] does no longer rely on selection probabilities, but picks k individuals at random and selects the best. This procedure is repeated until the mating pool is full, i.e. n parents are selected. In our view this offers a lot of advantages, as the selection process is only dependent on the ordering of fitness values. Thus, any fitness function inducing fitness values such that $f(x_i) > f(x_j)$ for an individual x_i fitter than x_j (in the case of maximization problems) performs the same. Moreover, the selection pressure can be easily adjusted by setting k to specific values. In most GA implementations with tournament selection $k = 2$ which is termed *Binary*

Tournament Selection [1]. By definition, this selection scheme guarantees survival of the fittest individual (*Elitist Strategy*), as it will win any tournament it competes in. Continuing the praise, the tournament selection algorithm is of linear time complexity [GD91].

In our view, tournament selection also resembles the closest analogy to natural selection, as it is in a sense coevolutionary, because the chance of survival is dependent on the environment (other individuals in this case), but not only on the fitness value assigned by a static fitness function. For the above reasons binary tournament selection is our method of choice and all experiments in this thesis are performed relying on that scheme.

1.4 Summary

After presenting the principles of natural evolution, we discussed how these mechanisms can be incorporated in computer models so as to (try to) solve optimization problems of exponential complexity. The various instances of evolutionary algorithms, evolution strategies, evolutionary programming, and genetic algorithms have been outlined, paying particular attention to the latter. Genotype representation, fitness functions, and the basic genetic operators, namely selection, mutation, and crossover have been discussed. The fundamental theorem of GAs with its implications to the dynamics of schemata, along with a short overview of analytical tools to investigate GA behavior have been given.

[1] In all our GA experiments with various problems a $k > 2$ never increased performance.

Bibliography

[Ada91] Roger L. P. Adams. *DNA Replication.* In Focus. Oxford University Press, New York, 1991. ISBN 0-19-963216-2.

[Bäc92] Thomas Bäck. The interaction of mutation rate, selection and self–adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 85–94, Amsterdam, 1992. Elsevier.

[Bäc96] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996.

[Bak85] James Edward Baker. Adaptive selection methods for genetic algorithms. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 101–111. Texas Instruments, Inc. and Naval Research Laboratory, Lawrence Erlbaum Associates, 1985.

[Bak87] James Edward Baker. Reducing bias and inefficiency in the selection algorithm. In John J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[BB92] Trevor Beebee and Julian Burke. *Gene structure and transcription.* In Focus. Oxford University Press, New York, 2nd edition, 1992. ISBN 0-19-963317-7.

[BBM93] David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[BD93] Alberto Bertoni and Marco Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61:307–314, 1993.

[Bea90] J. E. Beasley. OR–Library: Distributing Test Problems by Electronic Mail. *Journal of Operational Research Society*, 41(11):1069–1072, 1990.

[Bet80] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, University of Michigan, 1980.

[BG87] Clayton L. Bridges and David E. Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In John J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 9–13. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[BH95] Karthik Balakrishnan and Vasant Honavar. Evolutionary design of neural architectures – a preliminary taxonomy and guide to literature. Technical Report CS TR #95-01, Iowa State University, Department of Computer Science, Ames, Iowa 50011–1040, U.S.A., January 1995.

[BHPT80] O. Beyer, H. Hackel, V. Pieper, and J. Tiedge. *Wahrscheinlichkeitsrechnung und mathematische Statistik*, volume 17 of *Mathematik f. Ingenieure, Naturwissenschaftler et al.* Harri Deutsch, Thun, 2nd edition, 1980.

[BM94] Thang N. Bui and Byung R. Moon. Analyzing Hyperplane Synthesis in Genetic Algorithms Using Clustered Schemata. Technical Report CSE-94-026, Penn State University, University Park, PA, March 1994.

[Bri81] A. Brindle. *Genetic Algorithms for Function Optimization*. PhD thesis, University of Alberta, 1981.

[BT95] Tobias Blickle and Lothar Thiele. A Mathematical Analysis of Tournament Selection. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the 6th International Conference (ICGA95)*, San Mateo, California, 1995. Morgan Kaufmann Publishers, Inc.

[CS85] T. Cavalier-Smith. *The Evolution of Genmome Size*. wiley, New York, 1985.

[CS88] R. A. Caruna and J. D. Schaffer. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 153–161, 1988.

[Dar72] Charles Darwin. *The Origins of Species.* Collier Books, 1872.

[Dav89] L. Davis. Adapting Operator Probabilities in Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufmann Publishers, Inc.

[Dav90a] Y. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[Dav90b] Yuval Davidor. Sub–Goal Reward and Lamarckism in a Genetic Algorithm. In *Proceedings of the European Conference on Artificial Intelligence*, 1990.

[Dav91] Lawrence Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991. ISBN 0-442-00173-8.

[deD94] Christian deDuve. *Ursprung des Lebens: Präbiotische Evolution und die Entstehung der Zelle.* Spektrum Akademischer Verlag, 1994. ISBN 3-86025-187-2.

[DeJ75] K. DeJong. *The Analysis and Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[DG89] Kalanmoy Deb and David E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[EC96] M. Eaton and J. J. Collins. A Comparison of Encoding Schemes for Genetic Algorithms. In *World Congress on Neural Networks*, pages 1067–1070. International Neural Network Society, Lawrence Erlbaum Associates, Inc., 1996.

[Ede87] G. M. Edelman. *Neural Darwinism:The Theory of Neuronal Group Selection.* Basic Books, New York, 1987.

[Eig71] Manfred Eigen. Selforganization of Matter and the Evolution of Biological Macro–Molecules. *Die Naturwissenschaften*, 58(10):465–523, 1971.

[Fab95] Stefan Fabry. Frühe Introns – späte Introns? *Biologie in unserer Zeit*, 25(5):300–306, 1995.

[Fie96] Paul Field. *A Multary Theory for Genetic Algorithms: Unifying Binary and Nonbinary Problem Representations.* PhD thesis, University of London, 1996.

[FM96] Christoph M. Friedrich and Claudio Moraga. An Evolutionary Method to Find Good Building Blocks for Architectures of Artificial Neural Networks. In *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge–Based Systems*, pages 951–956, 1996.

[Fog91] D. B. Fogel. *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling.* Ginn Press, Needham Heights, 1991.

[For85] S. Forrest. Documentation for Prisoner's Dilemma and Norms Programs that Use the Genetic Algorithm. Technical report, The University of New Mexico, Albuquerque, NM, 1985.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* Wiley, New York, 1966.

[Fra57] A. S. Fraser. Simulation of Genetic Systems by Automatic Digital Computers, I. Introduction. *Australian Journal of Biological Science*, 10:492–499, 1957.

[Fut90] D. J. Futuyma. *Evolutionsbiologie.* Birkhäuser Verlag, Basel, 1990.

[GBD+94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, 1994.

[GD91] D. E. Goldberg and K. Deb. AComparative Analysis of Selection Schemes Used in Genetic Algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[GDKH93] David E. Goldberg, Kalyanmoy Deb, Hillol Kargupta, and Georges Harik. Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In University of New Mexico Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64. University of Illinois at Urbana-Champaign, Morgan Kaufmann, 1993.

[GG93] W. Gilbert and M. Glynias. On the ancient nature of introns. *Gene*, 135:137–144, 1993.

[Gol83] D. E. Goldberg. *Computer–Aided Pipeline Operation Using Genetic Algorithms and Rule Learning.* PhD thesis, University of Michigan, 1983.

[Gol85] David E. Goldberg. Alleles, Loci, and the TSP. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159. Texas Instruments, Inc. and Naval Research Laboratory, Lawrence Erlbaum Associates, 1985.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, 1989. ISBN 0-201-15767-5.

[Gol91] David E. Goldberg. Construction of High–order Deceptive Functions Using Low–order Walsh Coefficients. Technical Report No. 93001, Department of Engineering Mechanics, The Clearing House for Genetic Algorithms, University of Alabama, Tuscaloosa, AL, 1991.

[Gre86] J. J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[Gru92] F. Gruau. Genetic Synthesis of Boolean Neural networks with a Cell Rewriting Developmental Process. In D. Whitley and J. David Schaffer, editors, *Proceedings of the Third International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 55–74, Los Alamitos, California, 1992. IEEE Computer Society Press.

[GS87a] D. E. Goldberg and P. Segrest. Finite Markov Chain Analysis of Genetic Algorithms. In John J. Grefenstette, editor, *Genetic*

Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, pages 1–8. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[GS87b] D. E. Goldberg and R. E. Smith. Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 9–13. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[HE97] Ben S. Hadad and Christoph F. Eick. Supporting Polyploidy in Genetic Algorithms using Dominance Vectors. In *Proceedings of GA97*. Springer, 1997.

[HMS95] Reinhold Huber, Helmut A. Mayer, and Roland Schwaiger. netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms. In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Forschungsbericht Nr. 580, Dortmund*, August 1995.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[Hol88] J. H. Holland. The Dynamics of Searches Directed by Genetic Algorithms. In Y. C. Lee, editor, *Evolution, Learning, and Cognition*, pages 111–128. World Scientific, 1988.

[Hor93] Jeffrey Horn. Finite markov chain analysis of genetic algorithms with niching. IlliGAL Report No. 93002, Department of Computer Science, University of Illinois at Urbana-Champaign, February 1993.

[HP95] Frank Hoffmann and Gerd Pfister. A New Learning Method for the Design of Hierarchical Fuzzy Controllers Using Messy Genetic Algorithms. In *Proceedings Sixth International Fuzzy Systems Association World Congress, Vol. 1*, pages 249–252, 1995.

[HSG89] Steven Alex Harp, Tariq Samad, and Aloke Guha. Towards the genetic synthesis of neural networks. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 360–369, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufmann.

[HSM96] Reinhold Huber, Roland Schwaiger, and Helmut A. Mayer. On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks. In *World Congress on Neural Networks*, pages 1063–1066. International Neural Network Society, Lawrence Erlbaum Associates, Inc., 1996.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[KBH94] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. The Zero/One Multiple Knapsack Problem and Genetic Algorithms. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 188–193. ACM Press, 1994.

[Kel94] Kevin Kelly. *Out of Control - The New Biology of Machines.* Fourth Estate Ltd., London, 1994.

[Kit90] H. Kitano. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4:461–476, 1990.

[KK88] Roger D. Kornberg and Aaron Klug. Das Nucleosom. *Spektrum der Wissenschaft*, pages 74–80, 1988.

[Knu69] Donald E. Knuth. *The art of computer programming*, volume 2. Reading, 1969.

[Koz92] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection.* Complex Adaptive Systems. The MIT Press, Cambridge, MA, 1992.

[LBB+95] Harvey Lodish, David Baltimore, Arnold Berk, S. Lawrence Zipursky, Paul Matsudaira, and James Darnell. *Molecular Cell Biology.* Scientific American Books, 3rd edition, 1995. ISBN 07167-2380-8.

[Lev91] James R. Levenick. Inserting Introns Improves Genetic Algorithm Sucess Rate: Taking a Cue from Biology. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127. University of California, San Diego, Morgan Kaufmann, 1991.

[lMT93] Michael De la Maza and Bruce Tidor. An analysis of selection procedures with particular attention paid to proportional and boltzmann selection. In University of New Mexico Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 124–131. University of Illinois at Urbana-Champaign, Morgan Kaufmann, 1993.

[Mah93] Samir W. Mahfoud. Simple analytical models of genetic algorithms for multimodal function optimization. IlliGAL Report No. 93001, Department of Computer Science, University of Illinois at Urbana-Champaign, February 1993.

[Man93] Mauro Manela. *Contributions to the Theory and Applications of Genetic Algorithms.* PhD thesis, University College London, 1993.

[MFH91] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Cambridge, MA, 1991. MIT Press.

[Mic91] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Artificial Intelligence. Springer, Berlin, 1991.

[MM93] David Moriarty and Risto Miikkulainen. Evolving Complex Othello Strategies Using Marker–Based Genetic Encoding of Neural Networks. Technical Report AI93–206, The University of Texas at Austin, Department of Computer Sciences, Austin, TX 78712–1188, September 1993.

[MS97] Helmut A. Mayer and Roland Schwaiger. Towards the Evolution of Training Data Sets for Artificial Neural Networks. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 663–666. IEEE Press, 1997.

[MSH96] Helmut A. Mayer, Roland Schwaiger, and Reinhold Huber. Evolving Topologies of Artificial Neural Networks Adapted to Image Processing Tasks. In *Proceedings of the 26th International Symposium on Remote Sensing of Environment*, 1996.

[MTH89] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing neural networks using genetic algorithms. In J. David

Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[Müh92] H. Mühlenbein. How genetic algorithms really work: I. mutation and hill-climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 15–25, Amsterdam, 1992. Elsevier.

[Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia, 1992.

[NMP94] Stefano Nolfi, Orazio Miglino, and Domenico Parisi. Phenotypic Plasticity in Evolving Neural Networks. In *Proceedings of the First Conference from Perception to Action*, 1994.

[NV92] Allen E. Nix and Michael D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[PM88] Stephen K. Park and Keith W. Miller. Random number generators: Good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, October 1988.

[PM93] Stephen K. Park and Keith W. Miller. Another test for randomness. *Communications of the ACM*, pages 108–110, July 1993.

[Pre94] Lutz Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report TR 21/94, Universität Karlsruhe, Fakultät für Informatik, 76128 Karlsruhe, Germany, September 1994.

[Rog89] J. H. Rogers. How were introns inserted into nuclear genes? *Trends in Genetics*, 5:213–216, 1989.

[RPLH89] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[RWL94] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, March 1994.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth–Generation Computer Technology Series. Wiley, New York, 1995.

[SGE91] Robert E. Smith, David E. Goldberg, and Jeff A. Earickson. SGA-C: A C–Language Implementation of a Simple Genetic Algorithm. TCGA Report 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Department of Engineering Mechanics, Tuscaloosa, AL 35487, May 1991.

[Sha87a] Craig G. Shaefer. The ARGOT Strategy: Adaptive Representation Genetic Optimizer Strategy. In John J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–55. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[Sha87b] P. A. Sharp. Splicing of Messenger RNA Precursors. *Science*, 235:766–771, 1987.

[SHM95] Roland Schwaiger, Reinhold Huber, and Helmut A. Mayer. Land Cover Classification of Landsat Images Using Problem–Adapted Artificial Neural Networks. In *Proceedings of the International Workshop on Soft Computing in Remote Sensing Data Analysis, Series in Remote Sensing, Vol. 1*, 1995. ISBN 9810227574.

[SM87] J. D. Schaffer and A. Morishima. An Adaptive Crossover Distribution Mechanism for Genetic Algorithms. In John J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. Naval Research Laboratory, Lawrence Erlbaum Associates, 1987.

[Smi89] John Maynard Smith. *Evolutionary Genetics.* Oxford University Press, New York, 1989. ISBN 0–19–854215–1.

[Sti87] D. R. Stinson. *An Introduction to the Design and Analysis of Algorithms.* The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 2nd edition, 1987.

[Sys89] G. Syswerda. Uniform Crossover in Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[Tan89] R. Tanese. *Distributed Genetic Algorithms for Function Optimization.* PhD thesis, University of Michigan, 1989.

[Tan92] Andrew S. Tanenbaum. *Computer–Netzwerke.* Wolfram's Fachverlag, Attenkirchen, 2nd edition, 1992. ISBN 3-925328-79-3.

[TŽ90] A. Törn and A. Žilinskas. Global Optimization. *Lecture Notes in Computer Science*, 350, 1990.

[vLA89] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic Publishers Group, Dordrecht, 1989.

[Whi91] L. D. Whitley. Fundamental Principles of Deception in Genetic Search. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[Whi93] Darrell Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Colorado State University, November 1993.

[WM96] David H. Wolpert and William G. Macready. No Free Lunch Theorems for Search. Technical Report SFI–TR–95–02–010, The Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501, February 1996.

[Wu96] Annie Siahung Wu. *Non–Coding DNA and Floating Building Blocks for the Genetic Algorithm.* PhD thesis, University of Michigan, 1996.

[Yao93] Xin Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[YFU97] Tomohiro Yoshikawa, Takeshi Furuhashi, and Yoshiki Uchikawa. The Effects of Combination of DNA Coding Method with Pseudo–Bacterial GA. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 285–290. IEEE Press, 1997.

[ZMV+94] Andreas Zell, Guenter Mamier, Michael Vogt, Niels Mach, Ralf Huebner, Kai-Uwe Herrmann, Tobias Soyez, Michael Schmalzl, Tilman Sommer, Artemis Hatzigeogiou, Sven Doering, and Dietmar Posselt. *SNNS Stuttgart Neural Network Simulator, User Manual.* University of Stuttgart, 1994.