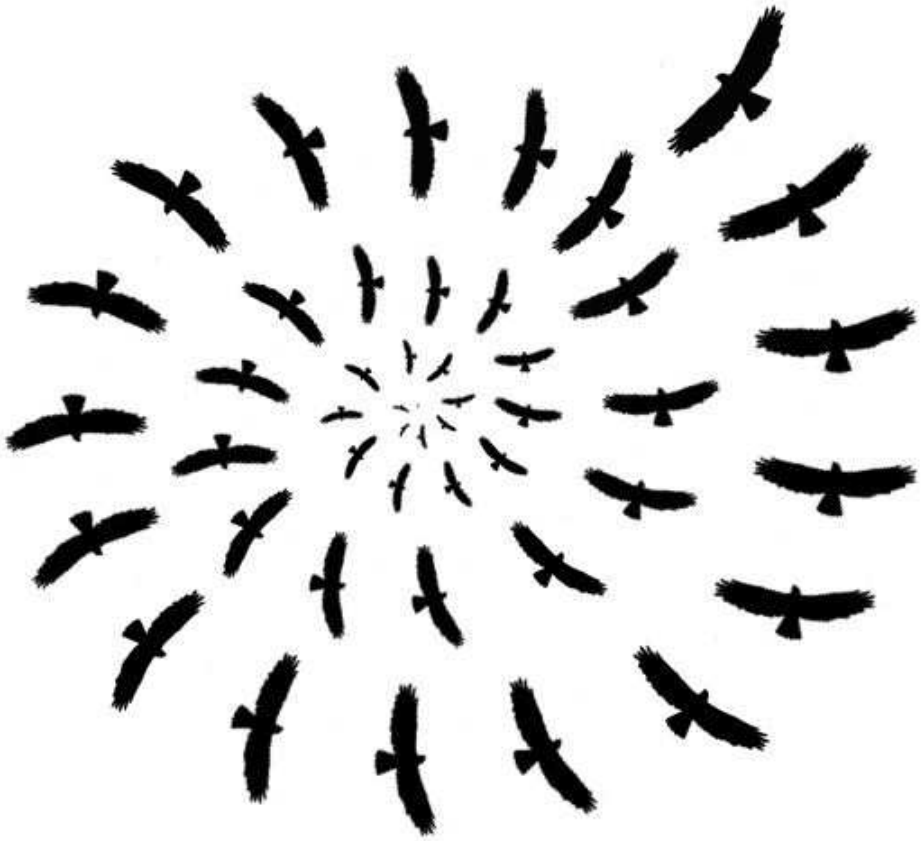
Martijn C. Schut



Scientific Handbook for Simulation of Collective Intelligence

M.C. Schut *Scientific Handbook for Simulation of Collective Intelligence.*

Version: 2 (February 2007).

http://www.sci-sci.org/

Cover artwork: N. de Carvalho Ferreira – *Zonder Titel.*

Contents

Foreword

Imagine that it is the year 2091 and your moon-Jeep is being repaired by a swarm of microscopic machines to fix some serious moondust damage. Do you trust them to do the job right? Imagine that is the year 2061 and the city of New York launches a new surveillance system consisting of a swarm of autonomic microflyers. Do you feel secure? Imagine that it is the year 2031 and there is the first android team that challenges a human soccer team for the ceremonial opening game at the world soccer championships. Which team do you put your money on?

These future scenarios have one common denominator: they all involve complex systems consisting of (many) interacting parts that are self organising and collectively intelligent. This book is about the understanding of the behavior and self organisation of complex systems: systems in which the interaction of the components is not simply reducible to the properties of the components. The general question that we address is: how should systems of very many independent computational (e.g., robotic or software) agents cooperate in order to process information and achieve their goals, in a way that is efficient, self optimising, adaptive, and robust in the face of damage or attack?

Acknowledgements

This book contains descriptions of studies that were based on research and teaching collaborations with the following people: Gusz Eiben, Konrad Diwold, Geoff Nitschke, Jan Treur, Tibor Bosse, Viara Popova, Mark Hoogendoorn, Catholijn Jonker, Betsy Sklar, Simon Parsons, Pieter Buzing, Nico Vink, Tamas Buresch, Maartje Spoelstra, Lieke Hoyng, Jelmer van der Ploeg, Joost Meijer, Olaf van Zon, Ronald Terpstra, Rogier Jacobs, Tudor Toma, Karin Rijnders and Albert van der Heide. Hereby I would like to thank these people for the pleasant cooperation.

– To introduce something altogether new would mean to begin all over, to become ignorant again, and to run the old, old risk of failing to learn.

Isaac Asimov

1 Introduction

Surprisingly, a book that is for large parts about self organisation does not organise itself. Therefore, we have structured and outlined it for you. Here we explain our motivations for writing this book, and give a brief overview of the contents.

Aims of this Chapter

This Chapter introduces this book to you. We have used an effective journalistic principle for doing this, namely the five-Wives-and-a-Husband concept – What, Why, When, Where, Who and How. Thus, after reading this Chapter, you know what this book is about, why we wrote it, how we explain and you learn about the what, why we wrote it now, for whom it is intended, and where we found the resources discussed in this book. We finish this Chapter with an overview of the book, containing brief descriptions of the Chapters.

1.1 Motivation

What

This book is about the understanding of the behavior and self organisation of complex systems: systems in which the interaction of the components is not simply reducible to the properties of the components. The general question that we address is: how should systems of very many independent computational (e.g., robotic or software) agents cooperate in order to process information and achieve their goals, in a way that is efficient, self optimising, adaptive, and robust in the face of damage or attack? In order to answer this question, we will look at natural systems that solve some of the same problems that we want to solve, e.g., adaptive path minimisation by ants, wasp and termite nest building, army ant raiding, fish schooling and bird flocking, coordinated cooperation in slime molds, synchronised firefly flashing, evolution by natural selection, game theory and the evolution of cooperation.

What Not

Sometimes it is just as important to tell what a book is *not* about. This is such an occasion. The topics we talk about are easily hypeable: self organisation, collectivity, emergence, agents, etcetera. Usually, things do not live up to their hype. So, for example, we do not explain how to make your organisation, business or company self organising or emergent. Neither do we give a complete theory or model for self organising or emergent sytems. Finally, we have not included traditional (or: mathematical) approaches to model and/or analyse collective intelligence, e.g., differential equations or dynamic systems theory.

Why

Our motivation is straightforward: there is currently no book that does the same as we do here. Although there are approximately a dozen books in the biological, mathematical, social and computer sciences on collective intelligence, such as [38, 49, 57, 100, 108, 110, 111, 118, 134, 187, 231, 302, 332, 373, 387], these are either aimed at modelling and understanding natural phenomenona or developing formal/computational theories; we here aim to eventually *use* these models and theories to design systems – as illustrated by the examples mentioned in the Foreword. For achieving this, we believe that much can be gained if the various disciplines involved in this topic interact with each other. This book is our attempt to contribute some first steps towards our before-mentioned aim.

How

So, how do we achieve our goal of getting you to understand the behaviour and self organisation of complex systems? We basically guide you through a number of different studies about these kinds of systems as such that after reading and learning about these studies, you can make yourself an informed picture about these systems. We make an important distinction between these studies which comes back in the structure of this book. Firstly, we consider *analysis* studies as those studies that have the aim to understand some phenomenon from Nature (including ourselves). Secondly, we identify *design* studies in which one aims to think up a suitable solution (in the form of an algorithm or protocol, for example) for some given design problem.

As design is concerned, let us mention explicitly that we aim to take a *problem oriented* approach. This means that we assume that you have some problem to be solved and are looking for a way to solve your problem. You may be a scientific researcher and such a problem would be abstract, like the Traveling Salesman Problem; or you are a computer network engineer and your problem is to find a reliable and robust communication protocol for very-large loosely coupled networks; or you are a roboticist and your problem is to get your robots cooperating as such to execute some complex task together.

In such a problem oriented approach, one may expect a solid methodology that leads you step-by-step through representing your problem, choosing solution parameters, running an algorithm to come up with a solution and apply this solution to your problem. We take another approach here. In this book we have gathered many related studies for your inspiration allowing you to learn by example. Additionally, we supply you with a number of different kinds of models and techniques, which you can consider as your toolkit. With this toolkit, and the learning experience from going through the case studies that are in this book, you are able to tackle your own analysis or design problem.

Finally, we have chosen to use simulation as our vehicle to research collective intelligence. On the one hand, this is because of fundamental theoretical aspects of the topic itself (which are explained in detail later); on the other hand, it is because of the fact that there is a multitude of simulation studies around on collective intelligence - both for analysis and design purposes.

Who

This book was primarily written with one audience in mind. We expect this book to be used as a textbook by Master and graduate students in the programmes of artificial intelligence, computer science and related areas. In particular, if you have ever played around with any of the artificial life simulations or testbeds and wonder where the science is in it, then this book is for you. Also, if you are an enthousiastic SPORE player and want to know more about the science behind it, have a read through the book.

But also, the contents of the book is relevant for the industry that is interested in eventually developing the future applications that are mentioned above. So, if you are thinking of developing a swarm surveillance system or a swarm of nanobots, read this book. This book sets you up with basic knowledge about how to do simulation studies in this area. The literature list at the end, which is referenced throughout the book, is a valuable contribution containing references to state-of-the-art articles, papers and books about this area of research.

Above all, for any audience, the book aims to deliver an enjoyable and fun playground for discovering and applying collective intelligence with a strong scientific foundation.

When

Although we are currently still far away from constructing either a de/prescriptive or normative theory on the topic of collective intelligence, much research effort is being spent on the topic. Therefore, we believe that the time is right to contribute to the field with this book. Also, specifically on the topic of simulation, the computational power is currently as such that first attempts are being made on constructing large-scale simulated worlds running on distributed network clusters. As a consequence, it becomes increasingly important to systematically set up such studies involving multiple disciplines in order to get tangible results.

Where

For this book, we have gone through a large amount of scientific literature on the topic of collective intelligence. This has resulted in a bibliography with over 350 references, which is included in this book. The bibliography contains over 120 scientific articles, 70 textbooks, and 70 conference papers in the areas of social sciences, biology, economics and computer science. These articles, books and papers are referenced and discussed throughout the book. If you are in an academic environment (with online access to journals and access to a scientific library), you must be able to find the references easily. At various points in the book we have also included internet links to research labs, groups and other resources (e.g., software).

1.2 Overview

This book is organised as follows. The book consists of three main Parts: Theory, Case studies and Howto. Additionally, two appendices contain descriptions of software packages and student projects. Finally, we close the book with the bibliography.

The first Part (Chapters 2 and 3) introduces the involved subtopics of collective intelligence – for example, self organisation, emergence, analysis and design. We also explain why and how to use simulation for researching collective intelligence. Finally, we give an overview of involved models and techniques to be used for analysis and design studies. The second Part (Chapters 4 and 5) overviews 27 research studies into the analysis and design of collective intelligence – including, for example, artificial societies, agent-based computational economics, collective robotics and insect-based computing. The third Part (Chapters 6 and 7) explains how to conduct a scientific simulation study and how to describe it in a report.

Part I

THEORY

– Nature is a collective idea, and, though its essence exist in each individual of the species, can never in its perfection inhabit a single object.

Henry Fuseli

– Emergence is in the eye of the beholder.

Marco Dorigo

2 Collective Intelligence

Take a brief look at the bibliography included at the end of this book. You immediately spot some terms that are used in many of the titles: emergence, self organisation, agents, local/global, micro/macro, etcetera. Because of recent wild growth of these terms and the fact that they are very hypeable, it is important to explain how we exactly understand these terms. This may help you to form a good understanding of them.

Aims of this Chapter

This Chapter provides you with the basics for studying collective intelligence. We describe the concept of collective intelligence and identify a number of properties (or: characteristics) of collective intelligence. The two major contributions of this Chapter are as follows. Firstly, we present a number of step-by-step *recipes* that you can use to model collective intelligence. You can use these recipes when getting started with your particular study of interest. Secondly, we make a division of the kinds of research studies of collective intelligence, according to which we structure the rest of the book: analysis and design studies.

2.1 What is it?

Google.com returns[1] about 1 million hits on the search term "collective intelligence". A quick browse through the first 20 or so hits shows that it takes all kinds with an interest in it: universities, institutes, companies, communities, new-age networks, awareness organisations, etcetera. The concept of collective intelligence therefore has a wide interpretation bandwidth, ranging from strict and pragmatic definitions (as explored below and throughout this book) to the other end of the bandwidth spectrum (e.g., "Collective intelligence is the capacity of human communities to evolve towards higher order complexity and harmony, through such innovation mechanisms as differentiation and integration, competition and collaboration",

[1] Measurement date: January 2007. The term "self organisation" OR "self organisation" yields approximately 1.2 million hits. The search term "complex systems" gives you exactly 10.5 million hits.

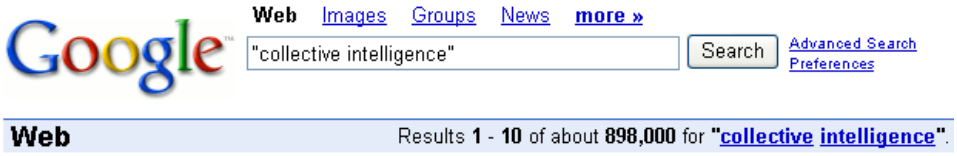


Figure 2.1: The search term "collective intelligence" yields approximately 1 million hits.

"Collective Intelligence: exploring the next step in human evolution", "Collective Intelligence is [...] dedicated to improving the efficiency of social ecosystems and accelerating the flow of capital to good", "Collective Intelligence, The Invisible Revolution").

A working definition that covers Collective Intelligence (CI) closest to how we use it in this book is coined by the Massachusetts Institute of Technology (MIT) Center for Collective Intelligence[2]:

Groups of individuals doing things collectively that seem intelligent

which actually expresses more and/or less than our intuition may say (but that is why it is a definition). Still, it gives us something to work with (hence, a *working* definition). Let us look at the terms "Collective" and "Intelligence" separately. For *Collective*, one can say that it is anything where multiple, interacting entities are involved. Alternatively, it means, a number of persons or things considered as one group or whole. For *Intelligence*, we can use numerous definitions. Within the study of artificial intelligence, the Turing definition is commonly used: if something or someone successfully passes the Turing test[3], then one is considered intelligent. Psychologists have for long been investigating intelligence and defining it, e.g., "a very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience. [...]" [141]. The combination of these two fundamental terms leads to interesting and fascinating ideas, models, theories and applications.

An interesting comment made by MIT-CCI is that collective intelligence itself is not a natural phenomenon, but rather a *perspective*. Like "team work" and "leadership", it is not something that can be directly observed as such. Instead, it needs the following ingredients: group of actors, set of resources, results of the actors' behaviour, and a way to evaluate these results. Then one can analyse how intelligently the group of actors acted with the resources available to them. This gives us a first start for thinking about collective intelligence, and we gradually work out this perspective in more detail throughout the book.

Noteworthy, NASA has also used the term collective intelligence [357, 356, 389, 388] to name a set of interacting individual learning algorithms that is designed in an automated fashion and forms an aggregate problem solver, called COIN. Later in this book, we explain the COIN framework in more detail. The framework has been successfully used for routing of internet traffic.

The big picture of collective intelligence includes fundamental ideas like evolution [64], culture [67] and order [187, 188]. Understandably, there are thus many different terms, concepts, frameworks and processes, currently under investigation, that are closely related to collective intelligence as defined above. Of the related research streams, we briefly touch

[2]http://cci.mit.edu/
[3]http://en.wikipedia.org/wiki/Turing_test

upon a number of them here: self organisation, complex adaptive systems, multi-agent systems, population-based adaptive systems, swarm intelligence and swarm engineering.

Self Organisation (SO) According to Farley and Clark [116], "A self organising sytem changes its structure, depending on its experiences and its environment". The organisation process itself is autonomous, thus there is no central coordination that manages the organisation. A working definition of self organisation is suggested by de Wolf and Holvoet: "Self organisation is a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control." [71, 72].

Complex Adaptive Systems (CAS) The Santa Fe Institute[4] in Santa Fe, New Mexico (USA), started in 1984 and dedicated to the research of complex systems – the study of *complexity science*. A complex adaptive system is a complex, self similar collection of interacting adaptive agents. Moreover, it is a key notion in complexity science. John Holland [164, 165, 166] observed CAS in many different contexts (for example, economics, biology, society) that eventually resulted in the theory and practice of evolutionary computation. Holland also set up the ECHO simulation tool[5] to investigate CAS behaviour.

Multi-Agent Systems (MAS) The field of artificial intelligence had throughout the 1960s till the 1980s developed the idea of a knowledge-based system (or: expert systems) in which expert knowledge was formalised and further used for decision support, consultation, etcetera. The systems worked in isolation and were passive with respect to the one operating them. When a number of these systems were combined with each other, a *multi-agent system* came into being. Such a system is "composed of multiple interaction software components (or: agents), which are typically capable of cooperating to solve problems that are beyond the abilities of any individual member" [391]. Later in this book we look at MAS in more detail.

Population-based Adaptive Systems (PAS) Population-based adaptive systems are systems that consist of many (thousands or millions) units that have adaptive, reactive, and self organising abilities. These systems resulted from the synergy of two research areas, namely intelligent multi-agent systems (as explained above) and evolutionary methods [103]. Both research fields contribute to each other in the context of PAS: the pro-activeness of intelligent agents is used in evolutionary systems (where the individuals typically are only passive) and the individual adaptive capabilities in evolutionary systems can be used in intelligent agents, allowing these to learn and adapt when necessary.

Swarm Intelligence (SI) Swarm Intelligence is an AI technique for search/optimisation and based on the idea of "swarming" (as observed in bird flocking or fish schooling). The term was initially coined by Beni and Wang [31] in the area of cellular robotics (i.e., robotics based on the theory of cellular automata) and has since then been worked out by Marco Dorigo [38], inventor of ant colony optimisation (ACO) – a metaheuristic for combinatorial optimisation problems. In SI, the intelligence of swarming as observed in nature is used for solving practical coordination problems. It is extremely successful in the area of intelligent

[4] http://www.santafe.edu/
[5] http://www.santafe.edu/projects/echo/

robotics; and the related ACO has also been used fruitfully for more abstract problems, like the Traveling Salesman Problem.

Swarm Engineering (SE) On a somewhat smaller scale than the research streams above, in terms of number of researchers working on it, Swarm Engineering is "a process where one creates a swarm of agents which complete a predefined task" [189]. The process consists of two steps: firstly, one generates a swarm condition (guiding the generation of a suitable swarm); secondly, a behaviour (set) must be generated that satisfy the given swarm condition. Kazadi, in [189], implemented this process and applied it to two robot application (plume tracking and object clustering), and the Traveling Salesman Problem.

2.2 Properties

What distinguishes collective intelligent systems from other systems? In the literature, one can find various answers to this question. For example, Bonabeau et al. [38] identify four basic ingredients, based on researching self organisation in social insects: positive feedback (amplification), negative feedback, amplification of fluctuations (e.g., random walks or errors), and multiple interactions. Heylighen [161] enumerates the following properties: global order, ditributed control, robustness, feedback, bifurcations, adaptation and optimality through perturbations. Holland [165, 166] identifies properties and mechanisms of complex adaptive systems: aggregation, tagging, nonlinearity, flows, diversity, internal models and building blocks. Mondada et al. [239] identify robustness, flexibility, scalability and performance as the key characteristics of socially inspired computing.

We add yet-another-list to these lists. We base ourselves on our experience with a number of research studies that we conducted and the study of similar lists like the ones above. We petnamed our list AEGIR[4] (short for AEGIRRRR), which stands for Adaptivity, Emergence, Global-local, Interaction, Rules, Redundancy, Robustness and Randomness.

Adaptivity – To adapt literally means 'to fit to', usually referring to (necessarily) changing one's own structure to deal with one's environment. In our context, this means that either an individual changes itself if necessary or the whole system changes itself. The former implies the latter, but not necessarily the other way around. On an individual level, it means that an individual changes its own behaviour. If this behaviour is specified by, for example, rules (see below), than (parts of) these rules are subject to change. But even if the individuals do not change, the system as a whole may still be adaptive. Consider, for example, an ant colony in which the ants do not change their own behaviour (e.g., for food foraging), but where the whole colony can change itself if so required: experiments have shown that a colony forages efficiently even when the food sources change over time [140].

Emergence – This is arguably the most convoluted property in this list; its simple meaning is "the whole is more than the sum of the parts" [63]. In complex systems, *something* happens when you move from the lowest abstraction level (individual) to the highest abstraction level (system) – and this something is often surrounded by "mystical overtones" [63]. And so a scientific dilemma emerges: when you have discovered the science of this something, the mystery disappears. It figures that Science may have a problem with this. We use the following working definition, coined by de Wolf and Holvoet: "A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions

between the parts at the micro-level. Such emergents are novel with respect to the individual parts of the system." [71, 72].

Global-local – We distinguish two aggregation levels: the *local* level that concerns the individuals in the systems, and the *global* level concerning the system as a whole. We need this distinction to position, for example, adaptivity and emergence. As mentioned above, adaptivity can happen on the local and/or global level. For the former, individuals may have adaptive behaviour, as that they adapt themselves to changing circumstances. For the global level, the system as a whole is able to adapt itself if responsive to changes in its surroundings. The latter does not necessarily mean that the indivuals have to be adaptive as well. Emergence is a concept, as explained above, that concerns going from the local to global aggregation level: at some point if you are going up the levels, "the system becomes more than its indiviual parts". Moving between these aggregation levels is a very fundamental scientific concept and subject of many research studies, for example Yamins who constructs a local-to-global theory for multi-agent systems [393, 394].

Interaction – Systems with many (thousand or millions of) individuals thrive on the way that these individuals interact with each other. For instance, how much do they interact, what do they interact about, do we only consider communication or are there other kinds of interactions? In a complex system, one cannot isolate a single part and analyse its behaviour to find out about the system behaviour; instead, one has to consider the interactions as well as the individual behaviours [368]. Other lists of properties explicitly consider *flows* to be a key property. This means that information flows through the system and some small cause somewhere in the system may eventually have very large consequences. We consider these flows as characteristics of interaction, but not separately. In combination with the properties of emergence (see above) and randomness (see below), we believe that the concept of flows is covered.

Rules – The most fundamental form of describing the behaviour of an individual (or whole system) is to use rules. Such rules are implications between inputs (observations) and outputs (actions). Such implications can be respresented in many different ways, e.g., if/then rules, trees or networks. The best representation depends on the particular domain and problem in hand.

Redundancy – This means that the same knowledge is represented in a number of different places in a system. This knowledge may be actual knowledge (e.g., rules), but it might also be information itself. This may be very straightforward: every individual works with the same rule set. In this way, the rules (considered to be knowledge) are everywhere in the system. It may also be more complex, in that there is knowledge or information that a number of individuals (but not everybody) share.

Robustness – A nice consequence of redundancy is that the system becomes robust. If the knowledge is represented at multiple places in the system, it means that if the system loses parts, chances are that the knowledge is still maintained somewhere in the system – making the system robust in case of damage or attack.

Randomness – Complex systems typically have some element of randomness in it that causes the system to show self organised critical behaviour: these systems exist somewhere on the edge of chaos where on one side there is chaos and disorder, and on the other side there is structure and order. Without an "external spark" (or: an inifitesimal driving rate [2]), such

Control Loop	
1:	initialise individuals
2:	initialise world
3:	while true
4:	for each individual
5:	observe the world
6:	perform an action
7:	end for
8:	world: determine new observations
9:	world: process costs and benefits for all individuals
10:	end while

Table 2.1: The control loop.

systems would be in balance: 1) litte disturbances have no consequences, 2) the system's response is proportional to the impact (action = reaction), and 3) only dramatic disturbances can cause state transitions. But with it, these systems get out of balance, moving over the edge of chaos; for example, the so-called "butterfly effect" can be observed in these systems[6].

2.3 Modelling

To be honest, in our experience, most researchers *just start somwhere* when constructing models of collective intelligence. Fascinated by the impressive list of properties of self organising systems as mentioned above, they enthousiastically start programming their simulations, write the most beautiful visualisations, and so on. Nothing beats enthousiasm. But some structure may be benificial if we want to learn systematically about these kinds of systems. Here, we give some fundamental recipes (or: algorithms) that you can take as a starting point for modelling collective intelligent systems.

We make a conceptual distinction between the *world* (or: environment) and *individuals* inhabiting that world. Sometimes these terms are really as straightforward as that, for example, an ant colony that learns to live in some hostile environment. In other, more abstract, studies, the world may actually be some search space defined by some optimisation criteria and the individuals inhabiting this world may be solutions for value optimisation within this search space.

We first present a basic recipe that sets you up with a simple world and simple individuals. After that, we present a number of extensions on this basic recipe that allow you to model: internal models, diversity, non-determinism and adaptivity. Although we present each as a separate extension of the basic model, it is possible to combine all extensions into one single model.

Note that the recipes will deliver you some *data sets*, e.g., a set of actions that an individual may choose to execute at some suitable point. But this does not give you an *operational model* yet, i.e., something in which you can press a Run button in order to execute the model. Therefore we assume the control loop as shown in Table 2.1. This loop basically says that, after initialisation, the execution (or: simulation) runs forever, and every iteration the individuals look around and perform some action, the world is updated and individuals are

[6] http://en.wikipedia.org/wiki/Butterfly_effect and http://www.imdb.com/title/tt0289879/

Recipe: **BASIC**	
1:	Determine action set for all individuals
2:	Determine observation set for all individuals
3:	Determine action → observation functions
4:	Determine costs for individuals for functions from 3
5:	Determine benefits for individuals for functions from 3
6:	Determine observation → action functions for all individuals

Table 2.2: The BASIC recipe.

re-evaluated. The control loop in combination with one of the recipes presented below, give you a model that you can execute – for example, a so-called microworld [286].

This control loop is definitely not the only or always best way to exectute your model. Neither is it ready-to-use. For example, one must assure, when desired, that the individuals do-their-thing in parallel. Also, one must arrange for the world to be updated correctly and timely such that one individual does not see a different world as another individual in the same iteration. Or, your individuals may perform multiple actions per iteration. It may even be that an iteration-based control loop is not the correct model for your particular study (however often it is used in similar studies). Thus, we take this control loop such that we have something to refer to when presenting the recipes, but be aware that it may take some playing around with before it suits your needs.

2.3.1 Basics

The basic recipe for modelling CI is shown in Table 2.2. It concerns 6 steps that take you from determining what individuals can do (actions, line 1) and can see or hear (line 2) up to deciding what they should do when (line 6).

Firstly, one decides what actions an individual can undertake. An action is something that causes a change in the environment. This may be non-trivial; for example, if you move one step forward, then this is considered to be an action because it changes your position in the world. In this first modelling step, the only thing you actually do is *naming* the actions: you do not need to specify results or costs. You will do this later, in the other steps. However, you may want to parametrise your actions; that is, extending them with variables, for example, `move(`*`s`*`)`, where *s* denotes the speed at which one moves.

Secondly, you determine the observations of the individuals. This is the available information that an individual may receive from the world (including other individuals). Note that it may be that your individuals actively look around themselves – in which case observation (or: observe) must be an explicit action. It may also be that your environment supplies your individuals (each iteration) with fresh information – then it is not necessarily an explicit action, but rather a step in the control loop of your world. Like with actions, the only thing you do here is specifying a set of observations by *naming* them. Just as for actions, it may be useful to parametrise observations: for example, `position(`*`x, y`*`)`, where *x* and *y* are coordinates.

Thirdly, we model the world – however, this might be somewhat counter-intuitive. We define a set of functions that map actions to observations. Consider the `move-forward` action. Assuming that you can observe your own position in the world, the result of this action is that in the next iteration you receive an observation with updated information about your position in the world. Note that at this point is an opportunity to decide whether your individuals

Recipe: **BASIC + internal models**	
1:	Determine action set for all individuals
2:	Determine observation set for all individuals
3:	Determine action → observation functions
4:	Determine costs for individuals for functions from 3
5:	Determine benefits for individuals for functions from 3
6:	Determine (internal model) set for all individuals
7:	Determine observation → (internal model) functions for all individuals
8:	Determine (internal model) → action functions for all individuals

Table 2.3: The BASIC + internal models recipe.

may perform multiple actions per iterations or not. Also, by means of these functions you can let an attempt to do an action fail or succeed. For example, you may decide to `move-forward`, but the function that maps this action to a new observation may let it fail – your position is still the same in the next iteration. Thus, in general, individuals attempt actions, and the world lets these actions fail or succeed.

Fourthly, doing something always comes with a cost. Thus the functions that were just defined, are assigned *costs*: an individual has to pay when performing an action. Herefore, an individual is assumed to have some property that expresses its *fitness*. Costs are deducted from this fitness. Fitness is a difficult concept. One might model it rather straightforward as it being energy that is accumulated and used throughout the individual's life. But, in evolutionary sense, it may also be the number of offspring an individual brings forth (as suggested by Holland [164]). This is more difficult to model, but sometimes necessary. Costs then influence this fitness negatively (and possibly indirectly).

Fifth, successful actions are beneficial for you – otherwise you would not undertake them. This means that you earn when an attempted action is succesful. Therefore, the functions from line 3 are associated with benefits. Note that this is not necessary, in which case the benefit can be 0. It may, for example, not make sense to assign benefits to moving. Benefits influence the fitness positively.

Sixth, you specify the behaviour of the individuals. This is done by means of a number of functions mapping observations to actions. You can understand these as being simple `if-then` rules. But actually, this can be any representation as desired, for example, a decision tree or bayesian network – as long as observations come in and actions come out. Note that in these models it is only possible to model the *outside* (or: behaviour) of individuals. Because we only allow actions to be caused by observations (line 6), this prevents you from modelling individuals with a memory or internal state (e.g., hunger, anxiety, and so on). Although this modelling is arguably the simplest (scientifically speaking), you often want to model the *inside* of your individuals too. Therefore you can extend your individual with internal models, which we explain next.

2.3.2 Internal Models

You may notice quite quickly, that you run into trouble with the basic model. It is, arguably, too basic. For example, sometimes you just need your individuals to *remember* something, or the individuals may not always behave the same *all the time*, or they may have *roles* that determines their behaviour and it is a pain to model these enumeratively in single observation-

Recipe: **BASIC + diversity**	
1:	Determine action set *for each individual*
2:	Determine observation set *for each individual*
3:	Determine action → observation functions
4:	Determine costs for individuals for functions from 3
5:	Determine benefits for individuals for functions from 3
6:	Determine observation → action functions *for each individual*

Table 2.4: The BASIC + diversity recipe.

action mappings. Therefore this extension allows for the system's individuals to have an *internal model* – see Table 2.3. The term was originally coined by Holland [164] and denotes a model that takes some input, filters it into patterns which can be used to change itself. An individual that has such a model can anticipate results if it receives input that it has received before. Although our internal models here cannot change themselves (this is dealt with later when we discuss adaptivity), they share much with the ones from Holland.

In Table 2.3, we have replaced line 6 from the basic model with lines 6–8. Basically, we have created an intermediate variable, called *internal model*, that is in between observations and actions. Such a model is something *inside* the individual. For example, it may represent a memory or some specific role. Note that the implementation of this concept is actually not as straightforward as with observations and actions. It depends very much on the specific goal that you have in mind. In case of a memory, it might be a data structure that is gradually filled when observations come in; for different roles, it may be a container with different role descriptions, which are activated based on particular observations.

2.3.3 Diversity

The basic recipe assumes that all individuals behave the same: all individuals have the same action set, observation set and policy functions. But this may not be correct for your study. Hence, you may want to model *diversity*. This requires to identify possibly dissimilar action and observation sets and policy functions for the (subsets of) individuals. The resulting recipe is shown in Table 2.4. Here, lines 1, 2 and 6 are not defined for all individuals (like in the basic recipe), but *for each individual*.

2.3.4 Non-determinism

Nothing in the real world is certain – and that is for certain. Therefore making models of the real world requires modelling uncertainty. We can make our models to allow for this uncertainty by including *non-determinism*: not every state of affairs, including human event, act, and decision is the inevitable consequence of antecedent states of affairs.

Table 2.5 shows that we can model non-determinism by assigning probabilities to the functions that we have identified for mapping actions to observations and vice versa. We are aware of the fact that this is a very pragmatic approach and can be scientifically argued. The idea is that on the environment level (lines 3 and 4) uncertainty exists as such that undertaken action may not always have the intended results. For example, a robot attempts to `move-forward`, but some slippage on the surface causes the robot to move somewhat more to the right rather than forward. This can be modelled by means of assigning a probability to

the (world) function that takes `move-forward` as input. On the individual level (lines 7 and 8), uncertainty manifests itself if individuals do not always undertake the same actions in the same situation – even without allowing for internal models (discussed above) or adaptivity (discussed below).

Note that we do not prescribe the origins of these probabilities. The probabilities on the environment level may have come from domain expert knowledge; or, your model may be directly coupled to physical environment (for example, in case of collective robotics) in which case these probabilities may fluctuate over time. On the individual level, if your individuals are non-adaptive, then these probabilities are hardwired (determined beforehand) into the system. They may also come from domain knowledge. Finally, be aware of the fact that non-determinism introduces inherent complexity in your model, but which is sometimes necessary.

2.3.5 Adaptivity

Adaptivity is the most complicated extension that we discuss here. It concerns the ability of individuals to change themselves if the environment requires so. Table 2.6 shows the recipe for adaptivity: we have replaced line 6 of the basic recipe. Now, line 6 says that you have to choose and implement a mechanism that actually *generates* the functions that map observations to actions. A number of these mechanisms is explained later in this book: evolutionary computation, neuro-evolution, co-evolution, among others. Some mechanisms already specifically take collectives into account: the Collective Intelligence (COIN) framework of Tumer and Wolpert or the Particle Swarm Optimisation method [99]. As mentioned, we discuss these mechanisms in detail later.

There may be many reasons to include adaptivity in your model. For one, you may have observed it in nature (e.g., an adaptive ant colony) that you want to model in order to understand it better. Or you wonder how some rule-sets that you discovered (about how birds flock, for example) have evolved as to what they are now.

In this way, the mechanisms as mentioned above (and explained in detailed later) are generally used for completely different purposes: mostly in the context of search or optimisation. For example, evolutionary methods are used as search meta heuristic to find optima in very large search spaces. Thus although the inspiration may have come from nature (evolution), the mechanism is not implemented/simulated to find out more about this phenomenon; rather, it is used as a technique for exploring large abstract search spaces in order to solve some problem. This brings us to a sharp distinction that we will make between analysis and design (problem solving studies), which we discuss next.

Recipe: **BASIC + non-determinism**	
1:	Determine action set for all individuals
2:	Determine observation set for all individuals
3:	Determine action → observation functions
4:	*Assign probabilities to each of the functions from 3*
5:	Determine costs for individuals for functions from 3
6:	Determine benefits for individuals for functions from 3
7:	Determine observation → action functions for all individuals
8:	*Assign probabilities to each of the functions from 7*

Table 2.5: The BASIC + non-determinism recipe.

Recipe: **BASIC + adaptivity**	
1:	Determine action set for all individuals
2:	Determine observation set for all individuals
3:	Determine action → observation functions
4:	Determine costs for individuals for functions from 3
5:	Determine benefits for individuals for functions from 3
6:	Choose and implement mechanism that generates observation → action functions for all individuals

Table 2.6: The BASIC + adaptivity recipe.

> "Attempting to discover the nature of life (or some other biological phenomenon) through digital naturalism is analagous to attempting to discover the laws of aerodynamics which govern flight through drawing many different pictures of imaginary birds, imaginary flying insects, etc. Once many such pictures of 'flight-as-it-could-be' have been rendered, one might spend years searching for principles which reval the nature of flight. [...]"

Figure 2.2: Bullock on Model Class Artificial Life (from [44, p. 22]).

2.4 Studying

Holland thought up the idea of *reproductive adaptive plans* which were structures that adapt themselves by means of some evolutionary mechanism [164]. Holland also suggested that these plans were actually hypotheses if considered in the context of natural systems, and algorithms in artificial systems. Dorigo makes a similar distinction: swarm intelligence exists in a natural form (where a natural phenomenon is of concern) and in a computational form (where one programs algorithms). Finally, in an attempt to categorise the research on artificial life, Bullock [44] identifies High Class, Model Class and Work Class Artificial Life (AL). High Class AL is concerned with the philosophy of artificial life; Model Class AL is concerned with the validaty of scientify theory; and Work Class AL involves the production of useful artifacts, most naturally understood as engineering. It is interesting to note that people may feel strongly about these different types, as Bullock demonstrates in the text in Figure 2.2.

It is important for us to also make such a distinction when it concerns collective intelligence. A good reason for this is that studying collective intelligence is versatile and interdisciplanary, and different researchers from different disciplines may have different study objectives. We describe here the distinction that we make between analysis and design studies. The case studies that are discussed later in the book are also structured based on this distinction.

2.4.1 Analysis Studies

The purpose of analysis studies is to learn about and to get better understanding of phenomena as observed in nature, including human nature. Some of the scientific disciplines undertaking such studies, in the context of collective intelligence, are the social sciences (e.g., [111, 110, 135, 211]), biological sciences (e.g., [49]), and economic/political sciences (e.g., [15]).

There are three simple steering behavior parameters in the BOIDS simulation of bird flocking [287]: seperation, alignment and cohesion. With specific settings of these parameters, flocking as observed in the real world can be simulated. The design-step that we propose means that we search for an appropriate parameter setting in order to get some desired group behaviour. For example, we may want to have the flock moving in a straight line, a circle, or a square. This design problem raises a number of questions. In increasing order of complexity, these are some questions. How do we tune seperation, alignment and cohesion to get this behaviour? Can we even get this behaviour with only tweaking the given parameters? If not, would it work to consider a more heterogeneous population where the parameters can vary per individual? If not, will it work to introduce new parameters (which ones?) With which we can get the desired behaviour? Currently, it is unknown how to define this problem, let alone to solve it.

Figure 2.3: Towards designing collective intelligence.

2.4.2 Design Studies

The purpose of design studies is problem solving (in its broadest meaning). This can mean to find a solution in the search space of an abstract formal problem (for example, solving the Traveling Salesman Problem) or to find a coordination protocol that lets a number of individuals cooperate with each other (for example, a set of robots that has to perform some collective task.) Design studies are mostly the terrain of computer scientists, artificial intelligence researchers and engineers.

In the context of collective intelligence, it is sometimes difficult to distinguish between analysis and design. Consider evolution for example. You may implement a model that has evolution in it for some purpose that is not explicitly clarified. A biologist using this model may want to study the mechanism of evolution itself, while a computer scientist is interested in using the mechanism as a heuristic within some optimisation process. For the biologist, it is important that the way that evolution is modelled is as realistic as possible: the model is the means and evolution is the ends. For the computer scientist, this is exactly the opposite: he can tweak and change the mechanism as much as he wants, because evolution is the means and the model (here: problem/solution) is the ends. However, note that the computer scientist may be solving a particular problem for which it is important that the *simulation* is realistic (even though the mechanism itself not). For example, if you want to develop robot controllers that are eventually implemented in real robots, the simulation should model the environment as good as possible.

The design studies that we have included in this book, are design studies as explained above: for instance, collective robotics, peer-to-peer network protocols, data clustering. In the last part of this Section, we want to briefly touch upon a long term aim: towards really designing collective intelligence. Consider the box in Figure 2.3 (a more practical example is in Figure 2.4). In this example, we show what we eventually aim for: some kind of engine that takes in some collective task description and outputs a solution that prescribes the (adaptive) behaviour of the involved individuals. As the example says: we do not even know how to define (represent) this problem, never mind how to solve it – this is currently cutting-edge research on desiging collective intelligence.

Assume that you are on a team to map out illegal wood cutting activities in the Amazon area in South America. Each member of the team is equipped with a light-weight plane with GPS navigation. The whole mission will take some given number of days. There is no central meeting place, neither at the start nor at the end of the mission. Each day you fly over some area and identify wood-cutting activities by writing down the GPS coordinates of your observation. You identify *all* such activities; after the mission, the reported activities will be compared with entries from a database with legal wood cutting activities. While flying, you can fly at either a relatively high or low altitude. Flying high means that you cover more area, but the information you collect is less detailed; flying low means the opposite. You will stay overnight at the place where you end up just before dusk. We assume, however unrealistic, that there is sufficient fuel supply throughout the Amazon area. Every night, the (partial) maps of all members are collected through telecommunication by some central authority, the collective map is updated and the resulting map is communicated back to all members. This is the only possible communication between the members during the mission.

Ask yourself two questions:

- What best strategy can I think up that will yield a good collective map as quickly as possible?
- How can we effectively use computer simulation in order to decide about these strategies?

Figure 2.4: A motivating example.

2.5 Summary

This Chapter introduced the concept of collective intelligence to you. We sketched what collective intelligence means 'out there' and we have explained what collective intelligence specifically means to us. Properties of collective intelligent systems were identified that makes these kinds of systems different from other ones. For modelling collective intelligence, a number of recipes were presented that give you a headstart when you want to model the basics as well as internal models, diversity, non-determinism and adaptivity. Finally, we made a distinction between analysis and design studies, where the purpose of the former is to understand natural phenomena and of the latter is problem solving.

Open Issues We do not postulate new theories or methodologies in this textbook. However, we consider it useful to mention the following two open issues concerning working towards such new theories and methodologies. Firstly, we need to be able to *formalise* collective intelligence modelling problems. With such a formalisation (as Holland did for evolutionary computing in the 1970s [164]), we are able to pinpoint research issues and relate different aspects of collective intelligence together. Secondly, specifically for the design of collective intelligence, we need to get a grasp on what kinds of problems we can tackle eventually with these kinds of techniques. Can we identify problem characteristics and requirements such that we know which techniques may solve these problems? For this, we need to make clear comparisons between the models and solution algorithms that are presented in this book. In one situation, one algorithm may perform best; in others an other one may perform best.

– Reality is nothing but a collective hunch.

Lily Tomlin

– The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work.

John von Neumann

3 Simulation, Models and Algorithms

How to study these collective intelligent systems then? Where to start? In general, for one, simulation is often used as the vehicle for the research. Therefore you must know why to use simulation and how to use it. However unfortunate, researchers may 'just start somewhere' for their investigation of such systems. One may benefit very much from a toolkit to start out with – just like chemists and biologists use microscopes, vats, Petri dishes, test tubes, etcetera. For computer science, such a toolkit is more abstract. Here, we supply you with such a toolkit – a collection of models and algorithms to analyse and design collective intelligence. Note that, very roughly, one may say that you need models in analysis studies and algorithms in design studies. However, you may want to investigate some adaptive process for which you use an algorithm; similarly you cannot have an algorithm without some kind of model. You need to study both, independent of your study.

Aims of this Chapter

This Chapter contains concise descriptions of models and algorithms involved with modelling and desiging collective intelligence. For both the models and algorithms, note that we only describe so-called artificial intelligence models and techniques (multi-agent based systems, rule-based systems evolutionary methods, neuro-evolution, etcetera) and have not included more traditional, mathematical, frameworks like system dynamics theory, differential equation models, etcetera. We start this Chapter with explaining simulation: what is it, what is the scientific relevance, why do we use it, and so on.

3.1 Simulation

Traditionally, one distinguishes a simulation from a model and a specific system [214]. In other words, you can analyse the (mathematical) model of a physical system by means of simulation. A *system* is a collection of entitities that act and interact together towards the accomplishment of some logical end [303]. Such a system can at any time be in some *state*

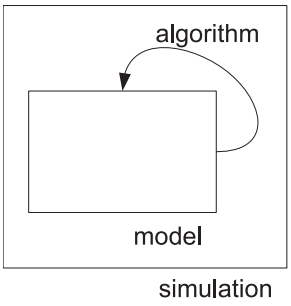


Figure 3.1: Diagram showing relation between simulation, model and algorithm.

that is a collection of variables describing the system at that time. The investigation of the workings of such a systems can be done by either experimenting with the actual system or with a model of the system. Such a model is representation of the system and can be studied as a substitute of the real system.

Let us explain our use of the terms simulation, model and algorithm with respect to each other. Figure 3.1 shows a simple diagram illustrating these terms and their relations. We that every simulation contains a model – stronger positions say that the simulation actually *is* the model, as explained later in this Chapter. In addition, the simulation may contain an adaptive component, by us called the algorithm. Note that this algorithm is *not* the control loop of the simulation. The algorithm can affect the components of the models on all possible aggregation levels: within the individuals (individual learning), over or between the individuals (evolutionary or social learning), in addition to the individuals (some kind of oracle), on the system level, etcetera.

To make this terminology somewhat more concrete, consider an artificial society in which the task of the individual agents is solely to survive on the basis of collected resources that exist in the environment (like Sugarscape [111]). The model that you use for this simulation can be simple rule-set to be executed by an agent: each agent is a rule-based system. If the agents are able to learn (some way or another), an adaptive algorithm is added to the simulation that makes this possible. For example, each of the agents may use reinforcement learning to dynamically change its rules in order for the best rules to eventually surface. The simulation then adds the control loop to the whole system (model + algorithm) and allows you to switch on the model and observe its development over time.

3.1.1 Types of Simulation

According to [214], there are three different dimensions over which one can categorise simulations, i.e., dynamism, determinism and continuity:

- *Static vs. dynamic* – A static simulation model is a representation of a system at a particular time. A Monte Carlo simulation is an example of a static simulation. Dynamic simulation models involve systems that evolve over time.

- *Deterministic vs. stochastic* – A simulation model is deterministic if it does not contain any probabilistic components: the output is *determined* only by the system inputs and the relations between these in the system itself. Deterministic simulations are not trivial: for example, modelling a chemical reaction by means of a set of complicated differential

equations (that are analytically intractable) is far from this. A stochastic simulation includes at least one probabilistic (i.e., random) element. Stochastic simulations produce output that is random itself, to be treated only as an estimate of the true characteristics of the model.

- *Continuous vs. discrete* – A discrete system is one for which state variables change at separated points in time. In a continuous system, these variables change continuously with respect to time. Few systems in practice are fully continuous or discrete. The decision to use a continuous or discrete model depends on the object of study.

Following [214], we call a simulation that is dynamic, stochastic and discrete a *discrete-event simulation*. Dooley [88] identifies two other different types of simulation: *system dynamics* and *agent-based simulation*. Although this typology was originally only considered within the area of organisational modelling by Dooley, we can generalise it to simulation studies in general. To further add to this general list, we also want to mention continuous, combined discrete-continous and Monte Carlo simulations. We do not discuss these types of simulations here but refer the interested reader to [214, p. 87–91]. We discuss discrete-event, system dynamics and agent-based simulation here.

Discrete-Event Simulation involves modelling a system as a set of entities evolving over time according to the triggering of events. This is a "bottom-up" approach: one defines the lower levels (or: individual parts) of the simulation (e.g., single events) and the overall behaviour of the simulation model comes forth from the interaction between the indiviual parts.

We describe this type of simulation in somewhat more detail, based on [214, Chapter 1]. Firstly, an event is an instantaneous occurrence that may change the system's state. Such events happen at certain points in time. One of the components of a discrete-event simulation is a *simulation clock* – this clock advances the simulation time as the simulation progresses. There is not neccessarily a relationship between simulation time and the time needed to run a simulation. In general, it is best to explicitly distinguish these two from each other in order to make reproducability of the results (on another computer) possible. There are two possibilities for advancing the simulation time: *next-event* time and *fixed-increment* time advance. For both, at any (real-)time, a discrete-event simulation has an *event list* containing the next time when each type of event will occur. If the advancement is fixed-increment, then when an event occurs, a new system state is computed and the clock is advanced to the time of a new most imminent event. This way, the system advances from event to event, and it skips periods of inactivity. The successive jumps are generally unequal in size. For fixed-increment time advance, the clock is advanced in increments of some pre-determined interval size. After each update of the clock, it is checked if any events occurred during the last time interval; these events then take place and a new system state is computed. If you simulate an economic system, in which the periods are normally per year, then fixed-increment is a good option.

System Dynamics involves identification of key "state" variables that define the behaviour of a system and relates the variables to each other through differential equations. In contrast with the other two types of simulation described here, system dynamics is a "top-down" approach: the differential equations describe the overall behaviour of the system without

necessarily going into the details of the individual parts. It was Forrester in the 1960s that founded the system dynamics approach, when he started applying principles of electrical engineering to every day kinds of systems [119].

Agent-Based Simulation involves "agents" that are the central primitive blocks of the simulation, and the indivudal behaviours of these agents and interactions with other agents are essential. Dooley [88] mentions correctly that complexity science gains increasingly more acceptance as the normative description of living systems, whereas agent-based simulation has the potential to become the dominant simulation-modelling paradigm for such systems. This is again a "bottom-up" approach to simulation: instead of events, the individual parts are agents and the overal system behaviour comes forth from the interactions between these agents. Note that we present agents at this point as an abstract entity of a simulation – the object of simulation is not necessarily a "multi-agent system". Vice versa, it is just as well possible to simulate a multi-agent system in a discrete-event simulation. Later in this Chapter we describe the multi-agent system model in more detail and return to this issue.

3.1.2 Methodology

The process of setting up a simulation study involves steps of problem formulation, data collection, model definition, experimental design, running the simulation, output data analysis and reporting of results [214]. Throughout this process, intermediate validation steps assure that the simulation model corresponds with the actual system under investigation.

Model definition concerns setting up a conceptual model of the actual system with respect to project objectives, performance measures, data availability, computer constraints, etc. Many tools exist nowadays to support modelers with this activity. Depending on specific user interest, one may choose from a variety of simulation languages and software packages. We provide a brief overview of such languages and packages later in this book. These tools provide natural frameworks for model construction. As such, they are based on formal system descriptions and include concepts like entities, states, events, time, variables, etc. The model definition includes validation of the simulation model: "the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" [214]. Validation is essential for assuring that the simulation model corresponds with the actual system. Various validation techniques exist, of which we mention one in particular. By letting the simulation program generate a trace, i.e., the state of the simulated system (e.g., state variables, statistical counters), it is possible to compare the states with hand calculations to check the validity of the program.

Analysis of output data is in practice still rather undervalued as the simulation process is concerned. Much time goes into model development and programming, rather than addressing the generated output results appropriately. A commonly made "error" is that a single run is made of some arbitrary length, supposedly to provide insight into the workings of the actual system. Instead, suitable statistical techniques must be used to design the simulation experiments and analyse the results. Since the output processes of simulations are almost all nonstationary and autocorrelated [214], classical techniques may not always be applicable. Still, statistical techniques exist for analyzing a single system (termination conditions, mean and average estimations, etcetera) and for analyzing multiple systems (measuring response differences, ranking, selection, etcetera).

Another methodological approach to simulation implementation is given by [88] in which the following stages are identified: conceptual design, code development, validation, experimental design, implementation, analysis, interpretation. More of such approaches can be found in various other research disciplines (e.g., economics and social sciences) that make use of simulation-based research. We do not give a comprehensive overview of these approaches here. In general, we think that a general scientific research methodology covers most of the different stages concerning building and conducting simulation experiments. We address, and tackle, methodological issues later when we discuss research methodology for research studies in general.

3.1.3 Distributed Simulation

So, what if you want to run a huge simulation? Simulations of collective intelligence typically involve many individual parts (or: agents, or: entities). Until now, we have discussed how to run simulations on a single processor. For huge simulations, you may have to distribute your simulation over multiple (in the order of tens to hundreds) processors (multiple within a single computer or cluster, or each one in a separate computer). Also, you may be simulating a parallel system – e.g., a flock of birds in which all the birds act in parallel. The field of "Parallel and Distributed Simulation Systems" gives outcome here [127].

In the context of collective intelligence, we can understand "distributed" in two different ways. Firstly, we are concerned with real systems that have many parts acting in parallel. Secondly, we may want to run simulations on a set of processors instead of only one. For the first one, many agent-based simulation software systems (that are discussed later in this book) can "mimic" this kind of parallelism. Although the simulation itself is sequential (by definition, because it runs on a single processor that sequentially processes operations), the control loop of the simulations is programmed as such that it is as-if the individuals acted in parallel. Although you may not notice this if you are programming or running your simulation, you have to be aware of it: it may still happen that *order effects* occur because of how the specific algorithm mimics parallelism.

Actually, in the context of parallel and distributed simulation systems (also as understood by Fujimoto [127]), researchers are mainly concerned with the second type of distribution: spanning a simulation over multiple processors. As [214] explains, there are a number of ways to do this. One way is to keep the simulation itself sequential, but delegating support functions (e.g., random-number generation, statistics collection) to different processors. A more advanced way is to divide the model into submodels and then delegate these submodels to separate processors. One major concern with this way is to take care of the correct time-ordering of actions: the operations of all the submodels must be synchronised with resepect to the model's overall actions. The simulation clock and overall events lists are not necessary anymore: instead, the submodels use *message passing* with time-stamps. There is still some sequentiality here that costs some efficiency: processors may have to wait on each other because of messages with information that have to be received before continuing the submodel. Also, deadlocks may occurs, on which must be check during the simulation.

Another way of distributed simulation works around this sequentiality by means of virtual time in combination with a time-warp mechanism [214, 127]. In this approach, each submodel continues without waiting for messages. When a message is received that should have been processed in the past, a *rollback* of the state variable occurs that recalculates the system state from the moment onwards that the message should have been processed. If a message was

sent to other processor in the time that is now being rolled back, antimessages are sent out. This time-warping can be made more intelligently and advanced making the simulation more efficient, depending on the particular simulation [127, Chapters 4-6].

3.1.4 Versus Empirical Experiments

Simulation experimentation is essentially different from *normal* experimentation. Law and Kelton [214] identify a number of differences – in practice, these differences are advantages of simulation experiments over experimentation in general. Firstly, some factors in the model that are actually uncontrollable in reality can be made controllable, e.g., customer arrival rate. Secondly, because of the deterministic nature of random-number generators, we have again more control over the experiments. Finally, in experimentation in general, you have to randomise treatments (different factor levels) and run orders (sequence in which the treatments are applied) to avoid variation caused by the experimental conditions. Assuming that a random-number generator is working properly, such randomisation is not necessary in simulation experiments.

3.1.5 Evolutionary Simulation Models

The term *evolutionary simulation model* refers to the "use of computer simulations as evolutionary models of adaptive systems" [44]. It denotes a subset of computer models within evolutionary biology and models for evolutionary simulation design within artificial life. Di Paolo, Noble and Bullock [87] established the term in order to define the scientific role of artificial life simulation: if you have constructed an artificial life simulation in which the interactions of many simple individuals cause complex patterns at the global level, and if these pattern seems like a real-world phenomenon, how do you demonstrate the scientific value of this model? Understandably, this relates much to the topic of this book. Hence, we briefly discuss this work in order for you to know somewhat more about these evolutionary simulation models.

A first possible answer to the question posed above is related to the strong and weak positions of artificial life. In strong artifical life, your work is an instantiation of the real-world phenomenon. (For weak artificial life, your work is a model of the real-world phenomenon.) Opposed to this (either instantiation or model view) is the position that simulation has no scientific value what so ever, since it only concerns the rearranging of symbols in some logical form and can therefore not lead to new knowledge. Di Paolo *et al.* [87] are not happy with these extremes and propose their position saying that "simulations are valuable tools for re-organising and probing internal consistency of a theoretical position".

A more careful classification is the following. As mentioned before, Bullock [44] identifies High Class, Model Class and Work Class Artificial Life (AL). High Class AL is concerned with the philosophy of artificial life; Model Class AL is concerned with the validaty of scientify theory; and Work Class AL involves the production of useful artifacts, most naturally understood as engineering. The kinds of AL simulations mentioned above (your simulation result reminds you of a real-world pattern and you wonder what the scientific value of this result is) fall into the Model Class AL.

Prior to the position of Di Paolo *et al.*, there were basically two approaches towards the scientific value of artificial life models. Firstly, you can identify it as an unique (new) object of enquiry. This means that it can only be used as a new technique for a new class problems,

possibly scientifically isolating the modelers and a risk for lack of rigour. Secondly, artificial life modelling is a new technique to attack problems that would traditionally be dealt with using existing formal logico-mathematical approaches. The question whether it is superior to existing methods can (as yet) not be generally answered and needs to be considered on each case individually.

The answer to the question how a simulation can be a model could now be answered by considering them as a kind of *thought experiments* "by using the relationships between patterns in the simulation to explore the relationships between theoretical terms corresponding to analogous natural patterns" [87]. As such, a computer simulation is a tool of theoretical enquiry (whereas "computer experiments" still have an empirical flavour simply because of this term itself). This way, a simulation may show that the relationships between theoretical terms is different than previously thought. In that case, systematic enquiry into the simulation model is necessary.

3.2 Models

We present a number of models of which you can choose from when constructing your simulation of collective intelligence: cellular automata, multi-agent based systems, boolean networks, NK-model, particle-based model, game-theory, formal logics and rule-based systems. Note that the descriptions of the models are on purpose kept short: with the references from the descriptions (mostly to textbooks), the reader can orientate him/herself in more detail on the specific model of his/her interest.

3.2.1 Cellular Automata

Cellular automata (CAs) are a way to study complex systems. Cellular automata can be complex systems themselves, and offer good ways of studying the behaviour of such systems. A CA is "an array of identically programmed automata, or cells which interact with one another in a neighbourhood and have definite state" [387]. Let us explore this definition step by step. An array is a way to spatially (in space) arrange objects. Arrays can have different dimensions. The most used arrays have either one or two dimensions. A one dimensional array could be imagined as a bookshelf with a lot of books standing next to each other. Then a two dimensional array would be a book-cabinet, with books ordered in two dimensions, different shelves, and more books on one shelf. Secondly, these arrays consist of identically programmed automata. By programmed we mean small objects which have a list with rules (that will be called *cells*). The rules bring us to the third part, the interaction. Interaction in CAs is based on the number of neighbours around a single cell. Finally, every cell has a state, most of the time cells have two states: on/off, dead/alive, 0/1. However, it is possible to build CAs with cells that have more than just one state; the number of states is never infinite, though. So it is always possible to know what the next state of a cell is. This is an important feature of a CA, as this makes it possible to always predict the next state of a CA. In other words, any (one) state can be computed from any other (one) state. One thing the definition does not state clearly is the passing of time in CAs. The basic idea is that a rule defines how we can go from one state of the CA (= one point in time) to the next state of the CA (= next point in time).

The idea of cellular automata emerged in the 1940s when Stanislaw Ulam (Los Alomos National Laboratory, USA) studied crystal growth and his colleague John von Neumann worked

on self replicating systems. Together they developed a small CA (each cell had neighbourhood size of 4 and 29 states) of which it was mathematically proven that some pattern would replicate itself endlessly. In the 1970s, John Conway thought up the *Game of Life*, which is basically a 2-dimensional CA with 2-state cells, where the rules define whether a cell lives or dies in the next state [48]. Finally, in the 1980s mathematician Wolfram developed a class of cellular automata that showed that complex patterns may emerge from very simple rule-sets in a CA. Worlfram left academia for about 15 years, in which he developed the Mathematica computer algebra system[1] [130] and wrote a 1280-page book on CA [387] showing that the concepts of CA are applicable to a very broad range of scientific disciplines.

Cellular automata have been succesfully used for, among others, the modelling of traffic flows [151], self organised patchiness in ecosystems [345], and the modelling of urban growth [27].

3.2.2 Multi-Agent Based Systems

The paradigm of *intelligent agents* within the artificial intelligence research area involves the investigation and design of situated agents that exhibit autonomous and rational behaviour with respect to their environment and to each other. Traditionally, agents are designed to work towards some given goals or to react to events that occur in the world.

The concept of an agent is widely used in a variety of software systems. The range varies from small systems such as personalised email filters to large and complex systems such as air-traffic control. Although the application range of agents is wide, in all such systems the key abstraction used is that of an *agent* and therefore we call them *agent-based* systems [391]. Intuitively, agents are systems that can decide for themselves what they need to do in order to satisfy their (design) objectives [392]. But this intuitive notion does not give us a very workable definition. On the one hand, it is too broadly defined and too many systems fall under it that we do normally not call agents, e.g., light-switches or thermostats. On the other hand, it is too narrowly defined, because systems we do call agents are not covered by the definition, e.g., email filters or user interface agents. A more practical, and arguably better, agent definition that works for us is: *a situated, computational and intentional system, which is capable of autonomous actions in its environment.*

The three characteristics of an agent-based system then reflect the aspects of agents we want to emphasise. We briefly describe these aspects. Firslty, an agent is *situated*: an agent is in continuous interaction with its environment. An agent perceives its environment through *sensors* and acts upon that environment using *effectors*. Secondly, an agent is a *computational* system. Since an agent is in interaction with an environment, that environment can, and in most situations will, demand action upon the agent within a certain amount of time. When designing an agent-based system, we have to take into account that an agent's resources are bounded. Finally, an agent is *intentional*: an agent is best described from an *intentional stance* [77]. Systems which are less complex, are better described from a *mechanical stance* and are not agents by our definition.

With respect to collective intelligence, multi-agent (based) systems have been succesfully used, among others, in combination with social simulation [59], for the modelling of (bounded) rational agents [390, 62], to investigate the concept of emergence [84, 83], in research on cooperation and competition [106] and for the (re)organisation of societies [138, 168, 279].

[1] http://www.wolfram.com/products/mathematica/

In general, agent-based systems can be used for a variety of purposes: for example, as the basis of simulation (as mentioned earlier), as a programming paradigm, and as a software engineering paradigm. To conclude this Section, we briefly overview these three purposes.

Agent-Based Simulation The field of agent-based simulation is very broad and does not denote one particular kind of simulation, but different ones. Firstly, it can denote that simulation is understood in the way as we defined earlier in this Chapter and that agents are used as the primitive elements. The system under consideration is thus not necessarily a (multi-) agent-based system. Secondly, the system that you model is actually an agent-based system and you want to investigate the workings of this sytem by means of simulation. Finally, by considering agents in the context of programming or software engineering, you can simulate such a software system in order to investigate its behaviour. The agent-based simulation is the simulation of a design-artifact, e.g., a network communication protocol, in a distributed system.

Agent Oriented Programming Similar to how objects can be the primitive of a programming language or paradigm, you can take agents to be such primitive building block. Such agents are proactive (take initiative), reactive (respond to environmental cues) with respect to their environment, and social with respect to each other [392]. An *object* is a computational entity that encapsulates some state, is able to perform actions or methods on this state and communicates by message passing [391]. There are some important differences between object- and agent-oriented programming [391]. The first is the *degree of autonomy*. Although an object can exhibit control over its own state, it cannot exhibit such control over its behaviour. The decision whether to execute a method lies within the object that invokes the method. With agents, the decision lies within the agent that receives the request. The second difference is with respect to the notion of *flexibility* in autonomy. This difference denotes that an agent can flexibly address its reactive, pro-active and social behaviour, whereas an object cannot do this. The third difference is that agents have their own *thread of control*. Despite the differences, we do not exclude the possibility of implementing agents in an object oriented model.

Agent-Oriented Software Engineering In a somewhat broader range than only the programming paradigm, agents can also be used as a useful concept in the whole process of software engineering – including requirement analysis, conceptual modelling, etcetera. Such engineering may be used in general for all kinds of software, but particulary[2] for software architectures that contain many dynamically interacting components, each with their own thread of control, engaging in complex coordination protocols.

3.2.3 Boolean Networks

As an appetiser for the next model (Kauffman's NK-model), we briefly explain[3] the concept of a *Boolean network*. The basis of such a network is a set of logical propositions: such proposition are sentences that can be evaluated either true (1) or false (0). The system for logical operations is called *Boolean logic*. Consider as an example of such a proposition the

[2] http://www.csc.liv.ac.uk/ mjw/aose/

[3] Based on http://www.calresco.org/boolean.htm.

sentence "All birds can fly". The idea of a Boolean network is that logical propositions are combined by logical operators (OR, AND, XOR, etcetera) in some networked fashion (relating the outputs of certain propositions to the input of others). If you call the propositions *gates*, such a network is basically the conceptual design of a modern digital computer.

The network thus connects in/outputs of gates with each other. The operation of the network is determined by the logic of the gates and the input connections. When we do not define what the particular meanings are of these gates and connections, you have a very abstract model that can be applied to all kinds of real-life situations: social interactions, gene regulation, plan dependency, etcetera.

A Boolean network has some *configuration* that defines how the propositions are connected to each other: it defines which gate output(s) is connected to which gate input(s) (and therefore also the propositions themselves). If you are interested modelling a complex system by means of a Boolean network, then you are actually thus defining such a configuration.

Kauffman was one of the first biologists that used the idea of a Boolean network to model genetic regulatory networks[4] (GRN) [185]. Such a GRN is *a collection of DNA segments in a cell which interact with each other and with other substances in the cell, thereby governing the rates at which genes in the network are transcribed into mRNA*. A Boolean network can model a GRN to relate the gene products (outputs) to the substances from the environment that affect it (inputs).

3.2.4 NK-Model

A *random Boolean network* (RBN) is a network is a Boolean network of which the configuration is unknown, and where the gate types and their connections are randomly initialised [131, 187, 188]. With respect to earlier discussed models, RBNs can actually be considered as a generalisation of cellular automata – the state of a gate can be affected by every node in the network, not necessarily only by its neighbours.

Following traditional terminology, we denote the number of gates (called *nodes* now) by N and the number of inputs to each node by K – hence, this type of Boolean network is also called a *NK-model*. In a RBN, the nodes are updated synchronously, like in a CA, by means of update functions. These update functions cause the system dynamics to flow through the network.

Assume that you have fixed N and K, and you are interested in the *behaviour* of this network: over time, what do the on/off states of the nodes do – stabilise, die out, fluctuate, etcetera? First of all, note that there are 2^{2^k} possible functions for each node; secondly, there is an extremely large number of possible networks. These two facts alone already prohibit statistical study of the network behaviour. Instead, we present some ways to *characterise* the network behaviour.

We can look at the end state of the system. Assume that we can *execute* a network, i.e., we choose an update function for each node, initialise the node states randomly and simply run this network – and see what it does. (Note that the network is completely deterministic now.) Over time, a network may end up in some state in which the node states do not change anymore, e.g., a system state in which 75% of the node states are 1 and the other ones 0 – this particular state is called a *point attractor*. Or, the system may converge to a set of states through which it cycles periodically – this is called a *cycle attractor*. To characterise

[4] http://en.wikipedia.org/wiki/Gene_regulatory_network

the behaviour of some given NK-model, we could thus analyse some number of (carefully selected) update functions for this particular model. If the network converges to such a point- or cycle-attractor, then this happens for all possible initialisation states. If not, then it is not a point- or cycle-attractor. The attractor *basin* is the set of states that flow towards the attractor.

More generally, we can look at the dynamics of the model *during execution* – not only at the end. If we can characterise these dynamics correctly, we may not have to wait until the system goes towards an attractor. Instead, we may be able to predict this, based on the dynamics that we have observed until now. A dynamical system can in general be in one three phases: ordered, chaotic or critical.

Let us visualise the network as a lattice where the nodes that do not change value any more are colored red and nodes that do change value are colored green – following [131] and [186, pages 166-167]. If the *ordered* phase, many nodes are red and some are green. The greens actually look like islands enclosed by the reds. In the *chaotic* phase, many nodes are constantly changing: constantly changing islands of reds enclosed by green seas can be seen. The *critical* phase is in between the chaotic and ordered phase: the green seas break into green islands and the red islands join and percolate through the lattice. These phases can be used to characterise the stability of a network.

Another way to characterise the stability of a network is to analyse how much the network depends on the initialisation or random changes in the network (changing nodes/connections while running the network). While you analyse this kind of stability, you may observe the so-called *Butterfly effect*: small local changes have large consequences[5].

There are more ways to characterise the dynamics of these NK-models, but we do not explain these here – we redirect the interested reader to the references included above. In general, the "NK-model research field" is concerned with: mathematical analysis of dynamics characterisation and phase transitions; explorations of model presented above (e.g., measurement of number and lengths of attractors, sizes and distributions of attractor basins, parameter dependences); extensions of the model (e.g., multi-valued networks, asynchronous updating, RBN classification); etcetera [131].

3.2.5 Particle-Based Models

In a significant number of simulations of collective intelligence, a *particle-based model* is used. For example, Farkas, Helbing and Vicsek have successfully modelled human panic behaviour in case of a crowd stampede based on a physical force model for many-particle systems [148]. This model was earlier used by the authors to model state transitions of physical matter. In this Section, we discuss this example in more detail, together with two other such studies, one on highway traffic and one on adaptive team formation.

Human panic behaviour has mainly been researched from the perspective of social psychology. Helbing, Farkas and Vicsek [148, 150, 114, 115] used a model of pedestrian movement to research the mechanisms and conditions of panic[6]. Based on the available socio-psychological literature, in combination with other resources (e.g., video materials), the authors identified a number of characteristics of panic behaviour, e.g., people move or try to move considerable faster than normal; indidivuals start pushing, and interactions among people become physical

[5] http://en.wikipedia.org/wiki/Butterfly_effect
[6] http://angel.elte.hu/~panic/

in nature; etcetera (9 characteristics were identified in total). A model was contructed of escape panic in a framework of self driven many particle systems. In this model, the behaviour in a crowd is influenced by a combination of socio-psychological and physical forces. The model includes parameters such as individual mass, desire to move towards some direction, desire to move at a certain speed, desired distance between individual and other pedestrians and the wall, etcetera. Some model assumptions were made (for example, average mass is 80 kg), because there was no suitable quantitative data available on escape panic. Based on this model, some phenomena of escape panic were observed in the simulation as were seen in the real-world: transition to incoordination due to clogging; 'faster-is-slower-effect' due to impatience; and mass behaviour. For the latter, the authors distinguished individual and herding behaviour; it was found that in an escape panic situation, a mix of these behaviours performed best (in terms of number of people escaping within some time frame).

Everyday traffic is a familiar "social phenonemenon in which diverse individuals compete with each other under certain constraints". Helbing *et al.* [149, 152] have simulated heterogeneous traffic and demonstrate that *cooperative, coherent states may arise from competitive interactions between individual vehicles.* In traffic, one can observe global phenomena (e.g., moving traffic-jam fronts or synchronised congested traffic) that emerge from the local behaviour of individual vehicles (e.g., acceleration, braking, overtaking, etcetera). In the simulation, there are two types of vehicles (cars and lorries) that differ in general characteristics (e.g., velocity, length, desired distance to vehicle driving in front, etcetera). Local rules define how the individual behaves (go faster if possible, take over if possible, etcetera). The model agrees well with known empirical features of traffic flows. The most important finding is that in the simulation it is observed that as the vehicle density increases, they end up in a state in which they all move at about the same speed, like the motion of a solid block. The authors analyse this result on the basis of the settings of the model parameters. A comparison of the results with empirical data from highway traffic in the Netherlands confirms the finding.

Adaptive team formation is concerned with dynamic situations in which teams of agents have to adjust to changing circumstances. Consider, for example, an immobile resource (e.g., a fort or castle) that is being protected by swarming protector agents against attacking intruders [381]. The protector have two tasks: prevent damage to the resource and to eliminate intruders. Assuming that a protector can only perform one task at a time, team formation basically means to choose the level of heterogeneity: how many protectors should be worried about damage to the resource, and how many should be eleminating intruders? Wiegand *et al.* [381] present an extension of the *physicomimetics* particle-based framework with heterogeneity. Physicomimetics [318] is a swarm control paradigm and can be used for the control of multiple agents. It is particle-based, which means that each agent is seen as a point-mass particle. Each particle has some position and velocity. It lives in a discrete-time simulation where it changes positions from time to time based on its previous position and its velocity. The velocities are dynamic and determined by artificial forces on the particles. The particles and forces are homogeous and symmetric in the classical model; the heterogeneity extension means that the particles have different masses, react differently to each other, interactive forces between the types of particles may differ, etcetera. Wiegand *et al.* show by simulation that the extended framework is able to find a suitable degree of heterogeneity for the protector scenario presented above. Good values for the many parameters of the heterogeneous control framework were found with an evolutionary method.

3.2.6 Game and Decision Theory

There is not much introduction necessary for the concept of games, because we all know what games are; we have all done computer games, board games, played soccer, baseball, etcetera. However, a more general idea of games goes further than such entertaining games: in the context of politics, economics, biology, etcetera [206, 395, 261, 379]. Such a general concept of a game[7] involves all those situations where "a number of players or decision makers interact, possibly threaten each other and form coalitions, take actions under uncertain conditions, and finally receive some benefit or reward or possibly some punishment or monetary loss". The study of such games is the area of *game theory*. The number of players is then two or more; if it is an one-player "game", then we simply talk about decision theory. Considering that the individuals in a collective system can be considered to be decision makers in such multi-player games, you can use game theory to let them make good decisions.

We briefly explain some general game-theoretic concepts. Whilst playing a game, it can be in some kind of *position*, e.g., like a board-position in a chess or checkers game. From this position, the players make *moves* that make the game go iteratively from position to position. For making such moves, players have some amount of *information* about the game and the other players. This is *perfect information* when the players know all past moves; if this is not known, the information is imperfect. Each player takes the available information into account and can decide on some *strategy* it will use prescribing the moves. Depending on your model of the game (extensive, strategic or coalitional), you may not want to take all the specific details (positions, moves) into account when deciding on an effective strategy. In the strategic model, the main concepts are strategies and *payoffs*. A payoff is basically the outcome of the game for yourself. This may be simply winning the game or not (as in chess of checkers), but usually there is some numerical value assigned to it. Based on these numerical payoffs (or: *utilities*), more game-theoretic concepts are based. For example, a *zero-sum* game is a game where the sum of the payoffs is zero no matter what moves the players make (i.e., gain for one player implies loss with same magnitude for another player).

There are many textbooks about game theory out there. You can also find many introductory texts and courses online[8] on game and decision theory. Therefore we decided on purpose to keep the description short here and redirect the interested reader to the available textbooks and online texts.

3.2.7 Formal Logics

Artificial Intelligence has a strong foundation in logics. As we saw earlier, the Boolean networks are also fundamentally based on logical expressions. Also many agent-based systems have been specified and analysed by means of logic. Hence, it is not suprising that in the area of collective intelligence, emergence, etcetera, many AI researchers have used formal logical models. To name a few examples: agent societies have been formalised using logic [12, 86]; agent-based social simulation has been compared with formal systems [117], and human organisations have been formally modelled [246, 305, 351]. As the formal modelling of (agent or human) organisations is concerned, Jonker and Treur [40, 39] have developed two formal languages that can be used to model such organisations – also with respect to dynamic aspects of the system behaviour. We look at these two languages in more detail.

[7] http://www.math.ucla.edu/~tom/Game_Theory/Contents.html
[8] http://www.gametheory.net/

In the Temporal Trace Language (TTL), a state ontology is a specification (in order-sorted logic) of a vocabulary, i.e., a signature. A state for ontology Ont is an assignment of truth-values true, false to the set At(Ont) of ground atoms expressed in terms of Ont. The set of all possible states for state ontology Ont is denoted by STATES(Ont). The set of state properties STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed time frame T is assumed which is linearly ordered. A trace or trajectory γ over a state ontology Ont and time frame T is a mapping γ : T $\rightarrow$ STATES(Ont), i.e., a sequence of states γ_t (t $\in$ T) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES (Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of dynamic properties DYNPROPEXP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner (for an organisation or part thereof, Ont is the union of all input, output and internal state ontologies of the roles in the organisation (part)).

Given a trace γ over state ontology Ont, the input state of a role at time point t is denoted by state (γ, t, input(r)); analogously, state(γ, t, output(r)), and state(γ, t, internal(r)) denote the output state and internal state of the role. These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate $\models$, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r)) $\models$ p denotes that state property p holds in trace γ at time t in the output state of the organism. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Trace for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg, \vee, \wedge, \rightarrow, \forall, \exists$. Within TTL abstractions can be made by introducing additional terms (e.g., predicates), which are definable in terms of the existing terms. In order to specify simulation models, a simpler temporal language has been developed, based on TTL. This language (the leads-to language) enables one to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the leads-to language $\alpha \rightarrow_{(e,f,g,h)} \beta$, means:

If state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold for a certain time interval of length h.

A specification of dynamic properties in leads-to format has as advantages that it is executable and that it can often easily be depicted graphically. Moreover, the language offers the possibility to express both qualitative and quantitative aspects of a process to be simulated.

Description of relationships between dynamic properties at different levels of aggregation, which 1) provides a basis for formalised biological, cognitive, or organisational theories and their validation, 2) enables evaluation of trajectories of dynamics (at any level of aggregation) against specified dynamic properties, 3) allows diagnosis of malfunctioning, and 4) allows relating dynamic properties to empirical physical/chemical/physiological/neurological data.

To aid simulation of processes, languages for specification of high-level executable models can be used, thus obtaining tractable dynamic models for phenomena that, at a too detailed level may be much more complex.

3.2.8 Knowledge Systems

Artificial intelligence in 1960-70s was mainly concerned with the development of so-called *knowledge systems* (or: expert systems, or: knowledge-based systems). Such a system is a program for extending and/or querying a *knowledge base* – this is a special kind of database for knowledge management. A knowledge system is a computer system which represents and uses knowledge to acccomplish a task [325]. The Physical Symbol System Hypothesis says that "physical symbol systems are necessary and sufficient for general intelligent action" [249]. This hypothesis, although debatable, justifies the interest in knowledge systems. The symbols in such a system are used for general intelligent actions: symbols are recognised by an (automated) recogniser, and meaning is assigned by an observer. Knowledge systems can be analysed on different levels: firstly, one can do this on the symbol level about the manipulating patterns of symbols; secondly, one can work on the knowledge level and analyse the meaning of the symbols regarding the task of the system.

It is interesting to consider knowledge systems with respect to collective intelligence. Although some collective systems may only need rather reactive individual behaviour to show global intelligent behaviour, others may require more sophisticated local reasoning. The area of knowledge systems offers a rich supply of methods and techniques thar perform such reasoning in some automated fashion. Current developments on knowledge systems mainly happens within the areas of knowledge management and, more general, the semantic web[9]. Later we return to knowledge systems, when we look at a specific class of systems (learning classifier systems) in which the program itself is not hardcoded, but discovered.

3.3 Algorithms

We present a number of (adaptive) algorithms that you can use in your simulation of collective intelligence when your simulation includes an adaptive component: evolutionary methods, co-evolution, learning classifier systems and neuro-evolution. Additionally, we present two algorithmic frameworks that combine model and algorithm with respect to collective intelligence: the COIN model and particle swarm optimisation. All the algorithms have some collective aspect; we purposely do not present other machine learning algorithms that do not have such an aspect (e.g., reinforcement learning).

As with the models, the descriptions of the algorithms are on purpose kept short: with the references from the descriptions (mostly to textbooks), the reader can orientate him/herself in more detail on the specific algorithm of his/her interest.

3.3.1 Evolutionary Methods

Since Holland [164] lay out his ideas about adaptation in natural and artificial systems, and de Jong [70] developed these ideas further into genetic algorithms, a complete research field has emerged in the last two decades on the topic of *evolutionary methods*. This class of heuristic search-methods was long characterised in all kinds of different variants with arguably many differences between them: evolutionary strategies, simulated annealing, genetic algorithms, genetic programming, etcetera. Recently, efforts are being undertaken to come to an unification of all these different views, e.g., [103].

[9] http://www.w3.org/2001/sw/

Evolutionary Algorithm	
1:	INITIALISE population with random candidate solutions
2:	EVALUATE each candidate
3:	repeat until (TERMINATION CONDITION is satisfied)
4:	SELECT parents
5:	RECOMBINE pairs of parents
6:	MUTATE the resulting offspring
7:	EVALUATE new candidates
8:	SELECT individuals for the next generation
9:	end repeat

Table 3.1: An evolutionary algorithm (from [103]).

An evolutionary method (or: algorithm[10]) is "a generic population-based metaheuristic optimisation algorithm". The heuristics that it uses are inspired by biological evolution: reproduction, mutation, recombination, survival of the fittest, and natural selection.

The generic algorithm is shown in Table 3.1. You can use such an algorithm for a wide range of problems. Let us say that you have to set up a consistent class schedule for a highschool, where each group can have at most one class at the same time and each group has to follow some fixed number of classes during a week. You start the algorithm with randomly initialising a population of solution candidates for your problem. To each inidividual, you assign a value saying how good it is. In the scheduling example, you can compute the number of constraints that is violated in the candidate solution. You want the algorithm to find a schedule in which the number of constraints is minimised, ideally null. *Evolutionary operators* create diversity in the population: *recombination* combines two solutions into a new one, *mutation* changes a single solution into a new one by applying some random changes to it. Natural selection is enforced upon the population as that only the best individuals at some given time can determine the make-up of the population. By iteratively going through and updating the population, it is expected that it eventually converges to one ore more good solutions.

Evolutionary methods have been used for a variety of different problems, e.g., optimisation, planning and scheduling, design, chemical engineering, mining, robotics, etcetera. Because of the fact that evolutionary methods are population-based (and inherently adaptive), they are a good candidate when deciding on an adaptive component in a collective intelligent system.

3.3.2 Co-evolution

One crucial component of an evolutionary algorithm is the *fitness function*: this function is used for evaluating an individual. Although fitness, in evolutionary biology, is originally measured by the number of offspring that an individual brings forth (according to Darwin [64]), in evolutionary algorithms this is often an abstract function that evaluates the candidate solution and returns a numerical measurement of the fitness (or: error). Such a function is often supplied by the designer of the algorithm.

This is a discrepancy between evolutionary biology and evolutionary algorithms – where does the fitness function come from in natural evolution[11]? If a natural environment does not

[10] http://en.wikipedia.org/wiki/Evolutionary_algorithm

[11] This description is based on the presentation of Dr. E.D. de Jong about co-evolution given at the DE-

give any information about the quality of individuals, then evolution is impossible. However, if the environment is *informative*, then we can define fitness on the basis of *co-evolution*. Assume, for example, that an environment contains rabbits, foxes (danger to the rabbits) and plants (resources for the rabbits). In this case, the environment is informative: finding plants increases the fitness, and evading foxes increases the fitness. Although we cannot determine an *absolute* measurement of fitness, we can define a *relative* one: how good is an individual at locating plants and evading foxes. We speak about co-evolution: an evolutionary setup where individuals are evaluated based on other (co-)evolving individuals. To be more specific: the *ranking* of the individuals is affected by co-evolving individuals.

A typical setup of a basic co-evolutionary algorithm then consists of two populations with first and second player strategies, respectively. The evaluation is then based on letting population 1 play population 2 and vice versa. The search process does not need an external fitness function, individuals used for evaluation develop as part of the process itself, and evaluation depends on the current state of the search process. There are a numer of successful applications of co-evolutionary algorithms: for sorting networks, co-evolution of virtual creatures (presented later in this book), for the majority function in cellular automata, in backgammon, etcetera.

A major concern in a basic co-evolutionary algorithm is that the fitness is determined by a changing set of opponents. This may lead to unstable evaluation and the basic co-evolution may therefore easily fail. This concern may lead to disengagement (one population beats the other one which gives not actual information about the individuals), intransitivity (individual A beats B, B beats C and C beats A) and over-specialisation (focus on a subset of the underlying objectives). These are currently, among others, important research topics within the co-evolution field of research[12].

3.3.3 Learning Classifier Systems

As the idea of knowledge based systems and expert systems arose in the 60s and 70s, with subsequent downfall in the 80s resulting from the emergent interest in reactive systems, the movement of machine learning algorithms saw the conception of Learning Classifier Systems (LCSs) in 1986 when Holland presented his "possibilities of general-purpose learning algorithms applied to parallel rule-based systems" [46]. These systems exploit the combination of evolutionary computation and reinforcement learning to develop sets of condition-action rules, see, e.g., [212, 213]. The essential difference between traditional knowledge based systems and LCSs is that rules are discovered instead of explicitly programmed. For over almost 20 years now, the LCSs have been part of the machine learning paradigm within the research area of artificial intelligence.

An LCS is in continuous interaction with its environment, from which it receives feedback in the form of reward. LCS models typically consist of four main components: 1) set of classifiers - these are condition-action rules representing the system's knowledge, 2) performance component - controlling the interaction with the environment, 3) reinforcement component - reponsible for distributing the received reward of the classifiers accountable for the reward, and 4) a discovery component - finding better rules, usually implemented by a genetic algorithm. There are two measures associated with classifiers: 1) prediction (also called strength)

COI2006 summerschool (http://www.decoi2006.nl/).

[12] See also http://www2.demo.cs.brandeis.edu/cgi-bin/coev-wiki

- estimate of amount of reward received when the classifier is used, and 2) fitness - quality of the information about the problem that the classifier conveys.

LCSs are mainly used for three important characteristics. Firstly, they are *adaptive* systems as such that they can learn at run-time in dynamic environments. Secondly, they are systems that can *generalise* as they are able to represent what they have learned compactly and they can apply what they have learned to previously unknown situations. This generalisation is achieved through the evolution of general rules matching many environmental states. Finally, LCSs are to some degree *scalable* as to how fast the learning time grows as the problem complexity increases. There is no clear evidence as to how general this goes for the system class of all LCSs, but it has been shown for some particular implementations. For example, based on such evidence, it has been suggested that the learning process grows as low order polynomial of the complexity of the problem.

LCSs have been applied to a number of domains. These include the areas of robotics, knowledge discovery and computational economics. Robotics has been the main testbed for experimenting with LCSs and is expected to remain such in the close future. For knowledge discovery, LCSs can extract compact descriptions of interesting phenomena described by multidimensional data. For the medical domain, LCSs have been implemented for the sake of knowledge discovery producing solutions that are easily interpretable and suggests interesting dependencies on one or a few attributes. For computational economics, LCSs have been used to model adaptive agents in artificial (auction-based) stock markets.

With respect to the machine learning paradigm, LCSs address three basic problems: 1) parallelism and coordination - different combinations of building blocks may be able to deal with a broad range of novel situations and can form the basis of the coordination and interaction of a large number of rules; 2) credit assignment - distributing rewards over action sequences is difficult and urge the need for local assignment (per action), which happens in LCSs; 3) rule discovery - as to make this process as effective as possible, past experience must be used in order to generate plausible rules for poorly understood situations.

3.3.4 Neuro-Evolution

In many real-world circumstances, it is hard to exactly define beforehand which actions are best in which situations. In these cases, you would just try some actions and see which ones are good and which one not. Gradually you will learn more about the situation and become better at choosing actions. A very powerful machine learning approach that basically does the same is *neuro-evolution*: a technique that evolves artificial neural networks with genetic algorithms [242, 323, 322]. It is a robust method in that it can deal with continuous states (not only discrete ones) and with partial observable states (there is no perfect information about the state). Successful applications include robot control and rocket control, automated driving and collision warning, coordination of multi-agent systems, game playing, and resource optimisation[13].

An artificial neural network (ANN) [1] is a computing paradigm that is loosely modelled after the cortial structures of the brain[14]. It consists of basic units that behave somewhat like neurons. These artificial neurons are always in some *state of activation* (simply binary (on/off) or have a continuous value, depending on the design); have an activation-threshold that defines when the neuron is acitivated (if activation is binary); and a (non-linear) activitation

[13] http://nn.cs.utexas.edu/keyword?neuroevolution
[14] http://en.wikipedia.org/wiki/Neural_network

function (if activitation is continuous (or: non-linear)) . Neurons communicate with each other via *synapses* that connect neuron in/outputs – each synapse has a *weight* connected to it expressing the relative importance for the neuron connected to it. A typical ANN structure (or: topology) consists of an input neuron layer and output neuron layer, with some number of hidden neuron layers in between. Consider an ANN that does handwriting recognition of numbers: it takes in an handwritten number as input and outputs a digital number. Assume that you lay a grid over the handwritten number, each cell in this grid is either blank or not. This gives you an input to the network of the size of the grid. The number of output neurons (or: nodes) is 10 (0-9). (The designer can determine the number of hidden layer him/herself.) The task of the ANN is to correctly connect output to input: when the written number is '8', the output neuron '8' should be active and the other output ones not. A neural network learns by iteratively going through a training data set; and applying its learned knowledge to an unknown data set afterwards. While going through the training data set, a learning algorithm updates the *connection weights* defining the relative influence of one neuron another.

In neuro-evolution, this learning algorithm is a genetic algorithm. The evolutionary operators of this algorithm thus work on the weight space of the network: recombination combines weight vectors of complete networks; mutation works on single networks' weights. The goal of a fixed-topology is then to optimise the connection weights that determine the functionality of a network. One particular kind of neuro-evolution also takes the topology of the network into account: the NeuroEvolution of Augmenting Topologies (NEAT) method is designed to take advantage of the structure to minimise the dimensionality of the weight space [323]. This methodology has been succesfully applied to a number of application, including the NERO video game where it accomplishes real-time neuro-evolution [322].

3.3.5 COIN

There are a number of algorithmic frameworks that already take collectives into account, and thus combine a model with an algorithm in some way. We discuss two of such frameworks here: this Section explains the COllective INtelligence (COIN) framework and in the next Section we discuss Particle Swarm Optimisation. The COIN framework was developed by Tumer and Wolpert at NASA during the last decade [357, 356, 389, 388].

A COIN is a multi-agent system where: 1) the agents each run reinformence learning (RL) algorithms[15]; 2) there is little to no centralised communication or control; and 3) there is provided a *world utility function* that rates the possible histories of the system.

The terminology of the COIN framework is outlined by Wolpert and Tumer according to six (preliminary) definitions. Firstly, there is a *microlearning algorithm* (the RLs employed by the individuals) and possibly a *macrolearning algorithm* (externally imposed run-time modifications to the COIN); the initial construction of the COIN is the *initialisation*. Secondly, time is assumed to be discrete and confined to integers; there is a lower bound on the time. Thirdly, The COIN contains discrete *nodes* that are at any time in some *state* that holds all variables that have any effect on the COIN. (The fourth definition concerns formal notations and is left out here.) Next, the universe in which the COIN operates is completely deterministic. There is an *environment node* that contains all variables that affect the dynamics of the system, but were not included in definition 3. Finally, there is a *world utility* that ranks the worldlines of the COIN; and there are *personal utilities*, which are defined similarly but on a

[15] Reinforcement learning is "learning what to so as to maximise a numerical reward signal" [331].

local level.

The goal of a COIN is then to optimise a global utility function by the collective behaviours of the individuals. When the components as defined above are given, a utility-based mathematical framework enables the achievement of global utility optimisation. This framework is quite extensive and described extensively by Tumer and Wolpert in [388]. The framework has been succesfully used for and the routing of internet traffic [389] and coordination of multi-rover systems[16] [357].

3.3.6 Particle Swarm Optimisation

Particle Swarm Optimisation[17] (PSO) is a population based stochastic optimisation technique inspired by social behaviour such as bird flocking and fish schooling [99, 98, 100, 108]. It shares many resemblance with evolutionary methods (populations of solutions that search for optima by updating generations). However, it does not have evolutionary operators (recombination and mutation), but the solutions are particles that fly through the problem state following the current best particles.

The particles form a swarm that flies through the (multi-dimensional) problem space, whereby the positions of the particles are updated each generation based on the particle's experience and that of its neighbours. Each particular has some velocity that determines its next position. The velocity is thus the driving force of the optimisation process. The velocity equation contains a *cognitive component* that expresses the experiential knowledge of the particle itself, and a *social component* that represents the socially exchanged information. The cognitive component is proportional to the distance of the particles from its own *personal best*. The neighbourhood of a particle has thus an influence on the direction of a particle. This neighbourhood can contain the whole swarm (in which case we talk about global best (*gbest*) PSO) or it only contains a subset (and we talk about local best (*lbest*) PSO).

The (basic) PSO algorithm is an iterated algorithm that terminates when some stopping condition holds (e.g., maximum number of iterations is reached, an acceptable solution has been found, no improvement is observed over a number of iterations). Each iteration, 1) each particle sets its personal best position, and the best position of the neighbourhood is set; 2) each particle updates its velocity, and its current position. Eberthart [100] suggests basic parameters and performance measures. The basic PSO parameters are: swarm size, neighbourhood size, number of iterations, and the acceleration coefficients (for controlling the stochastic influence on the velocity of a particle). The main performance measures are: accuracy, reliability, robustness, efficiency, diversity and coherence.

A nice feature of PSO is that (for two- or three-dimensional spaces) it visualises very well – executing the algorithm literally shows a swarm in search of optimisation[18]. PSO has been successfully applied in many areas, for example, function optimisation, artificial neural network training, and fuzzy system control.

3.4 Summary

This Chapter explained how simulation can be used effectively for the design and analysis of collective intelligent systems, and it enumated a number of generic models and algorithms that

[16] http://www-aig.jpl.nasa.gov/public/mls/multirover/

[17] http://www.swarmintelligence.org/

[18] See for example the Java Applets on http://www.engr.iupui.edu/~eberhart/web/PSObook.html.

can be of use when working on collective systems. We did not explain when and why to use which model and/or algorithm for a specific simulation of collective intelligence. The reason for this is that the science for this is not available yet – this is current state-of-art research in collective intelligence. The two last algorithm frameworks (Tumer and Wolpert's collective intelligence and Eberhart and Kennedy's particle swarm optimisation) are frameworks which already exploit the idea of collective intelligence. As shown throughout the book, we believe that the field of (simulation of) collective intelligence is broader the either of these frameworks covers. However, we do no exclude the possibility that the aspects that we discuss will be in some degree covered by these frameworks – this is under investigation.

Part II

CASE STUDIES

– The major pitfall here is 'wishful naming'.

Gusz Eiben

– It requires a very unusual mind to undertake the analysis of the obvious.

Alfred North Whitehead

4

Analysis of Collective Intelligence

Look around and you may observe collective intelligence everywhere. You can see it at every possible level - from the organisation or society that you take part in down to the molecules that make up your immune system (although you cannot actually observe these unless you would be coincidently looking through a microscope right now). Previously, many researchers have also looked around themselves and made studies of what they saw. We describe a number of such studies in this Chapter.

Aims of this Chapter

This Chapter aims to provide you with the description of a number of different case studies. These case studies have in common that they analyse collective intelligence in some kind of form as observed in Nature, where Nature can be understood in the widest sense: including ourselves. There is no specific reason why we chose for the particular case studies included here - these were the ones that we could describe good enough in the time available to us for preparing this book. We have categorised the case studies according to the scientific disciplines they are part of: social sciences, biological sciences and economic sciences. Although the studies, at the time that they were carried out, may not have had the analysis of collective intelligence as an explicit goal, we make it clear to you that there are many commonalities between the studies. We have made each description as such that after reading you have a basic understanding of the study, the phenomenon that was studied and possibly the model or simulation that was made. The descriptions each contain brief summaries of the used model, implementation[1] and conducted experiments. Additionally, we have included references to the literature (scientific articles, papers and/or books) for further and more detailed information about the studies.

[1]See Appendix A for an overview of the software packages that were used for the implementations (if applicable).

4.1 Social Sciences

Thinking about collective intelligence while looking around yourself immediately places it in the context of human society. We are here with approximately six billion people on the Earth and somehow we have to organise our society well. Social scientists research the human aspects of the world. It contains disciplines such as anthropology, political science, and sociology. In the last two decades, among others, Axelrod [15, 16, 17], Gilbert [59, 134, 135, 136, 281], and Epstein and Axtell [19, 111, 110] have put simulation on the map for the social scientist[2]. It is remarkable that in this social simulation literature, researchers often talk about *agent-based modelling* when presenting the topic of simulation in their field. While we present this kind of modelling in this book as one particular type of simulation, this distinction is not made by the social scientist. The agent-based model naturally agrees with society as a system (where humans are modelled as agents) [59]. Researchers have asked themselves whether evolutionary algorithms are realistic representations of social processes [53]. The concept of collective intelligence is not necessarily linked to simulation: Postmes *et al.* [277] investigate how social identity is formed with respect to social networks.

It is also noteworthy that the whole field of social sciences does not share the same opinion about these simulation studies. For example, Lansing [211] overviews critiques on artificial society models, e.g., that these models "reflect unconscious cultural assumptions and social prejudices of their creators". We do not specifically go into these critiques, but let it be a word of caution to the reader when reading this Section.

In this Section, we present three simulation studies in the social sciences: Schelling's segregation model, artificial societies, and the artificial Anasazi model.

4.1.1 Schelling Segregation

In his study, Schelling [300, 301, 302] investigated how local segregationist actions in a residential structure can influence the whole structure on a macro level, in terms of segregation. Segregation (i.e., the clustering of ethnic or social classes) is a phenomenon that has always been present in big cities. While it is something that was promoted by rulers in former times, with ethnic minorities allowed to settle only in specific parts of a town (e.g. settling laws for Jews), segregation happens in a self organised way nowadays. It is a very crucial socio-political and public economic issue, which has been a problem in the USA for a long time and is now also becoming an issue in Western-European countries [264]. It can lead to, for example, the degeneration of city centers into slums, because people prefer to live in the quiet suburbs, or a ghettoisation of originally ethnical blended districts (e.g. the segregation of Turkish migrants in the Netherlands [365]). Deficient Integration has been identified as one of the key factors leading to segregation and this discovery has led to increased awareness and action [264] For example, in the Netherlands, the budget to assist migration increased from just 9 million euro in 1970 to 1.1 billion euro in 2003 (see Commissie Blok, 2004).

With his model Schelling, tried to show that segregation takes place on an individual level. Agents within the model stay at a place or leave according to the status of their local neighborhood. Agents will stay at their position if they like their neighborhood, which depends on whether the number of agents in their neighborhood which are not of their kind is smaller than a certain fraction (e.g. 50% or 60%) of the total amount of agents. If agents

[2]Since 2003, there is an European association on social simulation (http://essa.eu.org/), where you can find information about social simulation workshops, conferences, books, and on so on.

are unhappy, they can move to a more satisfactory location, which leads from a mixed to a segregated environment.

Model

Originally[3] Schelling introduced a one-dimensional model for spatial segregation [300]. But we will focus here on the two-dimensional model which was developed later on [301, 302]. The model consists of an environment which is represented by a closed regular lattice. Each cell on the lattice can be occupied by exactly one agent. There are two types of individuals (O and X) which are initially distributed randomly in the the environment. It has to be noticed that some cells have to be left free in order to enable the agent's movement. After the initialisation, each agent determines if it is happy or not with its current position according to its neighborhood (a Moore neighborhood). Unhappy agents have the possibility to move to the nearest satisfactory position. This process is repeated until each agent is satisfied.

Implementation

When Schelling came up with his segregation idea in 1969, he used 2 sorts of coins placed on graph paper to demonstrate his theory. Nowadays several implementations exist in all kinds of programming languages such as Java and C++. The Schelling segregation model is also implemented for various multi-agent-plattforms such as NetLogo[4] and Mason[5].

Results

In Schelling's initial model, agents tolerated a ratio of up to 50% of foreign agents in their neighborhood [300]. These ratios were altered in later experiments (up to 66.66%). Nevertheless, the model always leads to a highly segregated state, which leads to the conclusion that segregation can take place even in a very tolerant environment.

4.1.2 Growing Artificial Societies

This study is a bottom-up approach to explore how social structures such as trade, migration or group formation can emerge as a result of simple individual interaction within a multi-agent system[6] [111].

The original Sugarscape artificial society [111] consists of 3 main components (agents, environment and rules) that will be explained in more detail in the next Section. A cellular automata approach is used to simulate the environment, and an agent model to simulate the agent's behavior. One of Sugarscape's key features is its bottom-up nature: an attempt is made to explain complex phenomena using simple assumptions. Epstein *et al.* [111] demonstrate this idea in their book where they evolve an artificial society, chapter by chapter, progressing from a very simple society to a society which performs complex tasks such as trading.

[3] More than 30 years onwards, the Schelling model is still used, analysed, changed, e.g., see [54, 398].

[4] http://ccl.northwestern.edu/netlogo/models/Segregation

[5] http://cs1.gmu.edu/~eclab/projects/mason/projects/schelling/

[6] This study was conducted by Epstein and Axtell and is part of the 2050 Project, a joint venture of the Santa Fe Institute, the World Resources Institute, and the Brookings Institution.

The society is evolved by introducing new simple laws to the environment or the agents. Using this approach, Epstein *et al.* want to show that social phenomena can be seen as the result of the interaction of different conditions on different levels of the model. Looking at social science from this perspective leads to the possibility of accounting for various phenomena in an interdisciplinary manner.

Sugarscape aims to provide researchers with artificial laboratories (i.e., artificial societies) that can be used to construct models and to run social experiments. In order to be able to simulate a wide range of social phenomena, Sugarscape is very flexible in terms of the rules applicable to the agents and the environment. Sugarscape has been used for an extensive amount of research in the area of artificial societies [4, 101, 112, 110, 144, 145, 146]. Artificial societies have, in general, been studied extensively, for example, in relation to social learning [11], and for investigating sexual signalling [251]. Different categories of artificial societies (open, closed, semi-closed, and so) have been identified by Davidsson [66]. Still, artificial societies are not without criticism, for example, see [11].

Model

As mentioned above, the Sugarscape model consists of three components:

- *Agents* – Agents are the inhabitants of Sugarscape. Agents exhibit internal states (e.g. metabolism-rate, vision, sex) and behavioral rules. Some states are constant (e.g. sex) while others can change over time through interaction with the environment or other agents (this depends on the model).

- *Environment* – The environment is a medium which is independent from the agents. Agents can operate on the environment and interact with it [111]. The environment also changes due to the agents (i.e., if sugar does not regrow and agents have to eat sugar, this will reduce the amount of sugar in the environment).

- *Rules* – There are 3 types of rules agent rules, agent/environment rules and environmental rules. Agent rules define the interaction between agents (e.g. reproduction). Agent/environment rules describe the interaction between the environment and the agents (e.g. consumption of sugar). Environmental rules describe the behavior of the environment over time (e.g. resource re-growth).

Let us now consider the simplest Sugarscape model: In order to survive agents have to consume a certain amount of sugar per simulation step. The amount of sugar which needs to be consumed is determined by their metabolism rate. The metabolism rates are constant and randomly initialised, which means they differ between agents. Each agent exhibits a certain visual range which is also randomly initialised. The visual range of an agent determines how much of its neighborhood it can perceive (note that agents can only observe the horizontal and vertical neighborhood). In order to find sugar, an agent needs to move through the environment. The following movement rules M determine an agents movement.

Movement-rules M:

- Find the position with the most sugar within your vision.

- In case of multiple options, choose the nearest location.

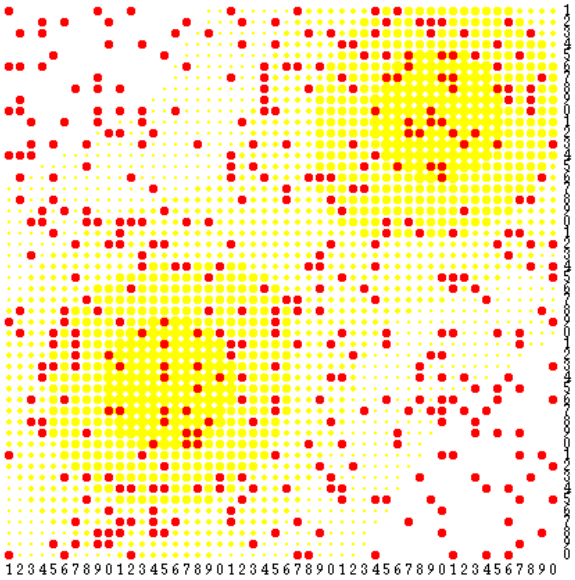


Figure 4.1: An example Sugarscape world (From: [111]).

- Go to this location.
- Collect the sugar on this position.

Collected sugar is stored in an agent's internal sugar repository. If the agent is on a sugarless patch or on a patch that exhibits less sugar then needed for its metabolism, the amount of sugar within its sugar-repository decreases. In order for agents to stay alive, patches in the environment which have been harvested can re-grow sugar with a certain re-grow rate. The re-growth rate has an influential effect on the agents' behavior, as will be shown in the Results section.

Implementation

Sugarscape was implemented using the object-oriented programming paradigm (supposedly in C++, but the book [111] does not mention that explicitly). As outlined above, Sugarscape consists of two main parts the agents and the environment. Both are implemented as distinct objects. These objects contain the states and methods necessary for the desired behavior. The object-oriented approach is used in order to enable an easy upgrade of agents (i.e., agents can be extended with new functionalities by creating a new agent object which inherits all the functions from the old agent object). Besides this implementation, Sugarscape has been implemented in Java[7] and for various multi-agent platforms such as Mason[8] or StarLogo[9].

Results

This section will discuss the effect of a variation in the re-growth rate on the simple Sugarscape model described above (for a detailed description see chapter two of Epstein and Axtell's

[7] http://sugarscape.sourceforge.net/
[8] http://cs1.gmu.edu/~eclab/projects/mason/projects/sugarscape/
[9] http://education.mit.edu/starlogo/samples/sugarscape.htm

book. [111]). As shown in Figure 4.1, the model consists of an environment which contains two sugar piles (in the lower left and upper right corners). Individuals are initialised randomly in the environment. In such a model, agents will move until they either die out (due to a lack of sugar) or reach a sugar-patch from which they can not see any better patches. This model results in two clusters of agents (one on each sugar pile).

Epstein and Axtell showed that the phenomenon of seasonal migration can be introduced in this model by varying the re-growth rates of the sugar piles. The extended the initial model with seasons (i.e., summer winter). If it is winter in the top half of Sugarscape, it is summer in the lower half, and vice versa. The seasons have an effect on the re-growth rate of sugar (i.e., sugar re-grows very slow in winter and very fast in summer). The same agents as in the initial experiment exhibited to this kind of environment exhibit a seasonal migration behavior. After clustering on the sugar piles they will move from one sugar pile to the other sugar pile depending on the season in order to optimise their influx of sugar.

4.1.3 Artificial Anasazi

This study tried to model[10] the historical devolution of the Anasazi population, who lived in the Long House Valley, Northern Arizona from approximately 1800 BC till 1300 AD [20, 85]. The aim of this study was twofold. It was intended to serve as a real world example of the usability of agent-based computational models in the social sciences (i.e., generative social science) [109, 110, 196] as initially proposed in the Sugarscape project of Epstein and Axtell [111]. The Anasazi case is well suited for this purpose, since a lot of real world data concerning the environmental and cultural conditions are available. It was also intended that the study would provide archeologists with thought-provoking ideas regarding the question of why the Anasazi population became extinct.

The Anasazi populated the Long House Valley, which is a part of the Navajo Indian Reservation in north-eastern Arizona [73]. It is thought that the introduction of maize to this area triggered the transformation from the sparse primitive population which inhabited the valley before the Anasazi culture. The cultural changes which mark the rise of the Anasazi are expressed in the form of a very good infrastructure (i.e., widespread roads), extensive trading networks, advanced pottery, the development from pithouses to stone masonry and adobe pueblos, and also changes in other ceremonial structures used by earlier inhabitants of the valley. The Anasazi are therefore seen as the genesis of the cultural configurations which define the modern Pueblo people.

Despite these advances, the population vanished around 1300 AD, which was thought to be caused by a climatic change which affected the maize crops cultivation.

Model

Within the model, two main components can be identified, namely the environment and the agents.

In order to set up a feasible environment, 3D satellite maps of the region were used. The end result is an 80×120 square grid, with each cell corresponding to approximately one hectare [220]. Each cell has a certain maize capacity which is based on its environmental attributes (i.e., hydrologic and depositional conditions, effective moisture and climate). A

[10] The Anasazi simulation model was developed at the Santa Fe Institute in cooperation with the Brookings Institution and the Arizona State Museum.

cell's maize capacity changes each year based on the paleo-environmental reconstruction of the historical conditions from A.D. 400-1450.

The environment is populated by agents that represent the smallest possible social unit within the Anasazi population (i.e., a household consisting of five people). They are initialised at known historical locations within the environment. An agent inhabits a residence site and a farmland site (both represented by cells within the grid). Farmland sites are exclusively used by one agent, while residence sites can be shared by several agents. Agents can perform two basic actions movement, and the building of a new household (i.e., reproduction). Movement is initiated if the agent is not able to survive using its current farmland site (i.e., maize consumption exceeds maize production). In such a case the agent will try to establish better conditions on another farmland site. When the agent reaches a certain age (15 years), it will establish a new household with a certain probability (this is a parameter of the simulation). This form of reproduction accounts for the fact that children within a household are expected to have reached marriageable age after that time, resulting in the establishment of a new household.

Whenever an agent moves, it has to choose a new farmland and residence site. The new farming site needs to be unfarmed and uninhabited, and it must also satisfy the resource requirements of an agent (i.e., high enough maize capacity). The residence location has to be within 1 km of the farmland and must be unfarmed. It also has to exhibit less maize productivity than the favored farmland. In case of multiple options, locations which are closest to the water resources are favored. If no site meets these conditions, the residence location conditions are relaxed. If the environmental conditions are too hard (i.e., a farming site satisfying the resource requirements can not be found), an agent will leave the valley.

Implementation

The Anasazi model was implemented using the Swarm modelling environment[11].

Results

Using the model described above, it was possible to reproduce the spatial and demographic features of the Anasazi in the Long house Valley during the time period of interest [20]. The results of the simulation differ from the true fate of the Anasazi, in that the agent population does not totally vanish under bad environmental conditions, which suggests a high pull-off effect (i.e., the movement of a family triggers the movement of other families). As pointed out by Epstein [110], the conclusion is not that this model solved the mystery of the Anasazi, but that agent-based modelling permits a new kind of empirical research and fosters interdisciplinary collaboration as seen in this project.

4.1.4 Human Learning Environments

This study[12] looked at learning methods (or: goal structures) that are used by teachers to teach a class or group of students. Such methods can be *individual* – students work on their own to learn new things, *competitive* – teacher makes student grades relative to the achievements of his/her peers, or *cooperative* – students work together on a task. The

[11] http://www.swarm.org/

[12] Master thesis project of M. Spoelstra under supervision of E. Sklar (City University, New York) and M.C. Schut (Vrije Universiteit, Amsterdam). Thesis can be obtained from the Vrije Universiteit, Amsterdam [321].

aim of the study was to compare these different goal structures with each other and to identify different factors that influence the learning behaviour of individual learners. The pedagogical literature describes factors that influence individual learning behaviour, e.g., ability and motivation – such factors were included this study. In particular, the work of the pedagogical theorist Vygotsky [372] contains the idea of a *zone of proximal development*: a mental frame indicating a level of development withing which information of a certain level can be best processed by a person. This "zone" was taken as a central concept in the simulation study. The study took place within in the context of the SimEd [315, 314, 333] research project. In this project, computer models are developed to study educational systems of three levels: school district (or: governmental), school house and classroom level.

Model

Based on an extensive literature research on the topics of pedagogy, e.g., [296, 181], multi-agent systems, and distributed AI, the research developed their simulation model. The model consisted of a set of learners and an environment. Each learner has some properties (like ability, understanding, zone, etcetera). The values of these properties prescribed how a learner should behave. The environment contained the three learning methods: individual, cooperative and competitive. For the cooperative educational structure, the Student Teams Achievement Decision theory (STAD) learning mechanism was used. This mechanism has five important characteristics, which all played a role in the model environment: class presentations, teamwork, quizzes, individual improvement scores and records, and team recognition.

Implementation

A computer simulation of the model was implemented in NetLogo. The implementation required to quantify the model variables. These variables concerned mainly "vague" concepts, e.g., ability, emotion, understanding, and these were hard to quantify. A pragmatic approach was taken; for example, emotion is a value between 0 and 1 and depends on how well the student performs on the a test. In cooperative learning, if the emotions of the learner's teammates is higher than its own, emotion is decreased by 0.01; and vice versa. The researchers put much effort in motivating their choices how the vairables were quantified, based on the available background literature on pedagogy. The learning environment contained concepts that had to be learned, varying in difficulty.

Experiments

In the experiments, the researchers investigated individual learning, competitive learning, and cooperative learning with respect to the learner's individual learning improvement and absolute performance (zone and passing rate). For cooperative learning, they also researched: team size, team composition, effect of team rewards, and competitiveness.

When comparing the three different learning methods, the researchers conclude that both high ability and low ability learners display similar learning behaviours in the individual and competitive goal structures. Also, most cooperative learning situations lead to an increase in the overall development of both high and low ability learners. These findings depend much on the specific settings of the model variables. Thus the researchers elaborate much on how

the interaction of the different variables that they discovered, links back to the pedagogical literature.

4.2 Biological Sciences

One of the stereotypical examples of collective intelligence in biology is as observed in insect colonies: ant foraging, dancing bees, and so forth. While quite some biological research in collective intelligence is about self organisation in such insect colonies [49], it stretches a broader field about animal behaviour in general [158] with respect to, for example, language and evolution. It is noteworthy that Nature magazine regularly publishes on research discoveries in this field [124, 203].

We begin this Section by briefly discussing two typical well-known biological examples of collective intelligence: bird flocking and fish schooling. We then present studies on primate dominance interaction about dominance structures in primate societies, self organised patchiness about patch vegetation patterns and division of labour in social insect societies.

Bird Flocking One of the earliest sucessful examples of the analysis of collective intelligence with computer simulation is the simulation of the *flocking of birds*. This phenomenon was first modelled in a computer simulation by Reynolds in the 1980s [287]. The computer simulation is called BOIDS and can be found online[13]. Reynolds identified three parameters based on which individual birds determine there steering behaviours: *separation* – steer to avoid crowding local flockmates; *alignment* – steer towards the average heading of local flockmates; *cohesion* – steer to move towards the average position of local flockmates. Based on these steering behaviours, the BOIDS model shows realistic flocking behaviour as observed in Nature. The BOIDS model is often cited as an example of principle of artifical life. For example, it has been used in more general type of creatures as entertainment software agents [142], inspired more elaborate models of parallel simulation of group behaviours [399], it was used in an autonomous disc jockey programma [171]. Flocking itself was investigated with respect to mobile agents by Tanner *et al.* [334].

Fish Schools Another typical biological phenomenon involving collective intelligence is *fish schooling* – see Figure 4.2. Although the movement of individuals fishes has been modelled [352], schooling is especially relevant with respect to collective intelligence. Hemelrijk and Kunz [159] have looked at an individual-based model that shows general spatial patterns with regards to density distribution and size sorting. The individual rules are simple: avoid what is close by, align to the ones that are at intermediate distance and move towards those further off. Krause *et al.* [205] have researched leadership in such schools. In fish schools (or: shoals), the ones that swim in front largely determine the movements of the school. The researchers discovered that the individuals in these front positions are usually characterised by a large body size and lower nutritional state. Finally, Kunz and Hemelrijk [209] show that fish schools can also operate without leaders, via processes of self organisation. The researchers investigated the interactions between a number of properties of the school and its individuals (body size and form, school size).

[13] http://www.red3d.com/cwr/boids/

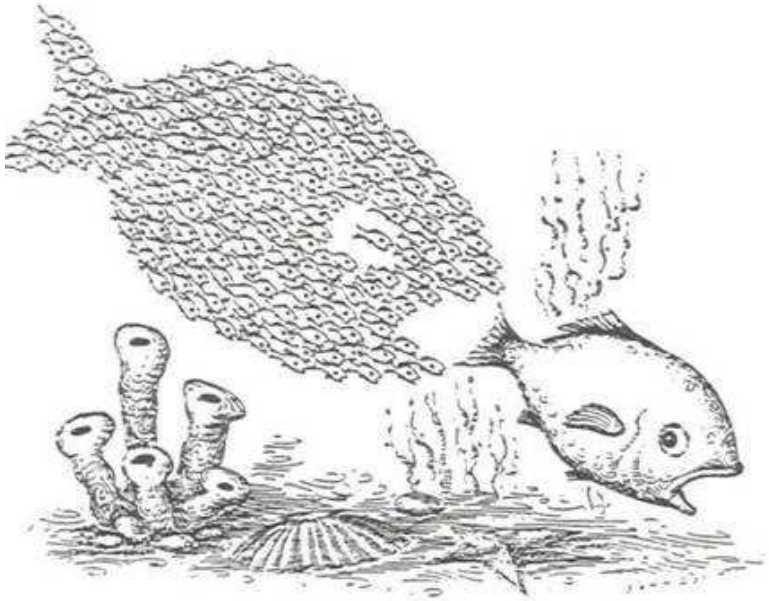


Figure 4.2: Fish schooling.

4.2.1 Primate Dominance Interaction

This study looked at the influence of cohesiveness on the emergence of dominance structures within primates through self reinforcing interactions[14] [153, 154]. Dominance has been identified as one of the key features in the organisation and social behavior of primates [93, 160]. Dominance hierarchies within primate groups are established by competitive interactions between group members. Individuals with a high dominance rank gain more benefits regarding feeding, security and reproduction within their group. How such hierarchies are established is still a subject of debate. The conventional belief is that such a hierarchy is the result of the (inherited) quality of the individuals [107]. From this point of view the hierarchy is something deterministic and rigid (i.e., a given group will always evolve the same dominance structure). The alternative belief, which this study takes into account, sees a dominance hierarchy as a product of self organisation which depends highly on chance and the self reinforcing effects of victory and defeat (i.e., the 'winner-loser effect') [52, 94]. The winner-loser effect can be observed in many species and states - losing or winning a dominance fight has a sustainable effect on a individual. Individuals that lose a dominance fight are likely to lose their next fight, even when they are superior to their opponent. Winning a fight on the other hand boosts the self-confidence of an individual, and therefore increases the probability of the individual winning the next dominance fight.
Another point which has to be taken into account is male-female dominance retaliation. Fe-

[14]The study was conducted by Hemelrijk *et al.* and is part of the "Evolution of Social Systems" project at the Artificial Intelligence Laboratory (University of Zuerich).

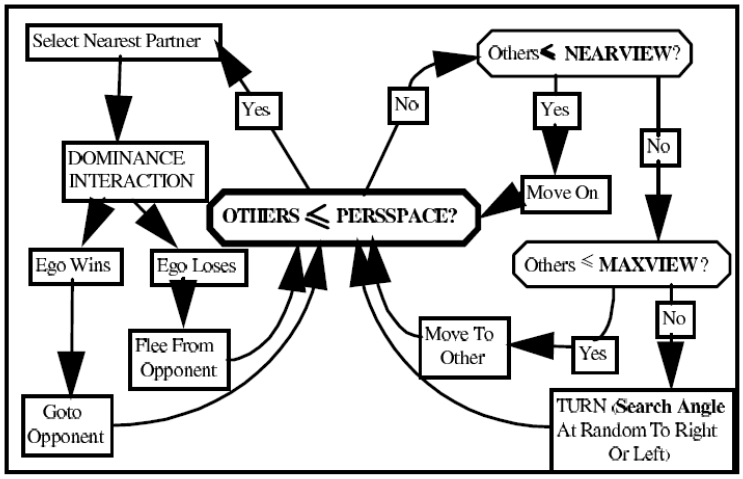


Figure 4.3: Flow chart for the behavioural rules of agents that are not attracted to another type (From: [155]).

males are often inferior to males in terms of size and fighting capacities and are normally considered to be subordinate to males. However, there are primate species in which this is not the case (e.g. pygmy chimpanzees). In such species, females are also likely to dominate males. It has been suggested that this is due to the fact that in these species females have a tendency to form groups and collectively fight against males [344]. The alternative explanation which this study tries to bolster is that female dominance in some primate species is not a result of a stronger "sisterhood" in such species but rather a result of cohesiveness.

Model

The DomWorld model is based on the ideas outlined above [153, 154, 155], and consists of a virtual world which is populated by two types of agents: weak-type and strong-type agents. These types differ in their initial fighting capacity. Strong-type agents are initialised with a higher dominance value and are therefore more likely to initiate dominance interactions.

The inhabitants of the world have two tendencies: to group and to perform dominance interactions. The reason for these tendencies is not explicitly specified. There are several possibilities for such behaviors, but they are irrelevant for this model.

Figure 4.3 show the behavioural rules of agents. When individuals violate the personal space (PersSpace) of other individuals this can lead to dominance interactions. An individual is likely to initiate a dominance interaction if the likelihood of its success is high (i.e., if it has a high dominance value). The dominance value depends on the self reinforcing effects of winning (and losing): An individual which succeeds in a dominant interaction will increase its dominance value and the opponent will reduce its dominance value. Defeating a higher ranked group member has more effect than defeating a lower ranked group member, allowing ranks reversals. A dominance interaction ends with the loser being chased away by the winner (for a precise mathematical formulation of the dominance interactions see [154]).

Variance in cohesiveness is achieved by varying the vision field of the agents; their tendency to group is equal. Agents with a wide vision field (e.g. 360 degrees) are able to observe more of their environment, which leads to a higher possibility of entering the personal space of

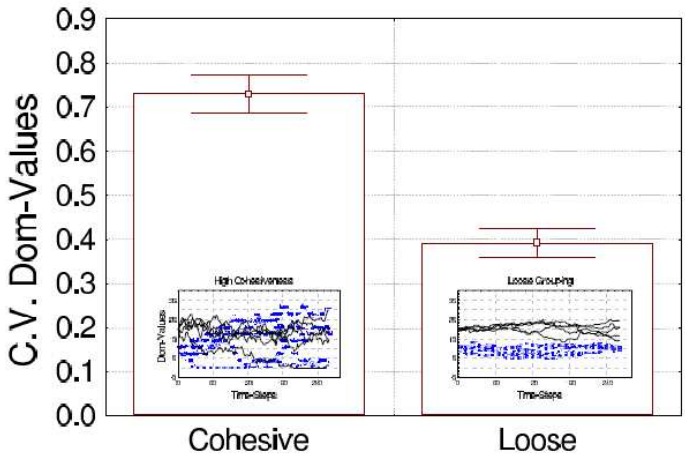


Figure 4.4: Differentiation of the dominance hierarchy, measured by the coefficient of variation of dominance values, in cohesive and loose groups (From: [155]).

other agents, due to their desire to group.

Implementation

The DomWorld model was initially implemented as a multi-agent model using object-Pascal, Borland Pascal 7.0 [154]. It consists of a 200 by 200 units toroid, which represents the world. The world is normally populated with 10 agents (5 weak-type and 5 strong-type) which move and interact in the world. Agents are initialised to random locations within a predefined subspace of 30 by 30 units. Agents have vision field of a a certain degree and a maximum range of sight (MaxView). The activity of an agent within each time-step is regulated by drawing random waiting times from a uniform distribution. The agent with the shortest waiting time is activated first. Normally the waiting time stays constant for all agents during a step, this is however not the case if a dominance interaction occurs within NearView of an agent. In such a case the waiting time is reduced, which leads to an increased probability of the agent being activated and being able to react/interact. A dominance interaction uses the rules described above.

Experiments

Hemelrijk *et al.* [153, 154, 155, 156, 157] were able to show that cohesiveness indeed has an effect on dominance – see Figure 4.4. In sparse groups (low field of vision), ranks are able to differentiate, meaning that low ranks will stay low and high ranks will stay high. This is due to the fact that it is unlikely that an individual with just a limited field of vision will bump into a higher ranked individual which already defeated it. This leads to a scenario where individuals of equal strength cluster.
This however is not the case for cohesive groups - in such groups individuals are constantly involved in dominance interactions. Weak-type individuals which have been lucky in the first rounds are therefore able to increase their dominance. This leads to the situation that is also observed in pygmy chimpanzees, namely that females are more dominant.

4.2.2 Self Organised Patchiness

This study[15] looked at arid ecosystems where self organised patchy vegetation patterns are a common occurrence [293, 292]. Here, short range facilitation (positive feedback on a small scale) and long range competition (negative feedback on a large scale) play a big role in the emerging of those patterns.

The patterns consist of patches of vegetation alternating with areas containing hardly any vegetation. A patch is a group of plants surrounded by bare soil. The specific shapes of the patches vary. A spotted pattern consists of bare soil with only spots of vegetation; a striped pattern has bare soil with stripes of vegetation. Another possible pattern is the so-called labyrinth. Here the vegetation patches have grown together to form a mazelike structure, which can still contain some isolated spots of vegetation. In bogs in North America and Eurasia "string patterns" are formed, made up of vegetated bands alternating with pools [290]. In savanna and (semi-)arid ecosystems the patterns range from bare gaps in vegetated areas, via labyrinths and stripes of vegetation, to isolated spots [291].

The research actually involved multiple studies over the time period of about a decade (1995-2006). Firstly, Thiery et al. [345] proposed a model to explain banded vegetation patterns that occur in arid and semi-arid zones. Thiery *et al.* call these "tiger patterns". They are generally found on gentle slopes. Rietkerk and Koppel [293, 292] considered plant-soil relations as an explanation for the existence of alternate stable vegetation states, i.e., states with and states without plants. HilleRisLambers *et al.* [162] explained the origin of vegetation pattern formation in semi-arid ecosystems. Rietkerk *et al.* [289, 290, 291] looked at various ecosystems that link self organised patchiness to catastrophic shifts. Those shifts are between alternate stable states, one with and one without vegetation. Finally, Rijnders [294] developed an agent based model that explicitly and separately contained the factors identified in previous studies of Thiery, Rietkerk, etcetera and extended the model with heterogeneity. The factors cause the short range facilitation and long range competition, and thus the patchy patterns: 1. the relation between the presence of plants and the capacity of soil to absorb water, 2. the flow of surface water, and 3. the infiltration of water into soil. Heterogeneity was defined in the environment: not every patch of ground behaves exactly the same; for example, some patches may be soil, others may have rocks on them.

Model

All studies mentioned above used Cellular Automata (CA) for modelling. We look at the model that Thiery [345] developed. This CA has a 2D grid. The grid represents the sloping surface on which the pattern will emerge. The top of the grid stands for the top of the slope. The cells represent trees, which can be in one of four possible states, which symbolise the tree's performance. State 0 represents that there is no tree, or a dead tree, state 1 a young or senescent tree, state 2 a small or a stressed adult tree and state 3 a well watered adult tree. A neighbourhood of 9×3 is used, with 6 cells located above the current cell, 2 below, and one column on each side. The new value of a cell is calculated using a convolution matrix; a 9×3 matrix, covering the entire neighbourhood, which contains a value at each location, except the one covering the current cell. This matrix is used to calculate the amount by which the value of the current cell will change.

[15] Master Thesis project of K. Rijnders under supervision of M.C. Schut (Vrije Universiteit, Amsterdam). Thesis can be obtained from the Vrije Universiteit, Amsterdam [294].

Implementation

The TIGREE model implemented the CA and was written with the VOYONS general purpose modelling software, which was developed by Thiery himself. The "VOYONS" application is available on request[16]. Only two specific subroutines were added, one for the fast computation of local interactions and one for the graphical display of tree-like symbols.

Experiments

The conclusion of [345] is that "the results demonstrate that almost all the structures observed in the field can be generated by this simple model". Slopes are, according to this model, an important factor in the formation of these patterns: "In semiarid and arid zones, the landscape imposes constraints on the orientation and rates of [water and nutrient] flows. In response to such constraints the vegetation cover tends to contract and to produce specific patterns [...]". This suggests that the patterns are produced because of the influence the shape of the landscape has on water and nutrient flows.

4.2.3 Division of Labour

This study looked at individual specialisation in insect society in in the context of division of labor [34, 129]. Division of labor, or polyethism, is one of the most interesting phenomena in social insect societies. It seems to be responsible for the ecological success of social insect societies, because it increases the efficiency of social insects [384] (i.e., many different tasks can be performed at the same time by many different individuals). There are two general concepts underlying the emergence of the division of labor in social insect societies. Individuals may work on different tasks depending on their age. It can for example be observed in bee colonies, where young individuals work on tasks in the hive (e.g. broad care) and old individuals work on tasks outside (e.g. foraging) [306].

The second type of division of labor is based on morphological differences among individuals of a social insect species. It is called morphological polyethism. The differences in workers' morphologies results in physical castes within a colony. Physical castes are most strongly developed in termites and ant species. Termite colonies usually contain three basic castes (workers, soldiers, and reproductives), which differ in morphology and the tasks they work on (for detailed information about termites see [383, pp. 103–119].

Colony-size is also thought to have an effect on division of labor: Small colonies have to exhibit a certain flexibility in order to cope with dynamical changes in the environment. Larger colonies are more robust to such changes due to their size. It has therefore been observed that specialisation (i.e., workers working just on one task) normally occurs just in species which exhibit large sized colonies [348, 184].

Model

A threshold reinforcement model (TRM) was used to model the division of labour [129]. TRMs are based on the assumption that workers respond to stimuli which they perceive in their environment. Workers have an intrinsic response threshold for each task, which determines how sensitive they are to the task-specific stimulus. The default state of a worker is to do nothing, but if a task-specific stimulus exceeds a worker's response threshold, then it

[16] E-mail: thiery@orstom.orstom.fr.

is very likely that the worker will react to that stimulus by performing an appropriate action. Workers can vary in their response thresholds due to many reasons (e.g. experience, genotype, physiological state [129]). In TRMs not only the stimuli but also the thresholds are updated every timestep due to a positive reinforcement process. As an individual works on a task, it becomes more expert in it, and thus its threshold associated with the task decreases. If it does not work on the task, it forgets how to do the task and thus the threshold increases. More precisely a TRM consists of N individuals which can work on m tasks $T_1, \ldots, T_m$. Each task T_j has a task-specific stimulus value $S_j \geq 0$. Each individual i has a task-specific threshold value $\theta_{i,j}$ such that $\theta_{i,j} \geq 0$. Let X be the state of an individual. The state of an individual determines what it is working on. Each individual has $m+1$ possible states, because it can work on one of the m tasks, or stay idle. If an individual is engaged in a task, it will do α units of work on this task during one timestep ($\alpha > 0$). At each timestep, the threshold values for each task T_j are changed for each individual i as follows:

- if i works on T_j, then $\theta_{i,j} = max\{\theta_{i,j} - \xi, 0\}$
- if i did not work on T_j, then $\theta_{i,j} = min\{\theta_{i,j} + \phi, {\theta_j}^{max}\}$

θ_j^{max} is the maximal threshold for task j (θ_j^{max} is necessary to prevent thresholds of getting infinitely large). ξ is the learning parameter and ϕ the forgetting parameter (Note that the standard assignment in TRMs is $\phi = 3.5, \xi = 4.0$). Individuals are initialised with a threshold of around 0.0 for both tasks.

In TRMs, the specialisation of an individual is not set apriori as, therefore such models are ideal to study the evolution and degrees of specialisation in populations with different colony sizes [233, 343, 129]. In order to do this, a measurement for a individual's specialisation is needed. Such a measurement for a system with 2 tasks was introduced by Gautris et al. [129]. In order to measure the specialisation of an individual, two more values (properties) are needed: Let C_i be the number of transitions divided by the total number of periods of work minus 1 for the individual i. The degree of specialisation F_i of the individual i is measured by $F_i = 1 - 2 \cdot C_i$. If an individual i has not switched between tasks, $F_i = 1$, thus that individual was highly specialised. If an individual worked on tasks in a random order, $F_i = 0$, thus that individual was not specialised. If an individual worked on the tasks in a alternating way, $F_i = -1$. The task-activity W_i of an individual i, is measured as the proportion of time steps it was working on a task j ($W_i = \sum_{j=1}^{m} W_{i,j}/t$).

Experiments

Gautris *et al.* [129] examined the differences between individuals' degrees of specialisation (measured over all time steps), in a two task model (i.e., T_1, T_2), for different sized colonies[17]. They showed that in colonies which are not too small (i.e., $N \geq 20$), specialists occur under medium ($D = 0.5$) and high ($D = 0.8$) demand situations.

The explanation for this effect was seen in the magnitude and frequency of stimuli differences. In small sized colonies, periods of high stimuli differences lasted longer than in colonies with more individuals. This is due to the fact that the stimulus has a wider range in large colonies, and can therefore decrease or increase faster. If $S_1 - S_2$ is large, the number of specialists for task 2 in the population will decrease, because individuals are more likely to work

[17] Note that this study involved a mathematical model – therefore this Section does not contain a description of the implementation.

on task 1 (due to the higher stimulus) and thus forget task 2. The same will happen for task 1 if $S_2 - S_1$ is large. In large colonies, stimuli differences do not last as long as in small colonies. This enables individuals to work sufficiently long on a task for the feedback of learning to take effect. The conclusion was that specialisation occurs only if the colony size exceeds some critical value. This conclusion is also supported by empirical investigations [348, 184].

4.3 Economic Sciences

In the social sciences, we saw that it is interesting to look at or explain social phenonemena from a *bottom up* instead of or in addition to a *top down* perspective. The same applies to economics. Traditional models of economics look top-down, for example, by means of equation-based models. Recently, the fields of computational economics, agent-based models, and others, offer a way to analyse economic phenomena bottom up.

In this Section, we present a brief overview of the field of agent-based computational economics, we explain the iterated prisoner's dilemma and we outline a research project on the economic impact of environmental dynamics.

4.3.1 Agent-based Computational Economics

Imagine that you want to research the behaviour of a traditional economic market. The particular market that you are interest is an auction-based market: products are sold by auction, like ebay. This particular market 'suffers' from two important notoriously difficult problems: people are sometime irrational, and auctions may have some strange complex behaviour. If you put these two things to the extreme, then disastrous events like the 1987 U.S. stock market crash may occur. For these things, "traditional economics" is left in the cold: it assume rationality of involved individuals and linear behaviour of involved mechanisms. Steiglitz and O'Callaghan [326] present a simulation study that researches these 'endogeneous price bubbles' (like the stock market crash). The novelty of their study is that they model *on the agent level* and yield results that can complement (not replace) theoretical explanatory methods.

In general, since the mid-1980s, economics has seen the upcoming of a research field called *computational economics* [7]. This field explores the intersection of economics and computation[18], including research on computational econometrics, computational finance, automated internet markets and so on. One such an area of interest in this field is *agent-based computational economics*[19] (or: modelling) (ACE). This area concerns "the computational study of economic processes modelled as dynamic systems of interacting agents" [17, 341], or in other words, "growing economies from the bottom up". It bridges and combines research from the disciplines of evolutionary economics, cognitive science and computer science. According to Tesfatsion [337, 338, 339, 340], there are four major strands within ACE: *empirical understanding* – why have certains patterns emerged despite the absence of top-down planning and control; *normative understanding* – how can good economic designs be discovered with ACE models; *qualitative insight and theory generation* – why did certain patterns emerge, but other ones not; *methodological advancement* – what methods and tools do ACE researchers need?

[18] http://comp-econ.org/
[19] http://www.econ.iastate.edu/tesfatsi/ace.htm

Research areas in ACE stretch further than only analysis, but also include design studies. The areas of interest within ACE are, among others, network formation, evolution of norms, development of computational laboratories, automated markets and software agents. As mentioned, some of these areas concerns the design of multi-agent systems – later in this book we look in detail at negotiation, trust, reputation and mechanism design.

A large part of ACE concerns the development of models. The key characteristics of such models are: 1) inclusion of *agents*, 2) these agents are situated in realistically rendered problem environments, and 3) behaviour and interaction patterns that can develop/evolve over time.

4.3.2 Iterated Prisoner's Dilemma

The *prisoner's dilemma* is a situation in which two prisoners (in separate rooms) each have the choice to confess (or not) to a crime that they are accused of. The four possible combinations of their choices determine the times that they have to serve. Each prisoner wants to minimise the time that he has to serve. If neither confesses, both serve half a year; if both confesses, both serve two years; if prisoner A confesses and B does not, then A goes free and B goes to jail for ten years; and vice versa. In game-theoretic terms, it is a *non-zero-sum game*: the total outcome of the player's actions is not zero, thus one player may come out *absolutely* better than the other player.

This dilemma was first thought up in the 1950s. In the 1980s, Axelrod [13, 14] explored an extension of this dilemma, whereby it was repeated some number of times, called the *iterated prisoner's dilemma* (IPD). The specific setting was a population of agents that would meet each other randomly and at every encounter play the prisoner's dilemma. Agents remembered whom they played, and could use this knowledge when faced with the same opponent. Axelrod was interested in what were good strategies: always defect, always cooperate, take the history into account, and so forth. The experimental setting was one in which Axelrod invited people to submit their strategies and he would put them together in an IPD tournament. The best (deterministic) strategy was tit-for-tat whereby an agent does the same action that his opponent did to him the last time. This proved to be a very effective strategy (although only four lines of code), beating over 60 other strategies.

The relevance of this dilemma is that, among other things, the evolution (and complexity) of cooperation can be investigated with it. These are also the titles of the books that Axelrod published about his findings. Axelrod discusses two real-world applications in [13]: the thrench warfare in World War I and the evolution of cooperation in biological systems.

4.3.3 Ecological Economics

Ecological economics is an approach to economics addressing the interdependence and coevolution between human economies and natural ecosystems. It is closely related to evolutionary economics (economics based on biology) and industrial ecologies (looking at the industrial system as an ecosystem). It considers economics to be a subfield of ecology.

Economist van den Bergh has looked at several aspects of ecological economics in relation to evolutionary modelling. For the management of renewable resources (e.g., fisheries), traditional economic models compute optimal harvesting strategies based on perfect rationality of the harvesters. Noailly *et al.* [250] show that, based on evolutionary machanims, there can be multiple strategies that are used simultaneously and yield optimal results. Another study

show that for network markets (products or services assembled from alternative combinations of basic products), equilibria are influenced by the topology and structure of the network – but these equilibria may not agree with the socially optimal ones [295]. In more general overview studies, van den Bergh discusses the micro-macro distinction in economic theory and modelling [359] and evolutionary modelling in evolutionary and ecological economics [361, 360, 358]. A more recent research project of van de Bergh concentrates on delivering an evolutionary model to research the impact of environmental dynamics on social systems[20]. The model contains a number of individual behavioral strategies that together form a complex system. One particular application of the model is to research how to distribute climate change regulations over a number of countries, where each country can autonomously determine its policy on climate change. There are existing economic equation-based models for this [255] that addresses the problem in a centralised manner.

Related work of van den Bergh and Janssen [362] looks at industrial ecologies. Janssen has also utilised the multi-agent based model to manage ecosystems [177]. Topics within this domain include land-usage decision making, integrating local's knowledge with scientific measurements, usage of multi-agent systems as role games, and so on. Other work of Janssen addresses issues of collective behaviour in general, e.g., [139].

4.4 Science of Philosophy

The science of philosophy is a science where collective intelligence can be understood ambigiously. On the one hand, one can analyse it in the same saw as we showed for social science and biology. In this way, we take a philosophical issue at hand, for example, the philosophy of mind, and investigate research questions in this area by means of computer simulation. An exmple is shown below, where we describe a research study on *shared extended mind*: the exploitation of patterns in the environment as external mental states. On the other hand, we may consider the computer itself to be a mind and investigate simulations as such. This second understanding of the philosophy of collective intelligence touches on the fundaments of artificial intelligence [82, 81, 80, 79, 78, 77, 76, 75, 163, 236]. In the second part of this Section, we present research, as suggested by Dennett, on an evolutionary account of free will.

4.4.1 Shared Extended Mind

Some types of animals exploit patterns created in the environment as external mental states, thus obtaining an extension of their mind [56, 259]. In the case of social animals the creation and exploitation of such patterns can be shared, which supports a form of shared extended mind or collective intelligence. Humans do it to: behaviour is often not only supported by an internal mind in the sense of internal mental structures and cognitive processes, but also by processes based on patterns created in the external environment that serve as external mental structures. Examples of this pattern of behaviour are the use of 'to do lists and 'lists of desiderata. Having written these down externally (e.g., on paper, in your diary, in your organiser or computer) makes it unnecessary to have an internal memory about all the items. Thus internal mental processing can be kept less complex. The only thing to remember is where these lists are available.

[20] http://www.few.vu.nl/ volker/

Especially in the case of social animals external mental states created by one individual can be exploited by another individual, or, more generally, the creation, maintenance, and exploitation of external mental states are activities in which a number of individuals can participate (for example, presenting slides on a paper with multiple authors to an audience). In some cases the extended mind principle serves as a way to build a form of social or collective intelligence, that goes beyond (and may even not require) social intelligence based on direct one-to-one communication. In such cases the external mental states cross, and in a sense break up, the borders between (the minds of) the individuals and become shared extended mental states.

Model

The main contribution of this work is a detailed analysis of this shared extended mind principle, and a formalisation of its dynamics. The principle is illustrated by a case study of social behaviour based on shared extended mind (a simple ant colony). The analysis of this case study[21] comprises multi-agent simulation based on identified local dynamic properties, identification of dynamic properties for the overall process, and verification of these dynamic properties.

The specification of the simulation model was based on the local dynamic properties [40] for the basic mechanisms. The world in which the ants live is described by a labeled graph, where the locations were the nodes and the edges represented ways to go from location to location. While passing an edge, pheromones are dropped. The objective of the ants is to find food and bring this back to their nest. The model involved multiple agents (the ants), each of which has input (to observe) and output (for moving and dropping pheromones) states, and a physical body which is at certain positions over time, but no internal mental state properties (they are assumed to act purely by stimulus-response behaviour).

The extended mind perspective introduces an additional, cognitive ontology to describe properties of the physical world, which essentially is an antireductionist step, providing a more abstract and better manageable, higher level conceptualisation. Such antireductionist steps can be useful in explanation and theory development. Indeed, following the extended mind perspective a high-level conceptualisation was obtained in this work.

4.4.2 Altruism

A basic assumption in the evolutionary explanatory framework is that an organism's behaviour serves its own interests, to improve its own well-being and production of offspring. For the explanation of the development of altruistic behaviour from an evolutionary perspective, one easily encounters a paradox; see, e.g., [316, pp. 17-23] and [366, 367]. As by definition altruistic behaviour is behaviour that is against one's own interests, this paradox can be formulated as: 'an organism serves its interests by behaviour which is against its interests'. One way to solve this paradox is by widening the scope in the temporal dimension. Then the occurrences of the concept 'interest' in the paradox get different time points attached, as follows: altruistic behaviour serves the organism's interests at a future time point by the organism's behaviour which is against its interests at the present time point. So, the organism's present behaviour is seen as an investment to obtain future revenues for the

[21] Since in this research the main aim was to get insight in the the shared extended mind principle and the case study was merely used as an illustration, we left out a description of the actual experiment.

organism itself [258, 350]. As long as the future revenues are at least as important for the organism as the present investment, this may work out fine, cf. [82, Chapter 7] and [3, 122].

In this case a basic assumption is that the environment of an organism has the potentiality or regularity to provide future revenues in return for present investments. This is a nontrivial characteristic of an environment, that often depends on the presence of other organisms in the environment. For example, other agents in the environment that offer the future returns when they are favoured by an agent, depending on their own 'intertemporal decision making' [222]. To estimate the risk of not getting the future revenues in return, the model of intertemporal decision making can be combined with a model for evolution of trust in other agents based on experiences with them: if the agent experiences over time that another agent does not provide services, the trust in this agent becomes lower; if it does provide services, trust becomes higher. Having such a dynamic environment model enables the agent to become better adapted to the environment. One of the main properties to verify is whether indeed agents with a cognitive system for trust-based intertemporal decision making do well over time, in contrast to agents that lack such a cognitive system.

Model

An artificial society of multiple agents was created performing trust-based inter-temporal decision making, several modelling approaches have been used. First, the LEADSTO simulation environment [40] was used for rapid prototyping, i.e., to create a high-level declarative specification of the simulation model, involving a small number of agents (six in this case). When this specification turned out to show coherent behaviour, it was used as a blueprint to create a large-scale version of the model in the NetLogo environment [36].

Results

The conceptual model was implemented in LEADSTO, modelling a society of six agents. In this model, three agents used a cognitive system capable of intertemporal decision making, whereas the other three agents were not equipped with this system. These three agents simply always selected the current reward, thus they never cooperated. Next, a large-scale simulation of the domain has been implemented in NetLogo, involving 200 agents. In this model, again half of the agents used the inter-temporal decision function, whilst the other half did not.

It turned out that the agents with this cognitive system enabling them to anticipate on the future show more altruistic behaviour. As a result, these agents get a bigger social network, and become in the end more healthy than the agents without such a cognitive system. This is in accordance with the theory of how altruism emerged in Nature as a result of more elaborated capabilities of mankind for inter-temporal decision making; e.g., see [82].

4.5 Summary

This Chapter overviewed a number of research studies into the analysis collective intelligence, i.e., studies with the objective to understand something from the real world. We divided the Chapter into three different scientific disciplines: social sciences, biology, and economics. The studies span a time period from the 1960s up till present day. Some of the researched phenomena (e.g., segregation, altruism) have a much longer scientific history though. We did not aim to cover these scientific disciplines completely with respect to collective intelligence.

The case studies are mainly to illustrate the ideas on collective intelligence outlined throughout this book. Having put these studies together, we hope to inspire you to see parallels and similarities between the different studies and so further the insights on the basics of collective intelligence. Within the context of this book, we aim for the reader to be able to make an informed judgement about the choices that the respective research made. For example, this research study used cellular automata as model, but why were agent-based models or boolean network not used? After reading this book, the reader should be able to answer these kinds of questions.

– The engineer's first problem in any design situation is to discover what the problem really is.

Unknown

5 Design of Collective Intelligence

What do with collective intelligence? If we know how it works, what can we use it for? This is basically the idea of designing collective intelligence. In the past decades, robots have learned how to cooperate based on how social insects do this, computers talk to each other based on how people gossip with each other, and Hollywood has animated crowd movements based on particle system theory. Although there are currently no standards, methodology, or alike for designing collective intelligent systems, there are many studies that undertake such designs. We present a number of such studies here.

Aims of this Chapter

This Chapter aims to present a number of research studies on the design of collective intelligence. The main discipline involved is computer sciences, although some studies may be of particular interest to economists. We have categorised the studies as follows. Firstly, we discuss collective robotics, where researcher have attempted to think up ways to let (mostly physical) robots cooperate with each other. Secondly, we present developments in the area of computer networks relating to collective intelligence: peer-to-peer protocols, autonomic computing, ecetera. The area of insect-based computing is presented – for example the use of ant foraging techniques for solving the travelling salesman problem. Next, we introduce a number of techniques that are much researched in computational economics. Finally, on a lighter but equally important level, we present collective intelligence in entertainment: games and movies.

5.1 Collective Robotics

A very concrete physical application area of collective intelligence is *collective robotics*[1] that "studies the different ways of using autonomous robot teams to efficiently fulfill predefined missions". We look at three different (partially overlapping) areas of interest within collective

[1] http://diwww.epfl.ch/lami/collective/

robotics: *swarm robotics* that takes the concept of swarming as its inspiration, *evolutionary robotics* that takes evolution as the mechanism for adaptivity in robotics, and *behaviour-based robotics* where robots are programmed on the behavioral level.

5.1.1 Swarm Robotics

If you are interested in programming collective intelligence into a group of robots, then swarming is a natural inspiration which may help you out for finding suitable coordination mechanisms. A number of researchers thought likewise and have used swarming as a central theme for the design of robot collectives. Within the last decade, there have been made a number of research efforts to achieve the "swarming" of intelligent robots. We briefly discuss some of these research efforts.

The European research project SWARM-BOTS[2] (2001-2005) had the specific objective to study new approaches to design and implement self organising and self assembling artefacts [239, 91, 349]. These artefacts, called *s-bots*, were composed of number of simpler, insect-like, robots. The project had one particular application: a well thought out classical search-and-rescue scenario. The scenario was divided into a number of different research challenges: coordinated motion, hole/obstacle avoidance, passing over a hole, moving on rough terrain, aggregation, (functional) self-assembly and adaptive division of labour. Each of these challenges could be investigated seperately, and were ultimately combined into finding a solution for the search-and-rescue scenario. The follow-up of the project is the SWARMANOID project[3] (2006-2009). This project has approximately sixty autonomous robots of the types eye-bots (specialised in sensing and analysing the environment from a high position), hand-bots (specialised in moving and acting in the space zone between the ground and ceiling) and footbots (specialised in moving on rough terrain and transporting either objects or other robots) which together form a *swarmanoid*.

Spears and Spears [147, 319, 317] have successfully used their *physicomimetics* model, that was explained earlier in this book, for the distributed control of swarms of robots (or: vehicles). To remind you, the physicomimetics "provides distributed control of large collections of mobile physical agents in sensor networks" [319]. The model treats agents as physical particles and drives the multi-agent system towards some desired configuration. In other words, based on virtual physics, the systems tries to minimise overall potential energy. For some specific set of applications, this model has significant advantages: emphasis on minimality, ease of implementation, and run-time efficiency. The possible applications include distributed sensing grids, and perimeter defense and surveillance.

McLurkin [230, 229] of the MIT Computer Science and AI Laboratory has worked on putting together a library of behaviour-based algorithms for programming swarms of robots. The algorithms have been succesfully used for swarms of robots that disperse throughout their environment. This dispersion may be used for, for example, exploration or surveillance of an area. The swarm-software that runs on each robot consists of behaviours that run concurrently. Example individual behaviours are `moveArc`, `bumpMove` and `followRobot`. These behaviours consists of equations defined on the actions that a robot can undertake (e.g., lateral separation of the robot's wheels, translational velocity). Each robot periodically (for example, every 250 ms) transmits information to its neighbours and a gradient-based

[2] http://www.swarm-bots.org/
[3] http://www.swarmanoid.org/

multi-hop messaging protocol carries information around in the swarm. Example group behaviours are `disperseFromSource`, `followTheLeader` and `clusterIntoGroups`. One of the studies involved experiments with over 100 physical robots that successfully located an object in 3000 ft^2 of indoor space and led a human to this object.

Finally, note that swarm robotics is very closely related to insect-based computing. While such computing may have been focused on, for example, metaheuristics for the travelling salesman problem, the discovered algorithms can also be used in swarm robotics. We do not go into detail of *insect-based robotics* here, although we discuss insect-based computing in general later. Thus be aware of the fact that robot coordination mechanisms can also get their inspiration from, for example, insect colonies, e.g., Krieger's work on ant-like mechanisms in cooperative robots [207] or Kube's work on cooperative transport [208].

5.1.2 Evolutionary Robotics

Evolutionary robotics is a design technique for the automatic creation of autonomous robots based on evolutionary principles (selective reproduction of the fittest) [253]. The basic idea of this class of robotics much resembles the evolutionary method that was explained earlier in this book. The control system of a robot (defining what the robot does under what conditions), is encoded as an artificial chromosome. The robot performs a number of tasks, after which it is evaluated and decided upon whether fit for reproduction or not. When decided fit, the robot controller is combined with other controllers and possibly mutated or duplicated, and the result ends up in the next generation, which goes into another testing round. This continues until some controller is found with which the designer is satisfied. The Khepera concept[4] is a design methodology for autonomous robots that was originally developed for researching adaptive and evolutionary algorithms in robotics. Nolfi and Floreano [253] present a number of tasks that robots can execute based on evolutionary learning, ranging from simple navigation to walking machines. Evolutionary robotics has been extended with co-evolution [253, Chapter 8]. Potter *et al.* [278] investigated the heterogeneity of robot controllers in a population of robots that co-evolved. It has also been used for analysis studies; for example Perez-Uribe *et al.* [275] investigated how the process of evolution of cooperation and division of labour in (simulated) ant colonies.

A related strand of research is *Embodied Evolution* (EE) that extends evolutionary robotics by by-passing the simulation step (that is often necessary in evolutionary robotics) [375]. Problems with transferring simulation results to physical robots are therefore avoided. In EE, the evolutionary algorithm runs directly on the physical robots: it needs some population of robots (arguably many more than present-day evolutionary robots according to the authors), continuous power delivery (for example, a powered floor) and a distributed evolutionary algorithm. In the evolutionary algorithm, individuals perform their evaluation autonomously. Therefore, the standard evolutionary algorithm is extended by some metric programmed into the robots. Simulation can still be used effectively in EE for investigating the evolutionary algorithm itself and the experiment setup, rather than providing a high-fidelity simulation of the robots and their environment. The methodology has been used for a range of examplary physical robot tasks, e.g., phototaxis (individuals move in response to a stimulus light).

[4] http://www.k-team.com/

5.1.3 Cooperative Robotics

We here enumerate work that is collective robotics, but we cannot classify it as either swarm or evolutionary robotics.

During the 1990s, Parker [269, 270] developed the ALLIANCE architecture, which is "a behaviour-based, fully distributed architecture that utilises adaptive action selection to achieve fault tolerant cooperative control in robot missions involving loosely coupled tasks". In *behaviour-based robotics*, a robot is provided a number of different simple basic behaviours. On execution, robots decide on appropriate behaviours based on the current state of the environment. These behaviours are typically layered with primitive behaviours on the bottom and increasingly more 'intelligent' behaviours. When the robot has enough time or energy, it may devote time and energy on executing more complex behaviours; if not, then it falls back on its basic behaviours. The ALLIANCE architecture allows robots to execute *behaviour sets* instead of only single behaviours. Behaviour sets are activitated or suppressed based on motivational behaviours. Within a team of robots, robots can communicate messages to each other that affect their motivations. Robot teams have been able to successfully perform several tasks with the ALLIANCE architecture, for example, box pushing.

A behaviour-based approach for cooperative robotics (for formation control) was also developed by Balch [22], where a group of reactive behaviours was autonomously combined at runtime by a team of robots. This approach was used in team coordination for a group of unmanned ground vehicles. It could successfully let the robots autonomously make line, column, diamond and wedge formations.

Willems [382] investigated a behaviour-based approach to robot design in the context of a collective task. The study investigated whether internal representations of the outside world were essential to produce intelligent behaviour. The study concerned a simulation of a number of different environments in which two types of robots (predator/prey) resided. The researchers observed cooperative behaviour despite the fact that the control structures of the robots were very simple.

Finally, Werger and Matarić [380] developed a behaviour-based control method for cooperative robot teams. They show that when robots are able to inhibit or supress *peer behaviours*, i.e., the same behaviour on another robot, efficient coordination can be achieved. In related work, Jones and Matarić [183] have investigated Large-Scale Minimalist Multi-Robot Systems (LMMS): systems that are "composed of groups of robots that each have limited capabilities in terms of sensing, computation and communication". They specifically looked at the capability of an LMMS to achieve some desired division of labour if the task of the system required so. The experimental setup involved a puck-collection scenario carried out by a number of physical robots in an area of approximately 315 square meters. This scenario involved red and green pucks, randomly dispersed in the area. Each robot was either a red-puck-collector or a green-puck-collector; it also has a memory of the number of observed red/green pucks/robots in the last x iterations. The following two transition functions define how a robot decides to change from a green-puck-collector to a red-puck-collector (and vice versa):

$$P(Green - Red) = \begin{cases} (GR - GP) * (1 - GP), & \text{if } GR \geq GP, \\ 0, & \text{otherwise} \end{cases}$$

$$P(Red - Green) = \begin{cases} (RR - RP) * (1 - RP), & \text{if } RR \geq RP, \\ 0, & \text{otherwise} \end{cases}$$

where GR denotes the number of green robots in the robot's memory, GP denotes the number of green pucks, RR denotes the number of red robots, and RP denotes the number of red pucks. (Lerman and Galstyan [216] have formally analysed these and other transition functions.) Jones and Matarić systematically changed the number red/green pucks in the area and show that the transition functions enable the robots to effectively change type if the circumstances require so.

5.2 Computer Networks

Present-day computer networks reach such sizes that centralised maintenance becomes increasingly difficult, if not impossible. Over the last decade, we have seen the upcoming of distributed and parallel methods and techniques that enable us to deal with this rising complexity. While the maintenance is a worrying factor, on a happier note, we can also effectively utilise the networks for distributing complex and hard computation tasks. We here present some technologies that address these issues: peer-to-peer protocols, self-star properties, grid computing and autonomic computing.

5.2.1 Peer-to-Peer Protocols

The rapid growth of the Internet, and the associated increase in public interest in sharing and accessing data or computing power, gave rise to a new form of networks for sharing data: the peer-to-peer (P2P) networks [235]. The difference between traditional forms of accessing and sharing data such as FTP, and P2P is that P2P networks are of decentralised nature. This means that the bandwidth and the computing power within such a network relies on the individual resources of the nodes within such a network, and not on a central server. This enables two or more nodes within a P2P-network to collaborate spontaneously by using appropriate information and communication systems without the necessity for central coordination [304]. In doing so, such systems exhibit good scalability in contrast to traditional systems.

This makes the nature of P2P networks different from traditional client-server models, since in such models communication is centralised (i.e., messages pass to and from a central server). An example of a non peer-to-peer system is an FTP (file transfer protocol) service. In such a service the client and server programs differ: the servers are the service providers, meaning that they react to and satisfy requests from Clients. Clients are used to initiate the download/uploads and communication with the servers.

P2P networks and their decentralised adhoc connections are useful for many purposes. One of their most prominent applications is to share content files (e.g. audio and video data). Besides the advantage of bypassing legal issues due to their decentralised nature (i.e., it is not possible to prosecute a central service within such a network, because the content shared resides exclusively on the peers themselves), such systems also tend to increase their performance when the number of the nodes within a network grows, since this leads to an increased availability of resources and bandwidth. In contrast, the opposite is the case in traditional non P2P services (e.g. FTP). In such systems, the service gets slower the more users are connected to it, because the server has to carry the bandwidth/resource demand alone. P2P systems are also used in systems dealing with realtime data, such as such as internet telephony (e.g. Skype) [327].

Newscast protocol	
1:	do forever {
2:	e = waitForEvent;
3:	if e is TIMEOUT {
4:	n_j = randomPeer()
5:	send x_i to n_j
6:	receive x_j from n_j
7:	x_i = aggregate(x_i, x_j)
8:	}
9:	if e is message x_j from n_j {
10:	send x_i to n_j
11:	x_i = aggregate(x_i, x_j)
12:	}
13:	}

Table 5.1: The Newscast protocol (from [178]).

Recently, an alliance has been suggested between P2P systems and other technologies which rely on decentralised systems, such as multi-agent systems. Since MAS are by definition operating to some degree as decentralised systems, their application within peer-to-peer systems is almost a natural fit. On the one hand, MAS could use the P2P infrastructure, and on the other hand P2P systems could be enriched with agency properties (e.g. autonomy, social ability, reactivity, pro-activity) and the ontologies already used in such systems [33, 201].

One problem that arises in P2P networks is the analysis of the information dissemination and aggregation within such networks. This is due to the fact that the total amount of data within such networks is enormous. Data is introduced to the network by the peers themselves, so the more nodes are connected to the network the more data is available. The content of the data in such networks is likely to overlap, i.e., many peers often provide the network with repeated versions of the same content (e.g., a popular piece of music may exist many times within the network). This is the main reason that such systems exhibit the need for decentralised algorithms which enable single peers to find demanded data within such a network. Another problem which arises is due to the adhoc nature of P2P networks. Peers frequently connect and disconnect from P2P services. When a peer joins, he introduces new resources to the system; while in contrast, when a peer leaves the system resources vanish. The available resources within such networks are therefore not stable but flexible and likely to change (i.e., a resource which was available five minutes ago might not be available any more).

The research area concerned with this topics is quite active, and several important algorithms have been introduced, such as an epidemic-style content based searching algorithm for P2P systems [370, 371, 371] (For a good overview of current ongoing research, see the proceedings of the Euro-Par conference [61, 248]). In this Section we will briefly introduce the ideas behind Newscast [179, 178, 180, 202], a P2P protocol for a highly distributed and non-deterministic form of collaborative information processing.

Newscast The aim of the Newscast protocol is to enable large scale computer networks to aggregate and distribute information among its peers in a decentralised manner. In monitor-

ing such a network, aggregation is sometimes needed to find out the average value of some parameter among the peers. Dissemination on the other hand is also crucial in order to spread information in a reliable way, such as for example alarm signals. Jelasity *et al.* [179, 178, 180] showed that a large network where each node n_i maintained a single number x_i was able to compute some aggregate of these values in a fully decentralised way. In order to do so, each peer incorporated the same Newcast protocol – see Table 5.1. The protocol works as follows: every $\triangle T$ time unit, a timer generates a TIMEOUT event. Whenever this happens, a peer selects a random peer j within the network and sends its own value x_i to this peer. The peer j which receives the value x_i from the peer i will send its own value x_j in return. Both peers use the received value to recompute their own value using an aggregation function. Jelasity et al. [179, 178, 180] showed that using this protocol the numbers of each peer converged exponentially to the overall average of all numbers within the network, as calculated using an averaging aggregation function $a(x_i, x_j) = (x_i + x_j)/2$. They also tested a maximizing aggregation function $a(x_i, x_j) = max\{x_i, x_j\}$ which converged super-exponentially fast. They therefore showed that it is possible to perform efficient non-deterministic information aggregation and distribution among peers in a decentralised manner.

5.2.2 Self-star Properties

Babaoglu [21] has coined the term *self-star* that covers a set of desirable properties of complex information systems allowing one to deal with scale, decentralization, heterogeneity, mobility, dynamism, and so forth[5]. The self-star research stream is embedded in the more general areas of self organisation in dynamic networks and large-scape information systems[6].

Topics within this very young research stream are, among others, scalability through self organisation, bio-inspired algorithms for peer-to-peer networks, evolutionary computing and autonomic computing, self-start topology control and self adaptive software.

5.2.3 Grid Computing

The aim behind grid computing is similar to the basic idea involved in P2P networks. While P2P networks mainly focus on the efficient and decentralised distribution of data, grid computing applies the idea of decentralization to actual computing power. Even though the increase in computing power of processors currently exhibits exponential growth - meaning that the possible computing power of new processors approximately doubles every 24 months (for more details see Moore's Law[7]) - there are several scientific problems which are very computationally expensive and can not be tackled by single machines, since this would take forever. As an example, consider the problem of protein-folding[8] or scanning received signals of radio telescopes to identify signals from extraterrestrial life[9]. In order to tackle such computationally expensive problems, computer grids are used. The idea behind computer grids is to split up the problem-space of a large-scale computation problem and distribute process execution across a parallel infrastructure to many computers within a network, thus using the resources of many separate computers. Depending on the number of computers within

[5]Notice that one can write *self-**, where the asterisk is a wildcard, to be substituted by -configuring, -organising, -managing, and so on.

[6]See http://www.cs.unibo.it/bison/ and http://delis.upb.de/.

[7]http://en.wikipedia.org/wiki/Moore's_law

[8]http://folding.stanford.edu/

[9]http://setiathome.ssl.berkeley.edu/

the grid and their capacity, up to tera-flop levels of computational power can be achieved by using such a parallel approach [120]. This enables the whole network to perform more computation at once than a single processor could.

The definition for what is and what is not a grid is quite fuzzy, leading Foster [121] to propose a checklist to define the criteria of a grid:

- A grid coordinates resources that are not subject to centralised control. This means that a grid should be able to incorporate and control resources from different control domains(e.g. operating systems, administrative units). Therefore, a grid needs to be able to deal with possible issues such as security or policy which can arise. Cluster management systems can not be considered as grids, because they normally exhibit complete knowledge and control of the system.

- A grid should use standard, open, general-purpose protocols and interfaces. In order to avoid classification as an application-specific system, the protocols and interfaces used within a grid should be standard and open.

- A grid delivers nontrivial qualities of service. The utility of a grid system should always be significantly greater than that of the sum of its parts, otherwise a grid would not be necessary.

In summary, a grid is a decentralised group of computational processing devices dependent on possibly different resources. This decentralised aspect makes grid computing an interesting application for self organisation.

5.2.4 Autonomic Computing

IBM® Research introduced in 2001 a new model of computing called *autonomic computing* that is flexible, accessible, and transparant with respect to the design and implementation of computer systems, software and support[10]. The inspiration for these new model is the autonomic function of the human central nervous system. This inspiration must lead to "a network of organised, *smart* computing components that give us what we need, when we need it, without a conscious mental or even physical effort".

Kephart and Chess [191] lay out this vision of autonomic computing. This vision is about how to deal with the looming software complexity crisis resulting from huge software source codes and the impossibility to manage large heterogeneous computer systems. The proposed model allows network administrators to express their high-level objectives based on which the system will autonomously manage itself. Within artificial intelligence, this relates much reasoning with undertainty in combination with a technique called *utility elicitation* [41, 274]. This technique is based on decision theory and gives one ways to obtain preferences from people. The research on *autonomic networks* is closely relating to autonomic computing model [199, 128]. Web services (application-to-application communication on the internet) can also make use of autonomic techniques [263]. Finally, multi-agent system theory can be used as an approach to reach the goals of autonomic computing.

[10] http://www.research.ibm.com/autonomic/

5.3 Insect-based Computing

Looking at a bee or an ant hive will always give one the impression that each individual is busy and working. They do not seem to need a break, but are always in motion, constantly working on tasks. What is more, their actions don't seem random, but highly organised; they seem to be coordinated and working together. Looking closely at a social insect colony, no central organisational structure can be found. There are no coordinators to tell other individuals what to do, or when to switch to other tasks. Nevertheless, such systems (as one might call them) can exhibit organisational features like temperature regulation, path optimisation, time control or migration, for which humans need managers, schedules or heuristics.

Social insects are also very appealing to computer scientists, since the are robust, minimal and very efficient. This realisation has led to a number of applications and algorithms which use the underlying principles of phenomena found in social insects (e.g. broad sorting) and apply them to computational problems [38, 37, 90, 283]. An event which promoted and boosted this field of research was the invention of the ant colony optimisation algorithm (ACO) by Marco Dorigo [89, 92]. His optimisation method (explained in more detail in the next Section) uses an ant-inspired heuristic to solve the travelling salesperson problem (TSP) and is successfully used in package routing.

Many applications have been proposed which make use of insect-inspired algorithms in various domains such as robotics [58, 91] and optimisation [89]. The following sections will look at three such applications (package routing, paintbooth scheduling and data clustering) which have been successfully solved using algorithms inspired by insect behavior.

5.3.1 Package Routing

The rapid growth of both information and information networks such as the internet - which can be seen as the nerve-center of communication infrastructure [244] - has fostered research into telecommunication networks, both in academia and industry [378]. A core problem within this domain is package routing. It constitutes the problem of routing information (e.g. a telephone call or data-packages) through a given telecommunication network. There are two major network forms to which package routing is applied.

Circuit-switched networks are the network type normally used in telecommunications (i.e., telephone networks). Two parties within the network are connected by means of cross-bar switches and therefore reserve a channel between these two parties. This is not the case for packet-switched networks, which are used for example for the Internet. In such networks no fixed connection is established between two parties via cross-bar switches, which results in a very flexible and scalable network architecture [190].

The process of routing and the requirements involved therefore depend on the network type. Within circuit-switched networks the routing process should lead to an optimal distribution of calls over the available switches, leading to a maximum number of possible calls. Areas within the network which are blocked due to maximal usage should be avoided in order to to minimise delays, but on the other hand the routing costs should be kept to a minimum (i.e., detours should be avoided). Sometimes, for example after thunderstorms, main connections are destroyed and a bottleneck is created within the communication system. In such cases a routing mechanism has to dynamically adjust to the new situation, requiring on-the-fly rerouting of packages through other parts of the system. The requirement of routing mechanisms in the case of packet-switched networks is to maximise the performance of

the whole network (i.e., to balance the usage of the network in order to avoid bottlenecks). Of course, the algorithm should also be able to dynamically adjust to breakdowns of hosts within the network. In the following subsection, an algorithm for insect-inspired package routing in packet-switched networks [89, 51] will be discussed. Such algorithms also exist for circuit-switched networks (see Subramanian *et al.* [329]).

Within the insect world, the behavior observed in ant colonies is an example of a solution to a similar problem. Ants have the remarkable ability to find the shortest path between their nest and a given food source. This is done by means of pheromones: while ants wander around, they deposit pheromones which are used by their colony members as guideposts. They therefore mark the way to the food-source. Pheromones are subject to a constant evaporation process, meaning that the amount of pheromone on a suboptimal pass to the food source will evaporate faster than the pheromone on the optimal food-source, since a single ant can use the shorter optimal path more often in the same amount of time. This will trigger other ants to use the same path (and those ants will also deposit pheromones), boosting the reinforcement of the path even more. In the end, all suboptimal paths to the food-source will have lost their dominance and all the workers will use the optimal path to the food source [38].

This idea inspired Dorigo and led to the invention of the ant colony optimisation algorithm (ACO) [89, 92]. The ACO is used for graph problems (such as the travelling salesperson problem) by incorporating the idea of pheromones outlined above to solve them. Virtual ants wander over the graph and decide which edge they will choose next on the basis of the length of the paths available and the amount of pheromone which they contain. The amount of pheromones on the edges is initialised equally and evaporates constantly. New pheromone is added to an edge if the edge was part of an solution. The amount of pheromone which is deposited depends on the goodness of the solution. Ant colony optimisation algorithms have been used to produce near-optimal solutions to the travelling salesman problem. One of their big advantages is that they can be run on dynamically changing graphs, which makes them suitable for network routing.

Method

In this section, we discuss AntNet, an ACO algorithm for asymmetric packet-switched networks [51, 89]. As outlined above, the objective of such routing algorithms is the maximisation of the performance of a complete network. Two types of virtual ants are used the forward_ant and the backward_ant. Ants have a maximum age which dictates how long they are allowed to remain within the network. When ants travel from one network node to another, they store the corresponding node addresses and trip times by pushing them on a stack.

Each node launches forward_ants to an arbitrary known destination at regular intervals. The probability of a node sending out a forward_ant to a destination within the network depends on the the amount of traffic generated for that node destination (i.e., nodes will launch forward_ants towards destinations which are important for them). On their way to their destination, forward_ants collect the addresses of the nodes passed and the time taken to reach them on its memory stack. The forward ant is used to experience the network conditions. When it reaches its destination, a backward_ant is created which inherits the memory of the forward_ant. The backward_ant backtracks the locations which where visited by the forward_ant (which is why a stack is used for the memory) and updates the routing information on the routing tables according to the trip time from the node to the destination.

If the path had a good trip time it will be reinforced more than a path with long trip times. The changes in the routing tables influence the future behavior of packages and forward_ants, but not the behavior of backward ants. For a detailed mathematical description of AntNet please see [51, 89, 378] .

Experiments

AntNet successfully applies the concept of stochastic spreading of data packets along all paths according to the goodness of the paths. There have been a number of experiments performed with different network topologies using 8, 13 and 57 nodes under various conditions (i.e., packet delays within the network). Besides being cost effective (less than 1% of the bandwidth is occupied by ant agents), the experiments have shown that AntNet outperforms all other algorithms normally used for package routing, such as Q-routing, PQ-routing, Shortest Path First and OSPF.

5.3.2 Paintbooth Scheduling

The famous quote of Henry Ford[11] that "Any customer can have a car painted any colour that he wants so long as it is black" does not apply in present-day production. Oversupply and market pressure oblige producers to produce cars that are as customisable as possible. The paint booth scheduling problem is a manufacturing process problem [243] involving the custom coloring of cars. The process of painting cars usually takes place directly after assembly, and the color of the car is selected by the purchaser in advance. The aim of paint booth scheduling is to schedule the painting of the cars in such a way that no production bottleneck occurs.

Normally it takes around 3 minutes to paint a car, but this is not the case however if there is a change in color (i.e., if the next car to be painted is not the same color as the last car painted). In such a case the booth has to be cleaned and re-stocked with the new color. An additional problem with changing colors is that the cost of paint changeover is also high. This results in the constraint that booth scheduling should minimise the number of such changeovers. Notice that this has to happen online, since new incoming orders can of course affect such a schedule.

Paint booth scheduling is just one of many manufacturing processes which requires careful planning. Generally stated, manufacturing processing planning involves the selection and sequencing of manufacturing processes in such a way that they satisfy a given set of constraints and achieve a high production rate. In recent years, the interest in agent-based approaches to such problems has increased [309]. Insect-based approaches have also been proposed [50, 55]. In this section we will focus on the paint booth scheduling approach suggested by Campos et al. [50].

The insect-inspired approach to paint booth scheduling is based on the response threshold hypothesis, which was discussed in the division of labor section in the last chapter. As already outlined, the response threshold concept relies on the diversity of the individuals' response thresholds for task-specific stimuli, within a population. Genotypic variation in a population can be a cause of different stimulus thresholds. Honeybee queens, for example, mate with 7-17 drones and this causes a genotypic variation within a honeybee population. Bees with different genotypes vary in their sucrose responsiveness, which influences their preferred foraging-task choice [299]. Nevertheless, bees maintain a task flexibility and are still able to switch tasks

[11] http://en.wikipedia.org/wiki/Henry_Ford

when faced with dynamical changes within the environment, in order to satisfy the colonys need.

Method

The algorithm by Campos *et al.* [50] assigns each booth an initial threshold for each color. Thresholds are bounded in order to enable threshold changes. If this were not the case, thresholds could increase and decrease indefinitely, which would lead to an inflexible behavior of the booth. In each round (i.e., when a new car should be colored) the global demand of each color is computed which constitutes the amount of cars which should be colored in this color. These demands are used to compute the probability of a booth to respond to an current coloring request.

Notice that this is a reformulation of the response threshold concept. If the demand for the current color exceeds the threshold of a booth for this color, the booth is very likely to respond to the request by coloring the car in the desired color. If this is not the case, the car will be colored in the same color as the car before it. This of course has an impact on the demand in further rounds. Colors which have been neglected for several rounds will increase their demand, which will lead to the booth responding to their request when the demand is sufficiently high.

When a booth has colored a car in a given color, its threshold for the color decreases (i.e., it specialises in the color) and its threshold for all other colors will increase(i.e., it de-specialises with respect to those other colors). For a detailed mathematical description of the algorithm, please see [50] .

Results

The performance of the algorithm is very depend ant on the initial values used for variables such as the initial threshold and the threshold boundaries. To avoid this problem, Campos et al. [50] used genetic algorithms to discover the best starting conditions. The experiments conducted contained 20 possible colors, 6 to 15 booths, and a maximum queue for each booth set to 5 cars. Possible booth defects were also modelled, meaning that a booth could become inoperable for some time during the simulation. The algorithm was able to outperform scheduling approaches based on fixed statistics which are used by car factories.

5.3.3 Data Clustering

Data clustering can be seen as one of the most important problems in the domain of unsupervised learning [176]. Its aim is to cluster data according to its similarity. This means that successful clustering of a set of unlabeled data should lead to a partition of the data into subsets, with the data within subsets being similar to each other and dissimilar to the data in other subsets. Clustering therefore imposes structure onto a collection of unlabeled data.

There are two ways in which clustering can be performed: hierarchical and partitional. Hierarchical clustering consecutively clusters the data within a set until one big cluster is formed which contains all the sub-clusters. Hierarchical clustering can be done in a bottom-up manner starting with each element as an independent cluster and merging them into larger clusters, or in a top-down manner, starting with the data set as a whole and dividing it into smaller subsets. Partitional clustering works differently, by clustering a given data set into

a given number of clusters. A very well-known example of partitional clustering is k-means clustering, which clusters a given data set into k sub-clusters.

An important point that must be mentioned is that the success of the clustering process, and therefore the result of data clustering, depends crucially on the metric used to quantify the similarity of data points within a data set. For this reason, a wide range of metrics exist, and a suitable metric should be chosen according to the problem to be solved.

Data clustering is used in many fields such as machine learning [238], bioinformatics [69] and neuroscience [330].

Clustering is of course also found in the natural world. Insects use a form of clustering in order to sort their brood [123, 307, 227] or the bodies of dead members of the colony [74]. In the ant species *Leptothorax uni-fasciatus* [123, 307], the brood is sorted according to its age and arranged concentrically around the nest center. Young brood items (eggs and microlarvae) can be found close to the nest center, while older brood (pupae and prepupae) are placed in intermediate areas. It has been suggested that this brood-sorting depends on the CO_2 level within the hive. The amount of CO2 is at its maximum in the center of the colony and concentrically decreases. Ants are able to sense the CO_2 concentration via their antennas, and can thus place the brood accordingly.

Ants also tend to create cemeteries, upon which the dead bodies of former colony members are disposed. Experiments have been conducted [74] where corpses were randomly distributed within a nest-site. The workers within a colony started to cluster these corpses into cemeteries. It is thought that once a worker picks up a dead body, it will carry it until it reaches an area where other dead bodies are lying. There it will drop the dead body. This initially leads to small clusters, and then to big clusters through positive feedback [38]. This concept is also used for insect-inspired data clustering, which will be explained in more detail in the next section.

Method

Denebourg *et al.* [74] used the observations described above to create an algorithm which can be used to sort or cluster objects. The algorithm is identical - in the clustering version objects are treated as being of a single kind (for example corpses) and are simply grouped into clusters, while in the sorting version different kinds of objects (for example brood members at different developmental stages) are sorted into homogeneous clusters. The general idea behind this algorithm is that isolated items are likely to be picked up by idle agents and dropped at locations where more items of that type are present. An idle agent picks up an item with the probability $p_p = (k_1/(k_1 + f_x))^2$ where f_x is the fraction of the objects x which the agent observed in the last T time-units and k_1 is a threshold constant. If $f_x << k_1$, p_p is close to 1, which increases the probability of an agent to pick up an item. In contrast, if $f_x >> k_1$, p_p will be close to 0 and therefore the likelihood of an agent to pick up an object will be low [38]. An agent which carries an object of type x moves around randomly and will deposit it with the probability $p_d = (f_x/(k_2+f_x))^2$, with f_x being again the fraction of objects of type x in an agent's neighborhood and k_2 being a threshold constant. Since f_x is not a constant, the agent will be more likely to drop an item if f_x is high, since a high f_x decreases the impact of the k_2 in the denominator. This was implemented in robots that could then successfully group objects in their environment. It should be noted that the objects to be clustered were of a very low dimensionality (differing for example only by their color, i.e., on a simple feature dimension), so a similarity metric was not crucial to their implementation.

Lumer *et al.* [227] extended this idea so that it could be applied to cluster data. As the dimensionality of real-world data is often very high (think for example of a signal measured over time), a suitable metric must be used in order to infer the similarity of the objects to be clustered. This means that determining f_x, the fraction of objects of a certain "type", is not as straightforward as in the previous example. Instead of f_x, a function $f(o_i)$ is used, which computes the average similarity of object o_i to the other objects within the agent's neighborhood (for the precise formula see Lumer *et al.* [227] or Bonabeau et al. [38]). The probability of picking up an object o_i is therefore $p_p(o_i) = (k_1/(k_1+f(o_i)))^2$ where k_1 functions again as a threshold constant. After picking up an object, agents will move with it until they find a site to deposit it. The probability of dropping an object is $p_d(o_i) = 2 * f(o_i)$ if $f(o_i)$ is smaller than the threshold constant k_2. $p_d(o_i) = 1$ if $f(o_i)$ is greater or equal than the threshold constant for dropping times a constant, $k_2 * s$. This means that an object will be dropped for sure when a certain object-similarity is measured in the neighborhood. Initially agents and objects are placed randomly on a grid. In order to speed up the simulation, different agent-speeds were introduced. The idea behind this was that fast agents could do the raw sorting by being able to move fast through the environment, while slow agents would then add the finishing touches.

Results

The algorithm for data clustering introduced by Lumer et al. [227] was able to successfully cluster high dimensional data to a certain extent. However, the approach is slow and very inefficient in terms of computational time in comparison to existing clustering methods.

5.4 Agent-based Computing

Consider yourself in need for some product. The easiest way to obtain this product is going to the shop and buy it. But imagine that the shop does not have your product and you have to get the product at a market close by. In this market, the prices are not fixed. Thus you haggle (or: *negotiate*) with the market salesman about the price. Still, you agree on the price and you have your product. But imagine that you cannot get the product on the market either. Then you would have to go online and you might find the product via ebay. On ebay, you can buy articles that are sold by *auction*. You put in some bids, win the auction and, again, you have your product. Finally, suppose that you are not so much in need for some product, but more for some service, for example, someone that helps you moving house. You cannot or do not want to buy this service. Thus, if you want to obtain such a service, you need to rely on *trust* and *reputations* of the persons offering this service.

The mechanisms and concepts mentioned above (negotiations, auctions, trust, reputations) are much researched and used in the field of multi-agent systems. These particular mechanisms are useful in situations where you have to reach (global) mutual agreement, and you can only define local interactions. In other words, they can be used in situations where you want to achieve collective intelligence in some way, but you cannot centrally control this. Sometimes there is a commonly agreed *currency* for these interactions (e.g., money), but in other situations this might also not be possible and one has to fall back on trust and reputation. We explain some of these mechanism in somewhat more detail.

5.4.1 Automated Negotiation

Automated negotiation involves the collection of methods and techniques to reach mutual agreement in multi-agent environments [204]. It includes methods like strategic negotiation, auctions, coalition formation, market-oriented programming and contracting. The key element is the *negotiation protocol*: some common agreed upon protocol through which agents negotiate with each other. How well a particular protocol performs depends on negotiation time, efficiency, simplicity, stability and money transfer. Kraus describes a number of negotiation models with game-theoretic background, an overview of auctions, contracting, and so forth. Still, no general guidelines can be given for which model is best in which situations: the choice of the specific technique for a given domain depends on the specification of the domain.

There exist a number of software frameworks that support automated negotiation. One such a framework is described by Bartolini *et al.* [24]. Concerning the negotiation protocol mentioned before, in practice it is important that this protocol agrees with standardised agent communication protocols (e.g., FIPA[12]). Bartolini *et al.* describe a negotiation software framework that both includes important aspects of negotiation and works on standardised communication protocols. Another negotiation software framework is the Multi AGent Negotiation Testbed (MAGNET) architecture[13] that was developed by Gini.

A particular branch of automated negotiation is concerned with participants that partially explain their motivations during the negotiation and can thus exchange *arguments* that influence each others' states [280]. *Argumentation based negotiation* is an extension of the traditional agent-based negotiation framework with which mutual agreements can be reached in situations where this is not possible in the traditional framework.

5.4.2 Trust and Reputation

A multi-agent system is a large-scale open distributed system. An open system is one where any agent can join or leave as it wishes. For such open systems, it is difficult to impose centralised or external requirements on the new participants. A number of coordination mechanisms has been succesfully used in these systems to let these open systems operative effectively. Tow main mechanisms are *trust* and *reputation*.

Ramchurn *et al.* [282] present an overview of the use of trust models in multi-agent systems. For these models, they make a distinction between individual-level (where each agent decides its trust in others itself) and system-level trust (where trust is regulated on the system level).

In follow-up work, Huyhn [170] presents the FIRE trust and reputation model that enables agents to evaluate the trustworthiness of their peers. The model integrates four important aspects of trust: interaction trust, role-based trust, witness reputation and certified reputation.

5.4.3 Computational Mechanism Design

The methods (negotiation, trust, repuation) presented above all get their inspiration from the real world. In everyday interactions with other people, we also rely on these concepts. But

[12] http://www.fipa.org/
[13] http://www.cs.umn.edu/magnet/

what if your agents are in a situation where such parallel with the real world is not available. But as a designer, you still want to enforce some general protocol or behaviour that leads to some system goal or objective. In other words, you actually have to define the rules of the game such that the result is some desired outcome.

Computational (or:automated) *mechanism design* is "the art of designing the rules of the game (aka. mechanism) so that a desirable outcome (according to a given objective) is reached despite the fact that each agent acts in his own selfinterest" [297]. In multi-agent systems, agents may pursue different goals, but must often still get to some common agreement in order to "live together in harmony". The automated mechanism design framework that was developed by Sandholm (in earlier work), takes sets of outcomes and agents (each with a given type and a utility function) as input and outputs a suitable mechanism. (Note that this an inverse version of a typical machine learning problem.) The author illustrates the framework with a hypothetical divorce settlement scenario. In similar work, Dash *et al.* [65] present an approach towards mechanism design based on auction theory.

5.5 Games and Movies

The area that is already making money from collective intelligence is the entertainment industry. Both in the game and movie industry, people successfully apply ideas on collective intelligence to generate intelligent non-player characters (NPCs) in computer games and to animate human, animal or cyborg crowds in games and movies.

This Section briefly discuss the RoboCup competition, the SimCity and SPORE games, the evolving creatures that are famous from the artificial life field, and the application of the mechanism used there for crowd simulation.

5.5.1 RoboCup

The aim of RoboCup[14] is to develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team by the year 2050. Since the mid-1990s [193] there are annual local and global events concerning this competition. The annual RoboCup competition is the most well-known and draws much attention from the media. The last years, this event involves a few hundred teams playing against each other. There are different leagues: for simulation, small size robots, middle-sized robots, 4-legged (e.g., Sony AIBO) and humanoid robots. Recently, the RoboCup event also includes a RoboCupRescue track where the aim of the involved robots is to perform search and rescue tasks in disaster situations. The RoboCup competition is always organised joinlty with a conference about related topics, e.g., robotics or multi-agent systems.

The scientific value of RoboCup is the delivery of new methods and techniques that enable agent systems to deal with real world complexities. It is expected to contribute to the research fields of artificial intelligence and robotics in the areas of: real-time sensor fusion, reactive behavior, strategy acquisition, learning, real-time planning, multi-agent systems, context recognition, vision, strategic decision-making, motor control, intelligent robot control, and many more.

[14] http://www.robocup.org/

5.5.2 SimCity

The SimCity video game[15] is an interactive computer simulation game in which the player is concerned with and actively participates in *community planning* in a virtual world. The Sims game[16] from the same company, focuses on the inhabitants of this virtual world and lets you control the behaviours of these individuals. Both these games give you a platform to explore collective intelligence in these contexts. Although the games are primarily meant for entertainment, the idea of a virtual world in which you can design and control the individuals also attracts research interest. For example, the NEW-TIES research project[17] takes such a virtual world in order "to investigate an evolving artificial society capable of exploring and understanding its environment through cooperation and interaction". The scientific contributions of this project will be new methods and techniques for developing large computer simulations on distributed computer networks; novel optimisation techniques combining individual, social and evolutionary learning; and new explanations for societal phenomena, for example, the Kula ring[18].

5.5.3 SPORE

The SPORE video game[19] takes The Sims even further back to the origin of life: it allows you to simulate "the complete history and future of life". The player can design and guide species over many generations that start from single-celled organisms and eventually becomes an intelligent species. The species develop in multiple phases: tide pool, creature, tribal, city, civilisation and space.

The technology of SPORE, based on *procedural content generation*[20], is largely unknown. The player is able to combine different components (legs, arms, and so) into a creature and this creature will behave according the combination of these components. For example, the creature's legs prescribe how it moves around on a particular surface.

Researchers have explored similar undertakings regarding simulations of the development of species. For example, Al *et al.* [4] created the AEGIS artificial world where the individuals have particular genes encoding the preferences for foraging, fighting and mating. In AEGIS, evolutionary learning enabled the population to deal with hostile environments. Also, Terpstra and van Zon [336] conducted a simulation study with virtual creatures into the cropping theory. This theory predicts an increase of the diversification of species in an environment with the introduction of effective predators. It is an explanation why there was a sudden diversification of species around the time of the Cambrian explosion (about 542 million years ago).

5.5.4 Evolving Creatures

In 1994, Sims introduced an at this time unique simulation that allowed the virtual evolution of block-creatures in three-dimensional worlds [310, 311]. The motivation behind this simulation

[15] http://simcity.ea.com/
[16] http://thesims.ea.com
[17] http://www.new-ties.org/
[18] http://en.wikipedia.org/wiki/Kula_ring
[19] http://www.spore.com/
[20] Procedural content generation is a technique that can create content on the fly and is often used in computer graphics applications.

stemmed from the complexity vs. control problem within the field of computer graphics and animation, meaning that it is hard to incorporate complex behavior within complex entities and still being able to control them.

In order to solve this problem Sims created a simulation that simulated behavior (such as walking or swimming) of virtual creatures in a simulated 3D physical world. The creatures are constantly exposed to evolution in order to trigger the convergence of a creatures behavior in the desired direction (over multiple generations of course). The morphology of each creature is based on its genotype which is a is a directed graph of nodes and connections so to say its "building plan". Creatures consist of cuboid 'limbs' which are interconnected with virtual muscles. They also exhibit virtual sensors and and incorporate an artificial neural network in order to process the input from those sensors and act on virtual muscles between their 'limbs'.

In his experiments Sims used a population size of 300 and a survival rate of 1/5. Simulations were initialised with an initial population of randomly created genotypes. Each individual was evaluated according to its abilities to fulfill a certain task (e.g. time needed to get from point A to point B). The best 1/5 of the generation was kept in the population for the next generation. Additionally surviving individuals were used to create offspring in order to refill the population for the next genration. Offspring was generated by copying and combining the genotypes of surviving creatures. The genotype of offspring was also exposed to evolution (i.e., probabilistic variation within the genotype).

In order to estimate the fitness of a creature Sims used a dynamics simulation to calculate the movement of creatures resulting from their interaction with a virtual three-dimensional world (i.e., articulated body dynamics, numerical integration, collision detection, collision response, friction, and optional viscous fluid effects).

Even though Sims was not able to evolve walking movement within his creatures several other forms of land and water based movements could be evolved (e.g. sea snake/fish like swimming, jumping and tumbling[21]). There are several other simulation programs which incorporate the same principles as the such as FramSticks[22] or Fluidiom[23]. Also, present research investigates these kinds of virtual creatures; for example, for the design of lifelike electromechanic robots [276] and to investigate salamander locomotion [172].

5.5.5 Crowd Simulation

Simulation of collective intelligence in the movies means *computer crowd simulation*: the computer rendering of large crowds based on *particle motion* or *crowd artificial intelligence*. Example animations can be seen in, for example, Batman Returns, the Lord of the Rings trilogy, and Spiderman. Video games also use these techniques for visual effects, e.g., on the Sony PLAYSTATION® 3. While crowd simulation as a term is also used within the academic community [247, 42], we here concentrate on the industrial and commercial interest within the movies and game industry.

The particle motion approach models the individuals as point articles, that are animated by simulation of wind, gravity, and so. These models resemble other models as presented in this book, for example, the physical models that were used for pedestrian and traffic modelling; as well as the particle swarm optimisation model. The crowd artificial intelligence approach

[21] Videos of the creatures evolved by Karl Sims are available at http://www.youtube.com/
[22] http://www.frams.alife.pl/
[23] http://fluidiom.sourceforge.net/

attributes *intelligent* behaviours to the individuals, for example, sight, hearing, emotion, and so forth. Individuals also have goals that they try to accomplish and can autonomously interact with other individuals. The particle approach is computationally cheap, but does not give such realistic results; the crowd AI approach is computationally expensive but gives more realistic results.

Noteworthy, of the researchers discussed in this book, both Reynolds (from the BOIDS simulation discussed above) and Sims (from the *virtual creatures* discussed above) are currently working in the entertainment industry. Reynolds[24] has, for exmaple, developed crowd simulation techniques for the Sony PLAYSTATION® 3 [288]. On his website, Reynolds maintains extensive annotated lists of links about, among others, steering behaviours, individual-based models and game research and technology. Sims heads the Genarts company[25] that develops special effects plugins that are used by movie studios.

5.6 Summary

This Chapter overview a significant number of research studies into the design of collective intelligence. Some studies are described in detail and explicitly explain the problem, solution method, and experiments. Others are described more generally. We show that the design of collective intelligence attracts attention in a wide area of applications, ranging from robots that can (collectively) avoid holes in the ground to large computer networks that can maintain themselves. We have not tried to completely cover all possible applications of collective intelligence design – there is just simply too much out there. We specifically explained the studies in such a way that in-between comparisons of *how* collective intelligence is applied, are possible, although the ultimate applications themselves may be very different. Also, in the context of this book, the reader must develop an ability to judge the decisions made in the various studies: why did the researcher choose a physics-based model and not an agent-based model? Also with respect to the used algorithms, the reader must be able to answer these kinds of questions after reading this book.

[24] http://www.red3d.com/cwr/
[25] http://www.genarts.com/

Part III

HOWTO

– Basic research is what I am doing when I don't know what I am doing.

Wernher von Braun

– Research is the process of going up alleys to see if they are blind.

Marston Bates

6 Research Methodology

If you have ever played around with an artificial life or agent simulation toolkit, you may recognise this feeling: it is easy to start out with and you can quickly have results that look good when you run your simulation. But you may wonder where the science is in here? How do you perform solid scientific studies using such tools? In this Chapter, we tackle a somewhat more general issue (the presented methodology is applicable to a much wider span of scientific research studies), but it very well applies to these particular studies. In our experience, this methodology has been an important factor for successfully completing a significant number of studies on collective intelligence.

Aims of this Chapter

This Chapter aims to systematically guide you through the process of conducting solid, valid and useful scientific research. We present a methodology that is based on the so-called $\cup$-model: going down on the left side by formulating research objectives, questions, hypotheses and identifying the relevant variables; on the bottom obtaining results by solid experimentation; and finally going up the right side by analysing results, verification or falsification of the hypotheses, and finally deciding whether the objectives were met. Everything is connected in this model and that is how it should be in practice too.

Tips and Tricks

We finish some Sections in this Chapter by some Tips and Tricks. These are practical pointers on how to conduct particular parts of your research. We first give you some general pointers about methodology.

- Always be aware of the fact that scientific research is a *creative process*. We can give you as much guidelines or rules of thumbs as we want, in the end the success of your investigation largely depends on your own creativity. Although this may sound a bit trivial, it gives you some bandwidth on how to put our given advice into practice. It

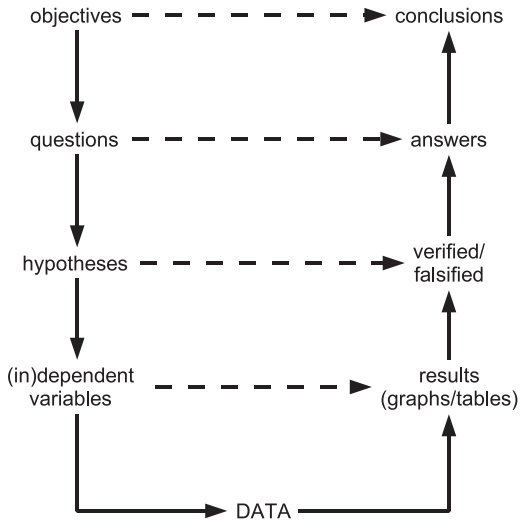


Figure 6.1: The ∪-model.

also means that while you are in the process of investigating something, your attention may be caught by a single detail that has nothing to do with the overall objective of your study. Or, in the words of Isaac Asimov: "*The most exciting phrase to hear in science, the one that heralds new discoveries, is not Eureka! (I found it!) but rather, 'hmm... that's funny...'*".

- Keep your level of ambition modest. Although you may be tempted to research the most amazing phenomena that you can think of or that you have seen, know that the research studies described in this book were projects taking many (wo)man-years. It is difficult to actually estimate the level of ambition before embarking on your research. And because of the fact that it is hard to determine beforehand, do not be afraid to make your ambitions more modest while you are actually already in the research process itself.

- In preparation of your study, you have to plan when you are going to do each part of the study. Traditionally, you also make an estimate how long each part takes (hours, days, weeks). As a general rule of thumb, double the total estimated time of all the parts together: this gives you a somewhat realistic idea of how long your project is going to take.

- The related research field of evolutionary computing is currently realising that scientifically valid experimentation is essential for generalisability of results, performance measures and reproducibility [102]. The articles and books on experiment research methodology for evolutionary computing, e.g. [25], are highly relevant to what we present here and thus recommended reading.

6.1 Research

Consider Figure 6.1 that shows the ∪-model of research. In this Section, we describe the first three steps of this model as shown on the left-hand side: formulation of objectives, questions and hypotheses. This can be considered as the preparation of your research.

6.1.1 Literature

The very first preparation of your research starts with a *literature study*. By doing this, you should be able the answer the question *what is out there*. In other words, how novel is your research, what is the difference to existing work and so forth. In case you are writing a report for a course or you are doing your Bachelor or Master final project, you may already have some literature provided to you by your supervisor. If you are getting further into research (MPhil, PhD programmes and beyond), you will go more and more into a specific research area and develop an idea of open research paths or novel research studies yourself.

Although we mention it here as the thing that you do as the very first in a research study, it is actually a continuous activity. You may best start off with reading 10-15 articles about the research area in order to get started. It may very well be that one of your objectives, research questions or hypotheses actually comes directly from the literature itself: in this study, you are going to do something that someone else identified as a missing piece of research. Along the way, while conducting your study, you will find more particular and more detailed aspects of your research that require you to look up more articles about these particulars and details. We also mention the literature study as a separate part of your research report in the next Chapter.

Where to find the related literature then? A literature *search* is a "systematic gathering of published information relating to a subject" [68]. Such a search is thus systematic, because although you may start off somewhat randomly, you gradually define the boundaries of the related work during your search *and know when to stop*. Also, the information must have been published. Within academic circles, this means that it can be found in journals, conference papers, scientific magazines, and similiar sources. Although you can find information in many more places (e.g., internet, popular science, weblogs), it is important that the information is academically *recognised*. This means that the information has been substantially reviewed before publication – assuring its scientific value[1].

You best start your literature search with reading one or more textbooks on some specific topic concerning your object of study. You may also start from a *review article* that overviews the current state-of-the-art of some specific research field or topic. From there, your search then basically follows (inductively) the bibliographies of the literature that you read. From the textbook or review article you may find 5-10 scientific articles that are related to your specific study. Each of these articles contains a bibliography, with which you can continue your search. There are many formats in which the information may come [68]: books, journals, conference proceedings, technical reports, theses, manuals or software. If you are in an academic environment (e.g., university or research institute) you will most probably have access to a library where you can find those information sources. Many publishers also give online access to their journals and proceedings. You often need a login and password to access

[1] On a side note, the system for scientific evaluation is so-called *peer review*: submitted papers and articles to conferences and journals are reviewed by your peers, i.e., persons that are equal to yourself in terms of scientific capabilities and level of knowledge.

such information, but often universities have such access. If you browse through the university's computers, then the website of the publisher will be able to automatically assign you the correct login and password, giving you online access to their journals and proceedings.

6.1.2 Objectives

You define your goals here: what do you want to achieve with doing this research? Do you want to show that your algorithm outperforms all other exisiting algorithms? Or do you want to understand why dominant primates are always sitting in the middle of the colony? You can express these objectives in natural language. That gives you a lot of creative freedom, but is good to consider some rules of thumb when formulating the objectives.

- You will be evaluated based on your objectives. In other words, you can define your own evaluation criteria here. For example, if you are doing a lab project, then you decide in the objectives when your project failed or succeeded. Or, if you are undertaking a big research project, then the project evaluators can judge the research quality based on these objectives. This is nice, but also has an impact on the formulation of the objectives. For example, the answer to the question if you have achieved the objectives should be simply yes or no; and, ideally, one should be able to actually *measure* if you have achieved your objective or not. If you begin your objective with "We aim to understand ..." then this cannot be measured - how do I, being the evaluator, know if you have understood ...?

- There are some options to choose from about what you are going to do in your research [240]: *characterisation* - if you are modelling a new phenomenon or programming a new algorithm; *optimisation* - to further model an already characterised system; *confirmation* - is the same system performing the same way now that it did in the past; *discovery* - what happens if we explore new variables; and *stability* - how robust are previously found results?

- Seize the formulation of the objectives as an opportunity to reach mutual agreement among the involved researchers about what the research is going to be about. Although this may seem trivial, with interdisciplinary research (biologists and computer scientists developing simulation models together) there are misunderstandings before you know it. Is it going to be an analysis or design study? The biologists may think that the team is embarking upon an analysis study, while the computer scientists have a design study in mind.

- In addition to models and results, every project is going to have physical deliverables, e.g., journal articles, conference papers, books, or products. It is good to make this explicit here: formulate an objective that you, for example, are delivering a book by successful completion of the project.

- You can have multiple objectives, but not too many. Somehow, three is a magical number here - to have three objectives seems in general a good thing. If you have too many objectives, you risk losing the cohesion in your research.

6.1.3 Questions

The second step in preparing your research is the formulation of the *research questions*. These question actually form the glue between the objectives (as explained above) and the research hypotheses (as explained below). As explained, the objectives are ideally only evaluated as success or failure. Research questions give you a way to formulate more *informative* inquiries into the content of your research. For example, assume that your objective is to demonstrate that algorithm A outperforms algorithm B on some test suite problems. You can either succeed or fail in achieving this objective - but the real information is in the details: how much faster is it, is there extra preparation necessary, and so on? Take the following advice into account while formulating the questions:

- At least one of your questions should address the positioning of your research. Thus why is it so important that you do this research right now? And is it really novel or did other researchers already do similar work? You should have some idea about this even before formulating your objectives, but there must be a point in the research that you are working this out. Eventually, the results of this will end up in a literature review of related work. If you have formulated a question about it, it gives you the possibility to spend research time on it.
- You have to be more precise and detailed here compared with formulating the objectives. However, you still have relatively much creative freedom. You will base your hypotheses on these questions; when you formulate your hypotheses, you will have to be very rigid (as explained below).
- Again, the magical number is three here where it concerns the number of questions. However, as a rule of thumb, work out each objective in three questions, thus in totally you would have nine research questions.
- Research questions in the exact sciences rarely address why-issues (as opposed to science of philosophy or social science). Usually the form of your question will be "How ..." or "What ...", indicating that you are interested in how something works or how things are related (not why).

6.1.4 Hypotheses

An hypothesis is *a specific statement of prediction*. A set of hypotheses actually forms your roadmap for analysis: it steers your research and it defines the borders of what you will and will not investigate. Although hypotheses are not carved in stone (and may need to be reconsidered when you are conducting your research), it is necessary to really put significant effort into writing down an explicit set of hypotheses. Again, you are adviced the following when it concerns formulating hypotheses:

- Hypotheses are in general statements about relationships between variables in your model. Typically, one of these variables is systematically varied (the indepedent variable) and the other one is measured (the dependent variable). The relationship can, for example, be correlational, causal, and so on.
- Scientifically speaking, there are actually two version of a single hypothesis: the *alternative* and the *null* hypothesis. The former is the one that actually states what you

expect will happen; the null hypothesis is the opposite of what you would expect. In your research, you will actually investigate the null hypothesis and not the alternative one (your prediction). Based on your results, you may then conclude to reject the null hypothesis, which leaves (by process of elimination) you to conclude that these results confirm the alternative one (your prediction).

- There are some resources available[2] containing tests for checking the formulation of your hypotheses (to be done before testing the hypotheses themselves). Because it is difficult to give general guidelines for hypothesis formulation, such tests form a good mechanism for checking your formulations.

- An hypothesis can always be evaluated true or false. This relates to the results that you gather while carrying out experiments. When your experiments are 1) based on your hypotheses, and 2) well designed (which is explained later), then you must be able to evaluate each hypothesis based on the obtained data.

- The hypotheses as mentioned above are called *directional* hypotheses: statements about how (in)dependent variables relate to each other. There is another type of hypotheses: *descriptive* hypotheses ask specific questions about some phenomenon. These kinds of hypotheses are phrased like questions and address parts of the earlier defined research questions. We advice to only use directional hypotheses and work out descriptive hypotheses earlier on the level of research questions.

Tips and Tricks

- When you are doing your literature study, it is good to keep a so-called 'Annotated Bibliography'. This is a list of references in which you include for every paper that you write: title, authors, objective contents (what is in the paper, sometimes summary of the abstract), relation to your study (state reason that you read the paper, even if it turned out not be relevant), and your opinion about the paper (were there missing gaps in the experiments, would you have done it otherwise, and so forth). Every item in this list should be about half a page max.

- There are many internet resources on how to formulate objectives, questions and hypotheses. This is not necessarily a good thing (but it does not hurt), because outside scientific research, these terms are used more loosely compared to how we defined them above (especially concerning hypotheses).

- Be aware that for computer scientists constructing simulation models for analysis studies, the hypotheses, ideally, come from the domain expert. You can do your best to find hypotheses by studying the literature, but domain expertise is essential for the right positioning and focus of the research.

- Remember that *true hypotheses are boring*. You are evaluated based on your objectives, not on proving your hypotheses! Although you may want your prediction to be correct, keep an open mind. Well formulated hypotheses have equal chance to be proven or disproven.

[2]For example, http://www.newfoundations.com/Hypothesizing/Hypothesize.html gives three rules (disconfirmability, relative simplicity and groundedness) which should apply to your hypotheses.

6.2 Model

The hypotheses that you have by now defined are going to determine largely what your model is going to look like. In this stage of preparing your experiment, you make a *model specification*. Note that this is not the actual specification of the program code yet (which we call "implementation" and is explained in the next Section). The deliverable that comes out of the modelling stage is the model specification, documented with a number of assumptions made beforehand and design choices made along the way. We do not give particular guidelines for structuring this specification, but let this depend on the specific choice of model (and algorithm).

Independent of whether you are conducting an analysis or design study, you need to have some model in which you can test the hypotheses. For simulation studies, such a model can, for example, be a cellular automaton, multi-agent based system or formal model. Each of these models asks a specific approach for building a model specification.

If you are conducting a design study, then you also need to have an algorithm. Deciding what algorithm is best suited and how much tinkering you have to do with it depends on the particular study. Often the algorithm itself is the main contribution of the research (e.g., to show that your algorithm outperforms other one) and thus motivated the research from the start on already.

In general, when embarking upon a simulation study, it is important to determine the right *level of model detail*: what to model and what not. It is seldom necessary to fully model every single detail of reality. We give some general guidelines for determining the level of detail (based on the guidelines presented in [214]). Firstly, since models are designed for specific purposes, clearly define the specific issues under investigation and the performance measures. The hypotheses that you have defined should guide you into determining what the specific issues are. Secondly, use expert knowledge and sensitivity analysis to help determine the detail level. Thirdly, start with a "moderately detailed" model and work this gradually out. It often happens that this takes place the other way around: start with very much detail and take things out that turn out to make the model too complicated. Fourthly, the level of detail should be consistent with the available data. Finally, available money and time are always factors in deciding on the level of detail.

6.3 Implementation

Implementation means the process of working out your simulation model into an executable computer program. We assume that you have basic programming skills, thus we do not spend much time on the actual implementation of your model. In general, you do not need to write up a simulation program from scratch; there are a sufficient number of simulation packages available in/with which you can write your program. Note that there are significant differences between these packages: some are routine libraries that you can use in your (JAVA) program, others are complete suites with their own programming language. It pays off to seriously think about what suits your needs best. Depending on the scale of your project, spend thus some time (ranging from days to possibly weeks) playing around with some simulation packages. There are several surveys available that point out the differences between a number of these packages, e.g., [133, 308, 347].

We advice to take the following remarks into conderation when deciding upon a package.

Firstly, get an idea about the learning curve: how quickly can I program something sensible that generates meaningful output? For example, if your project can take 6 months in general, you cannot spend 3 months on learning how to work with the package. Secondly, can you achieve what you want with a particular package? Some packages are well-suited for quick prototyping, while others enable you to create a well-designed program. This point is related to the learning curve and available time. Thirdly, since you get most insight into the behaviour of your model by playing around with it, it is important to quickly get some initial results – to start playing. You may thus consider to use a combination of packages: first *quick-n-dirty* prototyping in a package with a steep learning curve, and then write up the real implementation in a more complex package. Fourthly, the implementation is where you are really confronted with your modelling decisions: the program simply has to work. It happens all the time that while implementing you reconsider a decision, have to take a new decision, etcetera. You have to document the final decision that you ultimately made: you may also mention the different decisions that you took before. Later this will allow you to explain your code much better to yourself and others. Finally, keep always the option to "build something up from scratch" open. It does not pay off when you have to invest so much time into understanding some code that someone else wrote, while you could have done it yourself in a heartbeat.

Everything starts with good preparation. Before you start programming, it can be good to make some drafts (on paper) of what you are going to see on the screen when you will be running your experiment later. This helps you to make your ideas about the experiment more concrete. The independent variables are usually shown as sliders[3] (although these can also be put in initialisation files); the dependent variables can be shown in dynamics graphs (monitors). If you are doing the simulation study for someone else, then these *manual screenshots* are great communicators – especially if your user is not a computer literate.

When you are actually implementing, it is important to verify that your program is correct (or: has no bugs). There are a number of techniques for doing this, and some that are not related to programming in general, some are specific for simulation programs. This list is in [214, Chapter 5]. Firstly, you can program in modules, classes and/or subprograms: a main program that calls methods and procedures from libraries or other procedures. Note that some packages do not allow for object-oriented programming, but are strictly procedural. While this is okay for prototyping, in practice, this is a serious limitation for implementation of large-scale simulation programs. Secondly, have someone else going through your code, e.g., a domain expert. With explanatory comments in your source code and pseudo-code, you may be able to talk that domain expert through your code. He/she can doublecheck your decisions. Thirdly, run your simulation with some bogus input parameters to see if you get reasonable results. Fourthly, you can let a discrete-event simulation program produces *traces*: this is basically a core dump[4] of the state of your simulation at each iteration: contents of the event list, state variables, statistical counters, and others. This allows you to check the output step-by-step. Finally, you can observe an animation of your results.

[3] A *slider* is a component of the graphical user interface of a computer program. It is a (horizontal or vertical) bar over which you can move a pointer. The bar represents a value range and the pointer shows the chosen value. For example, you can use it to adjust the speaker volume.

[4] A core dump is the name often given to the recorded state of the working memory of a computer program at a specific time.

6.4 Experiment

You have now clearly defined hypotheses as well as a model to test these hypotheses with. But exactly *how* are you going to test these? For other research studies you may consider formal proofs or making calculations - here, we are going to run a number of (computer) experiments to test your hypotheses. The stages of such experimentation are: 1) experimental design and setup, 2) performing the experiment, 3) results and 4) analysis. Although we present these stages like a waterfall model (there is one way down and no way back), in practice you go back and forth through these stages. In this Section, we describe each of these stages in detail.

6.4.1 Design

Experimental design is an important part of the *scientific method*[5] and many researchers spend much time thinking about how to do this right, e.g., Montgomery [240]. In this Section, we explain the concepts of dependent and independent variables, define the basic elements of design (iteration, run, etcetera), and a number of design types (pairwise, factorial, etcetera). Note that here we mention only little basic statistics while explaining experimental design, and you can find extensive explanation of fundamental statistics related to this topic in [240]. Furthermore, you can find a good short review of basic probability and statistics relevant to simulation modelling and analysis in [214, Chapter 4].

Variables

Your experiments are set up in order to find out about relations between variables (as stated in your hypotheses). In your experiments, you systematically vary a number of variables, called *independent variables*, and observe the responses of some variables, called *dependent variables*. Here, we distinguish independent, dependent and other variables, which for the complete set of variables. For your study, you can best enumerate your specific variables in a table (as introduced later in this book when we explain how to write a report).

- **Independent variables** – The independent variables (or: factors) are the variables that you systematically vary in your experiments[6]. For example, if you want to know how a population size affects the average fitness of this population, the *population size* is an indedependent variable. It is good habit to define the type of this variable, for example, integer, real or cardinal. You also define a *range* for this variable. This can be [`min`, `max`] where each value between `min` and `max` is tested. The range can also be a specific set. For example, for population size, you may want to test only sizes from the set {40, 100, 400, 1,000}. The different values in this set are called *levels*.
- **Dependent variables** – The dependent variables (or: response variables) are expected to be affected by the independent variables in an experiment. In defining these variables, note that you have to be able to *measure* them. Although this definition may seem trivial, it sometimes takes long before you can decide on measurable dependent variables.

[5] The scientific method is a body of techniques for investigating phenomena and acquiring new knowledge, as well as for correcting and integrating previous knowledge; it is based on gathering observable, empirical, measurable evidence, subject to the principles of reasoning (Newton, 1687).

[6] In "traditional" design of experiments, the input parameters and structural assumptions composing a model are called *factors*. The independent variables are factors by this terminology, but strictly speaking there can be factors that are not independent variables.

Besides, they have to be *directly* observable in your experiment. Also, you have to determine *how* to measure them. Although you may be eventually interested in the average of some factor (for example, average age of the population), you cannot observe averages or sums directly in your experiment. The rule of thumb here is to keep the collected data as raw as possible and do all aggregated calculations (averaging, sum, and so on) afterwards. In simulation studies (with a graphical user interface shown on the computer screen), the dependent variables can usually be *monitored* while running an experiment. This can be a simple counter or it may be a more advanced visualisation like a dynamic line or bar chart (more about line charts later) that is filled while the experiment runs. Such monitoring is indispensable for large-scale experiments for which it is not feasible to do all analysis afterwards and re-run the experiment if necessary.

- **Other variables** – Besides the previously mentioned variables, there are typically many more variables in a simulation study. These are the variables that remain the same throughout the experiment, but (may) have impact on the results. Hence, these could be called parameters and are categorised in general as *other variables*. Since these parameters are essential for someone who wants to reproduce your results, these need to be enumerated here. If you research an artificial society, then the size of the world is an important factor (note that if you are interested in investigating this explicitly, it would have been an independent variable) and thus considered to be a parameter. Although it is sometimes difficult to decide what has and has no impact on the result, it is always best to go safe here. Leave out things only if you are sure enough that they have no effect: whether you clicked the start button with your left or right index finger can hardly have an effect on the results and, therefore, not a parameter.

Elements

Here, we define the terms (in decreasing order of granularity): iteration, run, series, set and experiment.

- The primitive step in a simulation is an *iteration* (or: timestep) in which all individuals can perform one or more actions. It is basically one iteration of the simulation control loop as discussed earlier in this book. Note that, statistically (design-wise) speaking, these iterations are *just* different levels of an implicit factor in your design: time.
- A *simulation run* is a sequence of iterations, usually representing progress of time in a simulation. For example, a run could consist of 1,500 iterations.
- If your model contains non-determinism (for example, you have Monte Carlo initialisation [240]), then you will have to do a number of runs. This is called a *series*. A series then consists of, for example, 10 independent runs.
- Consider that you obtain a series of runs for each value of an independent variable. For example, if *population size* is an independent variable and the possible values are {100, 200, 500, 1000}, then you have four series. These four series together are called a *set*.
- Assuming that you test your independent variables *ceteris paribus*[7], you will have sets for all independent variables. These sets together form your complete experiment.

[7] Ceteris paribus means: all other things (here: all other variables) being equal.

Types

An experimental design is now *a procedure for choosing a set of factor level combinations* [25]. Although we do not intend to give a complete overview of design of experiment (DoE) methodology here (see [240] and [214, Chapter 12] for this), we want to point your attention to the following ways to organise your experiments.

- **Simple comparative** – Assume that you want to test one single factor with two levels. For example, you have modelled and implemented an artificial society (for example, Sugarscape [111]) and are interested in the effect of *communication*, where individuals can either communicate in *centralised* or *distributed* fashion. Thus "communication" is the factor under investigation (independent variable) and the levels are {centralised, distributed}. The simplest design of experiment (DoE) that you can use here is to do simple comparative experiments: conduct two series of experiments, one with centralised communication and one with distributed communication. Say that the performance measure is the population size at the last iteration and each series consists of 10 runs. Then you have obtained 2×10 values. You can visualise these results with a dot diagram, histogram, box plots, or any other visualisation form (as explained below). See [240, Chapter 2] for an extensive explanation of a number of basic statistical concepts (such as random variables and sampling distribution) which you can use for simple comparative experiments.

- **Randomised complete block** – This type of design is not so much used for simulation studies. However, we include it here because it is, in general, one of the most widely used experimental designs – and, it is a design type to consider if you are researching in physical collective intelligence (e.g., collective robotics). It is especially meant to make the experimental error as small as possible: the *experimental error* is a statistical error that arises from variation that is uncontrollable and generally unavoidable[8].

  Assume[9] that you are designing a robot and it has to have some kind of thermometer – a tipping device that it can dip in a liquid substance and measure the temperature. You have four options (different tip types) and want to choose the best one. There is thus one factor (the tip type) that has four levels. You have decided to do four tests with each tip. A completely randomised (single-factor) design would consist of randomly assigning one of the 4 (types) $\times$ 4 (observations) = 16 runs to an experimental unit: put the tip in a liquid and oberve the temperature. This requires 16 different liquids – each of which has to be prepared or obtained *exactly the same*. The problem with this design is that it is hard to guarantee the sameness requirement; for example, they vary a bit in temperature because the temperature in the room changes during the time of experimentation. You can compare it with a skating competition where the first skaters do their round on freshly prepared ice, while the last ones have ice that the 18 skaters before them have skated on.

  A design that solves this is one where each tip is tested once on each of four liquids: this is a *block*. But now you run the risk of order-effects: the tip that measures the

[8] Note that this type of error has a much more important role in non-simulation experiment studies. In simulation, you can control the environment of your experiments relatively well (because you programmed it yourself). If you are experimenting with physical collective robotics, then this error is an important one to take into account.

[9] This example is based on the Hardness Testing Problem example from [240, Chapter 2].

liquid as the last one may include effects of the other three measurements. Therefore you randomise the order in the different blocks: for each of the four liquids, you test all the tips (hence, complete) in randomised order. This is the randomised complete block design.

- **Latin Squares** – Randomised complete block design works by blocking out variability due to a known and controllable nuisance variable. There are more designs that do this. Assume[10] that you are designing a fuel-driven robot and you have to decide on the formulation of the fuel with respect to robot's fuel usage. There are five different formulations for mixes of raw materials that make the fuel. The formulations are prepared by different operators. There are thus two nuisance factors: batches of raw material and operators. Ideally, you want to test each formulation exactly once in each batch of raw material, and each formulation prepared exactly once by each of the five operators. Such a design is called a *Latin square design*. These designs are suitable when there is one factor (batch of raw material) with p levels (formulations). A latin square then consists of a $p \times p$ table in which each cell is assigned one of the levels, indicated by a latin letter A, B, and so forth. Each letter occurs once, and only once in each column and each row.

- 2^k **Factorial** – The most used design for simulation for simulation studies is the *factorial* design. This design is useful when you want to study the effect of multiple factors. It is much used in the early stage of experimentation when you are still largely in the dark about the behaviour of the system. A factorial design enables you "to investigate all possible combinations of the factor levels in each trial or replication of the experiment" [240]. A simple solution for dealing with multiple factors is to use an one-factor-at-a-time design. However, note that 1) this is very inefficient, and 2) such design does not show possible interactions between the factors.

  If you have k ($k \geq 2$) factors, you define the *main effect* of a factor as the change in response produced by a change in the level of the factor. How does each of the factors affect the responses? The 2^k *factorial design* is a design where you choose two levels per factor and run simulations for each of the 2^k factor-level combinations (called *design points*). The two factor levels are usually indicated by '+' (high) and '-' (low) signs, that are assigned to particular values of the factor by the experimenter. Such a design gives you a *design matrix* where each row is a design point and a measurement of the response (giving you 2^k rows, hence the name of the design type).

  Calculating main effects now works as follows. Assume that you have 2 factors, hence $2^2 = 4$ design points (rows in the design matrix), thus 4 responses in total. You can now calculate the main effect of a factor by taking the dot product of the design point sign vector and the response vector divided by 2^{k-1}. Let the design point sign vector for factor 1 be $[-, +, -, +]$ and the response vector $[R_1, R_2, R_3, R_4]$, the main effect e_1 of factor 1 is $(-R_1 + R_2 - R_3 + R_4)/2$.

  The degree of interaction between factors is measured by the *two-factor interaction effect*. You can calculate this value for two factors by first multiplying their design point sign vectors, multiplying this resulting vector with the response vector, and then

[10] This example is based on the Rocket Propellant Problem example from [240, Chapter 4].

divide the result by 2^{k-1}. Two-factor interaction effects are completely symmetric and can be generalised to k-factor interaction.

- 2^{k-p} **Fractional factorial** – The 2^k factorial design may still involve too much experimentation if there are many factors. A solution for this is to first pick a subset of the factors and make a factorial design for this subset. This solution is called a 2^{k-p}*fractional factorial design*, where p factors were removed from the factor set. If p is 1, we speak of a *half fraction*, if p is 3, then we say an *eighth fraction*, and so forth. The main question with this design is how to decide upon the p factors - which ones go out? The experimental-design literature has elaborate a well-thought-out procedures for this, e.g., [240, Chapters 8 and 9]. An abbreviated "cookbook" procedure for determining this subset can be found in [214, Chapter 12].

6.4.2 Setup

The experimental setup involves deciding on everything that has to do with your experiment, necessary for reproducing the results – except for the part that you have described in the experimental design (such as independent and dependent variables, design type). Concerning variables, here you determine the values of the "other variables" (those besides the (in)dependent variables). Thus if the cost of performing some action is not a factor (independent variable), then this is the place to precisely determine the cost. You also determine things related to the simulation itself, for example the number of iterations per run. Finally, you state what your equipment consists of, for example the speed and type(s) of the computer(s) that you use for running the experiment.

Like with the factors in the experimental design, you can best enumerate all those variables in a series of tables and include these tables in your report.

6.4.3 Performing the Experiment

It may be too trivial to take the actual performing of the experiment up here in a separate section, because this usually only concerns pressing the Start button in your implementation of the simulation model. However, there are still some necessary considerations.

Firstly, make use of *scripts* that let the computer run your experiments and write the data to files. This way, you can run experiments while you are doing something else (e.g., sleeping). Some simulation software packages have the option to script the performing of your experiments, but it is not hard to program it yourself. Alternatively, you can use normal scripting languages (e.g., UNIX shell scripting, Perl, Visual Basic Scripting) and wrap this script around your program.

Secondly, pay considerable attention to the runtime of simulation runs. It is not uncommon that a simulation has a runtime of several hours or even days (exceptionally, simulations on distributed networks may take weeks). Sometimes this is necessary (because of the many factors to be investigated), but often significant speed increase can be achieved when the simulation is programmed efficiently. The key point is to seriously concentrate on your main objective: (dis)proving the hypotheses – despite all the temptations to create marvellous 3D visualisations of your artificial world: "In der Beschraenkung zeigt sich erst der Meister"[11] (Goethe, 1802). Besides code optimisation for improving runtimes of individual runs, an

[11] In English: "It is not until limitations are put on him that the master really shows himself".

efficient design also contributes hugely to speed improvement of the overall runtime of the experiment.

6.4.4 Results

Data is what makes your clock tick in experimentation. You implementation delivers a number of *data files* on which you are going to apply your output analysis later. It is important to write your results to files - despite the dynamic graphs that are included in most simulation software packages. You *never* store your results by means of screenshots - this may seem like a viable option at 3am in the morning, but makes it completely impossible to properly analyse the simulation output later.

It is essential to *always store your original data* - no matter how well you have eventually visualised your results. One of the most important parts of the scientific method is reproduction of results. When you or readers of your work want to reproduce your results (for example, in another implementation or as preparation for extending your research), the raw data is necessary.

Raw data (data files) itself gives you hardly any insight into the results - it is very hard to *interpret* the data and usually impossible to see patterns. For this, you need to *visualise* your data. We explain some ways to visualise your data next. But first we want to let you know some requirements of data visualisation (or: graphical display) according to Tufte [355, 353, 354]: show the data; induce the viewer to think about the substance rather that methodology, design or something else; avoid distorting what the data have to say; present many numbers in a small space; make large data sets coherent; encourage the eye to compare different pieces of data; reveal the data at several levels of detail; serve a reasonably clear purpose: description, exploration, tabulation, decoration; be closely integrated with the statistical and overall descriptions of a data set.

Visualisation

There are numerous ways to visualise your gathered data: for example, tables, graphs, or box plots. Because no reader of your work will appreciate pages and pages of raw data, you always have to do some *data reduction* in order to make your results comprehensible. This means that, for example, instead of showing the data of each of 10 independent runs, you average over the 10 runs and show this together with the standard deviation. While doing this, you always lose some information, but it is important to lose as little information as possible. Here we present some ways that are often used to visualise results.

Note that, independent of the specific visualisation, values of independent variables are almost always shown on the x-axis and values of dependent variables on the y-axis when you plot those variables against each other.

Dot diagram Imagine that you are investigating the effect of two negotiation mechanisms for large networks. The performance measure is the time it takes until mutual agreement is reached, and each series consists of 10 independent runs. After performing your experiment, you have 2×10 values. These values can be plotted as dots in a *dot diagram* where you show the names of the two negotiation mechanism on the x-axis and the performance measure on the y-axis. The advantage of the dot diagram is that is a very simple and straightforward

visualisation, and it shows the actual data (not aggregated); however, it is difficult to show data with many value points.

Histogram An histogram shows the (relative) frequencies of the observations. Although it does not show the actual data, you can see the central tendency, spread and general shape of the distribution of the data. You construct an histogram by dividing the x-axis into *bins* (usually of equal length) and draw a rectangle for each bin where the area of the rectangle is proportional to the number of observations that fill in that bin.

Box plots This type of plot (also called box-and-whisker plot) is very efficient.A box plot shows the minimum (lowest horizontal line), maximum (highest horizontal line), median (50th percentile, horizontal line within the box), lower quartile (25th percentile; part of the box under the median) and the upper quartile (75 percentile; part of the box above the median). For further explanation, see [25, Chapter 3].

Scatter plots In a scatter plot, you plot two variables against each other. Typically, the variable on the x-axis is an independent variable and the one on the y-axis is a dependent variable. Based on such a scatter plot, you can get an idea whether the two variables are related, they are (non-)linearly related, there are outliers, and so on.

Line graphs It is common practice in simulation studies to make plots where the x-axis displays the iterations and the y-axis thus shows the value development of some variable over time, e.g., the population size. Technically speaking, these plots could be considered scatter plots. In most simulation software packages it is very easy to make such time-series plots and let them dynamically change while the simulation is running.

Often the individual data points are connected to each other with lines – this gives you a *line graph*. Sometimes the actual data points are left out in such a line graph. It is important to realise that these lines are aggregates (derived from the original data) and that they do not display the data itself.

6.4.5 Output Analysis

Too many research studies stop just before the point where it starts getting interesting: with the analysis of the results. When a study has results in some graphs and plots, this is generally considered succesful and publishable. This is not good working practice. We now know, based on the ∪-model of research, that you are only half way when you have obtained results; you still have to go back up on the right side of the ∪-model. The two main stages at this time are output analysis and validation, which we explain here[12]. Note that these discussions are by no means intended to fully introduce you to the field of statistical inference; they are mere handgrips to get you started to analyse your results objectively.

With output analysis, we mean the *statistical verification of the hypotheses*. Assume that through experimentation you have obtained some data set. *Hypothesis testing* is the whole process of formulating a hypothesis, taking a random sample, computing a test statistic, and the acceptance (or failure to accept) the hypothesis. Whereas we earlier used the "hypothesis" to state a predicition, we can define a more narrow version of it: a "statistical hypothesis"

[12]This Section is largely based on [25, Chapter 3] and [240, Chapter 2].

is *a statement either about the parameters of a probability distribution or the parameters of a model.*

Suppose that you have two different algorithms A and B and you are interested in whether algorithm A is faster than algorithm B on some set of test problems. Thus your data consists of two sets of values, where each value is a runtime measurement for A or B for each test problem. Then you can define

$$H_0 : \mu_1 = \mu_2 \qquad \text{(null hypothesis)}$$

$$H_1 : \mu_1 \neq \mu_2 \qquad \text{(alternative hypothesis)}$$

where μ is the mean of the response at the factor level (here: runtime measurement), H_0 says that there is no difference between the algorithms and H_1 says that there is a difference. There exist a number of *significance tests* with which you can determine whether you can accept a hypothesis or not *beyond a reasonable doubt* – based on objective measurements and calculations. All these test are based on taking a random sample, computing a test statistic and rejecting, or failing to reject, H_0. These tests are based on the discovery of false falses (null hypothesis is rejected while it is true) called *Type I error*, and false trues (null hypothesis is not rejected while it is false) called *Type II error*. Statistically:

$$\alpha = P(\text{type I error}) = P(\text{reject } H_0 | H_0 \text{ is true})$$

$$\beta = P(\text{type II error}) = P(\text{fail to reject } H_0 | H_0 \text{ is false})$$

where α and β thus represent the respective errors. The value of the probability of the α-error is also called the *significance level*. Hypothesis testing then consists of specifying the significance level and design the test procedure as such that the probability of the β-error has a suitably small value.

Two-Sample t-Test

A possible test procedure for hypothesis testing is the so-called *two-sample t-test*. Other test procedures are the *two-sample z-test* and the *paired t-test* – see [25] for more details of these two other test procedures, and [240] for more test procedures and when to choose which test procedure. We explain the two-sample t-test here. For this test procedure we assume that the variances of two data sets are identical. We estimate the common variance, called S_p^2 (S_p squared), by calculating

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

where S_1^2 and S_2^2 are the two individual sample variances[13], and n_1 and n_2 are the sample sizes. Having this estimate, we can now calculate the test statistic t_0 as follows:

$$t_0 = \frac{\overline{y}_1 - \overline{y}_2}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

where $\overline{y}_1$ and $\overline{y}_2$ are the sample means[14]. The test procedure then compares t_0 to the t distribution with $n_1 + n_2 - 2$ degrees of freedom: we reject H_0 if $|t_0| > t_{\alpha/2, n_1+n_2-2}$. If you have decided on a significance level α, e.g., 0.05 (a so-called 95% confidence interval), you can determine this latter value by looking it up in a t-table[15].

[13] See http://en.wikipedia.org/wiki/Variance for how to compute sample variance.
[14] See http://en.wikipedia.org/wiki/Sample_mean for how to compute sample mean.
[15] See http://en.wikipedia.org/wiki/Student%27s_t-distribution.

Analysis of Variance (ANOVA)

Let us now assume that you are conducting an experiment with one single factor (with a levels, where $a > 2$). It is difficult to use a test procedure like the t-test, because you do not do any comparisons. Instead, you can analyse the variance of the obtained results. The statistic used in this case is called *ANalysis Of VAriance* (or: ANOVA). The variance of a random variable is a measure indicating how its possible values are spread around the expected value. The square root of the variance is called the *standard deviation*.

Say that you are building a new robot and that you have to determine the hardness of the wheel. There are four different wheels that you can choose from. You are interested in the wheel hardness that brings your robot the furthest in some fixed time interval. You do five runs (or: *replicates*) per wheel type (to be technically correct, the design is completely randomised). Your data will then be 20 values in total: five values per wheel type. You can plot this data in a box or scatter plot and get some idea how the wheel type influences the speed of the robot. But suppose you want to be more objective about this idea: you want to test for the differences between the mean speeds at all four levels (different wheel types). In other words, you want to test the equality of all four means. The right procedure for determining this is ANOVA – arguably "the most useful technique in the field of statistical inference" [240].

We can put this information in the following model (called the *effects model*):

$$y_{ij} = \mu_i + \tau_i + \epsilon_{ij} \begin{cases} i = 1, 2, \ldots, a \\ j = 1, 2, \ldots, n \end{cases}$$

where y_{ij} is the ijth observation, μ_i is the mean of the ith treatment (or: level), τ_i is the ith treatment effect and ϵ_{ij} is a random error component. We let H_0 express that the means μ_i for $i = 1, 2, \ldots, a$ are equal in this model: $\mu_1 = \mu_2 = \ldots = \mu_a$ and H_1 the opposite: some means are different.

Following [25], let the sum of all observations and the ith treatment $y_{i\cdot}$ be

$$y_{i\cdot} = \sum_{j=1}^{n} y_{ij}$$

where i corresponds to a level, j corresponds to an observation and n is the number of observations per treatment. Then, let the grand total of all the observations $y_{\cdot\cdot}$ be

$$y_{\cdot\cdot} = \sum_{i=1}^{a} \sum_{j=1}^{n} y_{ij}$$

where a is the number of factor levels. We can define means for both these measures:

$$\overline{y}_{i\cdot} = y_{i\cdot}/n, \text{ and}$$

$$\overline{y}_{\cdot\cdot} = y_{\cdot\cdot}/N$$

where $N = an$. With these measures, we can further calculate the *corrected sums of squares* for the total (SS_T), the treatments ($SS_{\text{Treatments}}$) and the error (SS_E):

$$SS_T = \sum_{i=1}^{a} \sum_{j=1}^{n} y_{ij}^2 - \frac{y_{\cdot\cdot}^2}{N},$$

$$SS_{\text{Treatments}} = \frac{1}{n}\sum_{i=1}^{a} y_{i\cdot}^2 - \frac{y_{\cdot\cdot}^2}{N},$$

$$SS_E = SS_T - SS_{\text{Treatments}}$$

With the treatments and error, we associate the mean squares (MS's): $MS_{\text{Treatments}} = SS_{\text{Treatments}}/(a-1)$ and $MS_E = SS_E/(N-a)$. If we take the ratio of these MS's, we get the *test statistic* F_0:

$$F_0 = \frac{MS_{\text{Treatments}}}{MS_E}$$

that is distributed as F with $a-1$ and $N-a$ degrees of freedom. The test procedure is now to reject H_0 if

$$F_0 > F_{\alpha,a-1,N-a}$$

where the latter can be looked up in a F-table[16].

6.4.6 Validation

Validation is "the process of determining whether a simulation model (as opposed to the computer program) is an accurate representation of the system, for the particular objectives of the study" [214]. It is very different from the output analysis that we explained above: output analysis is a statistical issue and concerns the estimation of a simulation model's (not necessarily the system's) true measures of performance.

There are different ways to do such validation, as described in [214, Chapter 5]: inspection, confidence-interval and time-series. These methods assume that you have a number of values as observed in your simulation and a number of values as observed in a real-world system. These data sets are then compared with each other. For example, if you simulate a queueing system for cash tills in a supermarket, you can make observations in the simulation as well as in reality.

For some simulations, such real-world system data may not be available. It is often very study specific what kind of data you are simulating. Therefore we advice you to make the validation of your results a collaborative effort with the domain expert. Thus, although we do not explain here the before-mentioned approaches in detail, we do want to emphasise that result validation is critical for the credibility of your results and should therefore not be skipped in a research study.

Tips and Tricks

- You may want to organise your analysis by identifying a number of different *findings*. A finding is phrased the same as an hypothesis, making it *a specific statement of discovery* (rather than prediction). These findings do not necessarily correspond one-to-one with the hypotheses that you have defined earlier - you can have discovered more (more findings) than originally stated in your hypotheses. This gives you a way to place these discoveries. If you phrase your findings well, than it is relatively easy to relate the proof/falsification of the hypotheses to your data - namely, via these findings.

[16] See http://en.wikipedia.org/wiki/F-distribution (External links).

- It is good habit to keep an experiment log while you are running the experiment. Some research disciplines (e.g., biology, psychology) know strict formats for such logs. In general, write down what conditions you tested and briefly what the results were. If you are in the middle of doing the experiments, this may be relatively easy - doing it afterwards is almost always impossible: even the next day you may not remember precisely which particular parameter setting yielded that terrific result you had at 3am this morning.

6.5 Summary

This Chapter explained how to properly set up a simulation study. The big picture can be seen in the so-called ∪-model: you start with the formulation of objectives and research questions, then define hypothesis, design, set up and conduct experiments yielding results, based on which you can validate/falsify the hypotheses, answer the research questions and conclude whether you achieved your objectives. Although presented as a one-way flow of activities (or: waterfall model), in practice you notice that you often jump back and forth between these different phases. This is not problematic, if you keep it within limits. At some point, you have to fix yourself upon a set of objectives, questions and hypotheses in order to get meaningful results within a reasonable time frame. Current working practice in the fields of artificial intelligence and computer science often neglects the serious consideration regarding experimental design and the objective analysis of output results – therefore we gave special attention to these activities in this Chapter.

– The skill of writing is to create a context in which other people can think.

Edwin Schlossberg

– Learn as much by writing as by reading.

Lord Acton

7 Writing a Research Report

So, by now you have clearly defined your objectives and questions, formulated hypotheses, did all your experiments, analysed the results and drew some bold and insightful conclusions. Thus the time has come to write everything down to let others know about the fruits of your laborous work. Right? Wrong! By now you should already have written down the largest part of your report. Theoretically, at this stage you are mainly concerned with organising and structuring all the separate pieces of text that you have written together in a single report. For example, while doing the literature review, you compiled an annotated bibliography, and you kept an experiment log while running the experiments. Also, you may have put substantial explaining comments in your source code. With these texts as a departure point, you are now ready to combine everything. Thus, it is important that you do not keep the writing for the last - read this Chapter *before* you start reading, implementing, experimenting, and so forthxs. If you have already done so, keep this advice for the next time that you conduct a study.

Aims of this Chapter

This Chapter aims to explain how to write a research report according to some general accepted academic guidelines. Basically, a report consists of 1) an introduction, 2) a literature overview, 3) a description of your new theory/model/algorithm, 4) a description of the implemention of this new thing that you did, 5) description of the experiments and results, 6) conclusions, and 7) a bibliography. This Chapter explains each of these parts in detail.

Tips and Tricks

We finish some Sections in this Chapter by some Tips and Tricks. These are practical pointers on how to write particular parts of the report. For example, we advice you to write the introduction after you have written most of the other parts of the report. Or, when you are writing up the analysis of your experimental results, you can best organise your findings in

an itemised list. There are some general tips and tricks that are good to keep in mind, which we give here.

- Be careful that your report does not become a chronological proze describing your activities from day one up till the last day. This does not make pleasant reading. Adopting the structure that we suggest here, should already prevent you from doing this. If you still want to report this, then include a separate appendix to your report in which you describe everything that you actually did. For a thesis or internship, such an appendix can be considered a trainee report or journal. There may be good reasons for writing and including this – for example, if your scientific results are not satisfactory, but you did all that you could as good as you could. Then you want to be evaluated based on these attempts, not on the fact you did not yield the intended scientific results. If you are more experienced in research, then you must be able to prevent this by defining your objectives correctly.

- Besides writing a report, you may want to disseminate your results in other ways. For example, you need to give a presentation about your work afterwards. We do not explain here how to give presentations. Also, before embarking on a research study, you may have to do more preparation than we have included in this book. For example, if you are a graduate student preparing for your final project, you need to choose a project, prepare a project proposal, choosing a supervisor, etcetera. We do not explain here how to give presentations or choose a project/supervisor/etcetera. A good book that explains this is [68].

- When your report will be read or evaluated by someone else (lecturer, supervisor, manager), keep this person in the loop when you are writing. A good setup is to do this chapter-wise, although some sometime a more granular level is necessary – per section or per subsection. When you show new material to this person, indicate what has changed. Thus your supervisor can focus on these changes, while not having to go through the whole document each time.

- Be very precise in language, spelling, etcetera. Keep this for the last thing to do. If possible, let a native speaker go through your report to check the language. Use a spell-checker to take the big mistakes out. Bad use of languages, spelling errors and typos keep your reader from focusing on the message of your report: the content. In the words of R.K. Milholland: "*Typos are very important to all written form. It gives the reader something to look for so they aren't distracted by the total lack of content in your writing.*".

- A colleague of mine told me that she often organised her reports just before the deadlines literally by means of scissors and tape. All the texts that she had written during the study were in one big document, but this document was completely unorganised. She knew that all the information was in there, but it still had to be put in the right order. So she would print out the document, take scissors and cut up the whole document and rearrange on the floor of her office. She then taped it together, and copied and paste everything back and forth on the computer. There are two lessons that you can draw from this: firstly, write down everything in one big document and do not stop until you have the feeling that everything you want to say is in there; secondly, this document

does not have to have the structure of your final report, but can still be completely re-arranged at the last minute.

- Choose a good word processor in which you will write up your report. In the end, this can be the difference between successful and failed writing. There is a multitude to choose from, for example, Microsoft Word, OpenOffice Writer, LaTeX.

- Our personal preference is to use LaTeX for writing your report. Although at the start it is a tough cookie to crack (it has got a very slow learning curve), your learning investments will eventually pay off big time. The key point of LaTeX is that you 'only' have to structure your document logically, and LaTeX will take care of layout, style, and so forth. This means that, in beginning, you can focus on the content of your work rather than layout, which is a big plus. In academic circles, the layout that LaTeX produces is concidered to be of very high quality. LaTeX may put you patience somewhat to the test – but if you are in the final stages of writing, its capabilities to reorganise Chapters, renumber Figures, change the make-up of the bibliography, etcetera, will be very rewarding and save you loads of time - which is good if you are working towards a deadline.

- You can find various books, tutorials and introductions on LaTeX online of which we recommend [260, 200] in particular.

7.1 Overview

When you are writing your report, you are actually structuring your own thoughts and ideas about the study as well as those of the reader. Before explaining each of the Sections in detail, we give you some general advice on structuring and writing the report.

Firstly, think about the report in terms of the five-Wives-and-a-Husband concept that we used earlier in this book – What, Why, When, Where, Who and How. What is your report about, why did you write it, how do you explain the what, why you wrote it now, for whom it is intended, and where you found your resources. It might be that this may actually not literally end up in the report somewhere, but you must have thought about it at some point – this influences for example the specificity of the text, use of language, and writing style. Before you start writing is an excellent time to do this.

Secondly, think up the skeleton of the paper in terms of Sections and Subsections. The skeleton may be nothing more than a text file with the headers of the (sub)sections filled in – the text is actually not there yet. If you have some ideas about what you want to say at some place, you can put some comments in that you later write out to full text. The skeleton is an evolving structure: it changes while you are writing. But it gives you a first idea of what you want to explain where. Having an initial skeleton should also answer the question whether you have a place for everything that you want to say. If you have already written pieces of text during the study itself (what we advice you to do), then you can already include these pieces in the skeleton.

Finally, the order in which you write the different parts of the report is in general to start with the literature review, model and implementation, then the experiments, and finally the introduction and conclusions. The Bibliography writes itself because you are including references throughout writing the different parts of the report.

7.2 Abstract

The function of an abstract is "to summarise briefly the nature of the research study, its context, how it was carried out and what its major findings are" [35]. The length of an abstract depends on the type of report that you write. If you are writing a scientific paper or article, then the abstract is normally around 150 words. If you are writing a thesis or technical report, it is at most around 200-300 words.

Because of the space limitations of an abstract, you need to think carefully about the structure of it. Keeping in mind what the function of it is, write it such that it reflects the content of your paper, but avoid it being a clear copy-and-paste of it. You may be compelled to take the sectioning of the paper and write a single line about each section, which then makes up your abstract. This is bad practice. For one, it does not make pleasant reading; secondly, the reader might see your paper as just an extension of your abstract – while it should be exactly the other way around. Thus do not make it a literal summary of the report, but restructure it. It sometimes works to start with your own motivation for undertaking the research study. For example, you observed a major problem in practice or with an algorithm, thought up a solution and you can show that your solution works good. Then start your abstract immediately with stating the problem, and work further from there.

Be aware that sometimes your abstract should be a teaser for the reader - (s)he reads the abstract and then decides whether or not to read the report. Thus try to write the abstract as such that it invites your reader to read further in the report.

Finally, avoid including references, specific jargon and acronyms – these will all follow later in the introduction. Also, make sure that your sentences follow logically on each other: that one sentence opens a gap or question by the user which is then filled or answered by the next sentence.

7.3 Introduction

The introduction basically states the objectives and research questions that you formulated. It also works these out further by giving motivation for them, it positions the study within the respective research field and relates it to the current state-of-the-art. Remember thus to explicitly (in bold-face or italics if necessary) state the research objectives and questions in the introduction.

The first paragraph of the introduction is usually a teaser – it states an observed problem or gap in some research field that no-one has noticed before. When someone reads this first paragraph, (s)he realises that your report is interesting because it brings up something that (s)he had not thought of before.

The last paragraph of the introduction traditionally says "This report is organised as follows. Section 1 introduces ...". In other words, you briefly talk your reader through your report such that (s)he gets an idea what can be expected. Do not overdo this, i.e., do not make promises that you cannot keep – often the disappointment of a broken promise outweighs the excitement about the promise anyway.

Tips and Tricks

- Leave the writing of the introduction to the last (chronologically). Most part of the introduction you will learn while you are actually doing the work (literature review,

research, analysis of results, etcetera). Although it is very tempting to start with the introduction and promise the world to your audience, you only know if you can deliver when you have done the work.

7.4 Literature

The literature Section is arguably the the most important part of the first part of your thesis. This is namely the first Section in which you are going to keep parts of your promises. If you have mentioned before that you aim to have a new algorithm that outperforms all other exisiting algorithms, these existing ones should be described here. You have to convince your reader that you are really doing something novel.

In your literature Section, you *do not refer explicitly to the current study.* It should be clear to the reader what the relation is between the discussed literature and the current study. You can refer to earlier work of yourself as part of the existing literature. But the presentation of exisiting work should be done objectively and disregarding the current study. Thus the thing not to do is to describe an algorithm, identify some anomolies and immediately mention that in the current study you solve the anomolies. This confuses the reader.

Writing the literature review is thus partially contradictory: on the one hand you cannot literally copy-and-paste the descriptions of other people's work (this would be be plagiarism), on the other hand your literature review has to offer something novel. One way to get around this is to split up the review in three different parts: firstly, you describe the exisiting work *in your own words*; then you discuss the anomolies/problems/and so on that either the authors of the original work identified or that you identified yourself; and finally you explain how the current study works towards solving these anomolies. The last part can be done in a summary of the Section that is then technically not really part of the review. The first part (objectively describing the existing work) may actually be a contribution on its own if it is the first time that this literature is brought and described together in some coherent context.

Tips and Tricks

- Adopt 'Cite-while-you-write': this means that you do not keep the organisation of your references to the very last thing. While you are doing your writing, include the references. Afterwards, this will take much more time than when you do it while you are writing.

- Remember the 'Annotated Bibliography' that we introduced before. You should be happy now that you made such a bibliography: this makes writing the literature review much easier.

- Some scientific disciplines, e.g., economics, let students start by reading some collection of papers (normally in the order of 20-30), on the basis of which a table is constructed that shows the coherence of the collection. This is a very good representation for identification of the coherence of the papers, combined with showing the contribution of your current study. You can compare such a table with a product overview that shows which properties the products have and do not have.

7.5 Model

As explained previously, when you were modelling your object of study, you ended up with a *model specification*. This specification is documented here. In case of an analysis study (with a model like a cellular automata, multi-agent system, or Boolean network) this specification is often some kind of framework or diagram. In case of a design study, this specification is often an algorithm.

We want to re-emphasise that the conceptual model is *not* the main loop of your computer program – this is implementation and presented in the next Section. In your model, you specify the observations and actions of your individuals and the interactions of the invididuals with each other and with their environment. This implies that you also specify the benefits and costs of individual actions.

Having said that the main loop is not your conceptual model, we have to weaken this statement for some particular application studies, e.g., artificial societies. In this case, the model may actually be very similar to the implementation. But even then it still holds that you specify and document the conceptual in terms of diagrams, Tables, Figures, and others instead of documenting the actual program source code.

Tips and Tricks

- Your model may contain mathematical formulas, e.g., describing the dynamics of your model. It is important that your reader can understand these formulas. This means that you have to define the variables, functions, and others, that appear in the formulas. Normally, one writes down the formula (centered on the page, between two white lines), followed by "where x denotes ..., y means ...". Although it may be considered trivial to do so, it is essential to do this correctly. Note that this remark does not only apply to this Section, but throughout your report where you use formulas.

7.6 Implementation

Although your experimentation should not solely depend on the implemention of your model, you have to describe the implementation explicitly. As mentioned before, with implementation, we mean the actual computer program that underlies your conceptual model (which is described above). It should be clear that you cannot simply dump your sourcecode in this Section. Neither does it suffice to let javadoc summarise all the comments in your code and dump this summary into this Section.

This Section should be in normal natural language and convince the reader that you implemented the conceptual model correctly. The best way to do this is to include the *pseudocode* of the main method of your program and describe the steps of this method. With these description, you can also explain the steps with more pseudocode.

If you have worked according to the methodology guidelines that we presented before, then you should have a list of decisions that you made during the implementation process. It is good to include this list in this Section. If some decisions have resulted in a revision of your conceptual model and have been further worked out in there, then you do not need to include those decisions here. But if you made decisions in your implementation concerning things that still may be ambigious in your conceptual model (you did it like this, but someone else may decide differently in another implementation) – then you need to include it here.

Like the report in general, you do not describe your implementation chronologically. Sometimes you want to describe that you first try this, which did not work; then you tried something else; etcetera – this does not make pleasant reading. You can better describe the last version of your code, including a separate list of the design decisions, as was mentioned earlier.

Tips and Tricks

- Although you are adviced not to include the sourcecode in your report (neither in this Section, nor as an Appendix), we do strongly advice to include the source code digitally with your report (for example, on a CD or accompanying website).
- The purpose of including a description of your implementation here is different than describing software in general - your results should ideally be independent of the implementation. If someone decides to implement your model on a different platform in a different programming language, your results should still hold. Implementation issues that do not get the same attention here compared to software documentation in general are, for example: bug, error description, software engineering information, limitations of the program, etcetera.
- If you are writing a scientific article or conference paper, it is good to put your sourcecode online. This stimulates repetition of experiments, and makes it easier for other people to continue with your work.

7.7 Experiments

The experiments have been extensively described earlier when we explained the methodology. We briefly summarise here what you should include in the report concerning the experiments. You can best structure this Section by including each of the following points in a seperate subsection. Firstly, for the experimental *design*, you have to state the independent variables, dependent and other variables including value ranges. You can do this best in one or more tables. You have to mention what type of experimental design you chose (completely randomised, factorial, etcetera), and the corresponding design specification, i.e., the Tables that show how you tested the different variables. Secondly, for the *setup*, you include a Table with all the other variables (i.e., not the independent and dependent variables because these are already reported in the design type) and the value that you fixed for the experiment. For example, you state that the size of the artificial world was 200×200. You also identify here the type and speed of the computers that you used for the experiment. Thirdly, you report the *results* of the experiment. You basically only include the graphs, plots, etcetera that you made based on the experimental results. Text-wise, this is a minimal part – it may literally say "In Figures 1-3 we show the experimental results.". If you have many graphs, put them separately in an appendix. What you should strive for, is that your reader can easily understand the results and easily do comparisons between graphs if so desired. If you have made the raw data available somewhere else (website or CD), then you have to mention that here. It is important to pay attention to making the graphs and plot. Below we describe this in more detail. Fourthly, you report the *analysis* of the results. This includes descriptions of the tests that you used for the testing of the hypotheses. Finally, you explain the *validation* of your findings - did you check with a domain-expert, did you consult the literature, and so forth?

Results

We give some special attention to how to do the visualisation of your results. This is normally in the form of tables, graphs or plots. For all these forms, there are some guidelines that you have to follow [68]:

- does it have a brief but clear and descriptive title?
- are the units of measurement clearly stated?
- are the sources of data used clearly stated?
- are there notes to explain any abbreviations?
- have you stated the sample size?
- does it have clear axis labels?
- are bars and their components in the same logical sequence?
- is more dense shading used for smaller areas?
- is a key or legend included (where necessary)?
- do the tables have clear colums and row headings?
- are columns and rows in a logical sequence?

We would like to add to this that you pay attention to the *value ranges* on the x- and y-axis for graphs, plots and diagrams. If your reader has to compare two graphs with each other, then the x-axis and the y-axis of both graphs *have to be equal*. For example, if the computation speeds of two algorithms are to be compared with each other and are plotted in two different graphs, the y-axis (showing the measured speed, because it is the dependent variable) of both graphs have to be equal to each other. In many software packages, you have to set the value ranges manually in order to achieve this. This is because most software packages will determine a best fit range for each plot separately, resulting in unequal value ranges between graphs.

Tips and Tricks

- For presentation of the results, you can choose from very many different forms - line graphs, bar graphs, circle diagrams, frequency diagrams, etcetera. Each kind of data is best represented in some particular form, but there is no general rule for what is best. There are some books that guide you into choosing the best representation, for example [355]. We recommend you not to make this into a separate study, but rather take some rules of thumb into consideration: in general, computer scientists tend to represent their results in line graphs, bar graphs (both with error bars to represent standard deviations) or box plots.

- Software-wise, you can choose from many different programs that will put your data into some graphical representation (as mentioned above). Maybe the most well-known is Microsoft Excel. Psychologists tend to use the statistical software package SPSS, which also allows you to run all kinds of statistical tests on your data. For the plotting, we advise to use GNUPLOT[1]. Like the LaTeX word processor, it takes some time to get the hang of it, but eventually it will give you so much flexibility that it is very rewarding.

7.8 Conclusions

You finish your report with a Section with conclusions. This Section is best structured by first briefly repeating the objectives and research questions, then a summary of the analysis of the results and close with a set of conclusions based on your findings. For the objectives and questions, you have to *explicitly* give answers and conclusions here.

Depending on the available space (in conference papers, the space is often limited by a max number of pages), you can spend a number of paragraphs with details on the model and the experiments. You can also close by explicitly mentioning what the *take-home message* of the report: from all, this is what your reader should take home with him/her.

Often, during the research you have many ideas for future work and extensions of the study. It is good to keep such a list on the side while you are doing your study – and include it in the report in this Section. It allows you to focus on one single issue within your current study, while it also gives other people to pick up your work and continue with it, based on the pointers for future work that you give here.

Tips and Tricks

- Always assume that your reader *only* reads the introduction and conclusions. If this convinces him/her, then (s)he will read the remainder of your report. It means that you have to write the introduction and conclusions as one single unit that the reader can understand without having read the rest of the report.

- Note that (besides the future work) the conclusions actually do not introduce anything that you have not mentioned before in your report. It basically repeats in other words the things that you have elaborated on earlier in your report.

7.9 Bibliography

During the study, you have read books, articles, papers, technical reports, Master and PhD theses, and others. With your report, you have to include a *bibliography* (or: references) in which you enumerate the references to the literature that you read. Throughout your report, you refer to (or: cite) the literature and in the end, you include the bibliography - exactly the same as we do in this book.

You do not necessarily must have read each item in the bibliography – it is not a reading list. The function of the bibliography is *to position your research within the context of related research*. This means that you may refer to books or articles that you deem related (based on

[1] http://www.gnuplot.info/

the title, abstract and/or introduction/conclusions), but have not read completely. Be aware not to exaggerate this: your aim should not be to generate the longest possible reference list.

You must be very precise concerning the *formatting* of this reference list. Scientific journals, conferences, etcetera have *standards* to which your reference list must comply. For your report, you need to stick with one such standard, depending on the scientific discipline that you are working in. For social sciences, this is the American Psychological Association (APA) style; for political sciences, this is the American Political Science Association (APSA) style; biologists often take the Council of Biology Editors (CBS) style; etcetera.

For computer science, the citation style as defined by the Institute of Electrical and Electronics Engineers (IEEE) style[2] is generally accepted as the way to format your references and reference list. Other organisations, e.g., the Association for Computing Machinery (ACM), that also publish journals and conference proceedings in computer science, have their own standards.

We advice you to adopt the IEEE stylefile as default, and possibly another style (e.g., ACM) if circumstances require so, i.e., if you are submitting your work to a journal or conference organised by the ACM.

Tips and Tricks

- Remember 'Cite-while-you-write': when you are writing your report, you must already have quite a big part ready of your bibliography. When you are then in the last stage of writing, you can pay attention to getting the references exactly right instead of going through your room or office looking for those papers that you read two months ago.

- It is very helpful to use a *reference manager* to keep your list of references organised. Examples of such reference managers that work with Microsoft Word and LaTeX are EndNote[3] and JabRef[4].

7.10 Summary

This Chapter explained how to write a report on your research study. The general structure of such a report is introduction, literature study, theory, experiments and conclusions. This Chapter has described each of these parts in detail. In the various Tips and Tricks throughout the Chapter, we also gave you some advice on specific software (e.g., word processors and graphic tools) to write up your report. Although some things may have been trivial for you, it is good to be aware of the ins and outs of report writing and stick to standards – this will improve the chances that your work is accepted (by supervisors or peer reviewers) significantly. Although a good outside can and should never compensate for a bad inside, it definitely contributes to the acceptance of a good inside.

[2] http://standards.ieee.org/guides/style/ > Section 19 Bibliography.
[3] http://www.endnote.com/
[4] http://jabref.sourceforge.net/

Part IV

APPENDICES

A Software

This book would not be complete with an overview of software that is related to the book contents. Here, this means an overview of agent simulation frameworks (or: toolkits, or: testbeds, or: environments). Recently, a wild growth emerged for such frameworks and there are currently dozens of frameworks around for agent-based simulation modelling.

Aims of this Appendix

We give an overview of a number of different agent simulation frameworks. Our aim is to allow you to make a good choice if you are faced with having to choose between a number of frameworks to implement some simulation model. We have explicitly not aimed for an exhaustive list – we selected specifically those frameworks that are either much used in scientific research and referenced in the literature, or relate specifically to studies that were described in this book. Some descriptions in the list contain detailed information about the learning curve, difficulties, advantages, disadvantages, etcetera. Other descriptions are more descriptive and only contain general information about the framework. Each description is accompanied by an internet link, thus it is always possible to find more detailed information yourself.

Introduction

Many agent simulation frameworks are currently around - various (survey) articles refer to dozens of such frameworks [133, 308, 347]. Various places on the internet[1] contain overview lists of these frameworks, including download and documentation links. Some of these frameworks are more generic agent toolkits (usually aimed towards computer scientists for engineering agent-based applications), some are specifically aimed at social simulation (usually aimed towards social scientists).

[1] http://www.econ.iastate.edu/tesfatsi/acecode.htm

The mentioned surveys present overviews of frameworks based on some evaluation criterion or in historical comparison with the development of other software packages. Serenko and Detlor [308] present an overview of 20 agent toolkits and evaluate the use of these toolkits in post-secondary sources. Gilbert and Bankes [133] give an overview of 10 agent-based modelling tools. It is interesting that they compare the development of these tools as observed over the last 10 years with the development of statistical software. Firstly there were implementations on mainframe computers; then general-purpose languages were developed; following were routine libraries to support purpose-built programs; finally resulting in the development of packages containing routine collections with a common standardised user interface. For social simulation, an extensive survey was performed by Tobias and Hofmann [347]. As a result of strict overall requirements, the pool in this survey was reduced from 21 general to only 4 fitting frameworks. The requirements included, among others, that the frameworks must be java-based and suited for social simulation. Still, the 4 frameworks are thoroughly evaluated. Together, the three surveys give a good overview of agent-based modelling frameworks.

A.1 Swarm

One of the first agent-based modelling libraries is the Swarm framework, originally developed at the Santa-Fe Institute. It is a set of code libraries to be used in agent-based simulation programs written in Objective-C or Java. The simulation comprises collections concurrently interacting agents. The use of Swarm is widespread, it is recognised by the scientific community and has a wide and strong user base; it has to be started from command line, and while a user interface can be created it is necessary to have quite some programming skills to use Swarm [347]. It supports a hierarchical modelling approach, where agents may consists of swarms of other agents in nested structures.

http://www.swarm.org/

A.2 RePast

Another widely used framework is RePast [256]. The RePast package was evaluated best in the survey by Tobias and Hofmann [347]. It performed well on the evaluation criteria, which were general (including license, documentation, support), support for modelling and experimentation (including support for modelling, simulation control, project organisation) and modelling options (including large number of complex agents, inter-agent communication, nesting of agents). RePast borrowed concepts from Swarm, but distinguishes itself along the way in that it has a number of pure implementations in several languages (java, .net and python) and built-in adaptive features (e.g., genetic algorithms and regression). It allows programmers to build simulation environments, build user interfaces and automatically collect simulation data.

http://repast.sourceforge.net/

A.3 NetLogo

NetLogo [36] is based on the StarLogo framework [285]. It is widely used in research and education across a wide range of disciplines (epidemiology, social sciences, material sciences,

earth sciences). The feature that distinguishes it from other frameworks is the HubNet, which is a technology that allows NetLogo to run distributed participatory simulations. This can be used, for example, in a classroom setting where students can run their own partial simulation and connect to other simulations. It also enables researchers to collaboratively explore a simulation. It has a community of thousands of users worldwide.

http://ccl.northwestern.edu/netlogo/

A.4 Newties

The Newties project is growing an artificial society using computer programming that develops agents–or adaptive, artificial beings–that have independent behaviours. The project is the first of its kind to develop a large-scale and highly complex computer-based society. The project's results may have larger implications for information technologies design, evolutionary computing systems, artificial intelligence and linguistics. The project is a significantly scaled-up experiment beyond any existing state-of-the-art social simulation. To study the highly complex beings and their behaviours, very large populations are developed supported by a distributed computing infrastructure and a shared, p2p platform. The large-scale system ensures that the environment and society is complex enough to allow significant behaviours and interactions to emerge.

http://www.new-ties.org/

A.5 Breve

Breve is a 3D simulation environment for multi-agent simulations and artificial life. It evolved out of a thesis project by Jon Klein at Hampshire College and was developed further into a Master's thesis at Chalmers University. Breve is under active development and intended for building large scale simulations of decentralised systems and evolutionary dynamics [194]. Breve is similar to existing agent packages such as Swarm or NetLogo with the additional the possibility of 3D simulations (note that this is under current development in NetLogo). Simulations in Breve are written in a procedural object-oriented programming language called 'steve'. The steve language is similar to programing language such as c or objective-c. Being relatively easy to use its aims at non-programmers but on the other hand it also tries to restrict experienced users as little as possible (therefore it is possible to interface with other languages, libraries and programs via steve). Breve is currently used to model a variety of phenomena (e.g. swarm movement in 3D space [320]). So far Breve is more used for educational purpose. The drawback of Breve is its buggy state. In general the system is stable, but it is likely to run into bugs and problems while programming with it.

http://www.spiderland.org/breve

A.6 Mason

MASON (Multi-Agent Simulator Of Neighborhoods) is a joint project between the ECLab Evolutionary Computation Laboratory and the Center for Social Complexity at the George Mason University. It is a java-based decentralised discrete-event simulation toolkit intendend for the simulation of many agents (10000+) [224]. Mason shares similar features with Swarm,

Netlogo and Breve. It consits of a small fast core which can be extended with 2D or 3D visualisation. Therefore models are independent from visualisation, which guarantees a fast simulation speed if needed. Further models are self contained and can run inside other Java frameworks and applications. Mason has been used in several scientific projects concerned with topics such as foraging behavior [224] and cooperative target observation [225].

http://cs.gmu.edu/ eclab/projects/mason/

A.7 Starlogo

StarLogo is a Logo clone which was developed by by Mitchel Resnick and other members of the MIT Media Lab [284, 286, 285, 57]. The Development started in 1994 using Macintosh (MacStarLogo) and in 1999 the first Java based version of StarLogo was released that could be used platform independent. StarLogo is intended as an educational language, which is why also the MIT Education Program was involved in the development. It gives students and teachers the opportunity to easily gain insights into many real-life phenomena such as traffic jams, ant colonies and so on. The main difference between StarLogo and the original Logo (which is also intended for an educational purpose) is that StarLogo aims at modelling decentralised systems - it allows you to control thousands of agents in and the grid the agents are active on in general. The StarLogo language is relatively easy to learn, and Starlogo is therefore well-suited for fast Artificial Life projects. The main drawback of StarLogo is that it is very resource intensive and also buggy. This is why it is not commonly used by the scientific community, in contrast a Logo clone discussed above NetLogo is heavily used for all kinds of scientific simulation.

http://education.mit.edu/starlogo/

A.8 FramSticks

FramSticks is a three-dimensional artificial life simulation project which was build in order to study the processes of evolution in a computer-simulated artificial world. It was initially developed by Maciej Komosinski and Szymon Ulatowski at the Poznan University of Technology [197, 198]. The framstick environment is populated by so called *stick creatures*. These creatures consist entirely of sticks interconnected with elastic joints. Creatures are treated as physical objects in the 3D world. They are exposed to all kinds of interactions such as for example static / dynamic friction and gravitation. Via the mechanism of evolution, creatures can evolve the ability of navigating through the environment. Selection favors creatures which can achieve fast navigation in a given environment. There are different types of environments possible in the FramStick simulations such as various kinds of surfaces and water. The whole framstick approach is similar to Karl Sims' simulation of virtual creatures [310]. To conclude it has also to be noted that the constraints within the FramStick environment limit the usage of the environment for simulations of self organising systems. Therefore FramSticks is more an artificial life project than an agent toolkit.

http://www.frams.alife.pl/

A.9 Ascape

Ascape is a Java API for simulating multi-agent systems on a single computer, which was developed by Miles Parker of the Brookings Institute in Washington, DC [271, 268]. It originates from the swarm research project and is one of many swarm like artificial life world models. Ascape is a very complex modelling language, which is hard to learn. This is due to the fact that it tries to be minimal in respect to resource usage and compact in respect to source-code, required for a simulation. This makes Ascape very compact and efficient, but also hard to learn, in comparison to, for example, NetLogo. Ascape is currently used for simulations in social sciences (see for example [18]).

http://www.brook.edu/es/dynamics/models/ascape/

A.10 CORMAS

Cormas is a programming environment dedicated to the creation of multi-agent systems, with a specificity in the domain of natural-resources management. It provides a framework for developing simulation models of coordination modes between individuals and groups that jointly exploit common resources. Cormas is a simulation platform based on the VisualWorks programming environment which allows the development of applications in SmallTalk.

http://cormas.cirad.fr/indexeng.htm

A.11 MOISE+

Moise+ is an organisational model for Multi-Agent Systems based on notions like roles, groups, and missions. It enables an MAS to have an explicit description of its organisation. Its predecessor MOISE had three main concepts: roles, role relations, and groups, which are be used to build, respectively, the individual, social, and collective structural levels of an organisation. MOISE+ extends this original structural dimension with concepts such as inheritance, compatibility, cardinality, and sub-groups. It has been used as the foundation of other simulation models, e.g., MOCA [6].

http://moise.sourceforge.net/

A.12 AgentSheets

(From the AgentSheets website:) AgentSheets is a revolutionary authoring tool that allows non-programmers to create agents with behaviors and missions, teach agents to react to information and process it in personalised ways, and combine agents to create sophisticated interactive simulations and models. Our customers use AgentSheets to create interactive games, virtual worlds, training simulations, information gathering and personalising agents, and other interactive content.

http://www.agentsheets.com/

A.13 LEADSTO

The software environment LEADSTO [39] was developed to model and simulate dynamic processes in terms of both qualitative and quantitative concepts. Underlying to the environment is the LEADSTO formal language, which is a declarative order-sorted temporal language, extended with quantitative notions like integer and real. The environment performs simulations of LEADSTO specifications, generates data-files containing traces of simulation for further analysis, and constructs visual representations of traces. Examples of processes that have been modelled using LEADSTO are human reasoning processes, eating regulation processes, conditioning processes, ant colonies, organisations (e.g., a factory), and component-based software systems.

http://www.few.vu.nl/~wai/TTL/

A.14 SDML

SDML ('strictly declarative modelling language') is a declarative MAS modelling language based on *autoepistemic logic*. SDML has object oriented features and was developed at the Centre for Policy Modelling at the Manchester Metropolitan University. It is intended for modelling multi-agent interaction within articulated social structures (e.g. organisations) [246]. The declarative nature of SDML is due to the fact that it allows to incorporate cognitive theories within agents. Being declarative, it enables the distinction between behavior and its underlying explanation. SDML is mainly used to model business strategies (e.g. [245]).

http://cfpm.org/sdml/

B
Projects

What are you going to do now with the knowledge that you gained from reading this book up till now? You might want to get your hands dirty and do just do the stuff that we described yourself. Or investigate something similar to what we have described. Or do your own thing - it is up to you.

Aims of this Appendix

Here we present a dozen different projects that you may want to engage in. The topics of the projects range from self organising behaviour shown by house sparrows and penguins up till epidemic modelling and small-world networks. Each project is introduced by a short description, possibly already with some concrete objectives and/or hypotheses. We give some first hints as what are the things you may consider modelling/implementing when you embark on the research. Finally, each project description is accompanied by approximately 5-10 references to scientific articles, papers and/or books. We expect the necessary time to carry out such a research project well (including literature review, modelling, implementation, experimentation and writing a report) to range from several weeks (as lab work for a course) to several months (as a final project on graduate level[1]).

[1]You may want to take a look at the following examples of Master's theses written at the Vrije Universiteit, Amsterdam, about similar projects: [47, 363, 45, 369, 175, 364, 32, 336, 321, 294]

B.1 House Sparrows

Description

The aim of the house sparrows project is to model the foraging behaviour of house sparrows. If a foodsource is spotted by a sparrow which is too big to eat it itself and dividable then the sparrows will call other sparrows by chirping. It has been argued that this behaviour has a positive effect on the sparrows energy bucket and also enables an optimal exploitation of the food source [28, 105].

Getting Started

- Use model parameter *predators*: Pretators might be useful to increase the environmental pressure on the sparrows (alternatively one could model a market place situation where the probability that sparrows have to fly away due to being interupted by people etc. increases the longer they are at a food source).
- Use model parameter *food*: Food should appear randomly on the grid and disappear with a probability (increasing by its availability).
- Use model parameter *sparrows*: Chirping behaviour, Fleeing behaviour, Feeding behaviour and reacting to other sparrows chirping. This should be easy to model in netlogo.
- Krink simulated the foraging behaviour of house sparrows. This paper can be used as a starting point. Besides that, the whole model can be seen as an extension of a pretator/prey model which has to be adapted (see Krink)

Resources

- [8, 9, 10, 26, 28, 105, 104, 137, 173, 174, 182, 217, 324]

B.2 Emperor Penguins

Description

The aim of the emperor penguins project is to model the huddling behaviour of male penguins which can be observed during incubation: After a female penguin lays an egg, it leaves the colony to hunt for 4 to 5 months. The egg is left with the male penguin which carries the egg the whole incubation time pressing it with its feed in a warm skin fold.

The male penguins are unable to hunt/eat during the incubation time. During the incubation time, they exhibit the so called huddle behaviour: male penguins cluster together in a large group that is in constant motion. The huddling behaviour minimises the exposure of individual penguins to the wind and enables the penguins to share body warmth [192].

Getting Started

- use model parameter *flow of wind*: the flow of wind should be modelled quite easily, since it is static and also described well in the literature.
- use model parameter *iciness experienced by each individual*: is a little tricky and can influence the behaviour of the agent (e.g the donkey behaviour described by van der Heide).
- use model parameter *individual behaviour*: should integrate the ability to initially cluster with other individuals and keeping the cluster in an huddle-like motion. The clustering should be easy to model for example using the heat bugs model as a starting point. The more challenging part is the huddling behaviour such that the cluster stays intact.
- The huddling behaviour has been simulated twice: by van der Heide and by Stead (see Resources for the papers). Stead used Starlogo for his simulation.
- You have to think about when a simulation considered to be successful - in van der Heide's case this is when the behaviour is observed. Stead also takes the energetic impact of huddling into account (i.e., how long a penguin could survive like that). Another thing of consideration would be the mean temperature of a penguin, the mean mass of a penguin cluster, and so forth.

Resources

- [192, 269, 363, 324, 283]

B.3 Honeybee Colony Migration

Description

The aim of the honeybee colony migration project is to model the migration of a honeybee colony. A colony migration usually takes place in spring when large colonies divide themselves for reproduction [306]. In such a case the old queen and around half of the workers leave the old hive and migrate to a new hive site, which has been determined by scouts before. The extraordinary thing about the migration is that that only 5% of the migrating bees (i.e. the scouts) knows the direction to the new nest site during the migration [97, 29]. Nevertheless a migration proceeds in an coordinated and optimal manner.

There are two hypotheses how a colony can accomplish a coordinated migration with only 5% of the migrating population being informed about the direction to the new nest site:

1. Scouts guide the migration by streaking through the swarm cloud in the direction of the goal. They thereby visually indicate the travel direction for the other colony-members (vision hypothesis).

2. Pheromones are released from the Scouts Nasanov glands at the front of the cloud of flying bees. This pheromones chemically indicate the travel direction for the other colony-members (olfaction hypothesis).

Getting Started

- Use model parameter *swarm size*: It might be interesting to see what effect the size of the swarm has on the migration behaviour, i.e., if small swarms migrate faster / better than large swarms.

- Use model parameter *type of migration*: It might be interesting to see what hypothesis leads to a more effective colony migration. How does a hybrid migration (vision and pheromones) behave?

- Start with implementing the second hypothesis, since it is easier to implement due to already existing models (see next point).

- The swarm behaviour during migration has already been modelled by Beekman et al. [30]. This model could be used as a starting point for creating your own agent based model of the migration process. There are also other models which might be useful as a starting-point for you simulation such as models of flocking behaviour in birds or fish schools [272], since they simulate swarm movement. Also models of foraging behaviour by ant colonies might be useful since such models incorporate the use of pheromones (hypothesis 2).

Resources

- [29, 30, 97, 306]

B.4 Brood Sorting in Ants

Description

The aim of the brood sorting project is to model the brood sorting behaviour which occurs in ant species such as *Leptothorax uni-fasciatus* [123, 307]. The sorting behaviour leads to concentric brood item arrangement around the nest center. It also leads to a different distribution of brood according to their age within the nest: young brood items (eggs and microlarvae) can be found arranged concentric around the nest center, while older brood (pupae and prepupae) is placed in a concentric manner in an intermediate area.

It is assumed that these patterns help to organise brood-care [123], since it enables a fast feeding of old brood, which is not necessary for relatively young brood. The mechanism causing the brood sorting phenomenon are still under investigations, but it has been suggested that level of carbon dioxide within the hive might play a role. The amount of CO_2 has its maximum in the center of the colony and concentric decreases. Ants are likely to have the ability to detect this gradient with their antenna and could therefore sort the brood according to the amount of CO_2.

Getting Started

- Brood-sorting models have already been implemented [234, 262], they might be a good starting point for your own model.
- The model has to incorporate two kinds of individuals: brood and workers. Besides that, there should be an additional differentiation within the brood group according to their age.
- A good starting point for implementation might be an already existing model (i.e. the Heat-bug model). The heat-bug model simulates heat diffusion (caused by agents) and evaporation over time. In a model concerned with brood-sorting, a similar behaviour can be observed looking at CO_2.

Resources

- [262, 307, 123, 210, 234]

B.5 The Evolution of Circadian Rhythms

Description

The aim of the circadian rhythms project is to investigate whether such rhythms can lead to activity patterns (niching [167]) within a prey world with a day/night cycle. Circadian rhythm are 24-hour cycles in the physiological processes of living beings, including plants, animals, fungi and cyanobacteria, which determine the sleeping and activity patterns of all animals, including human beings. These rhythms also affect brain wave activity, hormone production, cell regeneration and other biological activities [96, 385]. The circadian mechanism itself is encoded in the genotype of an individual [265]. It is assumed that it has also been a target of evolution which caused several changes during its evolution due to selection pressures in particular environmental niches [232]. It might therefore be interesting to investigate what effect circadian rhythms have on predator/prey models and if they lead to heterogeneous (in respect to activity) populations within such a model.

Getting Started

- A good starting point for the implementation might be a simple predator/prey simulations, which can be found in nearly any agent-system (e.g. NetLogo, Swarm, RePast). This model should then be extended with a day-light cycle. Individuals need a circadian rhythm encoded in their genotype (this can be easily done via a bitstring).
- Biological plausibility: have a look at the literature and make sure that the rhythms are implemented in a biological plausible way.
- Selection pressure: think about what kind of selection pressure might be useful in such a model, as an example if it is harder for predators to catch prey during the night this might lead to an evolution of nocturnal prey activity.

Resources

- [385, 265, 167, 96, 237]

B.6 Epidemic Modelling

Description

The goal of the epidemic modelling project is to develop an model for epidemic of a disease of your choice. The general goal of epidemics modelling is to understand and predict the dynamics of the epidemic under different parameter and/or initial conditions. Such information can than later be used to protect a population from a epidemic or to fight epidemics more effectively. To give an example, there have been several studies (e.g. [219, 218]) about the structure of sexual networks within the Western Society. These studies try to identify inflectors and the peer-groups of the inflectors within such a network. Information gained from such studies can be used to aim sexual education at the right peergroups and therefore reduce sexually transmitted infections. Besides the use of differential equation in modelling epidemics, there are also several models using cellular automata [312, 125] and agent based modelling in order to model epedemics [95, 342].

Getting Started

- A good starting point for the implementation might be a simple infector model. Such models exist within nearly any agent-system (e.g. NetLogo, Swarm, RePast), which should then be extended. You may also develop a cellular automata.
- Biological plausibility: have a look at the literature and make sure that the epidemic features of your disease are implemented in a bioological plausible way (How is the disease transmitted, incubation time, and so on).
- It is quite likely that if you model a common disease, there exists already a model. Have a look at these models.

Resources

- [342, 95, 312, 257, 241, 219, 218, 125]

B.7 Biological Pattern Formation

Description

Aim of the pattern formation project is to get familiar with pattern formation in biological systems and to model a pattern formation phenomenon in a biologically plausible way. In nature, pattern formation occurs in many distinct and fascinating contexts (e.g. seashells [231], or life in general [113, 346]). Studying pattern formation in systems, two levels have to be taken into account: on the macro level, features of recognised patterns should be parametrised, while on the micro-level, the local interactions processes that cause the pattern formation on the macro-level have to be identified. In recent years, cellular automata have become a popular means of studying pattern formation, alongside the partial differential equations, which are the classical approach.

Getting Started

- In order to get started you first have to find an interesting pattern formation phenomenon. You might try to reformulate a model which was proposed as a partial differential equation model to a CA (e.g. evolution zebra-stripes [143]).
- There are many CA models used the biological domain (see Ermentrout et al. [113] for a general overview of the usage of cellular automata in biological systems or Winfree *et al.* [386] for pattern formation in the DNA domain). You could start by reprogramming such an approach.

Resources

- [231, 346, 113, 298, 273, 228, 143]

B.8 Small-World Networks

Description

Aim of the small-world networks project is to analyse how such an structure can be introduced in an agent system and to derive a general concept of how to control these network properties. Another research question could be how such a network structure affects the agent system in return.
The small-world phenomenon describes a network in which each node can be reached from any other node using few intermediate nodes. Human social networks exhibit this phenomenon (e.g. the Bacon Number[2]). The first study of this phenomenon was initiated by Milgrim in 1960. He distributed several letters all over the US and people where asked to pass them on to people they knew until a letter reached the person it was addressed to. Milgrim found that the number of people the letters passed until they reached their final destination was surprisingly small[3]).

About ten years ago the topic gained in popularity again due to research by Watts and Strogatz [377, 328, 376]. The topic of small world networks is closely related to the theory of random boolean networks [223]. Today the small world phenomenon a popular topic [23, 43]. In particular, social science research and research concerned with cooperation [335] and communication (e.g. gossip, newscast protocols [179, 178]) within various systems is looking at the implications of the small world phenomenon.

Getting Started

- In order to study the network structure of an agent system you need some kind of interaction. Such a structure could arise via communication or via spreading a disease. You can therefore base your model on models already existing that exhibit such features.
- A useful tool to analyse a system for the small-world phenomenon is the average path length between the nodes (agents), and the clustering coefficient which describes how connected the neighboring nodes of a node are.
- Try to find the general properties that are necessary for the phenomenon. There are many parameters which might have an effect. For example, for the case of communication, this can be the communication protocol used.

Resources

- [23, 43, 377, 328, 376, 335]

[2] http://en.wikipedia.org/wiki/Bacon_number
[3] For more information on the experiment see http://cnx.org/content/m10833/2.2/.

B.9 The Art-Gallery Problem

Description

Aim of the art-gallery project is to create a multi-agent system that is able to observe a number of moving objects using a minimum number of agents. The art gallery problem (also known as the museum problem, prison yard problem) is a visibility problem which originates from computational geometry [126]: An art gallery represented by a polygon P is observed by a set S of security cameras, represented by points within the polygon. The set S of points is said to secure the polygon if for every point x in the polygon, there is some p in S such that the line segment between x and p does not leave the polygon.

A harder version of this problem is a topic in robotics [267, 268, 221, 224]: in the *moving art-gallery problem* not the gallery itself but moving objects within the gallery have to be observed. The movement of the object is not known in advance and the aim of a system is to observe the objects accurate with a minimum expense of resources (i.e. robots).

Getting Started

- The main focus in this project is on observing moving targets. Therefore the environment in which the observation takes place can be disregarded (you can assume a plain surface on which the agents move).
- Two types of agents are needed observer-agents and target-agents. The target-agents should exhibit dynamical movement patterns which can change, in order to test the quality of the observer-agents.
- Vision plays a crucial role in observing moving targets. Which effects have variations in the range of an agents vision and an agents field of perception on the performance of your system.
- The size of area has also an impact on the performance of your agents. In a small area with a high density of target-robots it might be easy for a relatively small group of observing robots to fulfill their task.

Resources

- [267, 268, 221, 224]

B.10 Multi-Asset Surveillance

Description

The aim of this project is to create a multi agent system that is able to observe a given environment using a minimal amount of agents. At first glance, the multi-asset surveillance project seems similar to the art-gallery project. Nevertheless there exist crucial differences between the two problems. While the central goal of the moving art-gallery project is to observe a given number of moving objects with a minimum expense of resources, in this problem the goal is to explore an unknown area with a minimum expense of resources as quickly as possible. Therefore this problem focuses on exploration and therefore distributing the given resources as optimal as possible so that they can explore the environment as distributed as possible.

The multi-asset surveillance problem has been tackled by research in robotics [195]. It is especially interesting for organisations within the aerospace industry (i.e. NASA), since such techniques could be used to explore unknown planets with unmanned vehicles or aircrafts [397, 396].

Getting Started

- The environment plays a central role in this problem, Your system should be able to perform in all kinds of different environments. You should also pay attention to environmental setups.
- Vision plays a crucial role. Which effects have variations in the range of an agents vision and an agents field of perception on the performance of your system (i.e. in exploring an environment)?
- The size of area has also an impact on the performance of your agents: larger environments might be harder to explore.
- Obstacles which are placed in the environment and obscure the view on the environment behind also have a significant impact on your system.
- A factor which determines how good your system performs is the time it takes a system to explore the whole area. It might also be interesting to look at the evolution of the area explored over time.

Resources

- [195]

B.11 Cellular Chitchats

Description

The aim of this project is to model a communication network within a cell from a seedling. When a seedling receives light, it will grow. In darkness it will stay small. Light is a signal for the cells of a plant to make more tissue, by which the plant can grow.

Cells receive signals from their environment, by which the cell changes its behaviour. Inside the cell numerous molecules can be found, of which many proteins. In complex networks these proteins can *communicate* with each other. When a signal, called a ligand, arrives at the surface of a cell, it is recognised by the extracellular part of a receiving protein, a receptor. Upon this activation, the intracellular part of the receptor activates proteins inside the cell. These molecules activate or inhibit other molecules again, and so, a cascade of molecules gives the signal through towards the nucleus. Such a communication chain is called a *signal transduction pathway*. All the functions of a cell (e.g., normal regulation, cell death, mitosis) are precisely regulated by a highly specific signal transduction pathway.

Getting Started

- Use model parameter signal: either make a cycle, like day and night, or a switch to set it on or off.
- Use model parameter cryptochromes. Behaviour when free, behaviour when bound.
- Use model parameter COP1. Behaviour when free, behaviour when bound.
- Use model parameter transcription factor HY5. Behaviour when free, behaviour when bound
- Use model parameter gene, which can be activated by the transcription factor. Show the amount of genes, i.e. this decides how much the plant will grow.
- Use glossaries online, like the ncbiotech glossary[4].

Resources

- [60, 5, 132, 374, 169]

[4] http://www.ncbiotech.org/resource_center/glossary/

B.12 Puckscape

Description

The aim of the Puckscape project is to extend an existing simulation concerned with adaptive task allocation, which was developed at the VU. Adaptive task allocation is the ability of a system to dynamically adjust labor to the current workload (e.g. the division of labour in insect societies).

Puckscape is based on the work of Jones et al. [183] who introduced an adaptive task allocation mechanism for large-scale minimalist multi-robot systems (LMMS). Adaptive task allocation was studied in the context of a foraging task: The robots had to navigate through an area, collecting a number of pucks which were scattered throughout their environment. There were two different types of pucks Red and Green. Robots had two foraging states corresponding to the two puck types they searched for and collected red pucks if they were in state Red, and green pucks if they were in state Green. The aim was to create a robot controller which enabled the robot-collective to dynamically adjust the division of labor according to the amount of each puck type present.

Jones et al. [183] were able to develop a heuristic which enabled the robots to exhibit adaptive task allocation. This heuristic was explicitly programmed into the system, i.e. handmade. Puckscape tackles the same problem; however, the intention is to find out whether adaptive task allocation can occur as a result of evolution within an agent population. Several algorithms involving evolution at the level of individual agents were tried, but the results were not satisfactory. This leads to the question of whether evolution at the population level might overcome the problems found with individual-centred evolution. Genetic programming has been successfully applied to the evolution of agent controllers (see Luke et al. [226]). It might therefore be a good idea to find out if genetic programming can be used to evolve a transition function within the Puckscape system.

Getting Started

- Puckscape was written in NetLogo. The simulation software is available from the course co-ordinator and should be a good starting point. Additionally, there are two working reports which provide more detailed information and experimental results.
- A genetic programming framework already exists in NetLogo[5].
- One of the main problems will be to ensure that the transition function outputs a probability value (i.e. between 0 and 1). The normalisation involved might not be trivial! The literature available in the area of genetic programming should be a useful reference.
- It may also be interesting to create a genetic programming framework that is extendable to an increased number of tasks.

Resources

- [226, 215, 216, 183, 313]

[5] http://www.cs.northwestern.edu/~fjs750/netlogo/final/index.html

B.13 Artifical Societies - Poisonous Food

Description

The aim of this challenge is to let the agents learn to distinguish between good and bad food. There are two types of food: nutritious food and poisonous food. The weight of poisonous food is such that to pick it up will cost more than the energy it will give. With a suitable adaptive algorithm, it is expected that the agent will learn to do a *test* action before a pick-up action.

A related study by Parisi and Nolfi [266] on evolving neural networks involved a similar kind of world. The artificial world concerned individuals that were neural networks with the task to distinguish between nutricious and poisonous virtual mushrooms.

Another related study [32] looked at a Sugarscape-like environment where the agents were partly neural networks and also had the task to distinguish between good and bad food. An interesting outcome of this study was that the neural network could not be trained sufficiently long enough within the lifespan of an agent in order to learn about the distinction task.

Getting Started

- This challenge is defined in the default NEWTIES world (see Software). You can download this world by downloading the NEWTIES simulation software.
- A disadvantage of this type of poisonous is that it (the poison) keeps working over time. Because the poisonous is implemented through weight, it will poison the agent every time the agent walks. This will maybe prevent the agent from walking. Another side-effect is that the plant is less poisonous, if it is immediately eaten.

Resources

- [252, 254, 266, 32]

B.14 Artifical Societies - Path Following

Description

The aim of this challenge is to let the agent learn to navigate through rough terrain. Two big patches of food are placed in the world and between those patches there is a rough terrain. The roughness and size of the terrain should be made such that agents have to recognise the road, otherwise they will die before reaching the other patch. In this rough terrain there are some 'roads', terrain that is less rough, over which the agents can walk.

Getting Started

- This challenge is defined in the default NEWTIES world (see Software). You can download this world by downloading the NEWTIES simulation software.

B.15 Artifical Societies - Development of Norms

Description

The aim of the challenge is the dynamic development of norms about ownership. Agents can choose between public ownership and private ownership norms of food: If a hunt (search for food) was successful, they can claim the food for themselves, or immediately share it with other agents. Conversely, if they were not successful, they can request others to share their food, or accept their private property rights to the food. As a sanctioning mechanism, agents can attack each other - either to defend their private property, or enforce their rights to publicly owned food on other agents. It is expected that in case there is plenty of food, there is no need to share food or request food and the population is dominated by agents that do not share food and are not communicative. Opposite when food is scarce, the population of agents will be dominated by agents that share food and request food. The amount of food available is fluctuating over time.

The research question is whether evolutionary learning can keep up with changes in the environment and change the distribution of different types of agents appropriately.

Getting Started

- This challenge is defined in the default NEWTIES world (see Software). You can download this world by downloading the NEWTIES simulation software.
- The amount of food and agents is variable during a simulation. Probably there will be periods in which food is scarce.
- Agents implement different strategies. We distinguish four types of strategies along two dimensions: normal (shares food, demands food), altruist (shares food, does not demand food), egoist (does not share food, demands food), and hermit (does not share food, does not demand food).
- There are two genes in this experiments: share_gene and demand_gene.
- The evolutionary parameters work as follows. For recombination, average the values of the parents; for mutation, use Guassuan noise.
- You can use the following monitors: agent population size, average value of the share food gene over the population of agents, average value of the demand food gene over the population of agents, average energy of the population of agents, and plant population size.

Bibliography

[1] Herve Abdi. A neural network primer. *Journal of Biological Systems*, 2(3):247–283, 1994.

[2] Christoph Adami. *Introduction to Artificial Life*. Springer, 1999.

[3] George Ainslie. *Breakdown of Will*. Cambridge University Press, 2001.

[4] A. Al, A.E. Eiben, and D. Vermeulen. An experimental comparison of tax systems in sugarscape. In S.-H. Cheng and X. Yao, editors, *Proceedings of the The First International Workshop on Computational Intelligence in Economics and Finance*, 2000.

[5] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, 2002.

[6] Matthieu Amiguet, Jean-Pierre Mueller, Jos?-A. Baez-Barranco, and Adina Nagy. The moca platform: Simulating the dynamics of social networks. In Jaime Simao Sichman, Francios Bousquet, and Paul Davidsson, editors, *Proceedings of the 2nd international workshop on multi-agent-based simulation*, volume 2581 of *Lecture Notes in Computer Science*, 2002.

[7] H.M. Amman, D.A. Kendrick, J. Rust, Michael Intriligator, and Kenneth Arrow, editors. *Handbook of Computational Economics, Volume 1*. Elsevier, 1996.

[8] Peter Andras and John Lazarus. Teamwork: Multi-disciplinary perspective. In Natalie Gold, editor, *Cooperation, Risk and Evolution of Teamwork*, pages 56–77. Palgrave Macmillan, 2005.

[9] Peter Andras, John Lazarus, Gilbert Roberts, and Steven J Lynden. Uncertainty and cooperation: Analytical results and a simulated agent society. *Journal of Artificial Societies and Social Simulation*, 9(1), 2006.

[10] Peter Andras, Gilbert Roberts, and John Lazarus. Environmental risk, cooperation and communication complexity. In E Alonso, D Kudenko, and D Kazakov, editors, *Adaptive Agents and Multi-Agent Systems*, pages 49–65, Berlin, 2003. Springer-Verlag.

[11] Mauro Annunziato and Piero Pierucci. The emergence of social learning in artificial societies. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Applications of Evolutionary Computing: EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM*, volume 2611 of *Lecture Notes in Computer Science*, pages 467–478. Springer, 2003.

[12] Alexander Artikis and Jeremy Pitt. A formal model of open agent societies. In Elisabeth Andre, Sandip Sen, Claude Frasson, and Jorg Mueller, editors, *Proceedings of the fifth international conference on Autonomous agents*, pages 192–193, 2001.

[13] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 1985.

[14] Robert Axelrod. *The Complexity of Cooperation: Agent-based models of competition and collaboration*. Princeton University Press, 1997.

[15] Robert Axelrod. Advancing the art simulation in the social sciences. *Japanese Journal for Management and Information System*, 12(3):16–22, 2003. Special Issue on Agent-Based Modeling.

[16] Robert Axelrod and Ross A Hammond. The evolution of ethnocentric behavior. In *Midwest Political Sciences Convention*, Chicago, IL, 2003. Revised Version.

[17] Robert Axelrod and Leigh Tesfatsion. A guide for newcomers to agent-based modeling in the social sciences. In *Handbook of Computational Economics, Volume 2: Agent-Based Computational Economics*. Elsevier/North-Holland, Amsterdam, The Netherlands, 2006.

[18] Robert Axtell. The emergence of firms in a population of agents: Local increasing returns, unstable nash equilibria, and power law size distributions. Technical report, CSED, 1999.

[19] Robert Axtell. Why agents? on the varied mativation fo agent computing in the social sciences. Technical Report CSED Working Paper No. 17, Brooking Institute, 2000.

[20] Robert L. Axtell, Joshua M. Epstein, Jeffrey S. Dean, George J. Gumerman, Alan C. Swedlund, Jason Harburger, Shubha Chakravarty, Ross Hammond, Jon Parker, and Miles Parker. Population growth and collapse in a multiagent model of the kayenta anasazi in long house valley. *Proceedings of the National Academy of Sciences of the United States of America*, 99(3):7275–7279, 2002.

[21] Ozalp Babaoglu, Mrk Jelasity, Alberto Montresor, Christof Fetzer, Stefano Leonardi, Aad van Moorsel, and Maarten van Steen, editors. *Self-star Properties in Complex Information Systems: Conceptual and Practical Foundations*, volume 3460 of *Lecture Notes in Computer Science*. Springer, 2005.

[22] T. Balch and R. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[23] Albert-Laszlo Barabasi. *Linked: How everything is connected to everything else and what it means for business, science and everyday life*. Penguin Books, 2003.

[24] Claudio Bartolini, Chris Preist, and Nicholas R. Jennings. A software framework for automated negotiation. In Ricardo Choren, Alessandro Garcia, Carlos Lucena, and Alexander Romanovsky, editors, *Proceedings of Software Engineering for Large-Scale Multi-Agent Systems 2004*, volume 3390 of *Lecture Notes in Computer Science*, pages 213–235. Springer, 2004.

[25] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation: the new experimentalism*. Springer-Verlag, 2006.

[26] Melissa Bateson and Alex Kacelnik. Starlings' preferences for predictable and unpredictable delays to food. *Animal Behaviour*, 53(6):1129–1142, 1997.

[27] Michael Batty. *Cities and Complexity: Understanding cities with cellular automata, agent-based models and fractals*. The MIT Press, 2005.

[28] Guy Beauchamp. Does group foraging promote efficient exploitation of resources? *Oikos*, 111(2):403–407, 2005.

[29] M. Beekman, R. L. Fathke, and T. D. Seeley. How does an informed minority of scouts guide a honeybee swarm as it flies to its new home? *Animal behaviour*, 71:161–171, 2006.

[30] M. Beekman, S. Jason, and M. Middendorf. Honeybee swarms: how do scouts guide a swarm of uninformed bees? *Animal behaviour*, 70:349–358, 2005.

[31] G. Beni and J. Wang. Self organizing sensory systems. In J.T. Tou and J.G. Balchen, editors, *Highly Redundant Sensing in Robotic Systems: Proceedings of NATO Advanced Workshop on Highly Redundant Sensing in Robotic Systems*, pages 251–262. Springer-Verlag, 1990.

[32] Joost Berculo. Neural netscape: A neural network agent society that optimises for its environment. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2006.

[33] D. Bertolini, P. Busetta, A. Molani, M. Nori, and A. Perini. Designing peer-to-peer applications: an agent-oriented approach, 2002.

[34] Samual N. Beshers and Jennifer H. Fewell. Model of division of labor in social insects. *Annual Review of Entomology*, 46:413–440, 2001.

[35] L. Blaxter, C. Hughes, and M. Tight. *How to Research*. Open University Press, 1996.

[36] P. Blikstein, D. Abrahamson, and U. Wilensky. Netlogo: Where we are, where we're going. In *Proceedings of the Annual meeting of Interaction Design and Children*, 2005.

[37] E. Bonabeau and Guy Theraulaz. Swarm smarts. *Scientif American*, March:72–79, 2000.

[38] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence : From Natural to Artificial Systems*. Oxford University Press, 1999.

[39] T. Bosse, C.M. Jonker, L. van der Meij, , and J. Treur. A language and environment for analysis of dynamics by simulation. *International Journal of Artificial Intelligence Tools*, 2007. To appear.

[40] T. Bosse, C.M. Jonker, L. van der Meij, and J. Treur. Leadsto: a language and environment for analysis of dynamics by simulation. In T. Eymann, F. Kluegl, W. Lamersdorf, M. Klusch, and M.N. Huhns, editors, *Proceedings of the Third German Conference on Multi-Agent System Technologies*, volume 3550 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2005.

[41] C. Boutilier, R. Das, J. Kephart, G. Tesauro, and W. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In Christopher Meek and Uffe Kjaerulff, editors, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003.

[42] A. Braun, S.R. Musse, L.P.L. de Oliveira, and B.E.J. Bodmann. Modeling individual behaviors in crowd simulation. In *16th International Conference on Computer Animation and Social Agents*, pages 143 – 148, 2003.

[43] Mark Buchanan. *NEXUS: Small worlds and the groundbreaking theory of networks.* W.W. Northon and Company, 2002.

[44] Seth Bullock. *Evolutionary simulation models: On their character and application to problems concerning the evolution of natural signalling systems.* PhD thesis, University of Essex, 1997.

[45] Tamas Buresch. Evolutionary versus lifetime learning of agent's controllers on a grid-world. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2004.

[46] Martin V. Butz. *Rule-Based Evolutionary Online Learning Systems.* Springer, 2006.

[47] Pieter Buzing. Vuscape: Communication and cooperation in evolving artificial societies. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2003.

[48] Paul Callahan. What is the game of life? On the Math.com website, 2000.

[49] S. Camazine, J.L. Deneubourg, N. Franks, J. Sneyd, E. Bonabeau, and G. Theraulaz. *Self-Organization in Biological Systems.* Princeton University Press, 2001.

[50] Mike Campos, Eric Bonabeau, Guy Theraulaz, and JeanLuis Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2):83–92, 2001.

[51] G. Di Caro. *Ant colony optimization and its application to adaptive routing in telecommunication networks.* PhD thesis, Universit? Libre de Bruxelles, 2004.

[52] I.D. Chase, C. Bartolomeo, and L.A. Dugatkin. Aggressive interactions and inter-contest interval: How long do winners keep winning? *Animal Behaviour*, 48:393–400, 1994.

[53] Edmund Chattoe. Just how (un)realistic are evolutionary algorithms as representations of social processes. *Journal of Artificial Societies and Social Simulation*, 1(3), 1998.

[54] Kan Chen, Elena G. Irwin, Ciriyam Jayaprakash, and Keith Warren2. The emergence of racial segregation in an agent-based model of residential location: The role of competing preferences. *Computational & Mathematical Organization Theory*, 11(4):333–338, 2005.

[55] Vincent A. Cicirello and Stephen A. Smith. Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-Agent Sytems*, 8:237?–266, 2004.

[56] A. Clark and D. Chalmers. The extended mind. *Analysis*, 58:7–19, 1998.

[57] Vanessa Stevens Colella, Eric Klopfer, and Mitchel Resnick. *Adventures in Modeling: exploring complex, dynamic systems with starlogo.* Teachers College Press, 2001.

[58] M. Collett, T. S. Collett, S. Bisch, and R. Wehner. Local and global vectors in desert ant navigation. *Nature*, 394:269–272, 1998.

[59] Rosaria Conte, Nigel Gilbert, and Jaime Simao Sichman. Mas and social simulation: A suitable commitment. In *Multi-Agent Systems and Agent Simulation*, volume 1534 of *Lecture Notes in Computer Science*, pages 1–9, Berlin, 1998. Springer.

[60] G.M. Cooper. *The Cell, A Molecular Approach*, chapter Cell Signaling. Sinauer Associates, 2000.

[61] José C. Cunha and Pedro D. Medeiros, editors. *Euro-Par 2005, Parallel Processing, 11th International Euro-Par Conference, Lisbon, Portugal, August 30 - September 2, 2005, Proceedings*, volume 3648 of *Lecture Notes in Computer Science*. Springer, 2005.

[62] Arianna Dal-Forno and Ugo Merlone. A multi-agent simulation platform for modeling perfectly rational and bounded rational agents in organizations. *Journal of Artificial Societies and Social Simulation*, 5(2), 2002.

[63] R.I. Damper. Emergence and levels of abstraction. *International Journal of systems Science*, 31(7):811–818, 2000.

[64] Charles Darwin. *The Origin of Species*. John Murray, 1859.

[65] R. K. Dash, D. C. Parkes, and N. R. Jennings. Computational mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[66] Paul Davidsson. Categories of artificial societies. In Andrea Omicini, Paolo Petta, and Robert Tolksdorf, editors, *Engineering Societies in the Agents World II*, volume 2203 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.

[67] Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1976.

[68] Christian W. Dawson. *The essence of computing projects: a student's guide*. Prentice Hall, 2000.

[69] M. J. L. de Hoon, S. Imoto, and S. Miyano. Open source clustering software. *Genome Informatics Series*, 13:250–251, 2002.

[70] Kenneth A. de Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Computer and Communication Sciences, University of Michigan, 1975.

[71] Tom de Wolf and Tom Holvoet. Emergence and self-organisation: A statement of similarities and differences. In S. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, and R. Nagpal, editors, *Proceedings of the International Workshop on Engineering Self-Organising Applications*, pages 96–110, 2004.

[72] Tom de Wolf and Tom Holvoet. Emergence versus self-organisation: Different concepts but promising when combined. In Sven Brueckner, Giovanna Di Marzo Serugendo, Anthony Karageorgos, and Radhika Nagpal, editors, *Proceedings of the workshop on Engineerings Self Organising Applications*, volume 3464 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.

[73] J. S. Dean, A. J. Jr. Lindsay, and W. J. Robinson. Prehistoric settlement in long house valley, northeastern arizona. In *Investigations of the Southwestern Anthropological Research Group: An Experiment in Archaelogical Cooperation*, 1978.

[74] J. L. Deneubourg, s. Gross, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ant and ant-like robot. In *Proceedings First European Conference on simulation of Adaptive Behavior: From Animals to animats*, 1991.

[75] Daniel C. Dennett. *Brainstorms: Philosophical essays on mind and psychology.* The MIT Press, 1978.

[76] Daniel C. Dennett. *Elbow Room: The varieties of free will worth wanting.* The MIT Press, 1984.

[77] Daniel C. Dennett. *The Intentional Stance.* The MIT Press, 1987.

[78] Daniel C. Dennett. *Consciousness Explained.* Little, Brown and Company, 1991.

[79] Daniel C. Dennett. *Darwin's Dangeous Idea: Evolution and the meanings of life.* Simon and Schuster, 1995.

[80] Daniel C. Dennett. *Kinds of Minds: The origins of consciousness.* Weidenfeld and Nicolson, 1996.

[81] Daniel C. Dennett. *Brainchildren: Essays on designing minds.* The MIT Press, 1998.

[82] Daniel C. Dennett. *Freedom Evolves.* Penguin Group, 2003.

[83] Jean Louis Dessalles, Serge Galam, and Denis Phan. Emergence in multi-agent systems, part ii: Axtell, epstein and young's revisited, 2006. Work in progress. http://perso.univ-rennes1.fr/denis.phan/worksInProgress/dgp2006.pdf.

[84] J.L. Dessalles and D. Phan. Emergence in multi-agent systems: cognitive hierarchy, detection, and complexity reduction part i: methodological issues. In Mathieu, Beaufils, and Brandouy, editors, *AE2005 : A Symposium in Agent-based Computational Methods in Finance, Game Theory and their applications*, volume 564 of *Lecture Notes in Economics and Mathematical Systems*, pages 147–161, 2005.

[85] Jared M. Diamond. Archaeology: Life with the artificial anasazi. *Nature*, 419:567–569, 2002.

[86] Virginia Dignum, Frank Dignum, Vasto Furtado, Adriano Melo, and Liz Sonenberg. Towards a simulation tool for evalutating dynamic reorganization fo agents societies. In *Proceedings of the Workshop on Socially Inspired Computing at AISB Convention*, 2005.

[87] E.A. DiPaolo, J. Noble, and S. Bullock. Simulation models as opaque thought experiments. In M. A. Bedau, J. S. McCaskill, N. Packard, and S.Rasmussen, editors, *Proceedings of the Seventh International Conference on Artificial Life*, pages 497–506, 2000.

[88] Kevin Dooley. Simulation research methods. In Joel A C Baum, editor, *Companion to Organizations*, pages 829–848. Blackwell, London, 2002.

[89] M. Dorigo and G. Di Caro. Ants colonies for adaptive routing in packet-switched communication networks. In A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 673–682, 1998.

[90] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(1):29–41, 1996.

[91] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L. M. Gambardella. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2-3):223–245, 2004.

[92] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[93] C. Drews. The concept and definition of dominance in animal behaviour. *Behaviour*, 125:283–313, 1993.

[94] L. Dugatkin and M. Druen. The social implications of winner and loser effects. *Proceedings of the Royal Society B: Biological Sciences*, 271(0):S488–S489, 2004.

[95] Jill Bigley Dunham. An agent-based spatially explicit epidemiological model in mason. *Journal of Artificial Societies and Social Simulation*, 9(1), 2005.

[96] Jay C. Dunlap, Jennifer J. Loros, and Patricia J. DeCoursey. *Chronobiology: Biological Timekeeping*. Sinauer, 2003.

[97] F.C. Dyer and T.D. Seeley. Colony migration in the tropical honey bee apis dorsata f. (hymenoptera: Apidae). *Insectes Sociaux*, 41:129–140, 1994.

[98] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[99] R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceeding of IEEE International Conference on Neural Networks*, volume 4, 1995.

[100] Russell Eberhart, Yuhui Shi, and James Kennedy. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[101] A.E. Eiben, D. Elia, and J.I. van Hemert. Population dynamics and emerging features in aegis. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1257–1264. Morgan Kaufmann, 1999.

[102] A.E. Eiben and Mark Jelasity. A critical note on experimental research methodology in ec. In *Proceedings of the 2002 Congress on Evolutionary Computations*, pages 582–587. IEEE Press, 2002.

[103] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[104] Mark A. Elgar. The establishment of foraging flocks in house sparrows: risk of predation and daily temperature. *Behavioral Ecology and Sociobiology*, 19:433–438, 1986.

[105] Mark A Elgar. House sparrows establish foraging flocks by giving chirrup calls if the resources are divisible. *Animal Behaviour*, 34(1):169–174, 1986.

[106] Euel Elliott and L. Douglas Kiel. Exploring cooperation and competition using agent-based modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 99(3):7193–7194, 2002.

[107] Lee Ellis. *Social Stratification and Socioeconomic Inequality: Volume 2: Reproductive and Interpersonal Aspects of Dominance and Status*. Praeger Publishers, 1994.

[108] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[109] Joshua M. Epstein. Agent-based computational models and generative social science. *Complexity*, 4:41–60, 1999.

[110] Joshua M Epstein. *Generative Social Sciences*. Princeton University Press, 2006.

[111] Joshua M. Epstein and Robert Axtell. *Growing Artificial Societies: social science from the bottom up*. The MIT Press, 1996.

[112] Joshua M. Epstein, John D. Steinbruner, and Miles T. Parker. Modeling civil violence: An agent-based computational approach. Technical Report 20, Brookings Institute, 2001.

[113] G. B. Ermentrout and L. Edelstein-Keshet. Cellular automata approaches to biological modeling. *Journal of theoretical Biology*, pages 97–133, 1993.

[114] I. Farkas, D. Helbing, and T. Vicsek. Mexican waves in an excitable medium. *Nature*, 419:131–132, 2002.

[115] I. Farkas, D. Helbing, and T. Vicsek. Human waves in stadiums. *Physica A*, 330(1-2):18–24, 2003.

[116] B.G. Farley and W.A. Clark. Simulation of self-organising systems by digital computer. *Institute of Radio Engineers Transactions on Information Theory*, 4:76–84, 1954.

[117] Maria Fasli. Formal systems $\wedge$ agent-based social simulation $= \perp$? *Journal of Artificial Societies and Social Simulation*, 7(4), 2004.

[118] Gary William Flake. *The Computational Beauty of Nature: Computer explorations of fractals, chaos, complex systems and adaptation*. The MIT Press, 2000.

[119] Jay W. Forrester. *Industrial Dynamics*. MIT Press, 1961.

[120] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure - Application Tuning and Adaptation*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2 edition, 2004. ISBN: 1-55860-933-4.

[121] Ian Foster. What is the grid? a three point checklist, 2002. http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.

[122] Robert H. Frank. *Passions within Reason: The Stretegic role of the emotions*. W.W. Norton and Company, 1988.

[123] N. R. Franks and A. B. Sendova-Franks. Brood sorting by ants: distributing the workload over the work-surface. *Behavioral Ecology and Sociobiology*, 30:109–123, 1991.

[124] Nigel R. Franks and Tom Richardson. Teaching in tandem-running ants. *Nature*, 439:153–153, 2006.

[125] Shih Ching Fu and George Milne. Epidemic modelling using cellular automata. In *Proc. of the Australian Conference on Artificial Life, 2003*, 2003.

[126] Z. Fueredi and D. J. Kleitman. The prion yard problem. *Combinatorica*, 14:287–300, 1994.

[127] Richard M. Fujimoto. *Parallel and Distributed Simulation Systems*. Wiley, 2000.

[128] Dominique Gati, editor. *Autonomic Networks*. ISTE Publishing Company, 2007.

[129] J. Gautris, G. Theraulaz, J.L. Deneubourg, and C. Anderson. Emergent polyethism as a consequence of increased colony size in insect societies. *Journal of theoretical Biology*, 215(3):363–373, 2002.

[130] Richard J. Gaylord and Kazume Nishidate. *Modeling Nature: Cellular Automata Simulations With Mathematica*. Springer, 1996.

[131] C. Gershenson. Introduction to random boolean networks. In M. Bedau, P. Husbands, T. Hutton, S. Kumar, and H. Suzuki, editors, *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, pages 160–173, 2004.

[132] J. Gewolb. How seedlings see the light. *Science AAAS*, August:1–1, 2001.

[133] Nigel Gilbert and Steven Bankes. Platforms and methods for agent-based modeling. *Proceedings of the National Academy of the United States of America*, 99(3):7197–7198, 2002.

[134] Nigel Gilbert and Rosario Conte. *Artificial Societies: the computer simulation of social life*. UCL Press, 1995.

[135] Nigel Gilbert and Pietro Terna. How to build and use agent-based models in social sciences. *Mind & Society*, 1(1):57–72, 2000.

[136] Nigel Gilbert and Klaus G. Troitzsch. *Simulation for the Social Scientist*. Open University Press, second edition, 2005.

[137] L-A. Giraldeau and G. Beauchamp. Food exploitation: searching for the optimal joining policy. *TREE*, 14:102–106, 1999.

[138] Norbert Glaser and Philippe Morignot. The reorganization of societies of autonomous agents. In *Proceedings of the Eight European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, 1997.

[139] Robert L Goldstone and Marco A Janssen. Computational models of collective bahavior. *Trends in Cognitive Science*, 9(9):242–430, 2005.

[140] D. M. Gordon. *Ants at Work: how an insect society is organized*. W. W. Norton, 1999.

[141] L. S. Gottfredson. Intelligence and social policy. *Intelligence*, 24(1):Special Issue, 1997.

[142] S. Grand and D. Cliff. Creatures: Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1(1):39–57, 1998.

[143] C. P. Gravan and R. Lahoz-Beltra. Evolving morphogenetic fields in the zebra skin pattern based on turing?s morphogen hypothesis. *Int. J. Appl. Math. Comput. Sci.*, 14:351–361, 2004.

[144] David Hales. Stereotyping, groups and cultural evolution. In J.S. Sichman, R. Conte, and N. Gilbert, editors, *Multi-Agent Systems and Agent-Based Simulation*, volume 1534 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1998.

[145] David Hales. Cooperation without memory or space : Tags, groups and the prisoner's dilemma. In Scott Moss and Paul Davidsson, editors, *Proceedings of the 1st Workshop on Multi-Agent Based Systems*, 2000.

[146] David Hales. Evolving specialisation, altruism, and group-level optimisation using tags. In Jaime Simao Sichman, Francois Bousquet, and Paul Davidsson, editors, *Proceedings of the 2nd workshop on multi-agent-based simulation*, 2002.

[147] Rod Heil. A trilaterative localization system for small mobile robots in swarms. Master's thesis, Department of Electrical and Computer Engineering, 2004.

[148] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–487, 2000.

[149] D. Helbing and B. A. Huberman. Coherent moving states in highway traffic. *Nature*, 396:738–740, 1998.

[150] D. Helbing, P. Molnar, I. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and Planning B: Planning and Design*, 28(3):361–383, 2001.

[151] D. Helbing and M. Schreckenberg. Cellular automata simulating experimental properties of traffic flows. *Physical Review E*, 59:R2505–R2508, 1999.

[152] D. Helbing and M. Treiber. Jams, waves, and clusters. *Science*, 282:2001–2003, 1998.

[153] C.K. Hemelrijk. Spatial centrality of dominants without positional preference. In C. Adami, R. Belew, H. Kitano, and C. Taylor, editors, *Artificial Life VI*, volume 6, pages 307–315. MIT Press, 1998.

[154] C.K. Hemelrijk. Effects of cohesiveness on intersexual dominance relationships and spatial structure among group-living virtual entities. In D. Floreano, J.-D. Nicoud, and F. Mondada, editors, *Proceedings of the Fifth European Conference on Artificial Life*, volume 1674 of *Lecture Notes in Computer Science*, pages 524–534. Springer, 1999.

[155] C.K. Hemelrijk. Emergent social phenomena in a competitive, virtual world ('domworld'). In C. C. Marley and E. Boudreau, editors, *Artificial Life 7 Workshop Proceedings*, pages 123–127, 2000.

[156] C.K. Hemelrijk. Social phenomena emerging by self-organisation in a competitive, virtual world ("domworld"). In K. Jokinen, D. Heylen, and A. Nijholt, editors, *Learning to behave. Workshop II: Internalising knowledge*, pages 11–19, 2000.

[157] C.K. Hemelrijk. The use of artificial-life models for the study of social organization. In B. Thierry, M. Singh, and W. Kaumanns, editors, *Macaque societies A model for the study of social organization*, pages 295–313. Cambridge University Press, 2004.

[158] C.K. Hemelrijk, editor. *Self-organisation and evolution of social systems.* Cambridge University Press, 2005.

[159] C.K. Hemelrijk and H. Kunz. Density distribution and size sorting in fish schools: an individual-based model. *Behavioral Ecology*, 16(1):178–187, 2004.

[160] C.K. Hemelrijk, J. Wantia, and L. Gygax. The construction of dominance order: comparing performance of five methods using an individual-based model. *Behaviour*, 142:1043–1064, 2005.

[161] F. Heylighen. The science of selforganization and adaptivity, 2001. http://pespmc1.vub.ac.be/Papers/EOLSS-Self-Organiz.pdf.

[162] R. HilleRisLambers, M. Rietkerk, F. van den Bosch, H. H. T. Prins, and H. de Kroon. Vegetation pattern formation in semi-arid grazing systems. *Ecology*, 82(1):50?61, 2001.

[163] Douglas R. Hofstadter and Daniel C. Dennett. *The Mind's I: Fantasies and reflections on self and soul.* Basic Books, 1981.

[164] John Holland. *Adaptation in Natural and Artificial Systems.* The MIT Press, 1975.

[165] John Holland. *Hidden Order: how adaptation builds complexity.* Perseus Books, 1995.

[166] John Holland. *Emergence: From chaos to order.* Oxford University Press, 1998.

[167] Jeffrey Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal Cooperative Populations.* PhD thesis, Illinois Genetic Algorithms Laboratory (ILLEGAL) University of Illinois, 1997.

[168] Michael N. Huhns and Larry M. Stephens. Multiagent systems and societies of agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 2. The MIT Press, 1999.

[169] Guillaume Hutzler. Cellular automata and agent-based approaches for the modelling and simulation of biological systems, 2006. http://shum.huji.ac.il/

[170] T.D. Huyhn. *Trust and Reputation in Open Multi-Agent Systems*. PhD thesis, Electronics and Computer Science, University of Southampton, 2006.

[171] Jesus Ibanez, Antonio F. Gomez-Skarmeta, and Josep Blat. Dj-boids: emergent collective behavior as multichannel radio station programming. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 248–250, 2003.

[172] A.J. Ijspeert, A. Crespi, and J.M. Cabelguen. Simulation and robotics studies of salamander locomotion: Applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3:171–196, 2005.

[173] Ian R. Inglis, Bjorn Forkman, and John Lazarus. Free food or earned food? a review and fuzzy model of contrafreeloading. *Animal Behaviour*, 53(6):1171–1191, 1997.

[174] Ian R. Inglis, Steve Langton, Bjorn Forkman, and John Lazarus. An information privacy model of explanatory and foraging behaviour. *Animal Behaviour*, 62(3):543–557, 2001.

[175] Rogier Jacobs. Data stream mining artificial societies. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2005.

[176] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

[177] Marco A. Janssen, editor. *Complexity and Ecosystem Management: The Theory and Practice of Multi-Agent Systems*. Edward Elgar Publishing Ltd, 2002.

[178] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Department of Computer Science, Vrije Universiteit Amsterdam, November 2003.

[179] M. Jelasity and M. van Steen. Large-scale newscast computing on the internet. Technical Report IR-503, Department of Computer Science, Vrije Universiteit Amsterdam, October 2002.

[180] M. Jelasity, M. van Steen, and W. Kowalczyk. An approach to massively distributed aggregate computing on peer-to-peer networks. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 200–207. IEEE Computer Society, 2004.

[181] T. Joerg. A theory of reciprocal learning in dyads. *Cognitive Systems*, 6(2):159–170, 2004.

[182] Cheryl A. Johnson, James W. A. Grant, and Luc-Alain Giraldeaub. The effect of patch size and competitor number on aggression among foraging house sparrows. *Behavioral Ecology*, 15:412–418, 2004.

[183] C. Jones and M. J. Mataric. Adaptive division of labor in large–scale minimalist multi–robot systems. In *Proc. of Int. Conferrence on Intelligent Robots and Systems (IROS-03)*, pages 1969–1974, 2003.

[184] I. Karsai and J. W. Wenzel. Productivity, individual-level and colony-level flexibility, and organization of work as consequence of colony size. *Proc. Natl. Acad. Sci. U.S.A.*, 95:8665–8669, 1998.

[185] Stuart Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:434–467, 1969.

[186] Stuart Kauffman. *Investigations*. Oxford University Press, 2000.

[187] Stuart A. Kauffman. *The Origins of Order: self-organization and selection in evolution*. Oxford University Press, 1993.

[188] Stuart A. Kauffman. *At Home in the Universe: the search for laws of complexity*. Penguin Group, 1995.

[189] S. Kazadi. *Swarm Engineering*. PhD thesis, California Institute of Technology, 2000.

[190] F. P. Kelly. Network routing. *Philosophical Transactions of the Royal Society Series A (Physical Sciences and Engineering)*, 337:343–367, 1991.

[191] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[192] R. Kirkwood. Emperor penguin (aptenodytes forsteri) foraging ecology. Technical report, ANARE Reports, 2001.

[193] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In W. Lewis Johnson, editor, *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM Press, 1997.

[194] Jon Klein. BREVE: a 3d environment for the simulation of decentralized systems and artificial life. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, pages 329–334, Cambridge, MA, USA, 2003. MIT Press.

[195] Sven Koenig and Yaxin Liu. Terrain coverage with ant robots: a simulation study. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 600–607, New York, NY, USA, 2001. ACM Press.

[196] Timothy A. Kohler and George G. Gumerman, editors. *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes*. Oxford University Press, 2000.

[197] M. Komosinski and S. Ulatowski. Framsticks: towards a simulation of a nature-like world, creatures and evolution. In D. Floreano, J.-D. Nicoud, and F. Mondada, editors, *Advances in Artificial Life: 5th European Conference on Artificial Life*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pages 261–265. Springer-Verlag, 1999.

[198] Maciej Komosinski and Marek Kubiak. Taxonomy in Alife. Measures of similarity for complex artificial organisms. In Jozef Kelemen and Petr Sosík, editors, *Advances in Artificial Life. Lecture Notes in Artificial Intelligence 2159*, pages 685–694. Springer-Verlag, 2001.

[199] Alexander V. Konstantinou. *Towards Autonomic Networks*. PhD thesis, Columbia University, 2003.

[200] Helmut Kopka and Patrick W. Daly. *A Guide to LaTeX*. Addison-Wesley, 1993.

[201] Manolis Koubarakis. *Lecture Notes in Computer Science*, chapter Multi-agent Systems and Peer-to-Peer Computing: Methods, Systems, and Challenges, pages 46–61. Springer Berlin / Heidelberg, 2003.

[202] W. Kowalczyk, M. Jelasity, and A.E. Eiben. Towards data mining in large and fully distributed peer-to-peer overlay networks. In T. Heskes, P. Lucas, L. Vuurpijl, and W. Wiegerinck, editors, *Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence*, pages 203–210, 2003.

[203] Alan H. Krakauer. Kin selection and cooperative courtship in wild turkeys. *Nature*, 434:69–72, 2005.

[204] Sarit Kraus. Automated negotiation and decision making in multiagent environments. In Michael Luck, Vladimir Marik, Olga Stepankova, and Robert Trappl, editors, *Proceedings of the Advanced Course on Artificial Intelligence*, volume 2086 of *Lecture Notes in Artificial Intelligence*, pages 150–172. Springer, 2001.

[205] J. Krause, D. Hoare, S. Krause, C.K. Hemelrijk, and D. Rubenstein. Leadership in fish shoals. *Fish and Fisheries*, 1:82–89, 2000.

[206] David M. Kreps. *Game Theory and Economic Modelling*. Clarendon Lectures in Economics. Oxford University Press, 1990.

[207] Michael J.B. Krieger, Jean-Bernard Billeter, and Laurent Keller. Ant-like allocation and recruitment in cooperative robots. *Nature*, 406:992–995, 2000.

[208] C. Ronald Kube, Chris A. Parker, Tao Wang, and Hong Zhang. Biologically inspired collective robotics. In Leandro N. de Castro and Fernando J. Von Zuben, editors, *Recent Developments in Biologically Inspired Computing*, chapter 15. Idea Group Publlishing, 2005.

[209] H. Kunz and C. K. Hemelrijk. Artificial fish schools: collective effects of school size, body size, and body form. *Artificial Life*, 9(3):237–253, 2003.

[210] E. A. Langridge, N.l R. Franks, and A. B. Sendova-Franks. Improvement in collective performance with experience in ants. *Behavioral Ecology and Sociobiology*, 56:523–529, 2004.

[211] J. Stephen Lansing. "Artificial societies" and the social sciences. *Artificial Life*, 8(3):279–292, 2002.

[212] P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors. *Learning Classifier Systems: From Foundations to Applications*. Springer, 2000.

[213] P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors. *Advances in Learning Classifier Systems*. Springer, 2001.

[214] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.

[215] Kristina Lerman and Aram Galstyan. A general methodology for mathematical analysis of multiagent systems, 2001. http://citeseer.ist.psu.edu/lerman01general.html.

[216] Kristina Lerman and Aram Galstyan. Macroscopic analysis of adaptive task allocation in robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1951–1956, 2003.

[217] Andras Liker and Zoltan Barta. The effects of dominance on social foraging tactic use in house sparrows. *Behaviour*, 139:1061–1076, 2002.

[218] F. Liljeros. Sexual networks in contemporary western societies. *Physica A*, 338:238–245, 2004.

[219] F. Liljeros, C. R. Edling, and L.A. Nunes Amaral. Sexual networks: implications for the transmission of sexually transmitted infections. *Microbes and infection*, pages 189–196, 2003.

[220] Matthew Laws Littler. Simulating the long house valley: An evaluation of the role of agent-based computer simulation in archaeology. Master's thesis, Arizona Anthropology Department. Available Through University, 1998.

[221] Z. Liu, M. H. Ang Jr., and W. Khoon Guan Seah. A searching and tracking framework for multi-robot observation of multiple moving targets. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 8(1):14–22, 2004.

[222] G.F. Loewenstein and J. Elster. *Choice over time*. Russel Sage Foundation, 1992.

[223] L. Lovasz. Random walks on graphs: A survey. In *Combinatorics, Paul Erdos is Eighty*, volume 2, pages 353–398. Budapest: Janos Bolyai Mathematical Society, 1996.

[224] S. Luke, K. Sullivan, G. C. Balan, and L. Panait. Tunable decentralized algorithms for cooperative target observation. Technical report, George Mason University, 2004.

[225] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan. Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 SwarmFest Workshop*, 2004.

[226] Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 1996. MIT Press.

[227] E.D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In D. Cli, P. Husbands, J. Meyer, and S. Wilson, editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3*, pages 501–508. MIT Press, 1994.

[228] Pradipta Maji, Chandrama Shaw, Niloy Ganguly, Biplab K. Sikdar, and P. Pal Chaudhuri. Theory and application of cellular automata for pattern classification. *Fundam. Inf.*, 58(3-4):321–354, 2003.

[229] James McLurkin. Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots. Master's thesis, MIT Computer Science and Artificial Intelligence Lab, 2004.

[230] James McLurkin and Jennifer Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Proceedings of the Distributed Autonomous Robotic Systems Conference*, 2004.

[231] H. Meinhardt, P. Prusinkiewicz, and D. R. Fowler. *The Algorithmic Beauty of Sea Shells*. Springer, 2001.

[232] M. Menaker, L. F. Moreira, and G. Tosini. Evolution of circadian organization in vertebrates. *Brazilian Journal of Medical and Biological Research*, 30:305–313, 1997.

[233] D. Merkle and M. Middendorf. Dynamic polyethism and competition for tasks in threshold reinforcement models of social insects. *Adaptive Behavior*, 12:251–262, 2004.

[234] D. Merkle, M. Middendorf, and A. Scheidler. Modelling ant brood tending behavior with cellular automata. In *Proc. of the Workshop on Modelling of Complex Systems by Cellular Automata 2005*, 2005.

[235] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, Technical Report HPL-2002-57, HP Laboratories Palo Alto, 2002.

[236] Marvin Minsky. *The society of mind*. Simon and Schuster, 1985.

[237] M. Mirolli, D. Parisi, W. Banzhaf, T. Christaller, P. Dittrich, J.T. Kim, and J. Ziegler. Artificial organisms that sleep. In *Proc. of ECAL 2003 : European conference on advances in artificial life No7, Dortmund*, 2003.

[238] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[239] F. Mondada, G.C. Pettinaro, A. Guignard, I. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L.M. Gambardella, and M. Dorigo. Swarm-bot: a new distributed robotic concept. *Autonomous Robots*, 17(2-3):193–221, 2004.

[240] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, Inc, sixth edition, 2005.

[241] Cristopher Moore and M. E. J. Newman. Epidemics and percolation in small-world networks, Jan 2000. http://arxiv.org/abs/cond-mat/9911492.

[242] D. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(1):373–399, 1997.

[243] R. Morley and G. Ekberg. Cases in chaos: Complexity-based approaches to manufacturing. In *Embracing Complexity: A Colloquium on the Application of Complex Adaptive Systems to Business*, 1998.

[244] Christos J. Moschovitis, Hilary Poole, and Theresa M. Senft. *History of the Internet: A Chronology, 1843 to the Present*. A B C-CLIO, Incorporated, 1999.

[245] S. Moss, H. D. Dixond, and S. Wallis. Evaluating competative strategies. *Intelligent Systems in Accounting, Finance and Management*, 4:245–258, 1995.

[246] S. Moss, H. Gaylard, S. Wallis, and B. Edmonds. SDML: A multi-agent language for organizational modelling. *Computational and Mathematical Organization Theory.*, 4(1):43–69, 1998.

[247] Soraia R. Musse, Christian Babski, Tolga Capin, and Daniel Thalmann. Crowd modelling in collaborative virtual environments. In J. M. Shieh and Shi-Nine Yang, editors, *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 115 – 123, 1998.

[248] Wolfgang E. Nagel, Wolfgang V. Walter, and Wolfgang Lehner, editors. *Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference, Dresden, Germany, August 28 - September 1, 2006, Proceedings*, volume 4128 of *Lecture Notes in Computer Science*. Springer, 2006.

[249] A. Newell and H.A. Simon. GPS: A program that simulates human thought. In E.A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, New York, 1963.

[250] J. Noailly, J.C.J.M. van den Bergh, and C.A. Withagen. Evolution of harvesting strategies: replicator and resource dynamics. *Journal of Evolutionary Economics*, 13(2):183–200, 2003.

[251] Jason Noble. Sexual signalling in an artificial population: when does the handicap principle work? In *Proceedings of the 5 European Conference on Artificial Life*, pages 644–653. Springer-Verlag, 1999.

[252] S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. Technical Report 95-15, Institute of Psychology, National Research Council, Rome, Italy, 1995.

[253] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: the biology, intelligence, and technology of self-organizing machines*. The MIT Press, 2000.

[254] Stefano Nolfi and Domenico Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5(1):75–98, 1997.

[255] William D. Nordhaus and Zili Yang. A regional dynamic general-equilibrium model of alternative climate-change strategies. *the American Economic Review*, 86(4):741–765, 1996.

[256] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1–25, 2006.

[257] Andrzej Nowak and Maciej Lewenstein. Modeling social change with cellular automata, 1996. http://cognitrn.psych.indiana.edu/rgoldsto/complex/nowak.pdf.

[258] Martin A. Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393:573–577, 1998.

[259] Timonthy O'Conner and David Robb, editors. *Philosophy of Mind: Contemporary readings*. Routledge, 2003.

[260] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The not so short introduction to latex-2e, 2006. http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf.

[261] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

[262] D. V. O'Toole, P. A. Robinson, and M. R. Myerscoughs. Self-organized criticality in ant brood tending. *Journal of theoretical biology*, 221:1–14, 2003.

[263] G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef. Performance management for cluster based web services. Technical report, IBM, 2003.

[264] Romans Pancs and Nicolaas J. Vriend. Schelling's spatial proximity model of segregation revisited. *Journal of Public Economics*, 2006.

[265] Dhanashree A. Paranjpe and Vijay Kumar Sharma. Evolution of temporal order in living organisms. *Journal of Circadian Rhythms*, 3:1–13, 2005.

[266] Domenico Parisi, Federico Cecconi, and Stefano Nolfi. Econets: Neural networks that learn in an environment. *Network: Computation in Neural Systems*, 1(2):149 – 168, 1990.

[267] L. Parker and B. Emmons. Cooperative multi-robot observation of multiple moving targets, 1997. http://citeseer.ist.psu.edu/article/parker97cooperative.html.

[268] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12:231–255, 2002.

[269] Lynne E. Parker. Local versus global control laws for cooperative agent teams. Technical report, Massachusetts Institute of Technology, 1992.

[270] Lynne E. Parker. Adaptive heterogeneous multi-robot teams. *Neurobiology*, 28:75–92, 1999.

[271] Miles T. Parker. What is Ascape and why should you care? *Journal of Artificial Societies and Social Simulation*, 4(1), December 2001.

[272] J. K. Parrish, S. V. Viscido, and D. Gruenbaum. Self-organized fish schools: An examination of emergent properties. *The Biological Bulletin*, 202:296–305, 2002.

[273] A. A. Pate, E. T. Gawlinkia, S. K. Lemieux, and R. A. Gatenby. A cellular automaton model of early tumor growth and invasion: The effects of native tissue vascularity and increased anaerobic tumor metabolism. *Journal of theoretical Biology*, 213:315–331, 2001.

[274] Relu Patrascu, Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. New approaches to optimization and utility elicitation in autonomic computing. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 140–145, 2005.

[275] Andres Perez-Uribe, Dario Floreano, and Laurent Keller. Effects of group composition and level of selection in the evolution of cooperation in artificial ants. In Wolfgang Banzhaf, Thomas Christaller, Peter Dittrich, Jan T. Kim, and Jens Ziegler, editors, *Proceedings of the European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 128–137, 2003.

[276] J.B. Pollack, G.S. Hornby, H. Lipson, and P. Funes. Computer creativity in the automatic design of robots. *LEONARDO*, 36(2):115–121, 2003.

[277] Tom Postmes, S. Alexander Haslam, and Roderick I. Swaab. Social influence in small groups: an interactive model of social identity formation. *European Review of social psychology*, 16:1–42, 2005.

[278] Mitchell A. Potter, Lisa A. Meeden, and Alan C. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the Seventh International Conference on Artificial Intelligence*, pages 1337–1343. Morgan Kaufmann, 2001.

[279] Michael J. Prietula and Kathleen M. Carley. Computation organization theory: Autonomous agents and emergent behavior. *Journal of organizational computing*, 4(1):41–83, 1994.

[280] Iyad Rahwan, Sarvapali D. Ramchurn, Nicholas R. Jennings, Peter McBurney, Simon Parsons, and Liz Sonenberg. Argumentation-based negotiation. *Knowledge Engineering Review*, 18(4):343–375, 2004.

[281] Ana Maria Ramanath and Nigel Gilbert. Towards a methodology for aget-based social simulation research, 2003. http://www.cimne.upc.es/simweb/formacion/ABSSSIG-feb03.pdf.

[282] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *Knowledge Engineering Review*, 19(1):1–25, 2004.

[283] Jayanthi Rao and S. Biswas. Biologically inspired mobility models for energy conservation in sensor networks. In *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on*, pages 133–140, 30-31 May 2005.

[284] Mitchel Resnick. Collaboration in simulated worlds: Learning through and about collaboration. *ACM SIGCUE Outlook*, 21(3):36–38, 1992.

[285] Mitchel Resnick. Starlogo: an environment for decentralized modeling and decentralized thinking. In *CHI '96: Conference companion on Human factors in computing systems*, pages 11–12, New York, NY, USA, 1996. ACM Press.

[286] Mitchel Resnick. *Turtles, Termites, and Traffic Jams: explorations in massively parallel microworlds*. The MIT Press, 1997.

[287] C. W. Reynolds. Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[288] Craig Reynolds. Big fast crowds on ps3. In Alan Heinrich and Douglas Thomas, editors, *Acm Siggraph Video Game Symposium: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 113–121. ACM Press, 2006.

[289] M. Rietkerk, M. C. Boerlijst, F. van Langevelde, R. HilleRisLambers, J. van de Koppel, L. Kumar, H. H. T. Prins, and A. M. de Roos. Self-organization of vegetation in arid ecosystems. *The American naturalist*, 160(4):524–530, 2002.

[290] M. Rietkerk, S. C. Dekker, P. C. de Ruiter, and J. van de Koppel. Self-organised patchiness and catastrophic shifts in ecosystems. *Science*, 305:1926–1929, 2004.

[291] M. Rietkerk, S. C. Dekker, M. J. Wassen, A. W. M. Verkroost, and M. F. P. Bierkens. A putative mechanism for bog patterning. *The American naturalist*, 163(5):699–708, 2004.

[292] M. Rietkerk and J. van de Koppel. Alternate stable states and threshold effects in semi-arid grazing systems. *OIKOS*, 79:69–76, 1997.

[293] M. Rietkerk, F. van den Bosch, and J. van de Koppel. Site-specific properties and irreversible vegetation changes in semi-arid grazing systems. *OIKOS*, 80:241–252, 1997.

[294] Karin Rijnders. Self-organizing patchiness. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2006.

[295] R. Roson and J.C.J.M. van den Bergh. Network markets and the struture of networks. *The Annals of Regional Science*, 34:197–211, 2000.

[296] H. Sachar and S. Sharan. The effects of school organization on the implementation of cooperative learning in secondary schools. *School Effectiveness and School Improvement*, 6:47–66, 1995.

[297] Tuomas W. Sandholm. Automated mechanism design: A new application area for search algorithms. In James Bowen and Francesca Rossi, editors, *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2003.

[298] N. J. Savill, P. Rohani, and P. Hogeweg. Self-reinforcing spatial patterns enslave evolution in a host-parasitoid system. *Journal of theoretical Biology*, 188:11–20, 1997.

[299] Ricarda Scheiner. *Sucrose responsiveness and behaviour in honey bees (Apis mellifera L.)*. PhD thesis, Technical University of Berlin Faculty VII Institute of Ecology, 2001.

[300] T.C. Schelling. Models of segregation. *American Economic Review*, 59(2):488–493, 1969.

[301] T.C. Schelling. Dynamic models of segregation. *Journal of Mathematical Sociology*, 1(2):143–186, 1971.

[302] T.C. Schelling. *Micromotives and Macrobehaviour*. W. W. Norton & Company, 1978.

[303] J.W. Schmidt and R.E. Taylor. *Simulation and Analysis of Industrial Systems*. Richard D. Irwin, Homewood, 1970.

[304] Detlef Schoder and Kai Fischbach. Peer-to-peer prospects. *Commun. ACM*, 46(2):27–29, 2003.

[305] Richard Seel. Emergence in organisations, 2005. http://www.new-paradigm.co.uk/emergence-human.htm.

[306] T. D. Seeley. *The Wisdom of the Hive*. Harvard University Press, 1995.

[307] A. B Sendova-Franks and N. R. Franks. Spatial relationships within nests of the ant leptothorax unifasciatus (latr.) and their implications for the division of labour. *Animal behavior*, 50:121–136, 1995.

[308] Alexander Serenko and Brian Detlor. Agent toolkits: A general overview of the market and an assessment of instructor satisfaction with utilizing toolkits in the classroom. Technical Report 455, Michael G. DeGroote School of Business, McMaster University, Hamilto, Ontario, 2002.

[309] W. Shen, L. Wang, and Q. Hao. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 36:563–577, 2006.

[310] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press, 1994.

[311] Karls Sims. Evolving 3d morphology and behavior by competition. In Rodney Brooks and Patty Maes, editors, *Artificial Life IV Proceedings*, pages 28–39. MIT Press, 1994.

[312] Hokky Situngkir. Epidemiology through cellular automata case of study: Avian influenza in indonesia. Technical report, Bandung Fe Institute, 2004.

[313] E. Sklar, M.C. Schut, K. Diwold, and S. Parsons. Exploring coordination propoerties withing populations of distributed agents. In *Proceedings of the AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management*, 2006.

[314] Elizabeth Sklar and Mathew Davies. Multiagent simulation of learning environments. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *Proceedings of the Fourth International Joint Conference of Autonomous Agents and Multiagent Systems*, 2005.

[315] Elizabeth Sklar, Mathew Davies, and Min San Tan Co. Simed: Simulating education as a multi agent system. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Joint Conference of Autonomous Agents and Multi Agent Systems*, 2004.

[316] Elliott Sober and David Sloan Wilson. *Unto Others: The evolution and psychology of unselfish behaviour*. Harvard University Press, 1998.

[317] Diana Spears, Wesley Kerr, and William Spears. Physics-based robot swarms for coverage problems. *International Journal on Intelligent Control and Systems*, 2007. To appear.

[318] W. Spears and D. Gordon. Using artificial physics to control agents. In *IEEE International Conference on Information, Intelligence, and Systems*, pages 281–288, 1999.

[319] William M. Spears, Diana F. Spears, Jerry C. Hamann, and Rodney Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17:137–162, 2004.

[320] Lee Spector, Jon Klein, Chris Perry, and Mark Feinstein. Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*, 6(1):111–125, March 2005.

[321] Maartje Spoelstra. Simulating the effects of goal structures in human learning environments. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2006.

[322] K. Stanley, B. Bryant, and R. Miikkulainen. Real-time neuro-evolution in the nero video game. *IEEE Transactions Evolutionary Computation*, 9(6):653–668, 2005.

[323] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[324] Gary Stead. An artificial life simulation to investigate the huddling behaviour of emperor penguins. Master's thesis, University of Sheffield, 2003.

[325] Mark Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann, 1995.

[326] K. Steiglitz and L. I. O'Callaghan. Microsimulation of markets and endogenous price bubbles. In *Proceedings of the 3rd International Conference on Computing in Economics and Finance*, pages 1–4, 1997.

[327] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*. Springer, 2005.

[328] Steven Strogatz. *Sync: The Emerging Science of Spontaneous Order*. Hyperion, 2003.

[329] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In Morgan Kaufmann, editor, *Proceedings of 15th Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 832?–839, San Francisco, CA, 1997.

[330] T. Susumu, Y. Anzai, and Y. Sakurai. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. *Journal of Neurophysiology*, 89:2245–2258, 2003.

[331] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction*. MIT Press, 1998.

[332] Tadeusz M. Szuba. *Computational Collective Intelligence*. Wiley-Interscience, 2001.

[333] Yuqing Tang, Simon Parsons, and Elizabeth Sklar. Agent-based modeling of human education data. In Hideyuki Nakashima and Michael Wellman, editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems*. ACM Press, 2006.

[334] H.G. Tanner, A. Jadbabaie, and G.J. Pappas. Stable flocking of mobile agents, part i: fixed topology. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2010 – 2015, 2003.

[335] T. Tassier and F. Menczer. Emerging small-world referral networks in evolutionary labor markets. *IEEE-EC*, 5:482–492, Oct 2001.

[336] Ronald Terpstra and Olaf van Zon. Validating the cropping theory with virtual creatures. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2006.

[337] Leigh Tesfatsion, editor. *Special Issue on Agent-Based Modeling of Evolutionary Economic Systems*, volume 5(5) of *IEEE Transactions on Evolutionary Computation*. IEEE, 2001.

[338] Leigh Tesfatsion. Agent-based computational economics. Technical report, Department of Economics, Iowa State University, 2002.

[339] Leigh Tesfatsion. Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, 8(1):55–82, 2002.

[340] Leigh Tesfatsion. Agent-based computational economics: modeling economies as complex adaptive systems. *Information Sciences*, 149(4):262–268, 2003.

[341] Leigh Tesfatsion and Kenneth Judd, editors. *Handbook of Computational Economics, Volume 2: Agent-Based Computational Economics*. Elsevier, 2006.

[342] E. Teweldemedhin, T. Marwala, and C. Mueller. Agent-based modelling: a case study in HIV epidemic. In *Proceedings of HIS 2004. Fourth International Conference on Hybrid Intelligent Systems, 2004.*, 2004.

[343] Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. Response threshold reinforcement and division of labour in insect societies. *Proceeding of the Royal Society of London*, 265:327–332, 1998.

[344] B. Thierry. Feedback loop between kinship and dominance: the macaque model. *Journal of Theoretical Biology*, 145:511–522, 1990.

[345] J. M. Thiery, J.-M. D'Herbes, and C. Valentin. A model simulating the genesis of banded vegetation patterns in niger. *The Journal of Ecology*, 83(3):497–507, 1995.

[346] D'Arcy Thompson. *On Growth and Form (Reprint)*. Cambridge University Press, 1992.

[347] Robert Tobias and Carole Hofmann. Evaluation of free java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), 2004.

[348] J. F. A. Traniello and R. B. Rosengaus. Ecology, evolution, and division of labour in social insects. *Animal Behaviour*, 53:209–213, 1997.

[349] V. Trianni and M. Dorigo. Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics*, 95(3):213–231, 2006.

[350] Robert L. Trivers. The evolution of reciprocal altruism. In *Natural Selection and Social Theory: Selected Papers of Robert Trivers*. Oxford University Press, 2002.

[351] Duane P. Truex, Richard Baskerville, and Heinz Klein. Growing systems in emergent organizations. *Communications of the ACM*, 42(8):117–123, August 1999.

[352] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 94*, Computer Graphics, pages 43–50. ACM, 1994.

[353] Edward R. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative.* Graphics Press, 1997.

[354] Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, second edition, 2001.

[355] Edward T. Tufte. *Envisioning Information.* Graphics Press, 1990.

[356] K. Tumer and D. Wolpert. Coordination in large collectives. In Y. Bar-Yam, editor, *Fifth International Conference on Complex Systems*, chapter 1. Perseus Books, 2006. To appear.

[357] Kagan Tumer and Adrian Agogino. Coordinating multi-rover systems: evaluation functions for dynamic and noisy environments. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 591 – 598. ACM Press, 2005.

[358] J.C.J.M. van den Bergh. Evolutionary modeling. In J. Proops and P. Safonov, editors, *Integrated Modelling in Ecological Economics.* Edward Elgar, Cheltenham, 2003.

[359] J.C.J.M. van den Bergh and J.M. Gowdy. Evolutionary theories in environmental and resource economics: approaches and applications. *Environmental and Resource Economics*, 17:37–57, 2000.

[360] J.C.J.M. van den Bergh and J.M. Gowdy. The microfoundations of macroeconomics: an evolutionary perspective. *Cambridge Journal of Economics*, 27(1):65–84, 2003.

[361] J.C.J.M. van den Bergh, A. Ferrer i Carbonell, and G. Munda. Alternative models of individual behaviour and implications for environmental policy. *Ecological Economics*, 32(1):43–61, 2000.

[362] Jeroen C. J. M. van den Bergh and Marco A. Janssen, editors. *Economics of Industrial Ecology: Materials, Structural Change, and Spatial Scales.* MIT Press, 2004.

[363] Albert van der Heide. The group movement of aptenodytes forsteri implemented as an 'emergent phenomenon'. Master's thesis, University of Groningen, April 2003.

[364] Jelmer van der Ploeg. Evolving an agent controller using genetic programming. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2005.

[365] Roland van Kempen. Turks in the netherlands. *Turks in the Netherlands*, 41:374–395, 1997.

[366] Matthijs van Veelen. Evolution in games with a continuous action space. Technical Report TI2001-068/1, Faculty of Economics and Business Administration, Vrije Universiteit Amsterdam and Tinbergen Institute, 2001.

[367] Matthijs van Veelen. Altruism, fairness and evolution: the case for repeated stochastic games. Technical Report TI2002-111/1, Faculty of Economics and Business Administration, Vrije Universiteit Amsterdam and Tinbergen Instititute, 2002.

[368] Tamas Vicsek. The bigger picture. *Nature*, 418:131, 2002.

[369] Nico Vink. Exploring communication dynamics in vuscape. Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2004.

[370] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable peer-to-peer gossiping protocol. In *Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003)*, 2003.

[371] Spyros Voulgaris and Maarten van Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par*, pages 1143–1152, 2005.

[372] L.S. Vygotsky. *Mind in Society: The development of higher psychological processes.* Armonk, 1978.

[373] Mitchell M. Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos.* Simon and Schuster, 1992.

[374] Haiyang Wang, Li-Geng Ma, Jin-Ming Li, Hong-Yu Zhao, and Xing Wang Deng. Direct interaction of arabidopsis cryptochromes with cop1 in mediation of photomorphogenic development. *Science*, 294(5540):154–158, 2001.

[375] Richard A. Watson, Sevan G. Ficici, and Jordan B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the Congress on Evolutionary Computation*, pages 335–342. IEEE Press, 1999.

[376] Duncan J. Watts. *Six Degrees: The Science of a Connected Age.* W. W. Norton & Company, 2003.

[377] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[378] Horst F. Wedde and Muddassar Farooq. A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. *J. Syst. Archit.*, 52(8):461–484, 2006.

[379] Juergen W. Weibull. *Evolutionary Game Theory.* The MIT Press, 1997.

[380] Barry Brian Werger and Maja J Mataric. Broadcast of local eligibility for multi-target observation. In *Distributed Autonomous Robotic Systems*, volume 4, pages 347–356. Springer, 2000.

[381] R. Paul Wiegand, Mitchell A. Potter, Donald A. Sofge, and William M. Spears. A generalized graph-based method for engineering swarm solutions to multiagent problems. In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J. Merelo-Guervos, L. Darrell Whitly, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*. Springer, 2006.

[382] Don Willems. Apparent cooperative behavior in simulated reactive robots. Master's thesis, Department. of Artificial Intelligence, University of Nijmegen, 2003.

[383] E. O. Wilson. *The Insect Societies*. Harvard University Press, 1971.

[384] E. O. Wilson. *Success and Dominance in Ecosystems: The Case of the Social Insects*. Oldendorf/Luhe, Germany: Ecology Institute., 1990.

[385] Arthur T. Winfree. *The Geometry of Biological Time*. Springer Verlag, 2001.

[386] E. Winfree. Algoritmic self-assembly of DNA: Theorectical motivations and 2d assembly experiments. *Journal of Biomolecular Structure and Dynamics*, 11:263–270, 1999.

[387] Stephen Wolfram. *A new kind of Science*. Wolfram Media, Inc., 2002.

[388] David H. Wolpert and Kagan Tumer. A survey of collective intelligence. In David H. Wolpert and Kagan Tumer, editors, *Collectives and the Design of Complex Systems*. Springer-Verlag, 2004.

[389] David H. Wolpert, Kagan Tumer, and Jeremy Frank. Using collective intelligence to route internet traffic. *Advances in Neural Information Processing Systems*, 11, 1999.

[390] M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, , Cambridge, USA, 2000.

[391] Michael J. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[392] M.J. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[393] Daniel Yamins. Towards a theory of "local to global" in distributed multi-agent systems (i). In Frank Dignum, Virginia Dignum, Sven KOenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldrdige, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 183–190. ACM Press, 2005.

[394] Daniel Yamins. Towards a theory of "local to global" in distributed multi-agent systems (ii). In Frank Dignum, Virginia Dignum, Sven KOenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldrdige, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 191–198. ACM Press, 2005.

[395] H. Peyton Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 2001.

[396] Larry A. Young, Greg Pisanich, and Corey Ippolito. Aerial explorers. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Jannuary 2005.

[397] Larry A. Young, Gregory Pisnich, Corey Ippolito, Rick Alena, and Nasser Kehtarnavaz Philip A. Laplante. Aerial vehicle surveys of other planetary atmospheres and surfaces : imaging, remote-sensing, and autonomy technology requirements. In *SPIE proceedings*

series (SPIE proc. ser.) International Society for Optical Engineering proceedings series, pages 183–199, 2005.

[398] Junfu Zhang. A dynamic model of residential segregation. *Journal of Mathematical Sociology*, 28:147 – 170, 2004.

[399] B. Zhou and S. Zhou. Parallel simulation of group behaviors. In *Proceedings of the 2004 Winter Simulation Conference*, 2004.