Emergent Models

in Hierarchical and Distributed Simulation of Complex Systems

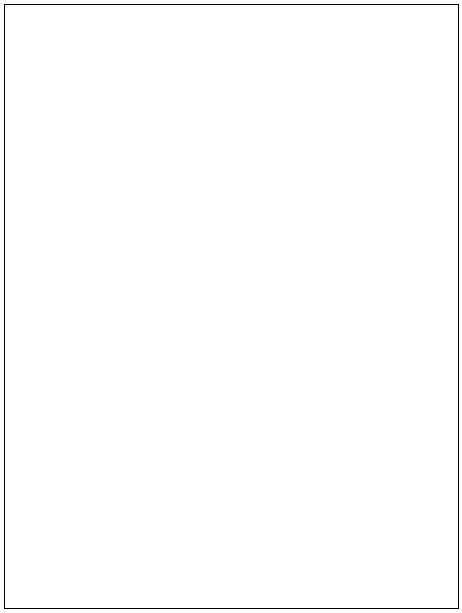
with Applications to Ecosystem and Genetic Network Modelling

A thesis submitted for the degree of

Doctor of Philosophy

at The University of Queensland in January 2005

with revisions in October 2005



Author: Henk Stolk, Research Scholar
Advanced Computational Modelling Centre / ARC Centre for Complex Systems / School of Information Technology and Electrical Engineering
Faculty of Engineering, Physical Sciences and Architecture
University of Queensland, Brisbane QLD 4072, Australia

Candidate's Statement of Originality

I state that the emergent models methodology developed in the present thesis is my original contribution.

Candidate's Statement of Contribution to Jointly-published Work

Succinct descriptions of the methodology have been published or presented elsewhere with my advisors Kevin Gates and Jim Hanan as co-authors, as stated in the List of Publications and Presentations hereafter. While I acknowledge their contribution, support, and inspiration, the emergent models methodology and its applications described in this thesis have been developed by me.

Statement of Contributions by Others

Some of the applications of the emergent models methodology developed in this thesis depend on particular models of biological systems developed by others, as stated in the Acknowledgements hereafter. My contribution consists of the new methodology and its application to these cases, in addition to cases developed by myself.

Signed: Brisbane, 19 November 2005

H.J. Stolk	K.E. Gates
Candidate	Principal Advisor

Acknowledgements

I acknowledge the support of my advisors Kevin Gates and Jim Hanan. The inspiring discussions with them and their pertinent comments on my draft writings have significantly contributed to the quality of the present text.

I also acknowledge the contribution of scientists who have developed biological models and shared these models with me to provide me with necessary material for applying the emergent models methodology. Part of the work on insect modelling is based on the monarch butterfly models originally developed by Myron Zalucki and Shane Dorosch. The work on the pea genetic networks is partly based on models developed by Christine Beveridge, Elizabeth Harding-Dun, and Paul Bell. A new pea flowering model has been formulated by myself.

Most of all, I thank my wife Simone and daughters Amaya and Viviane for their loving support and encouragement.

List of Publications and Presentations

Publications by the Candidate Relevant to the Thesis

Stolk H., Gates K. & Hanan J. 2003. Emergent Models in Complex System Simulations of Genetic and Biochemical Networks. 11th International Conference on Intelligent Systems for Molecular Biology, Poster Presentation, Brisbane, Australia.

Stolk H., Gates K. & Hanan J. 2003. Discovery of Emergent Natural Laws by Hierarchical Multi-Agent Systems. Proceedings of the IEEE / WIC International Conference on Intelligent Agent Technology, p. 75-82, Halifax, Canada.

Abstract

This thesis proposes and investigates application of the *emergent models* methodology as a general modelling and simulation methodology for complex systems consisting of many interacting entities. A multi-agent architecture is proposed as software environment to simulate complex systems. This simulation environment mimics a complex system's structure and processes through a model of interacting and concurrently executing software agents and processes.

With the emergent models methodology, software agents and processes exist at various hierarchical levels, like the entities in the complex systems they simulate. Properties and behaviour of agents at each level depend on those on a lower level. The expression *emergent models* means that emergent higher level or macro-level properties and behaviour are derived from lower level or micro-level properties and behaviour. Using evolutionary algorithms, macro-level agents are derived from micro-level agents and can be used in courser-grained simulations. This process can be repeated, thus enabling better understanding, as well as tractable simulations of very complex systems. The inverse problem of inferring micro-level properties and behaviour from observed macro-level properties and behaviour is also solved by the emergent models methodology, using evolutionary algorithms for complex system identification from observed data.

Deriving macro-level emergent models from micro-level models is illustrated with applications to ecosystem simulations, modelling ecosystems as complex systems of interacting plants, animals, and environmental processes.

The inverse problem of finding micro-level models explaining given macro-level data is illustrated by applications to genetic regulatory networks, regarded in the emergent models methodology as complex systems of interacting genes and gene products.

Consideration is also given to design and implementation issues for the development of a simulation environment with emergent models.

Table of Contents

List of Tables & Figures

Tables

Figures

1 Introduction

Complex systems consist of interacting *entities* or components. Generally, a number of different *levels* can be distinguished in a complex system, from the bottom level with the most fine-grained entities, through intermediate levels of composite entities, up to the top level of the whole system. Properties and interactions of entities at a lower or *micro-level* give rise to properties and behaviour of more course-grained entities at a higher or *macro-level*, but the latter are not obviously predictable from the former. At the highest level, there is one macro-level entity consisting of the whole system. This phenomenon of macro-level properties and behaviour arising out of micro-level properties and behaviour, without being immediately apparent from them, is known as *emergence*, a fundamental property of complex systems (see e.g. Cariani 1989; Cariani 1991; Baas 1994; Bonabeau et al. 1995; Baas 1997; Bedau 1997; Bar-Yam 1997; Holland 1998; Rasmussen et al. 2001; Kvasnička & Pospíchal 2002; Kubik 2003).

Historically, the concepts of complex systems and of emergence have played an important role in science, even if not always called by that name. Scientific research has often sought to understand one level of reality in terms of another level. This is illustrated by thermodynamics, describing properties of materials in terms of macroscopic parameters, and statistical mechanics, seeking to explain the laws of thermodynamics

through the microscopic application of Newton's laws (Bar-Yam 1997, p. 58-95). For instance, the ideal gas equation of state relating total number of molecules, pressure, volume, and temperature of a gas is explained by a model of the behaviour of gas molecules, specifically the kinetic theory of gases and pressure (Bar-Yam 1997, p. 74-85). Thus, much of the scientific effort has been dedicated to elucidate the relationships between components of complex systems and the emerging system level behaviour, typically using verbal or mathematical analysis.

Another example is the use of mathematical models in ecology to infer behaviour of populations from that of individual organisms (e.g. Okubo & Levin 2001). Strengths of this use of mathematical models are the generality and analytical explicitness of their results. A limitation is that many simplifying assumptions have to made to keep the analysis tractable. More recently, complex systems have been studied by means of computer simulations, and many observations have been made about emerging patterns, properties, and behaviour of a system as a result of these simulations (see e.g. DeAngelis & Gross 1992; Holland 1998; Ferber 1999; Wooldridge 2002). It is a strength of computer simulation that it makes it possible to build more complex models than was possible with traditional methods. Limitations are that the results of a simulation are sometimes hard to understand, and that they are typically limited to one specific problem.

In this thesis an effort is made to combine the strengths of the mathematical and simulation approaches to scientific discovery, while avoiding their respective limitations, by proposing a general methodology for using computer simulations to study how models of macro-level properties and behaviour of a complex system emerge from the properties and behaviour of the micro-level components of the system. This methodology starts with building multi-agent simulations, computer simulations composed of software entities called agents, which are relatively autonomous, but communicate with their environment and with each other (for detailed definitions see Section 3.2.1 and e.g. Holland 1998; Ferber 1999; Wooldridge 2002). Agents at different levels model micro-level and macro-level components of a complex system. Statistics produced by micro-level simulations are used to discover models describing properties and behaviour of macro-level components. Models discovered in this way are models of the properties and behaviour of higher-level agents as emerging from properties and behaviour of lower-level agents, or *emergent models*, and I refer to the proposed methodology as the *Emergent Models methodology*.

In practice, we often have the inverse problem: data on properties and behaviour of a macro-level entity composed of micro-level entities are available, and we would like to discover micro-level properties and behaviour of the composing entities. This is important to gain insight in the working of a complex

system. In the present work, I address this problem, focussing on genetic regulatory networks, where it is of tremendous importance to be able to discover exactly how they produce observed phenomena, such as genetically determined disabilities or illnesses.

I have selected *genetic programming* (discussed in more detail in Section 5.2.2; and see e.g. Koza 1992; Koza 1994; Koza et al. 1999; Koza et al. 2003) to discover macro-level emergent models, or to discover micro-level models from macro-level data, as it is an all-purpose method with sufficient flexibility to be applicable to many interesting problems. Genetic programming can be applied to many problems, as it performs a search based on trial and error, randomly mutating and recombining building blocks of possible solutions to obtain a best solution to a problem. The building blocks of solutions are provided by the programmer and can be defined in any desired way. In addition, any desired constraints can be imposed on the solutions to be found.

Other model fitting or machine learning methods could be used to good effect for particular cases.

1.1 Hypothesis and Objectives

Multi-agent (Ferber 1999; Wooldridge 2002) or individual-based (DeAngelis & Gross 1992) simulation seems to be a natural approach to develop complex system simulations.

However, an important shortcoming in existing complex system simulations is the lack of a general and practical method to model emergence. The development of a way to derive macro-level properties and behaviour from micro-level properties and behaviour would contribute to understanding emergence in complex systems. From a more practical perspective, knowledge about the macro-level behaviour of a system can be obtained in a much more effective way by simulation of course-grained macro-level entities than by running micro-level simulations of fine-grained entities. On the other hand, when data are available about macro-level properties and behaviour of a complex system, we would like to know how the system works in detail. Thus we need a way to derive micro-level properties and behaviour from macro-level properties and behaviour, which amounts to a method for reverse engineering of the system, or system identification.

Desirable characteristics of a simulation environment enabling implementation of such simulations include interoperability and flexibility. Large scale simulations should be possible by using communicating processes on different computers. If possible, existing simulation models should be used for components of the simulations.

To address these issues, I will demonstrate in the present work that complex systems and emergence can be modelled and elucidated using computer simulation, by

developing a methodology for modelling emergence in hierarchical and distributed multi-agent simulations of complex systems, combined with genetic programming as model discovery tool.

Specifically, I will explore how properties, behaviour, and interactions of a higher-level agent can be derived from properties, behaviour, and interactions of individual lower-level agents, by

developing algorithms to construct higher-level emergent agents from interacting individual agents and applying these algorithms to example problems.

The inverse problem is to infer something about micro-level properties and behaviour of the composing agents assuming macro-level data on properties and behaviour of composite agents are available. Therefore, I will examine how to derive properties, behaviour, and interactions of individual agents from higher-level agent properties and behaviour, by

developing and applying algorithms to construct interacting individual agents from higher-level emergent agents and applying these algorithms to example problems.

I will also discuss issues related to the *design and implementation of a simulation environment* based on the Emergent Models methodology, such as:

- *interactions* of agents with other agents and with the environment, including timing and synchronisation of *message passing* between agents in a distributed programming environment;
- *hierarchical structuring* of agents to manage the volume of message passing, with a view to making possible a distributed implementation of multi-agent complex systems simulations in areas such as ecology;
- *visualisation* of multi-agent simulations with hierarchically structured agents and distributed processing.

1.2 Outline of the Thesis

In this thesis I develop the Emergent Models methodology for complex systems simulation, and illustrate it with examples in simulation of ecosystems and of genetic regulatory networks.

This chapter has stated the present work's contribution, formulating its objective to develop a methodology for modelling emergence in hierarchical and distributed simulations of complex systems.

To place the work in context, Chapter 2 discusses the concepts of complex systems and of emergence as developed in the philosophy of science, while Chapter 3 reviews some illustrative examples of the role of these concepts in scientific practice, including mathematical modelling as well as simulation

of ecosystems and of genetic regulatory networks.

Chapter 4 considers design and implementation issues of a simulation environments for the Emergent Models methodology.

In Chapter 5 I describe the methodology, formulating an algorithmic approach to the discovery of emergent models in multi-agent simulations. This approach encompasses simulation methods for complex systems consisting of many interacting entities. A high level architecture for a software environment to simulate complex systems is proposed.

Chapters 6 and 7 elaborate illustrative examples in ecology and computational molecular biology. The applications to ecological dynamics developed in Chapter 6 illustrate discovery of macro-level population dynamics from individual insect behaviour. The applications to computational molecular biology developed in Chapter 7 illustrate discovery of micro-level interactions in genetic and biochemical networks from macro-level observed phenotype data. Specifically, sample simulations are implemented for the discovery of population dynamics from individual behaviour of butterflies, including interaction with plants, and for reverse engineering of genetic regulatory networks from observed phenotype data in pea plants.

Finally, in Chapter 8 conclusions and perspectives for the future are formulated.

2 Complex Systems, Emergence, and Emergent Models

The central theme of this thesis is emergence in computer simulations of complex systems. Therefore it is important to understand what complex systems are and why emergence is a central concept in studying them. In this chapter I examine the concepts of complex systems and of emergence as used in representative literature on complex systems and on the philosophy of science. I define how these concepts are to be understood for the purpose of the present work, arguing that computer simulation with can contribute to understanding how complex systems work and to clarifying the role of emergence.

2.1 Complex Systems, Levels, and Emergence

Complex systems can be defined in many ways (see e.g. Mitchell 2003). At a minimum, complex systems consist of entities interacting with each other to produce the behaviour of the system as a whole (see e.g. Bar-Yam 1997, p. 1). An important characteristic of a complex system is that the properties and/or behaviour of the whole are *emergent*, that is, they can not be simply inferred from the properties and behaviour of the components (see e.g. Bar-Yam 1997, p. 10; Holland 1998). Many relatively simple entities interact in

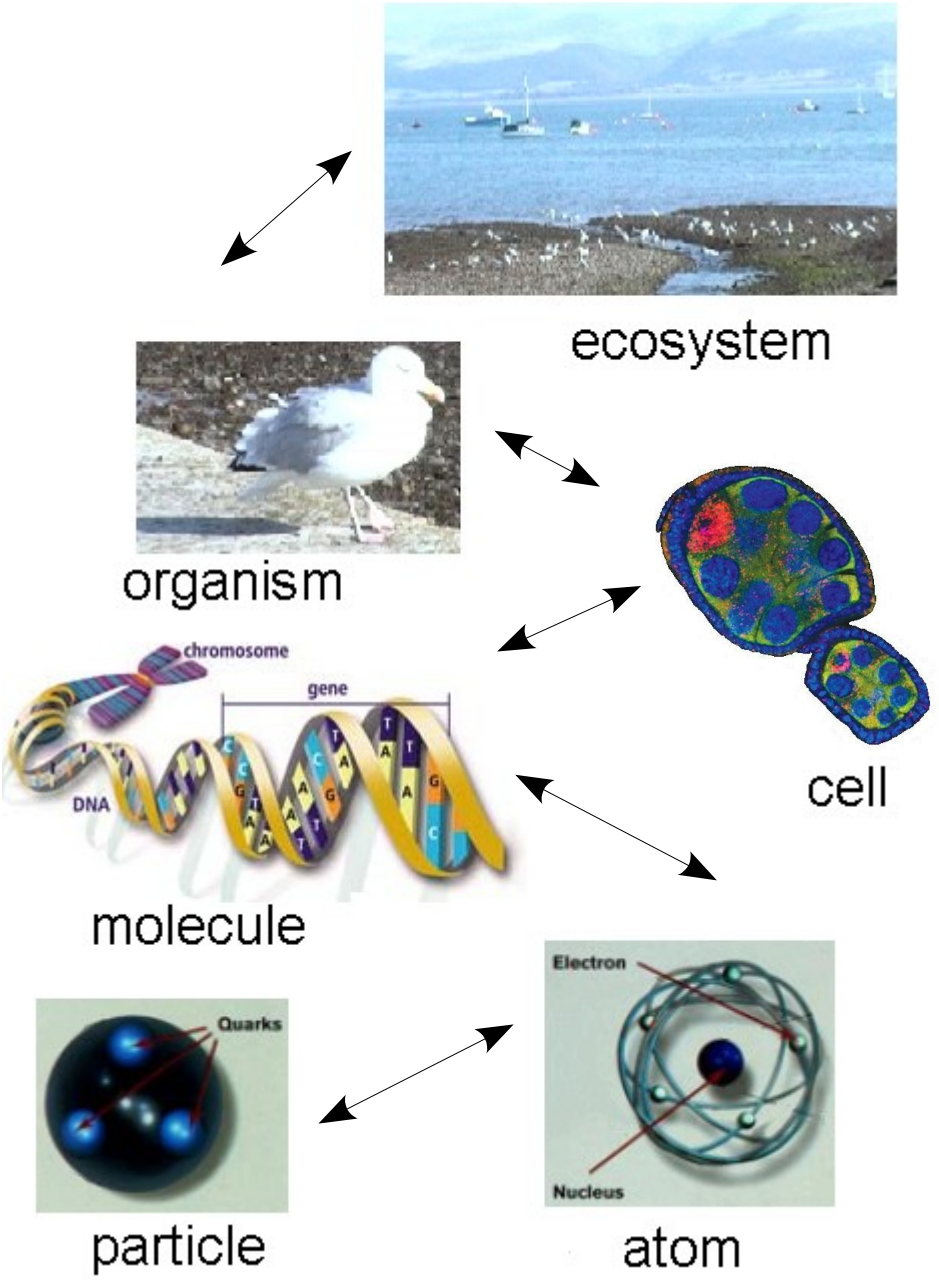


Figure 2.1. Levels in reality. Images: particle and atom from (CERN 2003); molecule from (ORNL 2005); cell from (Ferree et al. 2005); organism and ecosystem by author.

relatively simple ways to give rise to *emergent* phenomena that could not have been predicted easily from the definition of the entities and their interactions. To understand the behaviour of a complex system, we must not only understand the composing entities and their behaviour, but also how they interact and self-organise to determine the emerging state and behaviour of the whole (e.g. Bar-Yam 1997, p. 1; Lam 1997, p. 359; Holland 1998).

The foregoing implies that a complex system consists of a minimum of two *levels*, that of its parts and that of the whole system. Often there are more levels, in the same way that reality as a whole can be thought of as consisting of a series of different levels, entities of each higher level being composed of entities of the next lower level (see e.g. Salthe 1985; O'Neill et al. 1986; Allen & Hoekstra 1992). Some examples are illustrated in Figure 1, showing from the lowest to the highest levels atoms consisting of elementary particles, molecules consisting of atoms, cells consisting of molecules, organisms consisting of cells, and ecosystems consisting of organisms. This hierarchy is extremely simplified and only illustrative, as many more intermediate levels, or alternative hierarchies, can be imagined.

The Emergent Models methodology proposed in this thesis is meant to be as generally applicable as possible. Therefore it is sufficient for the purposes of the present work to assume that a complex system consists of different hierarchical levels, that higher levels are composed of lower-level entities, and that the

properties and behaviour of higher levels are determined by the properties and behaviour (including interactions) of the lower level entities. The methodology refers to emergent models, implying that models at the higher level are determined by those at the lower level, but that the relationship between higher and lower levels is not trivial.

2.2 The Concept of Emergence

Recently, philosophical discussions relating to complexity have begun to reemploy the concept of *emergence* (Cunningham 2001). Emergence had already been proposed and analysed as a philosophical concept by various philosophers and scientists in the 19th and early 20th centuries (see Kim 1999), including John Stuart Mill (1843, Bk. III, ch. vi), C. Lloyd Morgan (1927), and Alfred North Whitehead (1978) .

The concept of emergence can be construed in an ontological and an epistemological sense (Silberstein 2002).Ontological emergence claims that there are cases in which higher-level entities, properties, etc. are governed by higher-level laws not determined by the fundamental laws governing the structure and behaviour of their most basic parts, and thus it is *in principle* impossible to derive or predict higher-level phenomena on the basis of lower-level phenomena (Silberstein 2002, p. 91). Epistemological emergence, on the other hand, maintains that wholes or systems have features that

cannot *in practice* be explained from the features, or the laws governing the behaviour, of their parts (Silberstein 2002, p. 92).

Emergence is to be contrasted with the ontological assumption of reductionism that there is a most fundamental physical level of the world and that everything else can be reduced to elements of that fundamental level. Therefore, according to reductionism, the best understanding of a complex system should be sought at the level of the structure, behaviour, and laws of its component parts plus their relations, and science can in principle have a fundamental theory of everything (Silberstein 2002, p. 80-89). Emergentism, on the other hand, rejects the idea that there is a fundamental level of ontology, claiming that a whole has properties not understandable in terms of the properties of the parts and that there are non-reducible, or emergent entities, properties, and laws. Therefore the best understanding of a complex system must be sought at the level of the structure, behaviour, and laws of the whole system, and science may require a plurality of theories (Silberstein 2002, p. 81, 89-93).

Kim (1999) presents an interesting analysis of the typical emergentist claim of downward causation, the causal influence of emergent phenomena on lower-level phenomena. He concludes that higher-level properties can serve as causes in downward causal relations only if they are reducible to lower-level properties, but that, paradoxically, they are not really “higher-level” any longer in that case (Kim 1999, p. 33).

According to him, we may try to save downward causation by interpreting the hierarchical levels as levels of concepts and descriptions rather than levels of properties and phenomena in the world. We can then speak of downward causation when a cause is described in terms of higher-level concepts in relation to the concepts in which its effect is represented. The same cause may be representable in lower-level concepts and languages as well, and a single causal relation would be describable in different languages.

If Kim's analysis is correct, only an epistemological concept of emergence would make sense. However, we will see that it is not necessary to decide on the philosophical issue whether emergence should be understood in an ontological or in an epistemological sense, to develop a methodology for deriving emergent properties and behaviour from micro-level properties and behaviour. On the other hand, it may well be possible that such a methodology will contribute to clarifying philosophical issues.

In the present work a methodology is developed to determine higher-level properties and behaviour of a system from lower-level properties and behaviour. Therefore it is assumed that certain lower-level properties and behaviour imply specific higher-level properties and behaviour. In Holland's words, we can add levels to a basic system description, and “add new laws that satisfy the constraints imposed by laws already in place. Moreover, these new laws apply to complex

phenomena that are consequences of the original laws; they are at a new level" (Holland 1998, p. 190). With traditional methods it may be impossible or at least hard in practice to derive the higher-level properties and behaviour from lower-level properties and behaviour. *Impossibility or difficulty in practice* to derive (mathematically) the higher-level properties and behaviour from the lower level is enough to make the new methodology interesting. Without committing oneself a priori to a philosophical position about *impossibility in principle* to derive macro-behaviour from micro-behaviour, a methodology able to surmount practical difficulties is interesting for all *practical* purposes.

It is assumed that there is no emergence in some metaphysical sense of a higher level existing independently from lower levels. Indeed, if higher levels were completely independent from lower levels, it would have no meaning to say they emerge from lower levels. In order to make any inference at all between levels, one has to assume that regularities exist, so knowledge of lower levels makes possible knowledge of higher levels. It will be seen that the Emergent Models methodology can discover any existing regularity that can be expressed by building blocks used to describe emergent behaviour. These building blocks are defined as parts of a computer program that can be combined to constitute a complete program. Therefore, the Emergent Models methodology can in principle discover any emergent behaviour

that can be described by a computer program and corresponds to given micro-level system components and their interactions. By virtue of this it could shed some light on the would-be impossibility in principle to derive emergent macro-behaviour from micro-behaviour. Even if it is impossible or impractical with analytical methods to derive emergent macro-behaviour from the underlying micro-behaviour, this thesis will demonstrate that it can be done with a general purpose search strategy such as genetic programming.

So it seems that, if there are regularities relating a macro-level to its underlying micro-level, and if micro- and macro-behaviour can be expressed in computer programs, macro-behaviour can in principle be derived from micro-behaviour.

2.3 Modelling Complex Systems and Emergence

As discussed, there are systems in the 'real world' such as particles interacting through forces to form atoms, atoms interacting to form molecules, molecules forming cells, cells forming multi-cellular organisms, organisms interacting through communication to form ecosystems and societies. In addition, one can think of systems in a 'virtual world', consisting of *models* of the first kind of systems. There is of course but one world, and a model can be seen as a special system in that one world mimicking in some way the behaviour of another system. Thus, a model can itself be considered a complex system.

To illustrate this idea, the solar system can serve as an easily visualised example. The real solar system consists of a star, planets and other heavenly bodies interacting through gravity. Micro-level behaviour of a planet could be modelled by the gravity law of Newton, determining the motion of each planet at any instant of time. The macro-level behaviour of the whole system could be described by the laws of Kepler on the relationships between distances to the sun and revolution times of the planets. We can make a model of this system in a planetarium consisting of balls interacting through forces (in bars, motors, …) designed to mimic the action of gravity. Further, when thinking about the solar system, a complex system is formed in our own brain consisting of, at a low level, neurones and their interactions and, at a higher level, concepts and their interrelationships.

A detailed analysis of emergence in model worlds is found in (Holland 1998). He examines examples of emergence in board games and neural networks. These examples have in common, according to him, that they model the world, that the models consist of multiple interacting copies of a limited number of components, that the configuration of these components changes in time, and that interactions are constrained by a succinct list of rules (Holland 1998, p. 115-116).

In the context of the present work, model systems are computerised simulation models of complex systems. In these simulation models objects or agents represent the entities of a

complex system. Messages exchanged between these objects or agents model interactions between entities. An agent-based simulation model is itself regarded as a complex system, and exhibits characteristics like emergence in a model world, as is rightly stressed by Holland (1998, p.116-124). The foregoing discussion leads naturally to the idea of emergent models.

2.4 Conclusion: Emergent Models

In conclusion, a complex system as a whole is characterised by *emergent properties and behaviour*, which, due to the difficulty of exhaustive analysis, can often best be modelled by simulation of the system. So far, this has been done by running simulations and interpreting the simulation results as emergent properties or behaviour. However, the phenomenon of emergence itself has not been modelled in a satisfactory way. This is a serious shortcoming, as emergence should be regarded as a central concept in complex systems theory. According to Whitehead, it is even the explanatory purpose of philosophy "to explain the emergence of the more abstract things from the more concrete things" (Whitehead 1978, p. 20). As we have seen, the concept of emergence has been defined in the philosophy of science in an ontological and in an epistemological way. In the present work an agnostic point of view is adopted relative to these philosophical issues, defining emergent phenomena as properties and behaviour at a macro-level arising whenever certain properties and behaviour

exist at the micro-level, in other words a regularity between micro- and macro-level can be said to exist.

The present work develops a *novel approach to emergence* by regarding a simulation of a complex system as a complex system itself. Therefore, macro-level entities, objects or agents in a simulation have emergent properties and behaviour with respect to their micro-level composing entities, objects or agents. A similar view on emergence was formulated by Holland (1998), including the idea that a hierarchy of emergent models can exist in a model of a complex system. The present work builds on this idea and extends it by defining procedures for automatically deriving emergent macro-level behaviour from micro-level behaviour.

3 Emergence in Mathematical Models and in Computer Simulation

In this chapter I examine the role of the ideas of complex systems and of emergence in scientific theory and practice. In particular, I review research on mathematical modelling and computer simulation of systems in reality, with a view to assessing the usefulness of these approaches for gaining insight in relationships between micro- and macro-levels of a complex system. It will be seen that in both mathematical modelling and computer simulation useful ideas have been developed in this respect, but that both approaches also have important limitations.

Strengths and limitations of mathematical modelling and of computer simulation will be analysed, using examples in physics and in ecology, with a view to developing an approach building on the strengths of both mathematical modelling and computer simulation, while avoiding some of their limitations.

3.1 Emergence in Mathematical Models

This section reviews work in research on mathematical modelling of complex systems. An illustrative example from physics is a gas, seen as a complex system consisting of molecules. As an example in ecology, an ecosystem is a complex system encompassing living organisms and an

environment of physical and/or chemical processes. In both cases mathematical models have been developed. We will see how such models can provide insight into relationships between levels for these systems.

3.1.1 Mathematical Models in Physics

In physics there are several examples of reasoning about relationships between models of systems at microscopic and macroscopic levels (e.g. Kadanoff 1999). Here we will look at the relationship between statistical mechanics as microscopic model and thermodynamics as macroscopic model of an ideal gas. It will be seen that there is a similar relationship between classical mechanics as a model of a macro-world emerging from microscopic phenomena modelled by quantum mechanics. The purpose is to illustrate how mathematical reasoning has been used to derive macroscopic models from microscopic models.

The relationship between statistical mechanics and thermodynamics is clarified by the kinetic theory of gases and pressure (Bar-Yam 1997, p. 74-85). At the micro-level, a gas is composed of molecules. The standard theory analyses the case of an ideal gas, for which molecules are regarded as free-moving particles. Interactions of gas particles are neglected and they move in a container without collisions.

Pressure of the gas particles on the container walls is given by the force per unit area on the walls, where the force is equal to the actions of the wall needed to reverse the moments of incident particles between times t and $t+\Delta t$ (Bar-Yam 1997, p. 75), or

$$P=\frac{|\boldsymbol{F}|}{A}=\frac{1}{A\,\Delta t}\left|\sum_i m\,\Delta \boldsymbol{v}_i\right|$$

where:

P is the pressure of the gas;

$|\boldsymbol{F}|$ is the magnitude of the force on the wall;

A is a small (so flat) but still macroscopic area of the wall;

m is the mass of a particle;

$\boldsymbol{v}$ is the velocity vector of a particle.

Taking account of the fact that only the velocity component perpendicular to a wall of a particle contributes to pressure, the pressure P of a number N of particles with mass m and average square velocity $<v^2>$ in a volume V can be mathematically derived (Bar-Yam 1997, p. 75-77) as

$$P=\frac{N}{V}m\frac{1}{3}<v^2>$$

Macroscopic energy U of a gas with temperature T can be derived from the microscopic energy of all N particles, using probability arguments (Bar-Yam 1997, p. 66-73), as

$$U = \frac{3}{2} N k T$$

where:

k is the Boltzmann constant.

The energy can also be written as the number of particles N multiplied by the average kinetic energy of a particle (Bar-Yam 1997, p. 80), or

$$U = N \frac{1}{2} m < v^2 >$$

Substituting, we obtain

$$P = \frac{1}{V} \frac{2}{3} N \frac{1}{2} m < v^2 > = \frac{1}{V} \frac{2}{3} U = \frac{1}{V} \frac{2}{3} \frac{3}{2} N k T$$

or the macroscopic ideal gas equation of state relating the macroscopic quantities of pressure P, volume V, total number of particles N, and temperature T

$$P V = N k T$$

Thus, starting from a microscopic model of the properties and behaviour of individual gas molecules, a macroscopic model of the properties and behaviour of a gas can be mathematically derived.

A similar result can be obtained for a very simplified situation with a quantum mechanical model of gas molecules, as shown by Forster & Kryukov (2003), who derive the average quantum mechanical pressure $<P>$ of a particle in a one dimensional box as a function of the quantum mechanical mean energy $<E>$ and the volume V

$$<P>=2<E>/V$$

They conclude that laws in physics "commonly equate macro-quantities with ensemble averages, time averages, and quantum mechanical averages of micro-quantities" (Forster & Kryukov 2003, p. 1048).

Again it is clear that mathematical derivation can be used as a tool to establish relationships between micro- and macro-levels, at least when suitable simplifications (sometimes as extreme as a one dimensional box) are made.

This section has illustrated how reasoning between micro- and macro-levels in reality is part and parcel of the practice of science, and how this reasoning has typically taken a mathematical form in theoretical physics. Macro-level laws or

models can be derived rigourously from micro-level models., and general statements can be made about relationships between micro- and macro-levels. However, the systems studied in this way have to be very simplified, idealised systems, such as an ideal gas, otherwise mathematical analysis becomes intractable.

3.1.2 Mathematical Models in Ecology

Mathematical models incorporating reasoning between levels have also been developed in the study of populations of animals and plants (see e.g. Kareiva & Odell 1987; Turchin 1998; Huffaker & Gutierrez 1999, p. 463-534; Turchin & Omland 1999; Okubo & Levin 2001). In this case there are organisms (animals) at the micro-level and groups or populations of organisms at the macro-level. Here I examine typical models of this kind with the purpose of illustrating how mathematical reasoning can be used to derive macroscopic models of emergent population properties and behaviour from microscopic models of individual organism behaviour.

The examined models are mathematical diffusion models (Okubo & Levin 2001), in which individuals of a population are treated in the same way as particles of a fluid. A description of individual, typically random, behaviour of the individual particles or organisms gives rise to fluid level or population level dynamic laws. Thus, dispersal of animal populations has been analysed by starting with a direct analogy to the random walk or physical

diffusion, with additional consideration of intra- and interspecific population interaction (Okubo & Grünbaum 2001).

A simple one-dimensional random walk model starts with an equation governing the statistics of a particle (Okubo & Grünbaum 2001, p. 133-134), the probability $p(x,t)$ that a particle released from the origin at time $t=0$ reaches point x at time t being given by

$$p(x,t)=\alpha\, p(x-\lambda,t-\tau)+\beta\, p(x+\lambda,t-\tau)$$

where:

α and β are the probabilities that a particle will move to the right or the left in time unit τ, with $\alpha+\beta=1$.

Assuming λ is small compared to x and τ is small compared to t, and each term on the right-hand side of the equation can be expanded in a Taylor series, the following diffusion equation can be derived:

$$\frac{\partial p(x,t)}{\partial t}=-u\frac{\partial p(x,t)}{\partial x}+D\frac{\partial^2 p(x,t)}{\partial x}$$

where:

D is the diffusion coefficient;

u is the advection rate.

Multiplying p by the number of particles, this equation is transformed into an equation for particle concentration S:

$$\frac{\partial S(x,t)}{\partial t} = -u\frac{\partial S(x,t)}{\partial x} + D\frac{\partial^2 S(x,t)}{\partial x^2}$$

We see that, assuming a simple model of individual behaviour, a mathematical derivation of emergent population behaviour is possible.

More realistic models would take into account interaction between individuals, stimuli, and animal response to the environment (Skellam 1972, 1973, cited in Okubo & Grünbaum 2001). Indeed, more complex mathematical models have been developed to take account of aspects such as statistical correlation between steps, non-homogeneity of the parameters of the walk, the effect of non-random forces acting on an individual, interference between individuals, and non-uniformity of the use of space and time (Okubo & Grünbaum 2001, p. 134-140).

An example of such a more complex mathematical model is Patlak's model (see Okubo & Grünbaum 2001, p. 140-143; Patlak 1953), an extended random walk model, taking into account correlation between successive steps, non-isotropic environment, and external forces. Patlak's model has been applied by Turchin (1991) and Grünbaum (1998) to foraging insects.

Its assumptions about individual particle (or animal) movement lead to the following equation describing particle concentration S:

$$\frac{\partial S(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left(u_e(x,t)S(x,t)\right) + \frac{\partial^2}{\partial x^2}\left(D(x,t)S(x,t)\right)$$

where:

u_e is an advection term dependent on space and time;

D is diffusivity, also dependent on space and time.

Keller & Segel (1971, cited in Okubo & Grünbaum 2001, p. 151-158) have applied a biodiffusion model to chemotaxis in bacteria. Their model assumes a bacterial random walk with a step length l and a frequency of changing step direction $f(C(x))$ depending on the concentration $C(x)$ of a chemical attractant. They derive from these assumptions the following equation describing the chemotaxis of bacteria:

$$\frac{\partial S(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left(\chi(C(x))\frac{C(x)}{dx}S(x,t)\right) + \frac{\partial}{\partial x}\left(\mu(C(x))\frac{\partial S(x,t)}{\partial x}\right)$$

where:

$\chi \equiv -\frac{1}{2}l^2\frac{df(C)}{dC}$ is called the chemotactic coefficient;

$\mu \equiv \frac{1}{2}l^2 f(C)$ is the motility or diffusivity.

These quantities all vary with position. μ and χ depend on the step length and frequency of direction change.

The two terms in the equation thus describe the effects on bacterial density of diffusion or motility, depending on the concentration of the attractant, and of advection, depending on the concentration gradient of the substance. The frequency of direction change determines the advection behaviour. If that frequency decreases with increasing concentration of the attractant, advection is in the direction of increasing concentration.

An example of a more complex model of the tactic behaviour of micro-organisms is that of Oosawa and Nakaoka (1977, cited in Okubo & Grünbaum 2001, p. 152), where individual cells have internal state variables. Environmental conditions such as temperature gradients influence a cell's internal state, which in turn determines the cell's behaviour.

These mathematical models of biological diffusion provide examples of deriving macro-level behaviour of a population from micro-level behaviour of organisms. As in physics, relationships between micro-level and macro-level are defined with mathematical rigour, and the models apply to any system satisfying the assumptions made. However, these assumptions may involve oversimplification in order to make mathematical analysis possible. For example, every organism is assumed to behave in the same simple, idealised way.

3.1.3 Strengths and Limitations of Mathematical Modelling

An important strength of mathematical models is that, given their assumptions, they lead to general results. The main features of a system are described in a precise way, and it is clear exactly how well-defined assumptions about micro-level properties and behaviour imply equally well-defined macro-level properties and behaviour.

A limitation is that many simplifying assumptions have to be made to keep the analysis tractable. This is clear from the above described physical models of an ideal gas and from the ecological models of movement as a simple random walk with relatively straightforward extensions.

It could even be questioned whether mathematically derived macro-level behaviour in such highly simplified models is still emergent behaviour in the sense of being not easily predictable from micro-level behaviour. The very existence of a rigourous mathematical dependence of macro-level behaviour on the micro-level would seem to imply that we really have reductionist models here, not models with emergence.

One way to overcome the necessity of (over)simplifying is using computer simulation, which can incorporate any desired complexity. As we will see, this is especially the case for simulation with individual behaviour models.

3.2 Emergence in Computer Simulation

Where the previous section reviewed research on mathematical modelling of complex systems, this section considers computer simulation, focussing on multi-agent simulation, which is compared to other simulation methodologies. Individual-based simulation in ecology is also discussed as a methodology similar to multi-agent simulation in a particular application area.

Ecosystems are an interesting application domain of Emergent Models, because in this area relationships between levels have been analysed by mathematical modelling, as seen in the previous section, but have also been extensively studied using individual-based computer simulation of interacting living organisms and their environment.

3.2.1 Multi-Agent Simulation

This subsection describes multi-agent simulation. In the following subsections this simulation method is compared to other simulation methods, to assess its suitability for complex systems simulation and in particular for elucidating emergence in complex systems.

In *multi-agent simulation* active entities in the world and their behaviour are represented in a computer as software entities called agents, making it possible to represent a

phenomenon as the result of the interactions of a set of autonomous agents (Ferber 1999, p. 36; see also e.g. Holland 1998, p. 116-118; Wooldridge 2002). It can be applied to any system composed of individual entities.

Multi-agent simulation is a special case of multi-agent systems, which have been developed in the field of distributed artificial intelligence in computer science. A *multi-agent system* is a computing system of artificial entities in an environment or space. Agents and other objects are situated at positions in the environment. These agents and objects are linked to each other by relations. Agents can perceive, produce, consume, transform, and manipulate objects. The reactions of the world to agents' actions, or the “laws of the universe”, are also represented (Ferber 1999, p. 4, 11).

An *agent* is an entity with tendencies or objectives it tries to satisfy by acting in an environment and communicating with other agents. Doing this, it takes account of its resources and skills. Its actions depend on its perception and representation of the environment, and on communications it receives (Ferber 1999, p. 9). In other words, an agent is proactive, having goal-directed behaviour and taking the initiative to satisfy its objectives. It is also reactive, perceiving and responding to its environment. Finally, it has social ability, interacting with other agents (Wooldridge & Jennings 1995; Wooldridge 2002, p. 23).

According to the degree in which agents possess their

defining characteristics, a distinction can be made between cognitive and reactive agents. *Cognitive agents* have been developed in the tradition of artificial intelligence, emphasising knowledge and goal-directed behaviour (Ferber 1999, p. 16). *Reactive agents*, typically used in the artificial life tradition, are based on the idea that agents can be very simple and do not need intelligence themselves for the system as a whole to have intelligent behaviour. They have a stimulus-response behaviour, communicate through simple signal propagation, and have no internal representations of their environment (Ferber 1999, p. 27-28).

Examples of multi-agent simulation are a multi-agent population model representing individuals as agents and the number of individuals in a given species as a result of the behaviours of all agents (Ferber 1999, p. 36); a multi-agent simulation model of a human society representing individual people (or organisations and similar entities) as agents and phenomena such as growth of social complexity as a result of their behaviour (Wooldridge 2002, p. 259-263); or a multi-agent simulation of humans interacting with a technical system (Davidsson 2001).

A comparison with other simulation methodologies will clarify the distinctive characteristics of multi-agent simulation determining its suitability for simulation of complex systems.

3.2.2 Multi-Agent Simulation and Classical Simulation

Multi-agent simulation is developing in a context where computer simulation, generally based on mathematical relationships between variables measured in reality, has already made significant advances in theory and practice (Pavé 1994, cited in Ferber 1999; Zeigler et al. 2000). Nevertheless, in simulation models used so far, often based on differential equations and numerical simulation techniques, all variables and parameters are on the same level of analysis, so micro- and macro-levels cannot be linked. Therefore it is difficult or impossible to explain collective phenomena such as population size emerging from the interaction of individual decisions and behaviours. For example, parameters often represent complex behaviours in a single number and are difficult to estimate (Ferber 1999, p. 35-36).

In multi-agent simulations, on the other hand, representations are at the level of individuals, directly represented as agents and interacting with each other and the environment. Overall structures of complex situations on the macro-level emerge from interactions of individuals, or from models on the micro-level, thus breaking the level barrier in classical modelling (Ferber 1999, p. 36-37). According to another author (Davidsson 2001), multi-agent simulation has as advantages compared to traditional simulation that it is more natural for modelling humans and animals; that it supports

distributed computation in a very natural way; that it is possible to add or remove agents during a simulation without interruption; and that it is possible to specify the simulation model and software on a very high level (beliefs, intentions, etc.), increasing understanding by non-programmers.

Thus, multi-agent simulation has clear advantages for simulation of complex systems and studying emergence in such systems, compared to classical simulation.

3.2.3 Multi-Agent Simulation and Object-Oriented Simulation

Comparing multi-agent simulation and object-oriented simulation, there is a vague borderline between an agent and an entity that is just an object, an agent having higher scores than an object on the dimensions of reactivity; proactiveness; communication and social ability; spatial explicitness; mobility; and/or adaptivity (Wooldridge 2002, p. 23; Davidsson 2001). Agents react more than objects to their environment; are more proactive, having goal-directed behaviour; communicate with other agents; are located at a certain position and move about in the environment; and can better adapt to changing circumstances. So, while using agent-based concepts and methodology to develop a complex system simulation, objects can be considered as a limiting case of passive agents only responding to messages in the form of method calls from a controlling program or other agents or objects.

This situation is illustrated by the following two examples of agent-based simulations used for modelling purely physical systems, where it is debatable whether we really have agents or just objects. One of these examples is also illustrative of a possible approach to emergence in such simulations.

In the RIVAGE project at the French research organisation ORSTOM a multi-agent system is applied to model complex phenomena in hydrology (Servat et al. 1998a; 1998b). Individual water entities or water balls are considered as agents with interactions leading to water flows and emergent phenomena such as ponds and ravines. Explicit attention is given to viewpoints at different levels: water balls can form groups, which are themselves agents at a higher level, for example pond agents.

Another example of application of multi-agent simulation to a physical system is described by Breton et al. (2001). Sand grains optimise their own energy in a sand pile in a distributed fashion, taking account of the properties of neighbouring grains. This leads to better performing simulations than traditional methods based on solving the physical equations for the whole sand pile.

Thus, there appears to be no fundamental difference between multi-agent simulation and object-oriented simulation for simulation of complex systems, and both can be used to study emergent phenomena. We can consider object-oriented

simulation as a limiting case of multi-agent simulation with relatively simple agents. Alternatively, multi-agent simulation can be seen as a special kind of object-oriented simulation with relatively complex objects.

3.2.4 Multi-Agent Simulation and Artificial Life

Artificial life is a research area studying the biology of the possible by synthesising life-resembling processes or behaviour in computers or other media (see e.g. Emmeche 1994). It is assumed that artificially created components can exhibit a behaviour just as genuine as that of real-life organisms, because life is a process and the form of this process, not the matter, is the essence of life. Some artificial life researchers even claim to create life forms that are just as alive, but differ from life on earth as we know it. Like exobiology, which studies life supposedly existing on other planets, it is a "biology of the possible" (Emmeche 1994). Its objective consists of discovering general principles of life, not restricted to any particular instance of life. Artificial life is constructed using a bottom-up method leading to processes executed in parallel with self-organising, *emergent* behaviour (Emmeche 1994). Some researchers conceive life as an emergent property of an artificial computational chemistry (Adami 1998).

It is not clear how the strongest artificial life claim, that entities in an artificial life program are indeed living beings, can be substantiated. Unless and until this can be demonstrated,

computer simulations should be regarded as indeed simulations of something else, and not as a reality in themselves. Thus, artificial life simulations are really one kind of multi-agent simulation.

Like multi-agent simulation, artificial life offers obvious possibilities to analyse relationships of emergence between levels. Indeed numerous articles have been written by artificial life researchers on emergence.

Cariani (1989; 1991) distinguishes *computational emergence*, *thermodynamic emergence*, and *emergence relative to a model*. Computational emergence means that "emergent, complex global forms can arise from local computational interactions" (Cariani 1991, p. 776). Thermodynamic emergence is "the formation of new physical structures in the world at large" (Cariani 1989, p. 151). Emergence relative to a model is "the deviation of the behavior of a physical system from an observer's model of it" (Cariani 1991, p. 779).

Baas (1994; 1997) bases the definition of emergence on a general notion of (abstract or physical) structures as primitive entities. A set of first-order structures is complemented with observational mechanisms to obtain properties of the structures. Further adding interactions between these properties, second-order structures are obtained. Emergent properties are observed properties of these second-order structures that are

not observed properties of the first-order structures. Repeating the process, second-order structures can in turn form new second-order properties and interactions to give rise to third-order structures, and so on. Baas further distinguishes *deducible or computable emergence* , if a computational process can determine emergent properties from the lower level, and *observational emergence* otherwise (Baas 1994, p. 518-519. In (Rasmussen et al. 2001) these concepts are applied to dynamical hierarchies in molecular systems, by treating monomers as first-order structures, polymers as second-order structures and micelles, a kind of membrane-like polymer aggregates, as third-order structures. Polymers have emergent properties such as elasticity and radius of gyration, and micelles have a characteristic size distribution, an inside/outside and permeability.

In the conceptual framework defined by Bonabeau et al. (1995a; 1995b) levels of organisation and the importance of observation to detect emergence are emphasised in a similar way.

Bedau (1997) defines the concept of *weak emergence* for a system composed of micro-level parts with a time evolution governed by a microdynamic. Macrostates of such a system are “structural properties constituted wholly out of its microstates” and “typically average over microstates and so compress microstate information” (Bedau 1997, p. 377). A macrostate is defined as weakly emergent if and only if it can be derived from

the system's external conditions but only by simulation. Thus, “weakly emergent properties can be derived (via simulation) from complete knowledge of micro-level information” (Bedau 1997, p. 393).

Kvasnička & Pospíchal (2002) show how modularity in genotype-phenotype mappings can emerge in a simulation model of evolution of genotypes and phenotypes, and so implicitly use computational emergence as defined by Cariani.

Kubik (2003) defines *basic emergence* as a property of a multi-agent system, formally modelled as a grammar system. In such a grammar system he defines the sum of agents' behaviours as “a sum of conditions the agents can bring about in the environment if they act individually in the environment” (Kubik 2003, p. 51). Now a multi-agent system as a whole has the property of basic emergence if it can generate a behaviour that can not be generated by the superimposition or summation of individual agents' behaviours (Kubik 2003, p. 52-53).

For the purpose of the present work, emergent properties are higher-level properties that can be observed in simulation models. They are defined in a pragmatic way as properties an observer chooses to observe at a higher level of a system that can not be easily deduced from lower levels. This is consistent with most definitions described above and avoids the more metaphysical connotations of emergence implied in concepts such as thermodynamical emergence, or a kind of emergence in

nature that could not be modelled. Emergence is a useful concept if there are properties at higher levels of a system that are not easily derived from lower levels, even if such a derivation is possible in principle.

3.2.5 Multi-Agent Simulation and Active Walks

Active walk simulations (Lam 1997) can be regarded as another special kind of multi-agent simulation, intended to describe self-organising processes in nature and to treat problems in pattern formation, self-organisation, and the dynamics of complex systems. An active walker moves in a landscape, changing the landscape at each step, and its next step is influenced by the changed landscape. Each active walker communicates with and thus influences other walkers through the landscape (Lam 1997, p. 360).

Active walkers can be regarded as agents, having goals and interacting with their environment and with other agents/ walkers, with the restriction that communication between agents is always mediated by the environment. Many interactions between agents can probably be described in that way, so the idea of an active walker can be useful for complex system simulations. However, active walks do not appear to offer a fully general description of agent interactions, because of the constraint that all communication has to be described as modifying and reacting to a landscape.

3.2.6 Assessment of Multi-Agent Simulation

The foregoing arguments and examples demonstrate that multi-agent simulation can be a useful methodology to develop simulations of complex systems. It provides a good basis to support analysis of relationships between micro- and macro-levels in a complex system, and in particular of emergence of macro-level properties and behaviour from micro-level properties and behaviour.

An interesting example of a multi-agent simulation with explicit representation of different space and time scales, and emergence of at least macro-level properties (if not behaviour) from micro-level properties and behaviour, is provided by the hydrological simulation model described by Servat et al. (1998a; 1998b). Properties of group agents emerge from individual behaviour, but the group agents themselves are passive, really just descriptive of collective properties of individual agents.

In order to model emergence in a satisfactory way, a complex systems simulation methodology should explicitly address the relationship between levels in a simulation of not only the properties, but also the *behaviour* of micro- and macro-level agents.

3.2.7 Individual-Based Simulation in Ecology

This section examines individual-based simulation modelling in ecology as a methodology for complex systems simulation, focussing on its potential to elucidate emergence in ecological systems. While multi-agent simulation has been developed in the tradition of artificial intelligence research, individual-based simulation has been developed in that of ecological research.

In ecology it has been recognised since the 1980s that many simplifying assumptions used in mathematical models were not compatible with the reality of ecological systems (see e.g. DeAngelis & Gross 1992). One of the most important of these unrealistic assumptions was that individual members of populations can be aggregated into a single state variable representing population size, neglecting individual differences. A response to this inadequacy of models has been to develop models based on processes at the level of individual organisms, a number of which are reviewed in (DeAngelis & Gross 1992). In many of these models local interactions of individuals with their nearest neighbours are considered. Some also take into account hierarchical structuring of ecosystems (see also Ehleringer & Field 1993).

Individual-based simulation in ecology is really the same as multi-agent simulation. Modelling processes at the level of an individual as well as interactions between individuals, then

building a computer simulation on the basis of that model, amounts to nothing else than developing a multi-agent simulation of a natural system. For example, although not called by that name, the individual-based simulation models Simpdel, of panthers and deer, and SORTIE, of forests, can be seen as multi-agent simulations in ecology, if we consider animals and trees, respectively, as agents.

Simpdel (Abbott et al. 1997) is a spatially explicit individual-based simulation model of Florida panthers and white-tailed deer in the Everglades and Big Cypress landscapes. Deer are simulated as looking for food, growing, reproducing, etc. Vegetation is modelled taking into account growth, forage quality, etc. The landscape is distributed over several processors on a parallel computer.

SORTIE (Deutschman et al. 1997) is an individual-tree model of north-eastern United States forests. It explicitly represents the complex interplay between the local environment and each individual in the community. It has made possible analyses of the critical features controlling forest dynamics. However, the authors warn that the complexity of the models can make it difficult to understand them and their results. Visualising a complex model is regarded by them as a key part of model exploration.

Simpdel and SORTIE are typical individual-based ecological models. Individuals are modelled in a detailed way,

and simulation experiments can be conducted to observe the results of various assumptions. Although macro-level emergent properties and behaviour can be observed in the simulation results, there is no general way to derive these from the micro-level.

More explicit attention is given to relationships between levels and to hierarchical structuring of individual-based models by Palmer (1992), who aims to take into account the time and space scales of interactions in simulating nature. An example model consists of a top-level object called season, composed of several locations. Each location consists of several niches, in turn containing organisms. The behaviour of organisms is determined by resource availability in their niche. Organisms can live or die, reproduce, transform themselves into another type of organism, and also migrate to other niches or locations.

Palmer's model suggests a way of hierarchically structuring multi-agent simulations. However, it is not clear how higher level entities in Palmer's model determine their behaviour other than by delegating tasks to lower level entities. It would be desirable to consider higher level entities as agents in their own right, with their own properties and behaviour.

Scaling issues in ecology are also considered by Reynolds et al. (1993), who focus on various levels in an ecosystem from the individual plant to the ecosystem level. They regard ecosystems as hierarchically structured from leaf and other

plant organs through canopy, whole plant, and community to the whole ecosystem. For lower level models to be useful for higher levels, it has to be determined how lower-level effects can be translated to higher levels in the hierarchy (Reynolds et al. 1993, p. 130).

The authors take as an example the model STAND of long-term stand dynamics in chaparral plant communities, which operates on an annual time step and includes the simple phenomenological sub-model PHENPLT of plant growth (Reynolds et al. 1993, p. 134-135). PHENPLT describes the change in weight W of the average plant in the community as a function of a growth rate R and a maximum attainable weight W_{max}:

$$\frac{dW}{dt} = R(W_{max} - W)$$

At the hierarchically lower level of the individual plant, the model GEPSI, a detailed mechanistic plant growth model that runs on a daily basis (Reynolds et al. 1993, p. 135), is used to predict R for chaparral plants as a function of solar radiation L, nitrogen N, and carbon dioxide CO_2:

$$R = g(L, N, CO_2)$$

GEPSI was run under a wide range of these environmental

variables to calculate a response surface for R. A simple hyperbolic function g was then fit to this response surface and substituted for the fixed parameter R in PHENPLT (Reynolds et al. 1993, p. 135). In an application of the models to stand development after a fire, interesting interactions between carbon dioxide and nitrogen availability emerge that would not have been predicted from GEPSI models alone. For example, total biomass and leaf area index are increased significantly by carbon dioxide only when the stand is strongly nitrogen limited. This contrasts with the behaviour of individual plants predicted from GEPSI, which suggests that growth rates are most stimulated by carbon dioxide when nitrogen is abundant. It appears that the more open stands that develop under nutrient limitation are more responsive to carbon dioxide because they are less light limited. Stands with greater nutrient availability quickly attain a closed canopy where production is limited by competition for light (Reynolds et al. 1993, p. 135).

We have a good example here of combining models at different hierarchical levels. Stand-level effects of altered resource availability could not have been investigated in the original version of STAND because plant growth responses to resources were not represented. On the other hand, population and community effects of resource availability cannot be addressed by GEPSI, since community level processes like mortality and competition among plants are not included (Reynolds et al. 1993, p. 135). The advantages of combining

analysis at different levels are clear from this example, even if the relationship between micro- and macro-levels is modelled in a very straightforward way and limited to the dependence of a single parameter in the macro-model on a micro-level model.

In conclusion, it is clear that individual-based models and computer simulations make it possible to study complex ecological situations that would be difficult to analyse intuitively or even mathematically. However, it is typically difficult to assess to what extent such models of particular systems can be generalised, and how inferences between micro- and macro-levels can be made. Palmer (1992) and Reynolds et al. (1993) explicitly address the issue of relationships between levels in their respective specific models, making a first step towards the development of a general methodology for establishing relationships between levels in any complex system simulation.

3.2.8 Strengths and Limitations of Computer Simulation

Computer simulations can be carried out without many of the simplifying assumptions necessary for mathematical modelling. However, it is difficult to establish the generality of results of a particular simulation and even of the applicability of a particular simulation system to other cases (see e.g. Robinson et al. 2004). It would be desirable to have a general way to derive macro-level agent properties and behaviour from an individual-based simulation.

3.3 Conclusion

Comparing mathematical modelling and computer simulation as tools for studying complex systems and emergence, we have seen that both have strengths and limitations. A strength of mathematical models is that they lead to general and clearly understandable results. A limitation is that many simplifying assumptions have to be made to keep the analysis tractable. Computer simulations, on the other hand, can incorporate any desired complexity and be carried out without many of the simplifying assumptions necessary for mathematical modelling. However, their results are less general and can be hard to understand, if the simulation model is complex.

Therefore, a general methodology is needed to derive macro-level properties and behaviour from individual micro-level properties and behaviour in complex system simulations, combining the strengths and avoiding the limitations of both mathematical modelling and of computer simulation as they currently exist.

4 Simulation Environments for Emergent Models

Simulations of complex systems with emergent models have to be implemented. Therefore, design and implementation principles of complex systems simulation environments are considered in this chapter, taking into account the special requirements of simulations with emergent models.

Some relevant principles are highlighted by a hypothetical case study of ecosystem simulation. The components of an ecosystem are themselves complex phenomena. To name but a few, we have plants, animals, unicellular organisms, soils, water, atmospheric conditions, light. For many of these phenomena specialist simulation models have been developed and implemented in software systems. The modelling methodologies are different and include perhaps structural models for plants, behavioural models for animals, reaction-diffusion models for unicellular organisms, finite element models for soils and water, partial differential equation systems for meteorological conditions, and physical models for light. The implementation in software is probably in different programming languages and on different hardware platforms.

Suppose we want to simulate an ecosystem, without redoing all the work already done just to have yet another model, implemented in our own favourite language and running

on the computer we happen to have access to, but which could not be used in a different environment. Instead, we would like to access existing resources with all the specialist expertise and years of effort invested in them, and combine these as needed for the research problem under study. The vision is that of a researcher trying to get insight into a complex ecological phenomenon, using a control panel program on a computer with access to a local area network and the internet. The local computer has an installation of specialist software for plant modelling written in C. On the other desk in the room, there is a graphics workstation with excellent rendering capabilities and the latest version of a high-performance visualisation tool. A colleague in the physics department has just mailed about the latest version of an atmospheric model written in FORTRAN. The researcher now selects options from the control panel to specify the different software components to be used for the various parts of the problem to be solved, and runs the simulation. While watching the simulation develop, it is possible to interact with it, for example zooming in on interesting details, adding insects or other animals, and so on.

From this hypothetical case we can infer that relevant design and implementation principles of an environment for simulation of complex systems are related to its global architecture, to interactions between components of the environment, to timing and synchronisation of these interactions, to hierarchical structuring, and to visualisation of

results and control or steering of the simulations. For the purpose of demonstrating the methodology developed in this thesis, I have developed prototype applications rather than a complete simulation environment. The principles discussed in this chapter have been used while selecting the software for the prototype applications.

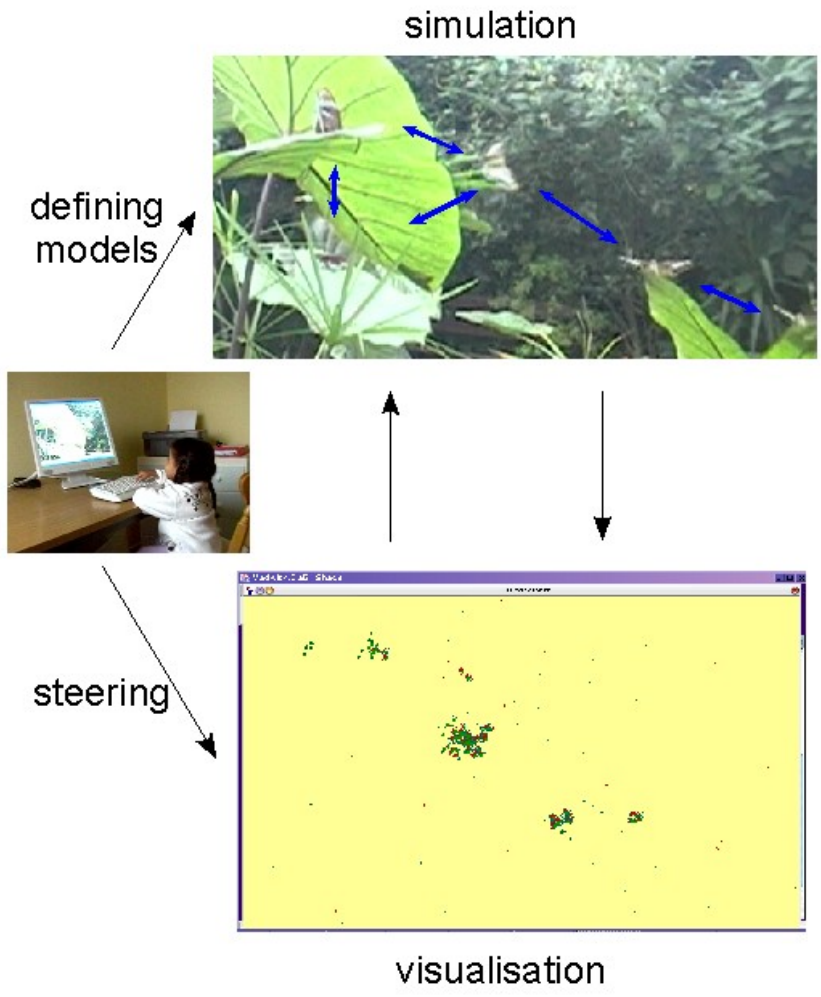


Figure 4.1. Global architecture: components and interactions.

4.1 Global Architecture and Interactions

The diagram in Figure 4.1 represents a structural view of a global architecture for a problem-solving environment for simulation of ecosystems. The user sets up the simulation by selecting agents and environmental processes to be modelled and then defining their interactions. After that is done, the user starts the simulation. During the simulation, all organisms (plants and animals) interact with each other and with the environment. Visualisation objects generate images, while the simulation is running, providing useful insights in the simulated processes. The user can observe these images and also interact with them to steer the simulation.

In this section I review the design and implementation of problem-solving environments for simulation of ecosystems and identify relevant design principles.

4.1.1 Problem Solving Environments

The individual-based models in ecology discussed in section 3.2.4 are implemented in a fairly straightforward way. In its implementation, the Simpdel program uses one specific computer and its communication facilities (Abbott et al. 1997). The SORTIE model is programmed as a stand-alone C application without attempts at parallel execution (Deutschman et al. 1997). A possible improvement of flexibility in such simulations could be achieved by using general purpose

communication facilities, enabling parts of the simulation to be executed on different machines and possibly programmed in different languages.

Efforts to improve flexibility have been made by developing integrated problem-solving environments. I here review several existing integrated problem-solving environments, including software systems that could be used as components of an integrated problem-solving environment, with a view to exposing the inspiration that can be drawn from them and also identifying shortcomings and opportunities for improvement. Often, these opportunities for improvement are made possible by technological development since the time of formulation of these environments. An introduction to the subject in (Houstis et al. 1997) identifies the integration of multiple interacting applications in multidisciplinary problem-solving environments as a major research challenge.

The *WISE* problem-solving environment (Knox et al. 1997) provides an interesting environment with the possibility to couple models from different sources to give results that would not be possible to obtain with each model separately. It uses object-oriented software to link research models from various sub-disciplines for more comprehensive ecosystem simulation and visualisation (Knox et al. 1997). WISE is a flexible system for coupling various models by a mechanism that allows any model instance to query another model instance dynamically for the most recent values of its state variables (Knox et al. 1997). It

also features shared visualisation tools and a simple encapsulation process for running a legacy model within the application (Knox et al. 1997). However, communication facilities are designed ad hoc and limited to a single host. For example, reporting state information between coupled models is done using message queues for inter-process communication on the same host, for which a message queue class was written, relying on the C++ Standard Template Library (Knox et al. 1997). Flexibility and scalability could be improved using general purpose communication facilities enabling different models and tools to be run on different machines.

The *SciNapse* code generation system (Akers et al. 1997) and the *Net Pellpack* problem-solving environment (Markus et al. 1997) are specialised problem-solving tools for particular classes of problems, not general purpose simulation environments. SciNapse transforms high-level descriptions of partial differential equation problems into customised, efficient, and documented C or FORTRAN code, bridging the gap between high-level problem description and source code (Akers et al. 1997). With Net Pellpack, scientists can solve complex partial differential equations with common web browsers that support Java applets communicating with a solver (Markus et al. 1997). The environment provides computational facilities for solving a target class of problems and a user interface, so users do not need specialised knowledge of the underlying computer hardware and software (Markus et al. 1997).

The *DEVS* simulation environment (Zeigler et al. 1997) is a high performance simulation environment with hierarchical features. It has its own communication library, and the accent is heavily on high performance in the context of a C++ programming environment (Zeigler et al. 1997). Therefore, to integrate existing models in the *DEVS* framework would require making them conform to a C++ environment and to the DEVS communication library. A general purpose simulation environment should have good facilities for integrating different components and not be dependent on one programming language or a special purpose communication library.

The *CPFG* L-system plant modelling environment (Prusinkiewicz & Hanan 1989; Prusinkiewicz et al. 1990; Mech & Prusinkiewicz 1996; Prusinkiewicz et al. 2000) has the same limitations of dependence on a particular programming language and an ad hoc communication library. In the CPFG environment plants are modelled as L-systems. Coupling with the environment is possible through environmental modules in the L-system, using custom communication functions written in C (Mech et al. 1998). Environmental programs also have to be written in C and must use the communication functions (Mech et al. 1998). Visualisation is done with a turtle interpreting the L-system symbols and calling routines in various graphics libraries through a dispatching C programming construct (Mech et al. 1998). Possible improvements of CPFG might include more software and hardware independence; parallel execution

of L-system, environment and visualisation components; and distributing computation on different computers in a network.

Taking account of the strengths and weaknesses of the reviewed problem solving environments, I now discuss design and implementation principles of architecture and interactions of such environments.

4.1.2 Design and Implementation of Architecture and Interactions

The design of the overall *architecture* of a distributed agent-based problem-solving simulation environment could be based on the *Model-View-Controller* design pattern (Buschmann et al. 1996, p. 125-143). The Controller component of this design pattern corresponds to the control panel of the problem-solving environment, the Model component to the simulation model and the View component to the visualisation.

For structuring a simulation system with *interacting agents*, the Active Object and Broker design patterns are relevant. The *Active Object* pattern (Schmidt et al. 2000, p. 369-398) is useful for implementing agents as software components with concurrent communication capabilities. An Active Object decouples method invocation on the object from method execution and thus enables an agent in an ecosystem to be modelled as an autonomous entity with its own behaviour, at the same time being able to respond to influences from its environment and from other agents. The *Broker* pattern

(Buschmann et al. 1996, p. 99-122) deals with interaction and communication. It can be used to structure a distributed problem-solving environment with decoupled components that interact by remote service invocations. It is implemented for example in the Object Request Brokers of the CORBA technology (see e.g. Aklecha 1999; Siegel 2000).

Therefore, a natural implementation of the kind of environment here considered would make use of active objects or software agents executing in parallel processes and communicating with each other. It should be possible to run simulations in parallel, both to gain realism (the natural processes modelled also act in parallel) and performance (parallel execution on different processors). Communication between the different processes should be flexible and not dependent on machine-specific libraries. A generic communication protocol should be used to make possible communication between models programmed in different languages and executing on different operating systems.

This would also allow the use of existing software for simulation of particular subsystems or components of a complex system, to be integrated in the distributed simulation environment through interfaces and wrapper code for legacy software, using the *Wrapper Façade* design pattern (Schmidt et al. 2000, p. 47-74). A wrapper façade is useful for integrating existing software in an agent/object-oriented environment, as it avoids accessing non-object-oriented APIs directly by

encapsulating their functions and data within the more concise, robust, portable and maintainable methods of object-oriented classes. In this way, functioning and tested software can be used in a flexible way as building blocks in the problem-solving environment with relatively modest effort.

In addition to a structural view of interactions between components, *timing and synchronisation* of interactions need to be considered. Figure 4.2, based on a diagram in (Mech et al. 1998), pictures a plant model and an environmental model communicating by synchronous message passing.

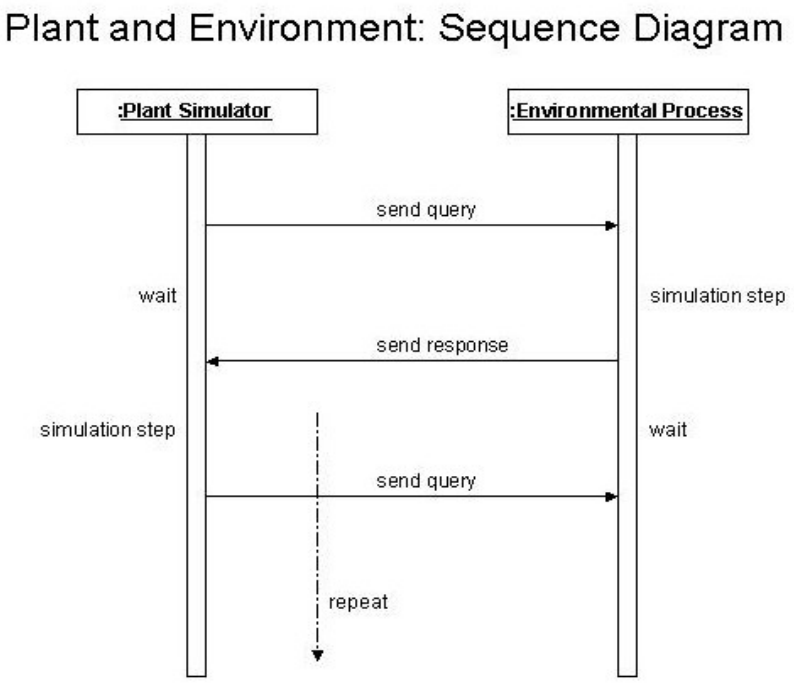


Figure 4.2. Timing aspects (from Mech et al. 1998).

The processes execute concurrently, but the plant simulator, before executing each simulation step, first sends a message to the environmental process, communicating information about its own state and requesting information computed by the environmental process. It waits for the environmental process to communicate its information and then executes the simulation step. The environmental process is suspended then and waits for the next message from the plant simulator. Other possibilities for communication between plant and environment would include asynchronous message passing: messages are sent and the sending process continues without first waiting for the response. This requires more sophisticated synchronisation mechanisms.

4.2 Hierarchy and Computational Complexity

In the preceding section I have discussed message exchange between components of a complex system, but said nothing about the *number of messages* to be exchanged. In general, interaction of all components of a complex system with all other components quickly becomes computationally prohibitive. One way of dealing with this problem is the use of a *hierarchical structuring of interactions*.

4.2.1 Hierarchical Structuring in Computer Simulation

Models of the interaction of plants with light in the environment provide a good illustration of the use of hierarchical structuring in computer simulation. Examples of such models are the radiosity models described in (Cohen & Wallace 1993). They are based on physical principles and calculate light exchange between surfaces in a scene. Each surface is divided into a number of small elements. To achieve a good approximation of light intensity everywhere, light exchange between each pair of elements would have to be considered. Thus, if the total number of elements is N, the number of pair-wise interactions is $O(N^2)$.

However, this computational complexity can be reduced by a hierarchical approach, analogous to algorithms applied to the N-body problem in physical dynamics described in (Greengard 1987; Cohen & Wallace 1993). The basic idea of these N-body algorithms is that the force due to a group of particles beyond some distance from another particle can be approximated with a single interaction. Likewise, pairs of such groups, separated by some distance, can be considered with a single interaction. For example, two widely separated galaxies each represent very large groups of particles, but the gravitational interaction between them can be approximated by the interaction of two particles, each representing the combined mass of the stars in the corresponding galaxy.

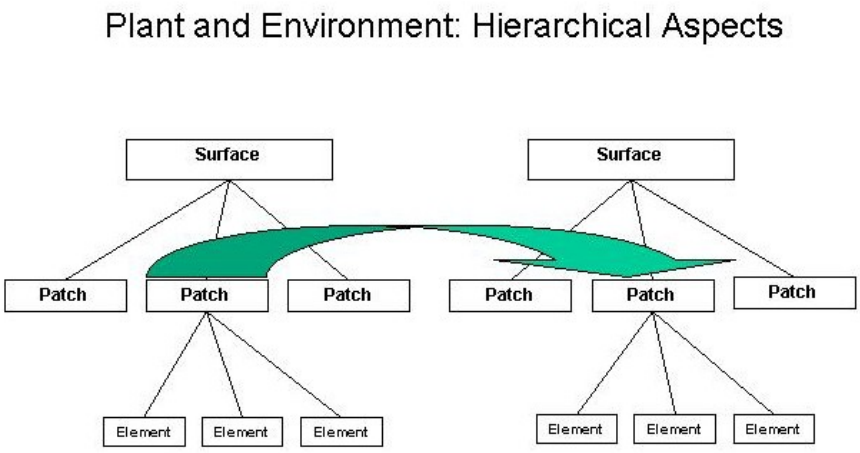


Figure 4.3: Hierarchical aspects, illumination example: light rays being exchanged between surfaces, patches of surfaces, and elements of patches.

In hierarchical radiosity algorithms a similar argument is applied to combine surface elements in larger surface patches. Thus, many interactions between small elements are replaced by fewer interactions between larger patches, as visualised in Figure 4.3. Computationally, this means that message passing can be reduced by limiting for example messages between different machines to messages at higher hierarchical levels.

In N-body algorithms, the effect (force) of all other, near and distant, particles on a particle is calculated by first calculating the magnitude of a potential field caused by the other particles. By clustering distant particles, efficient methods can be applied to derive the equations of the potential field. Once the information for the equations, typically a number of

coefficients of a polynomial, is derived, only that information needs to be communicated to the particle under consideration.

It would be interesting to explore if such a potential field approach could be generalised to other kinds of interaction between elements of a complex system. For example, in (Lam 1997) the principle of active walks is discussed (see Section 3.2.5). Here, components of a complex system do not communicate directly with each other, but they influence and are influenced by their environment, or by a potential field. This kind of modelling could contribute to reduce the message-passing complexity arising from each component interacting with each other component.

Implications of hierarchical structuring for design and implementation of a simulation environment are discussed in the next section.

4.2.2 Design and Implementation of Hierarchical Structure

In multi-agent simulations all agents can in principle interact with all other agents. Ways have to be found to reduce the number of messages exchanged, and especially the *message-passing complexity*, or the order of magnitude of the number of messages passed as related to the number of agents. If all N agents in a simulation exchange messages with all other agents, the number of messages exchanged is roughly proportional to the square of the number of agents and the

message-passing complexity is $O(N^2)$. Messages exchanged between different computers in a network are likely to have a higher overhead than messages between processes on a single computer, so their number is particularly important.

In one strategy to reduce message-passing complexity only messages between neighbours are allowed. One computer would correspond to one neighbourhood, and inter-machine communication would be limited to inter-neighbourhood communication.

Another strategy, well suited for the Emergent Models methodology, involves hierarchical structuring of agents and environment, using the *Whole-Part* design pattern (Buschmann et al. 1996, p. 225-242), which serves to model agents and their composition. A *Whole* object represents an aggregation of smaller *Part* objects. It forms a semantic grouping of its parts in that it co-ordinates and organises their collaboration. For example, a plant can be modelled as one object with a certain behaviour, but it can also be important to pay attention to the behaviour of its different organs, such as roots, stems, branches, and flowers. In a distributed implementation on computers in a network, each machine could execute a process corresponding to one branch in the hierarchy, and inter-machine communication would be limited to inter-branch communication.

This strategy is used in the Emergent Models methodology. In addition to providing insight in emergence in

computer simulations, the Emergent Models methodology involves hierarchical structuring of agents in higher level or group agents. A simulation of individual agents can be restructured as a simulation of group agents, leading to dramatic performance gains.

4.3 Visualisation and Steering

Using concurrency and interactions between an object-oriented simulation and an object-oriented visualisation system, the control, simulation, and visualisation functions of the software framework can be logically separated. For example, the simulation is running on one system. Sending appropriate messages, the user can obtain up-to-date information on the state of the system and use that information in the local visualisation system to render the appropriate scene. In the meantime, the simulation objects, being active objects supporting concurrent activities, can continue the simulation while at the same time being rendered.

Control of a simulation environment in general and a complex systems simulation environment in particular involves *computational steering*, a mechanism for integrating simulation, data analysis, visualisation, and post-processing (Parker et al. 1997). Possible approaches include program instrumentation (subroutine calls in programs to access parameters/results), directed scientific computation (subdivide a program into various

modules), and dedicated steering systems (e.g. SCIRun). Computational steering aims to achieve interaction and flexibility through control mechanisms (e.g. instrumentation, scripting languages), data distribution (single machine, remote sites, volume of data), data presentation (visualisation, custom programs), and user interfaces (GUIs, scripting, text files).

Simulation results need to be looked at, so design of *visualisation* is important. The *Pipes and Filters* pattern (Buschmann et al. 1996, p. 53-70) is useful for designing visualisation. An example of its application is the visualisation pipeline of The Visualisation Toolkit (Schroeder et al. 1998). It divides the task of the visualisation software into several sequential processing steps that are connected by the data flow through the system.

4.4 Emergent Models in a Simulation Environment

From the foregoing it is clear that a complex systems simulation environment with emergent models needs to have facilities and mechanisms for:

- constructing agents;
- interaction and communication between agents;
- synchronisation of interactions;
- hierarchically structuring agents;

- visualisation and steering of simulations.

The first three are generally provided by existing multi-agent development systems, and I have selected one of these to develop example applications. A multi-agent system has to be complemented by a mechanism for hierarchically structuring agents, or constructing higher-level agents from lower-level agents. I have selected a genetic programming library to provide this functionality in the example applications. Visualisation and steering facilities are necessary for a high performance and user friendly simulation environment of complex systems. I have used existing visualisation software in the example applications.

5 The Emergent Models Methodology

In this chapter I examine how multi-agent systems can be used for complex systems simulations with emergence. Possible algorithms for constructing emergent models in multi-agent simulations of complex systems are studied, leading to a proposed Emergent Models methodology.

Multi-agent systems, as described in Section 3.2.1, can be used to model and simulate complex systems of interacting entities in a straightforward way. An agent's properties can provide an instantaneous description of an entity, its autonomous behaviour can model an entity doing its own thing, and its interaction and communication capabilities can mimic interactions between biological entities. Grouping of agents has been proposed in research on multi-agent systems as an aggregation mechanism to describe properties of higher level entities, for example organisations consisting of individuals (Ferber 1999) or ponds consisting of water drops (Servat et al. 1998a; Servat et al. 1998b).

In this thesis I propose grouping of agents as a basis for modelling *emergent properties, as well as behaviour*. Holland (1998) proposes a similar idea. He uses the term *mechanism* to describe the elements, rules and interactions of complex systems (Holland 1998, p. 6). Mechanisms at one level interact with each other and become building blocks of persistent

patterns at a higher level. These persistent patterns can in turn become building blocks of persistent patterns at still higher levels of organisation (Holland 1998, p. 7-9).

Observation of such persistent patterns identifies *emergent properties*. In principle the value of any system variable with some invariance in time and space can be defined as a property. In practice it is hard to define what system variables should be measured and defined as emergent properties or, for that matter, as micro-level properties. The choice is usually determined by considerations like the quality, usefulness or simplicity of a model. For example, Copernicus formulated a model of the solar system regarded as a revolutionary departure from the Ptolemaic model, basically by redefining the *position property* of the planets. Defining this property as position relative to the sun instead of position relative to earth led to a model of the planets' movements radically simpler than was previously possible. Definition of exactly what is a good model and selection of properties making possible good models depend on little understood cognitive processes (see e.g. Holland 1998, p. 214-217, 232-233).

For the purposes of the present thesis it will be assumed that relevant emergent properties have been defined. Once these properties have been defined, it is usually straightforward to derive their values from those of micro-level properties. Typically one can use some aggregation procedure such as counting individuals in an area to calculate the population size

of that area, adding molecule masses to calculate a macroscopic mass, or averaging particle velocities to calculate a pressure. Modelling *emergent behaviour* then amounts to finding models describing relationships between emergent properties. Except in the simplest cases, examples of which have been described in Section 3.1, there is at present no systematic way to derive emergent models of macro-level behaviour from micro-level models. The Emergent Models methodology proposed in this thesis fills this gap.

An agent consisting of lower-level agents grouped together, a group agent, can 'learn' its own behaviour by looking at the results of the collective behaviour of its members and discovering regularities in that behaviour on a macro-scale. Thus, we can use *knowledge discovery* and *learning* techniques to define the group behaviour. As a general purpose knowledge discovery technique, *evolutionary algorithms* (e.g. Bäck et al. 2000) can be used. For example, assume that an equation describing the macro-level behaviour of a group agent is specified apart from some unknown parameters. An evolutionary algorithm can then be used to derive the optimal parameters from the micro-level behaviour of the individual agents. One kind of evolutionary algorithm, *genetic programming* (e.g. Koza 1992; 1994; Koza et al. 1999; 2003), is powerful enough to discover an unknown equation describing the macro-level behaviour when given building blocks to construct the equation and data on the behaviour to be

described.

Combination of these ideas, discussed in detail in sections 5.1 and 5.2, leads to the Emergent Models methodology proposed in Section 5.3. Finally, the implementation of the methodology used in this thesis is discussed in Section 5.4.

5.1 Multi-Agent Systems for Modelling Complex Systems

Agents have some clear advantages compared to traditional approaches for modelling complex systems. As an example we consider a fluid dynamics application. An agent modelling a droplet in a spray or cloud has flexible behaviour and dynamically determines its appropriate equations or rules. A droplet agent can say: I am far from obstacles, so follow the fluid velocity, or I am close to an obstacle, so drag is important; I am close to other droplets, so have to take into account hydrodynamic interactions; the flow is turbulent, so I diffuse with the eddies; I am at a surface, so I may deposit, splash or bounce. If even fluid droplets can usefully be modelled as agents, it is clear that agents provide a sufficiently general framework to model any entity observing the state of its environment and reacting by executing some action in the same environment. In the same vein, Holland stresses the obvious relevance of agent-based models for the study of emergence (Holland 1998, p. 116-118).

In an application developed in this thesis insects are modelled as agents observing the presence or absence of plants in their environment, and reacting by adjusting their movement behaviour.

In another application genes are modelled as agents observing the state of their environment, i.e. other genes' expression levels, and reacting to that environmental state by updating their own expression levels.

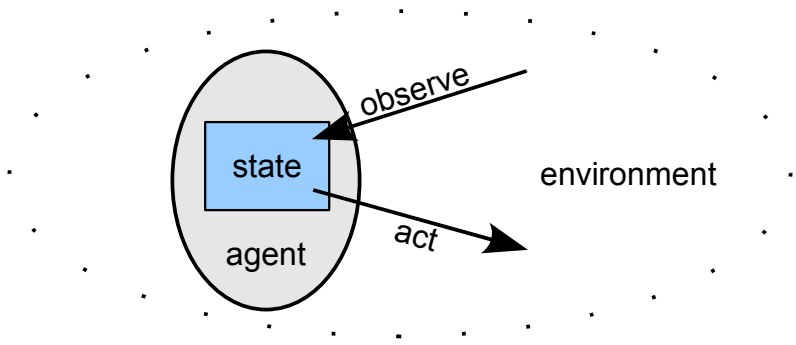


Figure 5.1. An agent with internal state.

As discussed in Section 3.2.1, more sophisticated agents are not only reactive, but have an intermediate internal state. Such agents observe their environment and, depending on the information gathered about the state of the environment, update their internal state. Their actions in the environment depend on

their internal state, as illustrated in Figure 5.1 (see e.g. Wooldridge 2002, p. 31-36). This resembles the behaviour of living organisms. Therefore organisms can usefully be modelled as agents with internal state.

Agents have a conceptual advantage in modelling systems of arbitrary complexity. Relatively simple agents interact to give rise to a complex system. Multi-agent simulations reveal the behaviour of the system as a whole, so, in principle, a system of agents can *discover emergent models* describing the macro-level behaviour. In particular, group agents can work out an appropriate macro-level model corresponding to given micro-level models of individual agents. At the next higher level, group agents can be individual member agents of a higher level group agent and the process can be repeated.

Biological systems are good examples of complex systems suitable for simulation using multi-agent approaches as described for example in (Ferber 1999; Ciancarini & Wooldridge 2001; Wooldridge 2002; Wooldridge et al. 2002; D'Inverno et al. 2002). Examining characteristics of biological systems, it becomes clear why this is so.

First, in a biological system there are autonomous organisms (plants, animals) with their own behaviour: growing, moving, feeding, responding to local stimuli, as well as satisfying internal goals. The idea of autonomous behaviour can even be extended to other physical entities or processes. Likewise,

agents have autonomy, i.e. they exhibit internal properties only modifiable by some action of the agent itself, and they have their own behaviour without being under the control of other program constructs such as a master program. Unlike an object, doing only something when a different object calls one of its methods, an agent, when created, starts doing something on its own. In principle, every entity in the world can be modelled as an agent.

Second, plants and animals interact with each other and with physical processes in their environment. Agents can simulate this by appropriate communication capabilities.

Third, biological entities are hierarchically structured. Micro-level entities act together to constitute macro-level entities. Macro-level properties and behaviour are derived from micro-level properties and behaviour. Yet, the macro-level can be described by a macro-model that does not include the micro-level. Agents can form groups, which can model macro-level entities. Properties and behaviour of group agents are emergent properties and behaviour, derived from properties and behaviour of their members. Yet group agent properties and behaviour are different from individual agent properties and behaviour.

A fundamental problem is the *emergence problem*: how can we derive properties and behaviour at the group level, or macro-level, from those at the individual level, or micro-level?

Multi-agent approaches with group agents have been used for simulation of physical systems for example in (Servat et al. 1998ab), where agents collectively form group agents with group membership determined from individual agents' properties and behaviour. In these simulations, whereas group membership can be said to emerge from individual behaviour, the multi-agent simulations do not automatically produce a description or model of the *emergent behaviour* of group agents. We just observe the results of the simulations and say they are produced by, or emerge from, all the individual actions. For example the gathering of a pond of water is described as an emergent result of actions of water droplets.

To elucidate emergence through computer simulation, we would like to have a systematic method of deriving macro-properties and -behaviour, in order to get models on the level of group agents in simulations as illustrated by Figure 5.2.

A group agent has to have a way of deriving its properties from the individual properties of its members. As seen in the introduction of this chapter, defining *group properties* is a non-trivial problem. However, if we assume that relevant properties have been defined, their values can usually be derived using some aggregation mechanism. For example group mass would be the sum of individual masses, group density the number of individuals divided by surface, etc.

The derivation of *group behaviour* is less straightforward.

Since a group agent does not have pre-established rules of behaviour, these rules must be discovered. Group behaviour could be described in general by a computer program implementing agent behaviour. For simplicity, let us confine our attention to the case of one method of the group agent implementing a function relating some group level properties to other group level properties, environmental influences, etc. A number of parameters of this function are unknown and the group agent should employ *knowledge discovery and learning* techniques to derive these parameters. In a more general approach, we can even assume that the shape of the function is unknown and also has to be discovered.

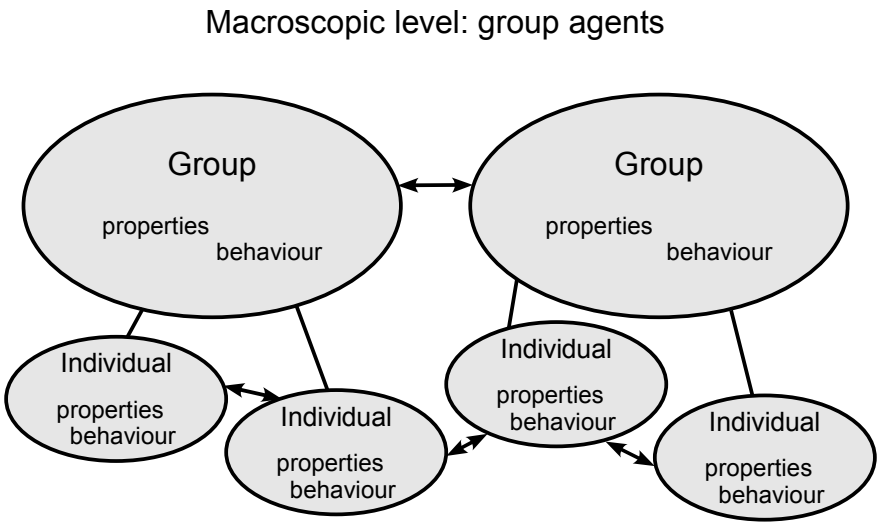


Figure 5.2. Levels in a multi-agent simulation.

Learning approaches have been combined with multi-agent systems in many studies, for example those presented in (Weiß & Sen 1996) and in (Weiß 1997). The main purpose of that work is to develop capacities of agents to attain a certain goal. In the present work, individual agents are just doing what they are modelled to do, and the group agents have to learn to perform better by mimicking the behaviour resulting from individual agents' behaviour.

Evolutionary algorithms have been applied to many problems (e.g. Stolk 1992; Schwefel 1995; Koza et al. 1999; Bäck et al. 2000). In the present work they are used in a novel way to model *discovery of emergent properties and behaviour in a multi-agent system*. In the next section, we have a closer look at evolutionary algorithms for this purpose.

5.2 Evolutionary Algorithms for Discovering Emergent Models

The problem of discovering emergent behaviour can be interpreted as a system identification problem: find a macro-level behaviour model that (approximately) reproduces the results of a given micro-level behaviour. Similarly, the inverse problem of discovering micro-level behaviour models corresponding to given macro-level data can be interpreted as a system identification problem: find micro-level behaviour models that (approximately) reproduce given macro-level data.

Evolutionary algorithms are general problem solving algorithms inspired by the evolution of organisms, interpreted as an optimisation process. They utilise the reproduction, random variation, competition, and selection of contending individuals in a population to find an optimal, or nearly optimal, solution to a problem (Fogel 2000). In general, evolutionary algorithms do not find globally optimal solutions, but only approximate solutions. System identification aims to identify the essential characteristics of a system, so an approximate fit to given data is an advantage, as an exactly optimal fit to probably noisy data is undesirable. Therefore, evolutionary algorithms often are a good tool to solve system identification problems.

Here I examine how evolutionary algorithms can be used to solve the system identification problem of discovering emergent behaviour.

In an evolutionary algorithm each individual in the population typically represents a potential solution considered for a given problem. Descendants of individuals are generated by randomised processes of mutation, or erroneous self-replication of individuals, and recombination, or exchange of information between individuals. Individuals are evaluated according to a fitness measure related to the problem in a selection process favouring the reproduction of better individuals (Bäck 2000).

Various kinds of evolutionary algorithms have been

developed, such as evolution strategies (Bäck et al. 2000, p. 81-88; Schwefel 1995), evolutionary programming (Bäck et al., p. 89-102), genetic algorithms (Bäck et al., p. 64-80; Holland 1975), and genetic programming (Bäck et al., p. 103-113; Koza 1992). The remainder of this section focusses on evolution strategies as a method for finding approximately optimal parameters of a function, and on genetic programming as a method for finding in general a computer program or function solving a given problem.

5.2.1 Evolution Strategies

A simple version of an evolutionary algorithm for optimising a function, known as an *evolution strategy*, searches for a combination of arguments of that function optimising the function value (Schwefel 1995). In analogy to real evolution, it starts from a (supposedly non-optimal) solution. In each iteration it randomly mutates that solution, and selects the best solution from the old and the mutated one to continue with. The algorithm iterates until a termination criterion is satisfied, which can be a specified number of iterations or a specified accuracy (distance from the optimum) of a found solution.

For concreteness, consider minimisation of a real valued function F of n real variables:

$$F : \mathbb{R}^n \rightarrow \mathbb{R}$$

1 Initialisation	Set $g=0$. Define a point $\boldsymbol{x}_g \in \mathbb{R}^n$ where g is the generation or step number.
2 Mutation	Construct $\boldsymbol{x}_{g,new} = \boldsymbol{x}_g + \boldsymbol{z}_g$ where $\boldsymbol{z}_g \in \mathbb{R}^n$ is a vector of random variables with the role of mutation.
3 Selection	Decide $\boldsymbol{x}_{g+1} = \boldsymbol{x}_{g,new}$ if $F(\boldsymbol{x}_{g,new}) \leqslant F(\boldsymbol{x}_g)$ $\boldsymbol{x}_{g+1} = \boldsymbol{x}_g$ otherwise. Increase g to $g+1$ and go to step 2 as long as the optimum has not been reached (or approached within a user-defined accuracy).

Table 5.1. An evolution strategy algorithm.

The algorithm in Table 5.1 will find the point in $\mathbb{R}^n$ where the function value is (approximately) a minimum.

In this algorithm individuals (candidate solution points $\boldsymbol{x}_g \in \mathbb{R}^n$) are mutated by adding a vector of random variables $\boldsymbol{z}_g \in \mathbb{R}^n$. The probability distribution of $\boldsymbol{z}_g$ can be adapted during the search in such a way that intermediate search results lead to more effective search.

Suppose the components $z_i (i = 1 \ldots n)$ of $\boldsymbol{z}_g$ are normally distributed with expectation value μ_i and variance σ_i^2. The distribution function of $z_i (i = 1 \ldots n)$ is

$$W(z_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(z_i - \mu_i)^2}{2\sigma_i^2}\right)$$

Convergence of the algorithm can be improved by the 1/5 success rule: from time to time during the search obtain the frequency of successes, i.e. the ratio of the number of new $\boldsymbol{x}_g$ values with lower function values to the total number of mutations; if the ratio is greater than 1/5, increase the variance, if it is less than 1/5, decrease the variance (Schwefel 1995).

Another way to implement reinforcement of the search direction is described in (Stolk 1992) and works by adapting the expectation values of $\boldsymbol{z}_g$ at each step as follows. Suppose the

expectation value vector is $\boldsymbol{\mu}_g$ at step g and we have obtained a new, better, search value $\boldsymbol{x}_{g+1}$ by taking a step $\boldsymbol{z}_{g+1}$.

Now make

$$\boldsymbol{\mu}_{g+1} = \boldsymbol{\mu}_g + \gamma(\boldsymbol{z}_{g+1} - \boldsymbol{\mu}_g)$$

Intuitively, we shift the greatest likelihood of further mutations somewhat in the direction of successful new values. How much importance we attribute to a new success, is determined by the value of γ, a constant with $0 \leqslant \gamma \leqslant 1$.

An evolution strategy can be applied to the *optimisation of a dynamical system* such as a control system by making the following interpretations (Stolk 1992). Assume the system behaviour is determined by a vector of parameters $\boldsymbol{p} \in \mathbb{R}^n$. Each particular combination $\boldsymbol{p}_g$ of parameters gives rise to a time evolution $u_t(\boldsymbol{p}_g)(t = t_0 \ldots t_{max})$ of some measured (output) variable u of the system. Assuming an optimal system evolution is given by $U_t(t = t_0 \ldots t_{max})$, we can define for each parameter combination $\boldsymbol{p}_g$ the deviation of the system evolution from the optimal evolution by a function such as

$$E(\boldsymbol{p}_g) = \sum_{t_0}^{t_{max}} (U_t - u_t(\boldsymbol{p}_g))^2$$

Equating E with our previous function F and $\boldsymbol{p}_g$ with point $\boldsymbol{x}_g$, an evolution strategy can find a parameter combination optimising the behaviour of the system.

The system to be optimised can also be a *simulation system*. The measured variable now is an output variable of the simulation, and optimal behaviour of the simulation is a time evolution of the output as close as possible to real data. Thus, an evolution strategy can also be used to optimise a simulation model with a number of unknown parameters.

For the purpose of this thesis, the system to be optimised consists of group agents whose behaviour is described by *macro-level equations.* The results of the group agents' behaviour lead to values of output variables of a simulation as dependent on input variables. Optimal behaviour of the group agents, as described by the macro-level equations, is defined as a time evolution of their output variables as close as possible to the evolution of the same variables as computed by the micro-level multi-agent simulation. The goal is thus to find equations at the macro-level that can be substituted for the simulation with micro-level agents.

5.2.2 Genetic Algorithms and Genetic Programming

A *genetic algorithm* transforms a population of individual objects, each with an associated value of fitness, into a new generation of the population, using the principle of survival and reproduction of the fittest and analogues of biologically occurring genetic operations such as crossover (sexual recombination) and mutation (see e.g. Holland 1975; Koza et al. 1999). In its basic form a genetic algorithm consists of the three steps of initialisation, generation, and result designation (Koza et al. 1999, p. 21-22), as described in the algorithm in Table 5.2.

Genetic programming is an extension of genetic algorithms in which the genetic population contains computer programs to solve problems (Koza et al. 1999). A genetic programming search for solutions of a problem starts with an initial population of computer programs composed of functions and terminals appropriate to the problem. The functions are frequently merely standard arithmetic functions and standard logical functions. The terminals typically include the external inputs to the program as variables, and may also include constants and zero-argument functions. During the search, individuals representing possible combinations of functions and terminals are mutated and recombined, until a good solution is obtained.

1 Initialisation	Randomly create an initial population of individuals.
2 Generation	Iteratively perform the following substeps until the termination criterion has been satisfied: a) assign a fitness value to each individual using the fitness measure for the problem; b) select one or two individuals from the population with a probability based on fitness; c) create individuals for the new population by applying genetic operations to these individuals with specified probabilities: i. reproduction: copy the selected individual to the new population; ii. crossover: create new offspring individuals for the new population by recombining parts of two selected individuals at a randomly chosen crossover point; iii. mutation: create one new offspring individual for the new population by randomly mutating randomly chosen positions of one selected individual.
3 Result Designation	Designate an individual (e.g. the best-so-far individual) as the result of the genetic algorithm.

Table 5.2. A genetic algorithm (Koza et al. 1999, p. 21-22).

A well studied problem in genetic programming is the symbolic regression problem, in which a function is sought that best approximates given data (see e.g. Koza 1992; Luke 2002). To solve this problem genetic programming works with a population of functions, which are represented as trees of arithmetic operators and terminals, as shown in Figure 5.3.

In the *initialisation* step of the genetic programming algorithm a number of such trees is constructed at random. In the *generation* step operators of reproduction, crossover and mutation are applied.

Reproduction operates on one individual computer program (or function) selected with a probability based on fitness and makes a copy of the program for inclusion in the next generation.

Crossover operates on two parental computer programs selected with a probability based on fitness and creates one or two new offspring programs consisting of parts of each parent. In the tree representation crossover means randomly selecting and exchanging subtrees of both parents, as illustrated in Figure 5.3. The offspring is inserted into the next generation.

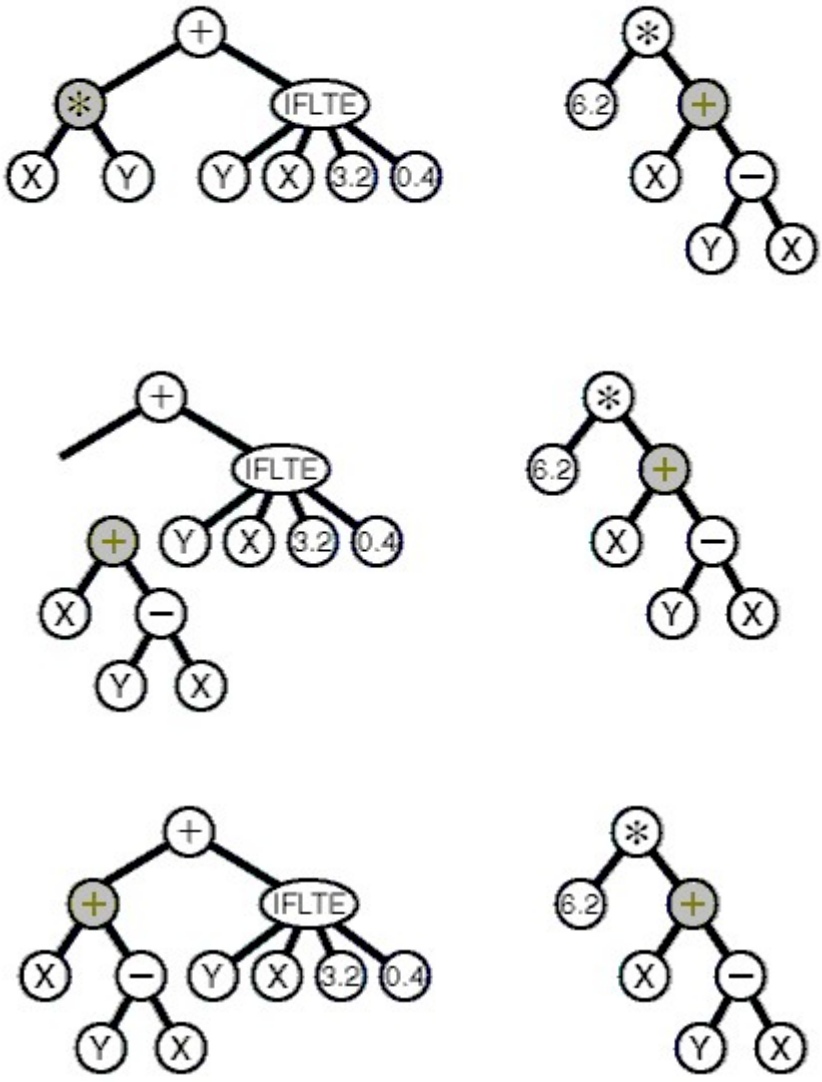


Figure 5.3. Tree representation of functions and the crossover operation (Luke 2002). Before crossover, the left hand function is $F(x, y)=(x*y)+(if\ x\leq y\ then\ 3.2\ else\ 0.4)$ and the right hand one $G(x, y)=6.2*(x+(y-x))$. After crossover, the left hand function is $H(x, y)=(x+(y-x))+(if\ x\leq y\ then\ 3.2\ else\ 0.4)$.

Mutation operates on one parental computer program selected with a probability based on fitness and creates one new offspring program to be inserted into the next generation. In the mutation operation a point is randomly chosen in the parental program. The subtree rooted at the chosen mutation point is deleted from the program, and a new subtree is randomly grown.

Thus, genetic programming can find a function approximating given data, by using building blocks defined by the programmer to construct functions combined with a measure for determining how the data are approximated by a particular function. Like evolution strategies, it can be used to find a macro-level equation that can be substituted for the micro-level multi-agent simulation. Genetic programming can achieve this with less a priori information than evolution strategies. An evolution strategy optimises parameters of an otherwise known function, while a genetic programming algorithm is only constrained by the building blocks and fitness measure given to it, possibly complemented by additional constraints that can be imposed in a flexible way by the programmer.

5.3 The Emergent Models Methodology

We have seen how complex systems can be modelled using multi-agent simulation. We have also seen how evolutionary algorithms, and in particular genetic programming, are general purpose algorithms for solving a variety of optimisation problems.

In the present work I combine ideas from multi-agent simulation and from evolutionary algorithms in a novel methodology for discovering emergent macro-level regularities or patterns in simulations of complex systems. These macro-level regularities are models of the behaviour of macro-level agents, in other words *emergent models*. I therefore call the new methodology the E*mergent Models methodology*.

It defines ways, summarised in Table 5.3, to derive macro-level behaviour from micro-level properties and behaviour, discovering models at the macro-level implied by those describing the micro-level.

This methodology can also be applied to the inverse problem of discovering micro-level behaviour of the composing entities of a complex system from data on its macro-level properties and behaviour. Table 5.4 summarises how the inverse problem is solved.

Describe the complex system to be studied as a multi-agent system with agents at different levels, at least a micro-level and a macro-level.
Define how micro-level agents behave, how macro-level or group agents are composed of micro-level agents, and how group agent properties depend on micro-level properties.
Run simulations with the micro-level agents, preferably a number of different runs with a representative set of initial conditions and parameters.
Collect data on group agent properties from the micro-level simulations.
Define building blocks of group agent behaviour, for example variables, constants, and operators to be used in functions describing the behaviour, and a fitness function measuring how well group agent behaviour approximates micro-level simulation results.
Execute a genetic programming algorithm with the building blocks and running a group agent simulation for each fitness evaluation, to discover a description of group agent behaviour approximating micro-level behaviour results.
Run simulations using group agents with the discovered behaviour, to obtain results comparable to the average behaviour of micro-level simulations, so more robust than a single micro-level simulation, and needing a small fraction of the time and computing resources of a micro-level simulation.

Table 5.3 The Emergent Models methodology: micro – macro.

Describe the complex system to be studied as a multi-agent system with agents at different levels, at least a micro-level and a macro-level.
Define data on macro-level agents' properties and/or behaviour to be approximated, how macro-level agents are composed of micro-level agents, and how macro-level agents' properties and/or behaviour depend on micro-level properties.
Define building blocks of micro-level agent behaviour, for example variables, constants, and operators to be used in functions describing the behaviour, and a fitness function measuring how well results of a micro-level agent simulation approximate the macro-level data.
Execute a genetic programming algorithm with the building blocks and running a micro-level agent simulation for each fitness evaluation, to discover a description of micro-level agent behaviour approximating the macro-level data.

Table 5.4 The Emergent Models methodology: macro – micro.

The Emergent Models methodology as defined in the present work only deals with discovering emergent behaviour, while assuming the emergent properties used as building blocks of that behaviour are already known. I anticipate that in future extensions of the methodology the problem of discovering emergent properties will also be tackled.

An Emergent Models simulation mimics a complex system's processes and structure by modelling it as interacting and concurrently executing software agents and processes. These agents and processes exist at various hierarchical levels. Properties and behaviour of agents at each level depend on those on a lower level, and mechanisms modelling the emergence of higher level behaviour from lower level properties and behaviour are defined.

As an example of such mechanisms, genetic programming algorithms are used to discover macro-level emergent models from micro-level properties and behaviour. These algorithms are also applied to the inverse problem of deriving micro-level models from known macro-level behaviour.

In the Emergent Models methodology models of complex systems are implemented as multi-agent simulations. In order to model emergence in a satisfactory way, a complex systems simulation methodology should explicitly address the relationship between *properties and behaviour* of micro- and macro-level agents.

This is done in the Emergent Models methodology by including in a simulation *group agents* composed of individual agents, somewhat similar to the group agents in Servat's model, discussed in Section 3.2.1.5. However, unlike the group agents in Servat's model, group agents in an Emergent Models simulation are active agents. They not only have group level

properties emerging from individual behaviour, but also a *behaviour* emerging from the properties and behaviour of their individual member agents. They resemble the composite mechanisms defined by Holland (1998, p. 188-201). To discover this emergent behaviour in an Emergent Models simulation, evolutionary algorithms and in particular genetic programming are used.

As discussed in Section 3.2, a multi-agent simulation or individual-based simulation can be very complex. It can be hard to understand how the simulation works and to have confidence in simulation results. With the Emergent Models methodology macro-level models can be automatically formulated to describe the same system behaviour as a multitude of models of micro-level agents and their interactions. This macro-level system description can be much more succinct and intuitively understandable than the micro-level description, making it possible to achieve a better understanding of and increased confidence in multi-agent simulations.

In addition, multi-agent simulations tend to require extensive computation. A simulation with emergent models can improve its own performance while learning its own macro-behaviour, substituting more and more macro-level laws for the equivalent micro-level behaviour. As many micro-level interactions can be reflected by a few macro-level interactions on much larger space and time scales, performance gains of orders of magnitude can potentially be achieved.

I will demonstrate the discovery of macro-level behaviour from micro-level behaviour in Chapter 6, applying the Emergent Models methodology to problems in ecology. Ecosystems are a suitable application area, being complex systems consisting of plants and animals interacting with each other.

In the Emergent Models methodology evolutionary algorithms can also be used to solve the inverse problem of discovering micro-level behaviour of individual agents in a multi-agent simulation, when their macro-level behaviour is known.

I will demonstrate this in Chapter 7, applying the methodology to individual plants. A plant can be described as a complex system, with macro-level properties determined by micro-level models of, for example, cells or genes. In particular, I interpret a genetic regulatory network as a multi-agent system and use genetic programming to discover genetic network models that can give rise to observed macro-level behaviour of the whole plant organism.

5.4 Implementation

As seen in Chapter 4, a complex systems simulation environment with emergent models needs to have facilities and mechanisms for constructing agents, interaction and communication between agents, and synchronisation of interactions. *MadKit* (“Multi-Agent Development Kit”) is a multi-agent development system written in Java satisfying these

requirements. It has been developed at the University of Montpellier (see Ferber et al. 2003), and is used in the present work to develop some applications described in Chapter 6.

We have also seen that a multi-agent system has to be complemented by a mechanism for hierarchically structuring agents, or constructing higher-level agents from lower-level agents. *ECJ* ("Evolutionary Computation with Java") is an evolutionary computation system written in Java with genetic programming capabilities. It has been developed at the University of Maryland and George Mason University by Sean Luke and collaborators (see Luke 2002), and is used in the applications described in Chapter 6 to derive behaviour of agents at higher levels from behaviour of agents at lower levels, and in Chapter 7 to derive lower-level behaviour from higher-level data.

Visualisation of results has been done using observer agents developed with MadKit, as well as the charting capabilities of VTK ("Visualization Toolkit"; see Schroeder et al. 1998) in chapters 6 and 7. VTK is visualisation software written in C++, but also providing Java wrapper classes. The scope of the present work is limited to defining the Emergent Models methodology and demonstrating its application to complex system simulation. Therefore more sophisticated visualisation and steering facilities have not been developed.

5.4.1 Multi-Agent Simulation with MadKit

For the multi-agent simulations developed in the present work I use the MadKit synchronous engine (Ferber et al. 2003), which provides facilities for scheduling and monitoring agents. A general MadKit agent has its own thread and communicates with other agents through messages. In the synchronous engine agents do not have their own threads and can be directly referenced. This enables the development of efficient prototypes, while retaining the possibility of upscaling a MadKit simulation to a distributed simulation using the agent implementation with one thread per agent and communication through messages.

To program individual agents I have written the abstract `Organism` class, a subclass of the MadKit `AbstractAgent` class. All particular agent models are subclasses of the `Organism` class. Group agents are implemented as the `SubPopAgent` class, also a subclass of the MadKit `AbstractAgent` class, or as subclasses of the `SubPopAgent` class. Group agents have capabilities for collecting data on the results of the behaviour of their individual member agents, for example population density, births, deaths, immigration, and emigration. These data can be visualised on the screen or written to a *MySQL* database using a Java MySQL database driver. Detailed information about the MySQL software is available on website http://www.mysql.com.

5.4.2 Emergent Model Discovery with ECJ

The data produced by the group agents of a multi-agent simulation, implemented with MadKit, are used by a genetic programming algorithm, implemented with ECJ, to discover a model fitting these data. The problem to be solved, including information such as the number of functions to be discovered and the fitness measure to be used, is defined in a subclass of the ECJ `GPProblem` class. Building blocks to be used by the genetic programming algorithm are defined by the `Operation` class and its subclasses, which represent functions and variables to be used. The `Operation` class is a subclass of the ECJ `GPNode` class.

A genetic programming algorithm has a number of parameters that can be set to determine the exact working of the algorithm. Examples are the population size, the number of generations for which the algorithm is run, and the random seed used to construct the initial population. Parameter values in the ECJ software are set in parameter files. Default values are provided in the files `ec.params`, `simple.params`, and `koza.params`. Specific values for the present work complement or override the default values and are defined in files with filenames related to the problem and extension `.params`. Solutions discovered by the genetic programming algorithm are written in files with the same filename as the `.params` files, but with extension `.stat`.

As the purpose of the applications developed in the context of the present work is to demonstrate the possibility of using genetic programming for the Emergent Models methodology, rather than developing genetic programming theory, parameter values used in the experiments were set in a pragmatic way to obtain results with a good fit.

6 From Micro-Level to Macro-Level in Ecology

Using the Emergent Models methodology formulated in Chapter 5, it should be possible in a multi-agent simulation to discover the emergent macro-level behaviour of the system. The applications in this chapter demonstrate how macro-level group agents can learn and work out an appropriate macro-level model corresponding to given micro-level models of individual agents.

Although emergence can be found anywhere, biology is a particularly interesting field to look for it. Ecosystems typically consist of so many interacting entities that individual-based simulation quickly leads to prohibitive computation time and resources, if done at a realistic scale. Other examples with similar complexity are the nervous system network of interacting neurones, tissues consisting of interacting cells, and genetic regulatory networks. In this chapter and in Chapter 7 the general approach formulated in Chapter 5 for complex system simulation is applied to problems of interest in various fields, in order to test and verify its general applicability.

This chapter illustrates how macro-level emergent models can be derived from micro-level models in applications for some problems in ecology. In the first application pesticide droplets are modelled as micro-level agents constituting a macro-level

cloud or spray. In the other applications organisms are modelled as agents interacting with each other to constitute a population, or subpopulations in turn making up a global population. The subpopulations can be described as agents whose properties and behaviour give rise to those of the global population.

6.1 Emergent Models in Insect-Plant Interactions

One area where Emergent Models can be used, is that of insect-plant interactions, as described e.g. in (Shorey & McKelvey 1977; Payne, Birch & Kennedy 1986; Beer, Ritzmann & McKenna 1993; Miller & Miller 1986; Huffaker & Gutierrez 1999; Woiwod, Reynolds & Thomas 2001). It has the advantage that both the individual and population levels are fairly easily observable. Modelling of the behaviour of individual insects and that of insect populations or subpopulations is important to predict herbivory, oviposition and other factors determining damage caused by insects to vegetation.

Insects move in the atmosphere according to behavioural principles such as anemotaxis, counterturning, chemotaxis, etc., as described e.g. in (Payne, Birch & Kennedy 1986). Processes and forces governing their movement include visual stimuli, olfactory stimuli, as well as self-steering behaviour. Consequently, there is not one simple equation governing the insect's motion, and an insect should be modelled as an agent with the capability to adapt to different circumstances.

All insects together constitute a population of insects, for example living in a canopy consisting of leaves. A computer simulation could in principle calculate the behaviour of all insects interacting with all leaves and other insects, but the results would be hard to interpret. It would also be a computationally intensive task. We would like to be able to reason on the level of the insect population interacting with the whole canopy, or a relatively small number of insect subpopulations interacting with each other and with a small number of plant patches, for instance.

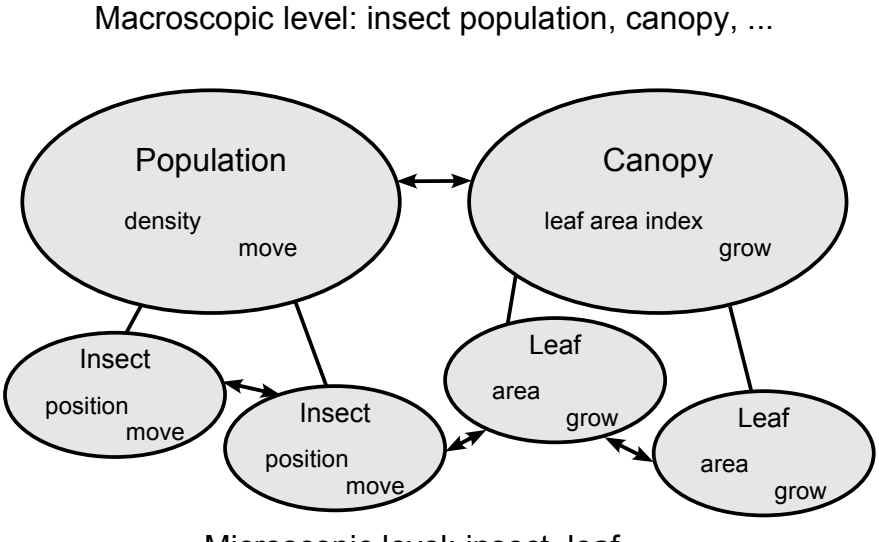


Figure 6.1. A canopy agent composed of leaf agents and a population agent composed of insect agents. Each agent is characterised by name, properties and behaviour.

To achieve this, a population or subpopulation of insects can be modelled as a higher level group agent. The group agent has properties such as the insect population density distribution. Interesting behaviour of the group agent could include a function relating insect arrivals on surfaces (plant leaves, soil) to concentration of an odour source, as well as wind velocity and direction. This is an example of macro-level behaviour to be derived from the micro-level behaviour of individual insects.

Figure 6.1 shows a way a population or subpopulation can be modelled as a group agent composed of individual insect agents, and a canopy as a group agent composed of individual leaf agents.

In this example, the leaf and canopy agents are passive agents with the role of receiving numbers of insects. Their properties influence the way insects arrive. A leaf agent has properties like position, size, and inclination. Canopy agent properties such as leaf area index are derived from leaf agent properties by aggregation. Insect agents have individual properties like size, mass, odour sensitivity, oviposition readiness, etc. Macro-level properties of a population agent depend on the properties of all individual insect agents. Macro-level behaviour is a result of individual insect behaviour such as movement, oviposition, etc.

Interaction of insect agents with leaf agents on the micro-level (e.g. arrival of an insect on a leaf) leads to interaction of

the population agent with the canopy agent (e.g. arrival of a number of insects on an area of canopy). So micro-level behaviour, including interactions between micro-level agents, determines macro-level behaviour (e.g. evolution of the population density distribution of insects).

I have conducted computational experiments to show how emergent macro-level models can be derived from micro-level models in simulations of pesticide spraying of crops, insect behaviour in an odour plume, and behaviour of butterflies interacting with plants. The experiments were conducted by modelling fluid droplets, insects, and plants as micro-level agents, and fluid clouds and (sub)populations of insects as macro-level agents.

The pesticide spraying experiment is meant to illustrate the Emergent Models methodology with an example that is as simple as possible. Micro-level agents model pesticide droplets and their behaviour is described by physical laws only. Together the droplet agents constitute a cloud agent behaving according to the emergent model found in the experiment.

In the next experiment the micro-level agents are insects interacting with environmental conditions determined by an odour plume. Together the insect agents constitute a population agent. The aim of the experiment is to find a macro-model describing the interaction of the population with the odour plume.

In the last experiment three levels are considered. Agents at the micro-level are butterflies and individual plants. At an intermediate level there are agents modelling subpopulations of butterflies occupying plant patches. The macro-level is the whole system of total insect and plant populations.

Plants were treated as passive entities in all experiments in this chapter, but the methodology could easily be applied with active plant and canopy agents as well.

Genetic programming was used as a general method to derive macro-level behaviour from micro-level properties and behaviour.

6.2 Droplets and Cloud: Pesticide Spraying

An application of ecological simulation is pesticide spraying (visualised in Figure 6.2). Accurate modelling of the behaviour of pesticide sprays and droplets is important to be able to direct the pesticides to the places where they are most effective and to minimise waste and environmental pollution.

When spraying a crop or a forest, a cloud or spray consisting of many pesticide droplets is deposited in a canopy consisting of many leaves. Thus it is a computationally complex problem to calculate the interactions of all droplets with all leaves. A possible solution would group droplets in clouds and leaves in clusters, effectively reducing interactions between droplets and leaves to interactions between cloud and canopy.

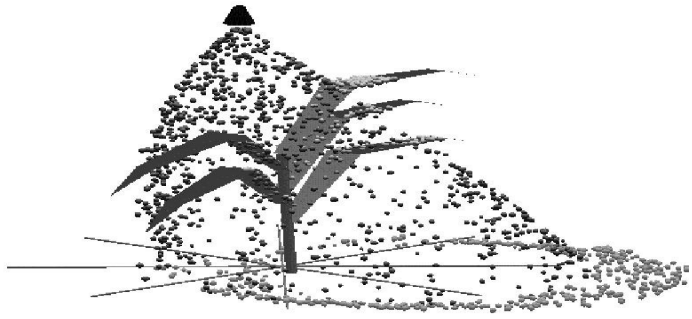


Figure 6.2. A simulation of pesticide spraying (from Hanan et al. 2000).

Droplets move in the surrounding air (a fluid) according to fluid dynamic laws, as described e.g. in (Friedlander 2000; Sirignano 1999; Zapryanov & Tabakova 1999). Processes and forces governing their movement include diffusion through Brownian motion, convection with the fluid, drag forces, gravity, deposition on surfaces, coagulation, collisions with surfaces and the resulting processes (splashing or bouncing), as well as hydrodynamic interactions between different droplets. All these processes can occur in both laminar and turbulent fluid flow. Consequently, there is not one simple equation governing the droplet motion, already a reason to model a droplet as an agent with the capability to adapt to different circumstances.

All droplets together compose the whole spray. The spray

has properties described by, for example, a droplet distribution function or a volume distribution function. Behaviour of the spray we might be interested in includes for example a function relating deposition volumes on surfaces (plant leaves, soil) to initial velocity and droplet distribution at the nozzle, as well as wind velocity and direction. This is an example of macro-level behaviour we would like to derive from the micro-level behaviour of individual droplets.

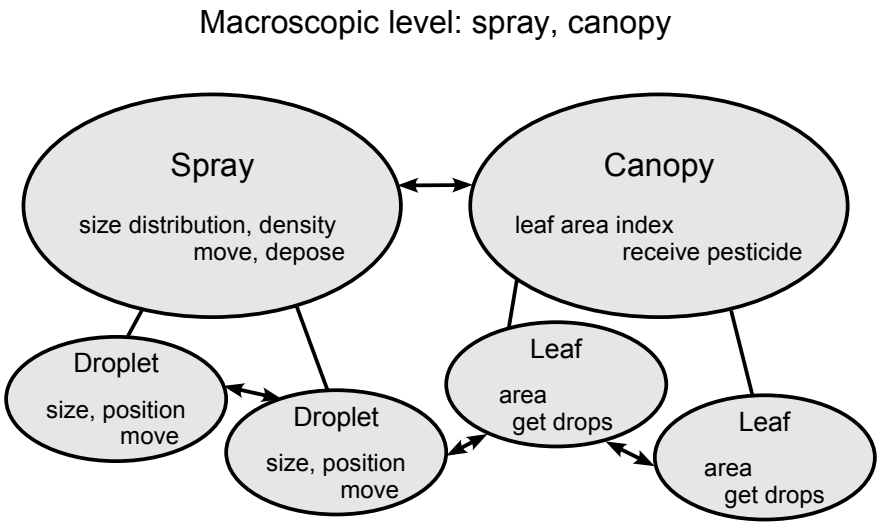


Figure 6.3. A canopy agent composed of leaf agents and a spray agent composed of droplet agents. Each agent is characterised by name, properties and behaviour.

It was hypothesised that an evolutionary algorithm, in particular a genetic programming algorithm, could derive a macro-level description of behaviour of a pesticide clouds from the simulated micro-level behaviour of the droplets composing the cloud.

Method

A spray can be modelled as a spray group agent composed of individual droplet agents, and a canopy as a canopy group agent composed of individual leaf agents, as shown in Figure 6.3.

In this example, the leaf and canopy agents are passive agents with the role of receiving volumes of pesticide. Therefore, we are mainly interested in their properties influencing the way pesticide is deposited. A leaf agent has properties like position, size, and inclination. Canopy agent properties such as leaf area index are derived from leaf agent properties by aggregation. droplet agents have individual properties like size, mass, and volume.

Macro-level properties of a spray agent depend on the properties of all individual droplet agents, such as position and size. Macro-level behaviour is a result of individual droplet behaviour such as movement, splashing, and deposition. This leads to evolution of the size distribution of droplets. Interaction of droplet agents with leaf agents on the micro-level, such as

deposition of droplets on leaves, leads to interaction of the spray agent with the canopy agent, or deposition of a volume of pesticide on an area of canopy.

Suppose we are only interested in relationships between certain selected macro-properties. For example, when spraying a field of plants, we would like to know the relationship between, on the one hand, a certain flux of pesticide introduced in the atmosphere at a certain position and with a certain speed and, on the other hand, volumes of pesticides deposited in the canopy and on the ground. A typical application involves computer modelling of aerial spraying of forests in Canada (Picot & Kristmanson 1997). The computer model involves following droplet trajectories from the spraying aircraft's wake to the canopy or the ground. The results of the computer simulations include input quantities such as the spray system flow rate, temperature, air pressure, and wind speed, as well as output quantities such as volume of pesticide deposited in the canopy and on the ground at different grid locations.

I attempted to derive a macro-level function describing deposited volumes V of pesticide as determined by input variables from individual droplet behaviour.

The macro-level function to be derived was assumed to have the following general form:

$$V = F(Acr, Std, Atm, Ato, Frm) \qquad (6.1)$$

where:

Acr is a set of aircraft properties (weight, wing span, flying height, speed);

Std is a set of stand properties (location, roughness height, displacement height, mean height, base of foliage crown, foliage density and vertical distribution, height of ground cover);

Atm is a set of atmosphere properties (temperature, pressure, wind velocity, relative humidity, height of mixing layer);

Ato is a set of atomiser properties (location on wing, number, flow rate, droplet spectrum);

Frm is a set of pesticide formulation properties (% non-volatile, tracer concentration).

The input variables determine the behaviour of each droplet. In the case of variables that are properties of another agent, such as the stand properties, they can be thought of as messages sent by that agent (the canopy agent in the case of the stand) to the droplet agents or to the spray agent.

Some effects determining droplet agent behaviour are:

- a droplet receives an initial downward velocity v from the aircraft wake described by

$$v = \frac{8L}{\pi^3 \rho u b^2}$$

where:

L is aircraft weight;

ρ is air density;

u is aircraft speed;

b is wingspan.

- droplet trajectories are calculated with droplet initial location given by atomiser position on the aircraft, and assuming that the droplet follows the motion of the air, with the addition of a gravitational settling velocity;
- droplet deposition on foliage in the canopy is determined by the probability that the droplet trajectory will cause it to interact with a foliage element, and the probability that the interaction will result in deposition by inertial impaction or interception.

Modelling each droplet as a droplet agent with a behaviour incorporating all these effects enables us to simulate the spray as a *multi-agent system*. The spray agent's behaviour incorporates macro-level equation (6.1), which is not yet determined. A number of runs of the micro-level simulation is executed with randomly chosen values of the input variables to produce data from which an average behaviour can be derived. The data produced by these runs are kept in a database and subsequently used by a *genetic programming algorithm* to

determine the macro-level equation, as follows.

First, define a set D of input data values over which we want to optimise the macro-level equation. Now define an error function measuring the deviation of the output of equation (6.1) from the multi-agent output as

$$E = \sum_{d \in D} |U_d - V_d|$$

where:

U is the volume as computed by the multi-agent simulation;

V is the volume as computed by the macro-equation.

The problem to be solved is to derive an emergent model of relationships between emergent properties, and the present example is constructed to demonstrate the principle in as simply a situation as possible. In principle a model containing any number of macro-level variables could be discovered. However, to simplify the problem as much as possible, we make a fairly arbitrary choice and look for a macro-level model consisting of a functional relationship between just two macro-level variables. In particular, the function to be discovered will describe how V, defined as the number of droplets deposited at a distance between 400 and 500 m from the position of the spraying plane, depends on wind speed W (in m/s), or

$$V = F(W) \qquad (6.2)$$

This particular choice for the equation to be derived is a choice out of numerous possibilities and will serve to demonstrate the principle of discovering a macro-level equation. Many other choices, including more complex models with larger numbers of variables, could be made without altering the principle of the demonstration. The possibility of making an arbitrary choice of properties to be included in the macro-level model also underlines the fact that there is no unique model describing the macro-level phenomenon. The choice of variables to be included in the macro-level model depends on its intended use.

A *genetic programming* algorithm can perform a symbolic regression in order to find an equation minimising the deviation of equation (6.2) from the multi-agent simulation results. Once this equation is known, the spray agent can calculate its own behaviour by evaluating the macro-level equation.

In a series of 100 simulations of a spray of droplets launched from an aeroplane, micro-level droplets behaviour was represented by the droplets' trajectories as determined by hydrodynamic laws, including turbulence effects.

To construct the equation, the genetic programming algorithm could use the wind speed variable W, random constants, as well as the standard arithmetic operators of addition, subtraction, multiplication, and division.

Results

The plot in Figure 6.4 shows the individual droplet trajectories of a spray of 50 droplets. Droplets are released from an aeroplane situated at the left hand side of the diagram, and are transported by the wind in the right hand horizontal direction. Hydrodynamic forces, including random turbulence effects, determine transport in the vertical direction.

In Figure 6.5, the data points represent pesticide deposition (number of droplets deposited) at a given distance of 400 – 500 m from the source, as a function of wind speed (in m/s). Each data point is the result of a different run of the multi-agent simulation with randomly selected wind speed. The function line represents the equation found by a genetic programming algorithm performing a symbolic regression to find the equation best fitting the data points. The result was obtained after 10 generations of a population of 1024 individuals.

The equation found was

$$V = \frac{(-1.206 + 18.50 * W)}{(-0.8883 - W) * (-0.7853 - 0.8104 * W^2)}$$

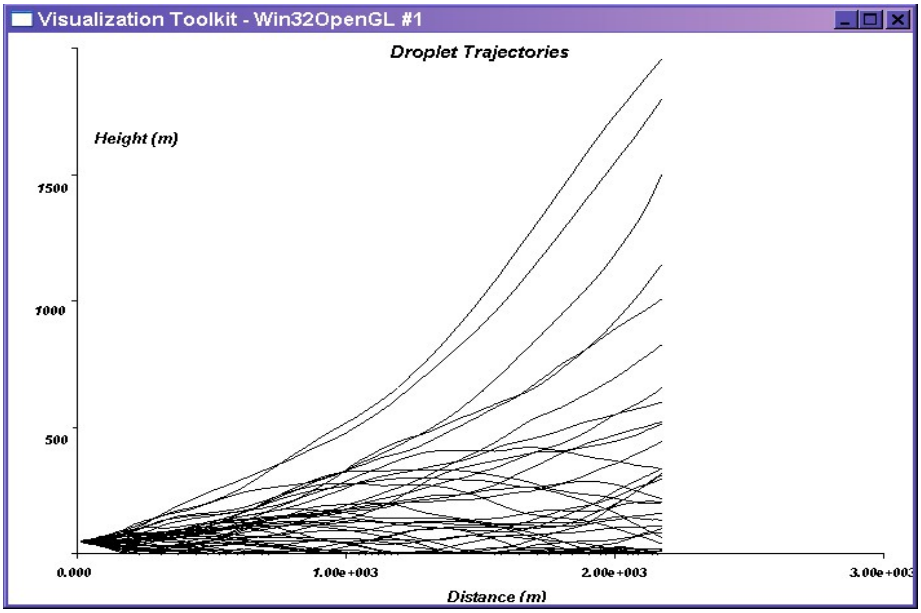


Figure 6.4. Individual droplet trajectories.

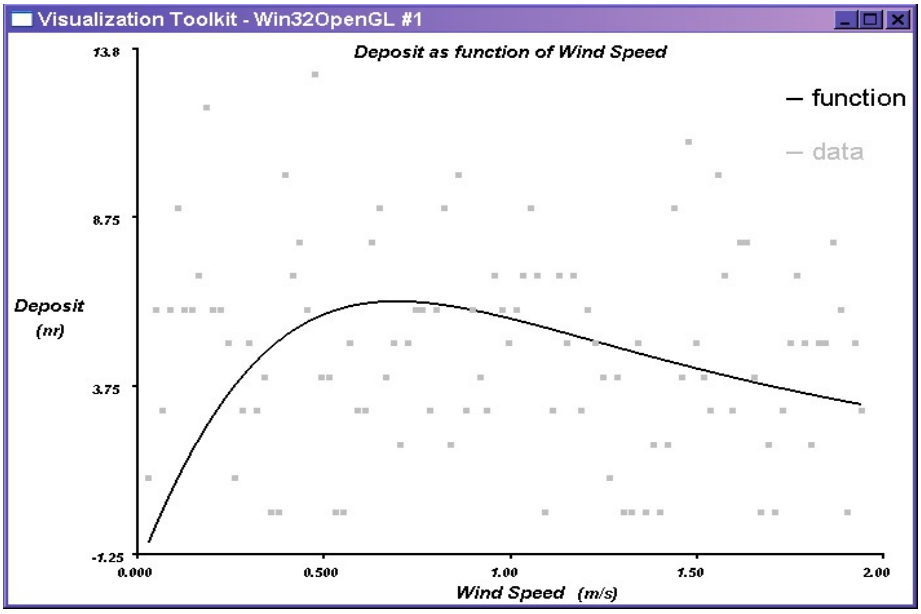


Figure 6.5. Emergent model of pesticide deposition (number of droplets deposited) at a given distance from the source, as a function of wind speed.

Discussion

In this example of applying the Emergent Models methodology, a group agent representing the cloud of pesticide automatically derives a relationship between input and output quantities. This has been realised as follows using genetic programming.

- It is assumed that a macro-level equation describing a relationship between input and output variables can be constructed using known variables and arithmetic operators. No other knowledge of this equation is required, so genetic programming is more general than an evolution strategy, which only determines unknown parameters of a function.
- Using a Monte Carlo-like approach, a number of combinations of input variables is selected at random, to be used as inputs for both the multi-agent simulation at the micro-level and the macro-level equation. As opposed to observational or experimental data, we have the advantage that input variables can easily be drawn from the complete space of all possibilities.
- An initial population of candidate macro-level equations is constructed at random. In contrast, for an evolution strategy only an initial guess for the unknown parameters of the function is required.
- The output variables are computed for each selection of

input variables, both by running the micro-level multi-agent simulation and as a result of the macro-level equation. We take the results of the micro-level multi-agent simulation as the 'real result'. Comparing the outcome of the macro-level equation with this target, the deviation is computed.

- The genetic programming algorithm now replaces a number of less fit individuals (functions with large deviations) in the population with fitter individuals. After a number of runs, an approximately optimal function will be found and the macro-equation will yield about the same results as the full multi-agent simulation.

Because of the simplicity of this example, the problem of finding a macro-model is reduced to a curve-fitting problem, for which other methods than genetic programming are obviously available. However, genetic programming is a general method that can also be used to discover more complicated macro-models.

One can further observe that, because of the small number of droplets in the simulation, data values are very dispersed and the fit of the macro-level function is not particularly good. Finally, no effort has been made to obtain physically realistic solutions, and there are some negative function values. However, this example only serves to illustrate the basic idea of Emergent Models as simply as possible. The

genetic programming algorithm just finds a solution that approximates the data supplied to it and thus has the basic limitation that the found solution can only be as good as these data. In order to obtain realistic solutions applicable to a given problem, care has to be taken to use data which are representative of situations encountered in reality. A statistical approach can be used to obtain quantitative measures of the representativeness of the data and the limits of applicability of the derived solutions. It is also possible to impose suitable constraints on possible solutions produced by genetic programming to ensure more realism. Examples of imposing such constraints will be studied in the following, in particular in Chapter 7.

In conclusion, this simple example demonstrates how emergent models can be derived for a cloud of droplets, starting from micro-level multi-agent simulations of individual droplets behaviour.

6.3 Individuals and Population Movement: Insects in an Odour Plume

In this example we study how micro-level movement behaviour of individual insects gives rise to macro-level behaviour of an insect population. Often one will only be interested in relationships between certain selected macro-properties. For example, when considering insects in a field of

plants, we would like to know how numbers of insects at a certain location depend on variables such as odour concentration of a pheromone, wind speed, and leaf area index.

The application developed in this example involves computer modelling of insects moving under the influence of an odour plume. The computer model produces simulated insect trajectories. The odour plume is given as a static set of odour concentrations and its behaviour is not modelled. Extending the model with an odour plume modelling component would be relatively straightforward.

The results of the computer simulations include the odour concentration at the source of pheromone as an input quantity and the number of insects arrived at a location in the canopy near the source after a certain time as an output quantity.

It was hypothesised that a macro-level relationship between odour concentration and number of arrived insects could be derived from micro-level insect behaviour, using a genetic programming algorithm.

Method

We want to find a function describing arrived numbers of insects I as determined by the odour concentration at the source C, as follows:

$$I = F(C) \tag{6.3}$$

The input variable determines the behaviour of each insect. In the case of variables that are properties of another agent, such as the canopy properties, they can be thought of as messages sent by that agent (the canopy agent) to the insect agents or to the insect population agent.

Effects determining insect agent behaviour are:

- an insect has an initial location and velocity determined at random;
- insects make counterturning (zigzagging) movements in a random fashion in the absence of odour or visual cues: in this experiment it was assumed that an insect turns every 50 time steps with an angle drawn from a uniform distribution between 150 and 210 degrees;
- when smelling an attracting odour, insects adapt the counterturning angle and frequency in such a way that they tend to move upwind: the angle is increased or decreased to be closer to the direction of the odour source and the number of time steps between turns is made inversely proportional to the odour concentration;
- insects make their speed inversely proportional to odour concentration, so they slow down and eventually stop when approaching the odour source.

In this way insect arrival is determined by the interaction of insects with the odour plume. Modelling each insect as an insect

agent with a behaviour incorporating all these effects enables us to simulate the population as a multi-agent system. The population agent's behaviour incorporates macro-level equation (6.3), which is not yet determined. Micro-level simulations were run with 40 different odour concentration values to produce data, which were kept in a database. A genetic programming algorithm used these data to discover the macro-level equation, as follows.

For the set of input data values over which the macro-level equation was to be optimised, an error function was defined to measure the deviation of the output of equation (6.3) from the multi-agent output as

$$E = \sum_{d \in D} \left| I_d - J_d \right|$$

where:

I is the number of arrived insects as computed by the multi-agent simulation;

J is that number as computed by the macro-equation.

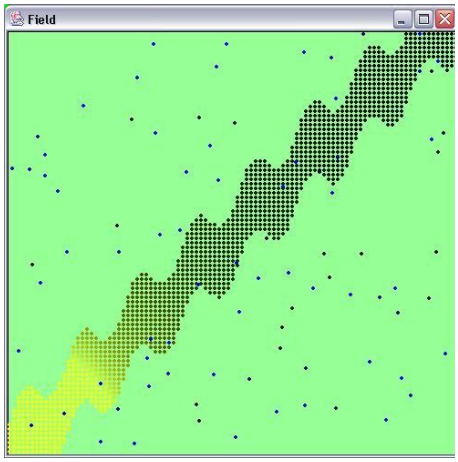
In the same way as in the spray example, the population agent can now calculate its own behaviour by evaluating the macro-level equation found by genetic programming.

To construct the equation, the genetic programming algorithm could use the odour concentration variable C,

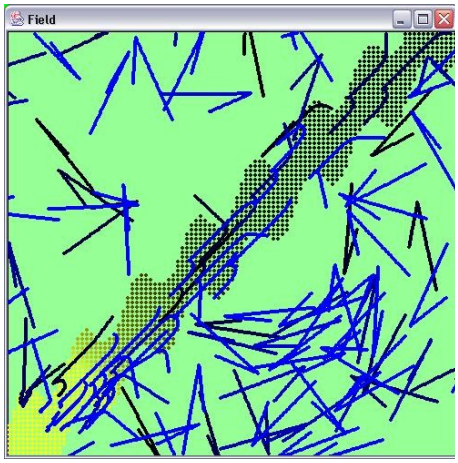
random constants, as well as the standard arithmetic operators of addition, subtraction, multiplication, and division.

Results

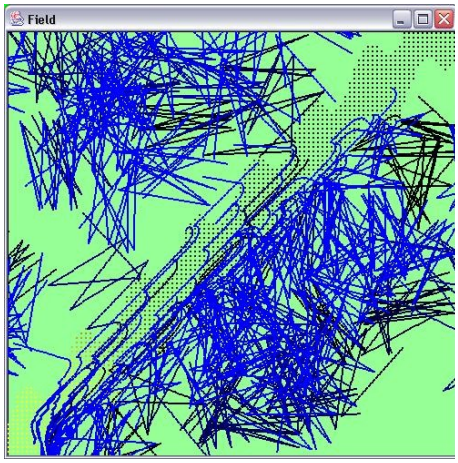
The plots in Figure 6.6 show random initial positions and trajectories of a number of insects behaving in this way. A pheromone plume comes from the left hand bottom corner of the plot. Concentration diminishes with distance from the source. The first plot shows the odour plume and random initial positions of the insects. Darker colour of the plume indicates lower concentration. Odour concentration is assumed to be inversely proportional to square distance from the source. The second plot shows insects' trajectories during the first 50 time steps of the simulation. Insects start moving with random counterturns in the absence of odour. Once they get to a position where they smell odour, they start moving in an upwind direction. Odour being propelled by the wind, that is also the direction of the odour source. After 500 time steps, many insects have found the odour plume and moved in the direction of the source, and a number of them have arrived in the neighbourhood of the source, as shown in the third plot. Insects decrease their speed with increasing odour concentration and thus tend to stop movement in the neighbourhood of the source. It is assumed they will then orient themselves visually to arrive at the source itself, but that is not modelled in the present simulation.



Initial positions of insects.



Positions of insects after 50 time steps.



Positions of insects after 500 time steps.

Figure 6.6. A simulation of insects interacting with an odour plume

Running a number of simulations with different odour concentrations at the source, a genetic programming algorithm found the relationship in Figure 6.7 between source odour concentration and number of insects arrived at positions within a small distance from the source after some number of time steps. This result was obtained with a population of 1024 after 50 time steps. The result shows that fewer insects arrive as the concentration increases, due to the fact that higher concentration leads to lower insect speed. This is a behaviour feature normally useful to make the insects stop when near the source. Here we see clearly that it also implies one cannot assume to trap more insects just by increasing odour concentration of a bait.

Discussion

This example of applying the Emergent Models methodology demonstrates how a group agent representing a population of insects can automatically derive a relationship between an environmental input and a macro-level property of the population. The macro-level emergent model of a population of insects has been derived using genetic programming, starting from micro-level multi-agent simulations of individual insect behaviour.

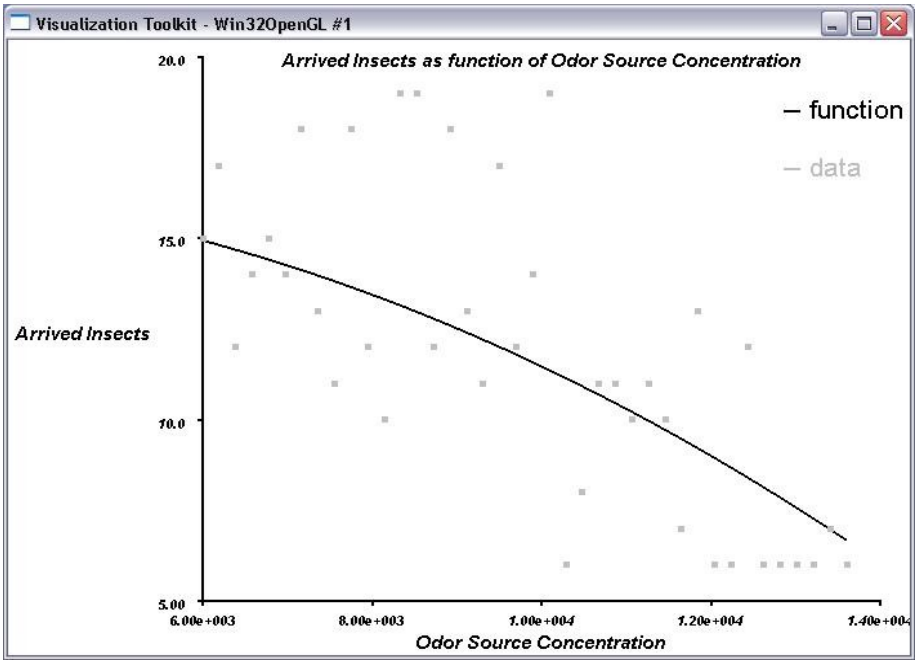


Figure 6.7. Emergent model of number of arrived insects as function of odour concentration.

As the example in Section 6.2, this example is a simple one designed to demonstrate a principle rather than produce exact results, so in Figure 6.7 a large scatter of data is observed, due to the limited number of insects in the simulation. It seems likely that increasing the number of agents would reduce the scatter, but lead to a similar distribution and derived function. This is an experimental matter and the quality of the derived solution will depend on the representativeness of the data. In realistic applications care should be taken to design the number of agents in a simulation, the random components of

their behaviour and so on in such a way that statistically significant results are obtained.

A more important methodological limitation is that the population agent only exhibits a behaviour emerging from its micro-level members, but does not itself interact with other group agents. The example in Section 6.4 will show how emergent group agent behaviour can be used in higher level simulations.

6.4 Individuals and Population Dynamics: Monarch Butterflies and Milkweed

Metapopulation dynamics is an interesting area in ecology to apply the Emergent Models methodology. Metapopulation dynamics distinguishes three levels in an ecosystem, with organisms at the lowest level, subpopulations of organisms at an intermediate level, and the total population consisting of subpopulations at the highest level. Thus, emergent models of subpopulation agents could be discovered from properties and behaviour of organism agents, and an emergent model of the total population could be discovered from properties and behaviour of subpopulation agents. Potential applications would include estimating the potential for survival, or the expected time to extinction, of a population as a function of landscape characteristics. This could be done much more effectively by using simulations with subpopulation agents (with agent models

derived from previous individual-based simulations) to predict total population behaviour, than by using individual-based simulations directly.

A fairly simple system to study metapopulation dynamics consists of monarch butterflies and milkweed plants. Milkweed is the only host plant of the monarch butterfly for feeding and egg laying (Zalucki 1983). In addition, milkweed grows in patches, so the butterflies tend to concentrate in and around milkweed patches, giving rise to a system where metapopulation dynamics are easily studied. Figure 6.8 shows a simulation of monarch butterflies interacting with milkweed patches. The butterflies fly a distance d at each simulation step, while turning an angle Θ (picked at random). If butterflies observe the presence of milkweed plants, they decrease d and increase the average value of Θ, as described in (Zalucki 1983). In addition, butterflies lay eggs on milkweed plants, which hatch after some time, producing newly born butterflies. Butterflies die with a certain probability, which is lower in milkweed patches than in the non-milkweed area outside the patches.

I have carried out computational experiments to derive a population dynamics model at the level of subpopulations from simulations of individual butterflies in interaction with milkweed plants. A subpopulation is defined as the population of one milkweed patch.

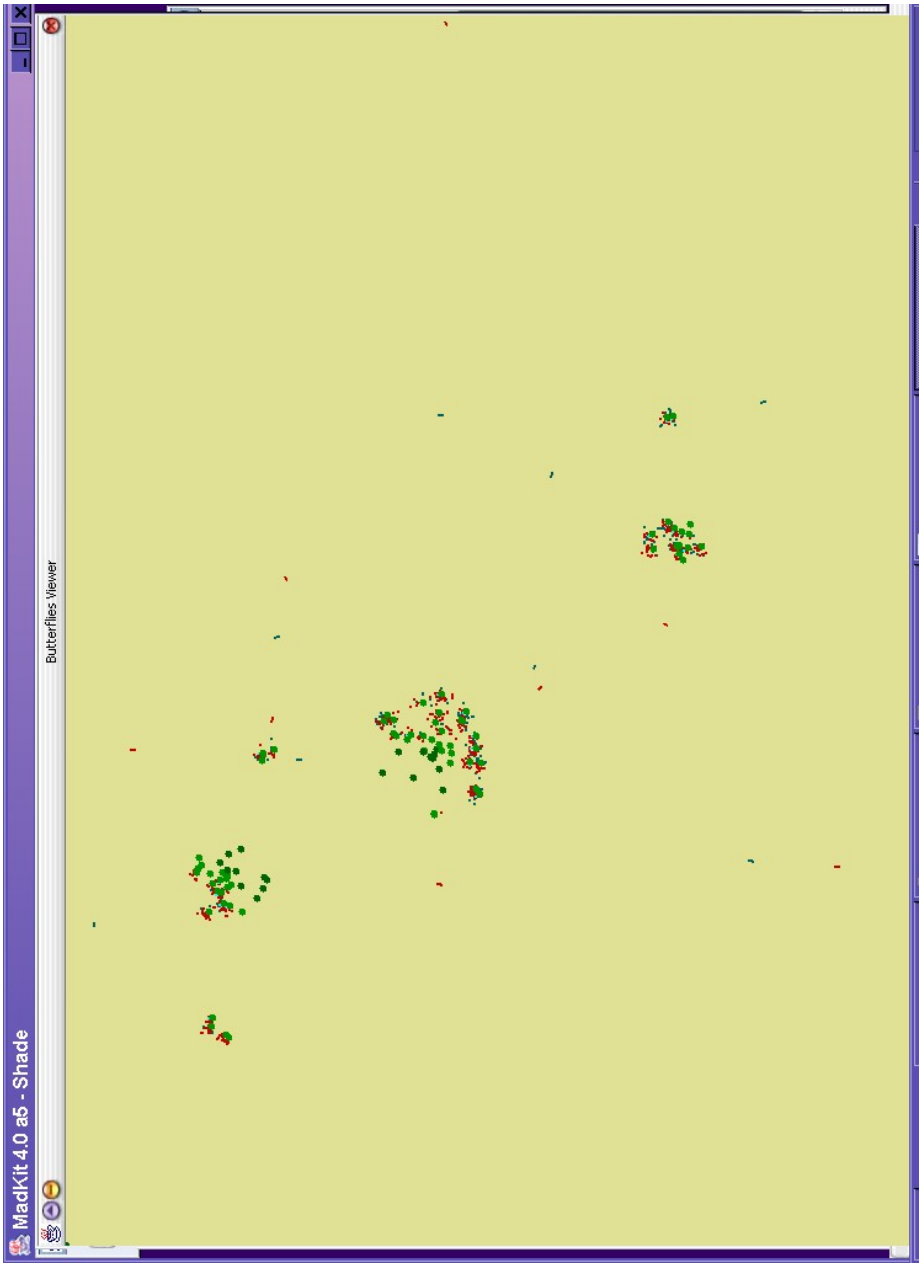


Figure 6.8. Monarch butterflies and milkweed patches.

I subsequently used the subpopulation models in a multi-agent simulation with subpopulation agents, hypothesising that simulations with subpopulation agents would yield results similar to those of simulations with individual insect agents, but with a much better performance. In particular, I derived relationships describing births, deaths, immigration, and emigration for each patch as functions of patch populations. Birth and death functions modelled patch subpopulation agents' autonomous behaviour, while immigration and emigration functions modelled interactions between patch subpopulation agents.

Method

A micro-level simulation was carried out to simulate movements, as well as births and deaths, of a population of monarch butterflies. The simulation started with 100 monarch butterflies at random positions and 100 milkweed plants, planted in 6 patches of various sizes. Patch positions and positions of plants within patches were also determined at random.

To determine its movements, at each time step of the simulation a butterfly observed its environment to know whether it was in a plant patch. It then moved a certain distance in a direction corresponding to a certain turning angle. When outside plant patches, it increased or decreased its turning angle with an angle drawn from a uniform probability distribution with a maximum of 0.1π radians, and moved a distance of 2 m.

When in a patch of plants, this maximum turning angle was 0.4π radians, and the moving distance 0.1 m. The exact values of these parameters are fairly arbitrary. It was expected that increasing turning angle and decreasing velocity when in a plant patch would cause the butterflies to spend more time in plant patches than outside patches. This kind of behaviour has been observed in monarch butterflies (Zalucki 1983).

To model births and deaths, each butterfly started with 70 eggs. At each time step, if it was in a plant patch, it laid an egg on a plant selected at random with a probability determined by the plant's attractiveness, which was set to an arbitrary value of 0.25 for each plant. Eggs hatched 10 time steps after being laid with a probability of 0.02 to produce a new butterfly. Butterflies dies at each time step with a probability of 0.05 outside plant patches and a probability of 0.01 in plant patches.

The simulation was run with time steps of 1 minute for a total simulated time of 38 hours. Subpopulation numbers, as well as numbers of born, died, immigrated, and emigrated butterflies for each patch and for the non-milkweed area, were collected in a database after each simulated hour and are shown in Appendix 1, Table 1.

A genetic programming algorithm was then run to find models for each patch relating births, deaths, and emigration for that patch in each simulated hour to the subpopulation of the same patch at the beginning of the hour. In the non-milkweed

area, eggs were not laid, so births did not occur, and models including only deaths and emigration were sought. Immigration was assumed to be equal to total emigration from elsewhere. It was also assumed that immigration and emigration only occurred between patches and the non-milkweed area, and not directly between patches. So immigration for a patch was equal to emigration from the non-milkweed area to that patch, and immigration to the non-milkweed area was equal to emigration from all patches.

Therefore, immigration of a patch was constrained to be a function of the subpopulation of the non-milkweed area. Births, deaths, and emigration of a patch were constrained to be a function of the patch subpopulation itself. The building blocks of the genetic programming algorithm were the usual arithmetic operators of addition, subtraction, multiplication, and division, along with the relevant population variables, as well as random constants. The data, especially for births and deaths, suggested some kind of (damped) periodic behaviour, so a sinus function was also allowed to be used by the genetic programming algorithm.

Using the derived behaviour models of the subpopulation agents, a simulation was run to predict the evolution in time of the total butterfly population. The time evolution produced by the subpopulation agents simulation was compared to that produced by the individual butterfly agents simulation. Computer time needed by both simulations was also recorded.

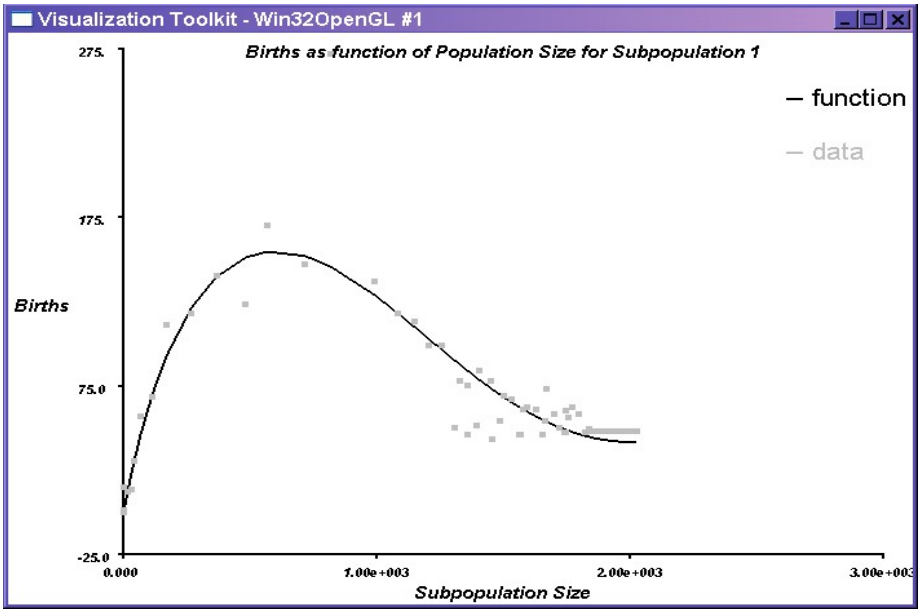
Results

The genetic programming algorithm found relationships for births, deaths, and emigration of each subpopulation at each time step as a function of its own subpopulation level at the beginning of the time step. Relationships for immigration, taken to be a function of the non-milkweed subpopulation level P_0, were also found. These results were obtained after 50 generations of a genetic programming population size of 1024.

The relationships found for births B_1, deaths D_1, and emigration E_1 of subpopulation 1 at each time step as a function of its own subpopulation level P_1 at the beginning of the time step, and for immigration I_1 as a function of the non-milkweed subpopulation level P_0, are shown in Table 6.1. Subpopulation 1 is the subpopulation of the biggest patch in Figure 6.8. Emergent patch subpopulation behaviour is graphically shown for subpopulation 1 in Figure 6.9.

The subpopulation level time step was 1 simulated hour, as opposed to the individual level time step of 1 simulated minute. Similar relationships were found for all patch subpopulations.

Results of the subpopulation level multi-agent simulation are compared with results of an individual butterfly level multi-agent simulation in Figure 6.10. Computer time needed for both simulations is compared in Table 6.2.



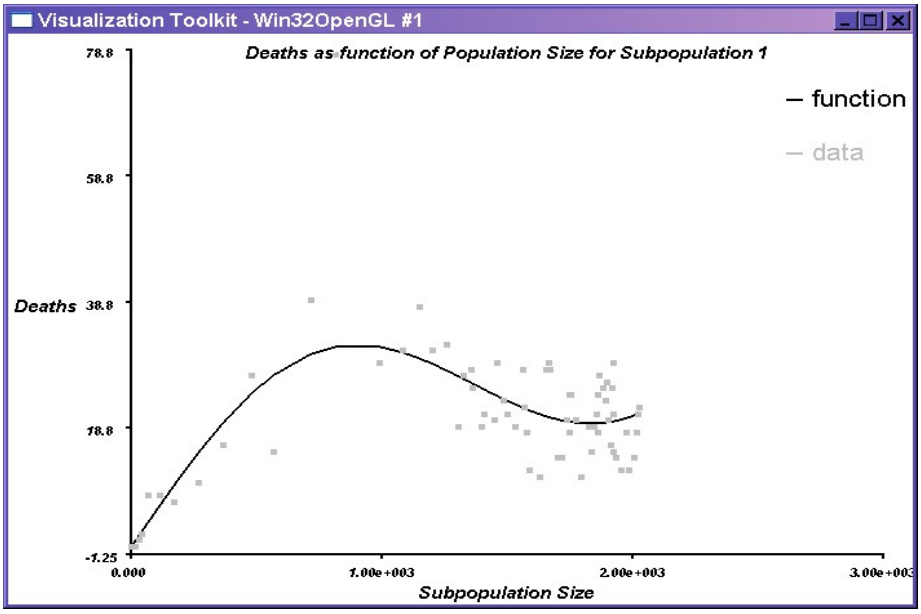
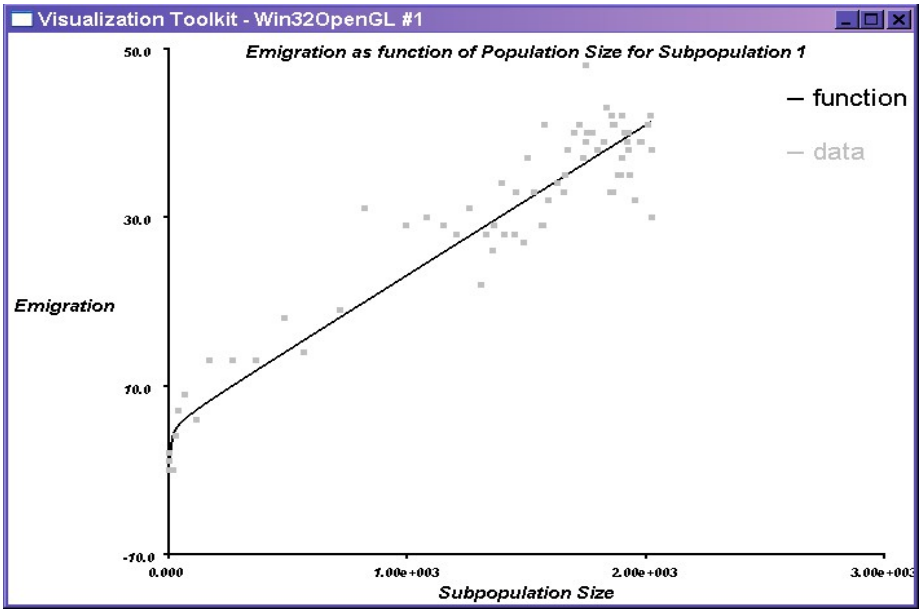


Figure 6.9a. Births and deaths as function of patch subpopulation size for patch 1.



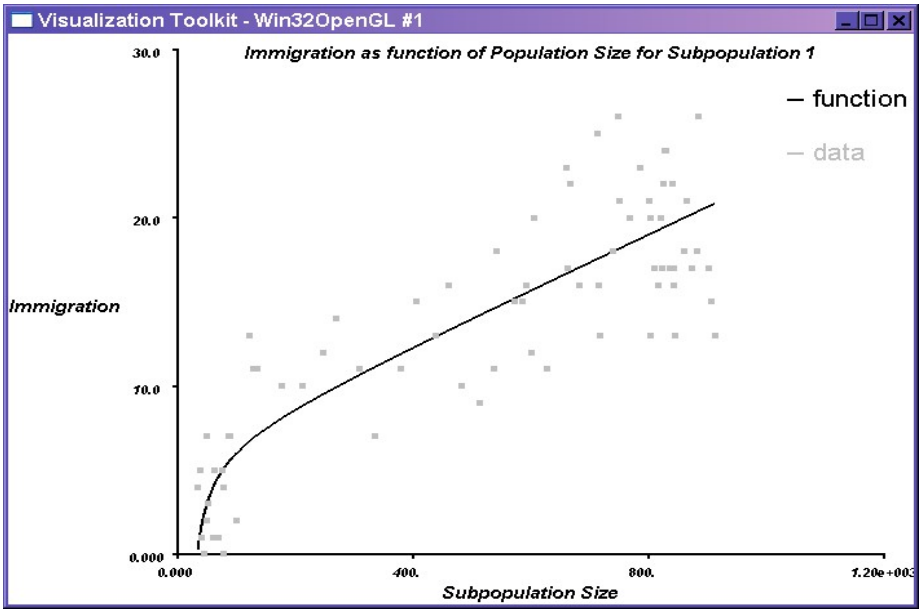


Figure 6.9b. Emigration as function of patch subpopulation size and immigration as function of non-milkweed area subpopulation size for patch 1.

$B_1 = 75.29 + \frac{\sin(330.2 - 0.002403 * P_1)}{0.003912 + 1.250\text{E-}5 * P_1}$
$D_1 = -0.1796 + 0.01893 * P_1 + 17.15 * \sin(0.002297 * P_1)$
$E_1 = 5.364 + 0.01779 * P_1 - \frac{38.66}{6.290 + P_1}$
$I_1 = \frac{-402.0 - P_0}{-61.34 - \frac{14489}{-323.5 + 9.845 * P_0}}$

Table 6.1. Subpopulation behaviour model for subpopulation 1.

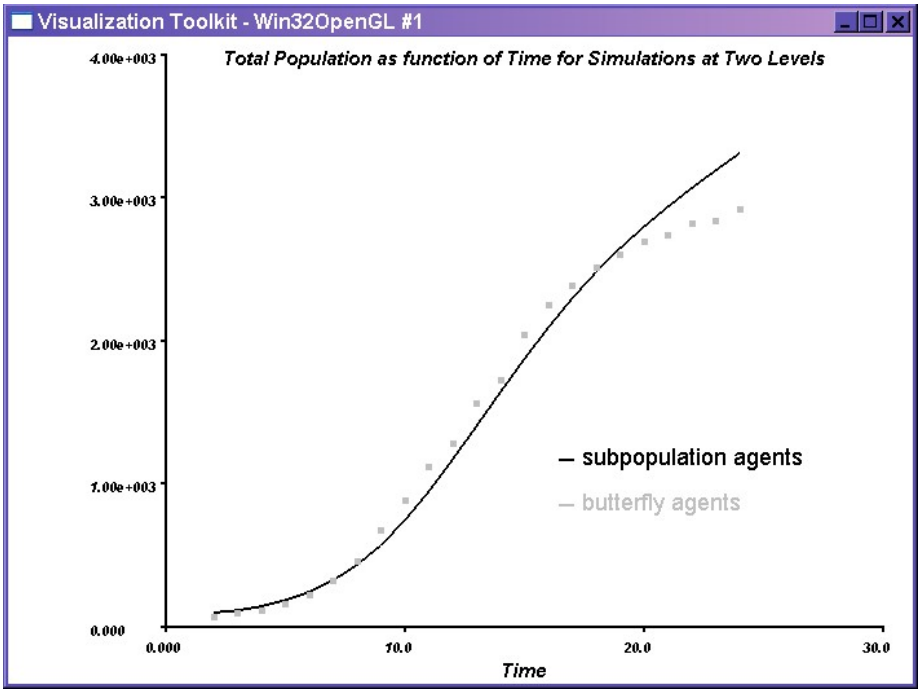


Figure 6.10. Results of subpopulation level and individual butterfly level multi-agent simulations.

simulated hour	elapsed computer time in seconds	
	individual butterfly agents	subpopulation agents
1	5	0.04
2	7	0.05
3	9	0.05
4	11	0.05
5	13	0.05
6	15	0.06
7	18	0.09
8	19	0.09
9	22	0.11
10	25	0.12
11	29	0.12
12	34	0.13
13	44	0.13
14	66	0.13
15	109	0.13
16	181	0.14
17	293	0.14
18	462	0.14
19	695	0.14
20	1000	0.14
21	1400	0.14
22	1890	0.14
23	2457	0.14
24	2982	0.16

Table 6.2. Computer time of individual agents simulation and group agents simulation.

Discussion

This example demonstrates how, in a three level system, intermediate level group agents can derive their behaviour from micro-level agents. Once that is done, simulations can be run with the group agents instead of the individual agents, yielding similar results.

A complex system can thus be understood at the level of subsystems, in addition to individual components.

Further, the group agent simulations use much larger space and time scales than the individual level simulations, enabling a performance that is better by an order of magnitude.

In order to focus on the simulation's multi-level character in this example and to keep its implementation as simple as possible, the derivation of the subpopulation agents' macro-level behaviour is based on data generated by only one micro-level simulation for only 24 time steps. Because of these limitations the fit of the macro-level function to the data is no longer very good after the time span used for the experiment.

It would be a straightforward exercise to run a representative number of micro-level simulations with different initial conditions and for a sufficient number of time steps, as was done in the examples described in sections 6.2 and 6.3. The derived macro-level behaviour would then be an average behaviour for all possible micro-level simulations, making the group agent simulation results more robust.

6.5 Conclusion

In conclusion, with the Emergent Models methodology one can find emergent macro-models corresponding to micro-models of ecological systems. Given properties and behaviour of micro-level entities, for example droplets or organisms, it is possible to derive properties and behaviour of macro-level entities such as pesticide clouds or subpopulations. We have seen in Section 6.1 how this can work for a very simple system of a cloud consisting of droplets, the droplet's behaviour being governed by physical equations. In Section 6.2 the droplets have been replaced by insects with movements defined by a simple behaviour model describing insects' response to environmental conditions as determined by an odour plume. Finally, in Section 6.3 a three level system has been described consisting at the micro-level of insects interacting with plants, at an intermediate level of subpopulations of insects corresponding to plant patches, and at the macro-level of the entire system of insect population and plant patches.

Once a model of subpopulation behaviour has been derived, “subpopulation-based” simulations can be run instead of individual-based simulations to obtain essentially the same results for the behaviour of the whole system or population.

In this way, understanding of behaviour at the macro-level can be increased. In addition, simulation performance can be improved dramatically, at least after a “learning phase”, during

which the macro-level behaviour is discovered by the genetic programming algorithm.

The examples in the present chapter have been kept as simple as possible to demonstrate the possibility of using the emergent methodology in a few illustrative situations. In practical applications many precautions will have to be taken to ensure realistic results. One such precaution is the use of Monte Carlo designs, running statistically significant numbers of micro-level simulations to ensure representativeness of the discovered emergent models, including a good fit to typical system behaviour. Further, micro-level simulations will have to be run for adequately defined ranges of initial and external conditions, and for long enough time spans, to cover all situations of interest and ensure applicability of the discovered emergent models to these situations.

Most importantly, emergent models should not be limited to simple functions fitted to data, but model macro-level agent behaviour in a realistic way. If building blocks are specified that can be used to construct complete definitions, i.e. agent-oriented behavioural specifications, of emergent agents, the methodology can be applied to much more complex systems than the ones examined in this chapter. Emergent models are only limited by the building blocks used to construct them.

It is important to understand that the Emergent Models methodology is not just a system identification tool. Rather, it

uses a set of technologies, such as multi-agent simulation and system identification tools, to achieve the purpose of deriving emergent properties and behaviour from lower level properties and behaviour in a complex system. Other tools than the ones used in the present thesis can be integrated without methodological difficulty, for example other system identification tools than genetic programming, geographic information systems to situate multi-agent simulations in space, etc.

To realise the full potential of Emergent Models, macro-level agents and their emergent properties will have to be defined in an adequate way to make possible the discovery of useful emergent models. In particular, emergent properties will have to be defined in such a way that useful models become possible. As discussed in Chapter 5, defining emergent properties is far from straightforward and still very much an art.

7 From Macro-Level to Micro-Level in Genetic Networks

On a lower level than an ecosystem, living organisms in turn consist of interacting objects such as organs, cells, and genes. Therefore, another application area of the Emergent Models complex systems simulation methodology is the individual living organism. This chapter focusses on the relationship between interacting genes and phenotype.

In molecular biology it is particularly interesting to derive micro-level models from macro-level data, as data are often available at the macro-level (the level of the phenotype), while the genetic and biochemical mechanisms at the micro-level (the molecular level) remain unknown. This chapter illustrates how micro-level models can be derived from macro-level data in prototype applications for some problems in molecular biology with genetic and biochemical networks.

To make clear what a *genetic network* or, to use a longer but more descriptive expression, a *genetic regulatory network* is, consider that multi-cellular organisms consist of many cells containing the same set of genes. Yet these cells are very different, because the genes are not expressed in the same way in each of them. Cell architecture and behaviour are determined by gene products, not the genes themselves. Genes can regulate the production of gene products by other genes in such

a way that not all genes are expressed in all cells all the time (Ptashne & Gann 2002, p. 3). A set of genes and gene products, together with their regulatory interactions, constitutes a genetic regulatory network. A model of such a network describes interactions between DNA, RNA, proteins, and small molecules in an organism, through which gene expression is controlled (De Jong 2002). Various formalisms have been proposed to model genetic networks, including directed graphs, Bayesian networks, Boolean networks and their generalisations, stochastic master equations, ordinary and partial differential equations, qualitative differential equations, stochastic master equations, and rule-based formalisms (De Jong 2002, p. 69). These formalisms have in common that gene expression and gene product levels are represented as nodes, and their interactions as links in a network.

For example, in a *Boolean network* (Kauffman 1993; De Jong 2002) the state of a gene is described by a Boolean variable with value $true$ or 1 for an active gene (gene products present) and value $false$ or 0 for an inactive gene (gene products absent). Interactions between genes are represented by Boolean functions calculating the state of a gene resulting from activation and/or inhibition by other genes. When the evolution of a Boolean network is calculated during a number of time steps, an attractor -steady state or state cycle- is typically reached.

Differential equations have also been widely used to model genetic networks (De Jong 2002, p. 77-89). Levels of gene products are modelled more realistically than in Boolean networks, as real values. Regulatory interactions, such as activation or inhibition, between gene products are modelled by rate equations.

In the same way as a genetic network, a biochemical network models interactions in an organism between molecules that are not directly related to its genes. A genetic or biochemical network consists of interacting elements and can be considered a *complex system*. In a genetic network the micro-level consists of DNA, RNA, and protein molecules, interacting with each other in excitatory or inhibitory ways (Bower & Bolouri 2001). The macro-level is that of the organism's phenotype as determined by the expression of its genes.

Therefore, a genetic or biochemical network can be simulated in a straightforward way as a *multi-agent simulation*, representing genes and gene products as interacting agents on the micro-level. These interacting agents model a genetic and/or biochemical network, which can describe the properties and behaviour at a higher level of a cell, an organ, or a whole organism.

If parts of such a network are relatively autonomous subsystems or modules related to specific functions in the

organism (for example a biochemical clock mechanism), these modules can in turn be described as agents whose properties and behaviour give rise to those of the whole organism. In this case there are three levels, a micro-level of molecules, an intermediate level of modules, and a macro-level of the whole organism.

Now, can we derive micro-level models from macro-level data? Given a set of macro-level data about the phenotype, can we find a genetic network with a pattern of activity explaining the phenotype data of the whole organism or its organs? In the case of models with three levels, there are two distinct problems. One of these is to find a network of genes and gene products plus their interactions, that explains properties and behaviour of a module. The other problem is to find a set of modules and their interactions explaining properties and behaviour of the whole organism.

To show how these problems can be solved, I conducted a number of computational experiments with models of genetic networks regulating branching and flowering in pea (*Pisum sativum L.*) to determine to what extent the characteristics of such genetic network models can be automatically derived from data collected in biological experiments. The data used described effects on the pea phenotype caused by mutations of the genes thought to regulate branching and flowering.

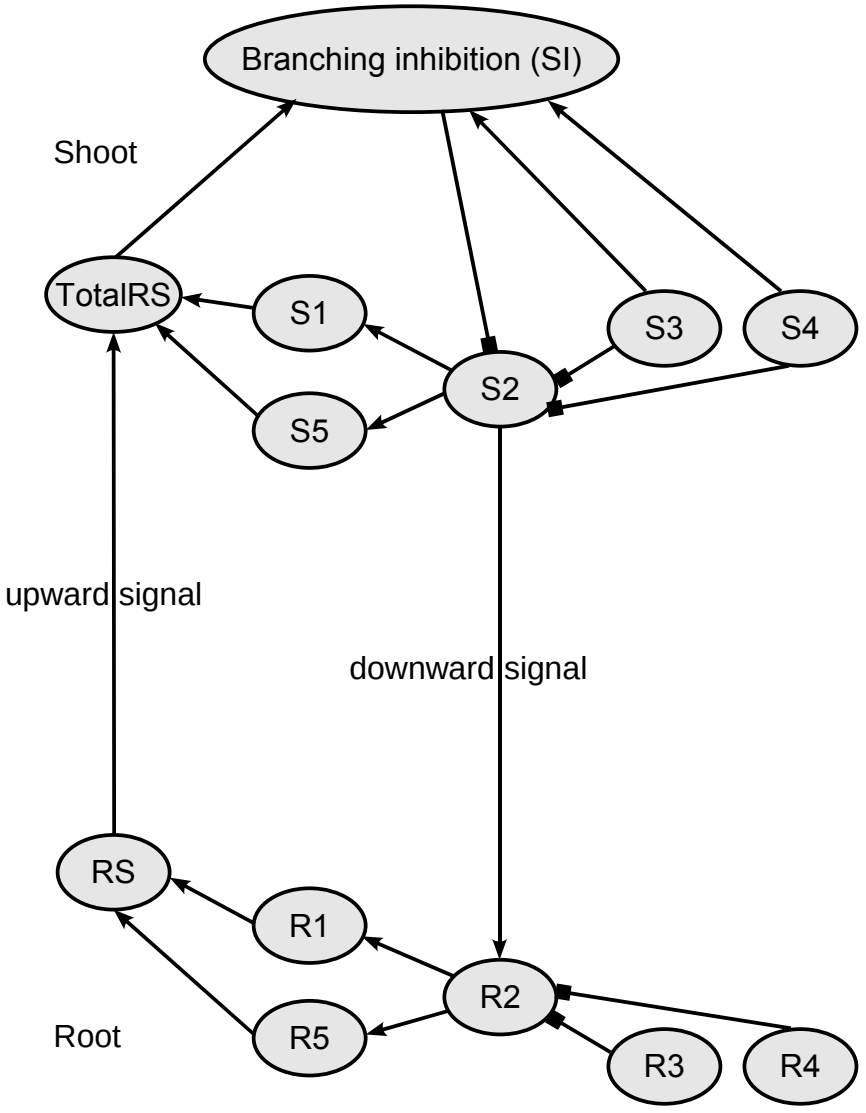


Figure 7.1. Network of pea genes determining branching (based on Harding 2003).

7.1 Branching in Pea: Genes, Signals, and Phenotype

A model of a genetic network regulating branching in pea, formulated in the context of a biological research project (described in Harding 2003), is schematically presented in Figure 7.1, showing relationships between genes, gene products, signals, and phenotype. The model was developed to explain experimental data on the branching phenotype of several root-shoot grafts, a root of one genotype being grafted to a shoot (scion) of the same or a different genotype. The experiments also included grafts of one root with two shoots - a scion and a cotyledonary shoot – the cotyledonary shoot in general having the same genotype as the root. Because of their general shape such grafts are called Y-grafts, in contrast to I-grafts of one root and one shoot. Similarly, two roots could be grafted to a single shoot to get a two-root graft.

In Appendix 2, Table 1, the phenotype of a plant is described by the level of branching inhibition. A plant with wild type root and scion has by definition a branching inhibition of 1. Smaller values of branching inhibition imply more branching. Observed values (columns E) are compared to model results.

The model consists of the equations in Table 7.1. The original model has been slightly modified here to bring out more clearly the distinction between gene presence and gene product activity, as well as the modular organisation of the plant.

$s3 = S3$	
$s4 = S4$	
$s2 = \frac{2}{1+si} \times S2$	(3.1)
$s1 = S1 \times s2$	(3.3)
$s5 = S5 \times s2$	(3.4)
$c3 = C3$	
$c4 = C4$	
$c2 = \frac{2}{1+ci} \times C2$	(3.2)
$c1 = C1 \times c2$	(3.5)
$c5 = C5 \times c2$	(3.6)
$r3 = R3$	
$r4 = R4$	
$sr = splitD \times d_2 \times s2$	(2.1)
$cr = splitD \times d_2 \times c2$	(2.1)
$r2 = \frac{2}{1+r3 \times r4} \times R2 + sr + YGraft \times cr$	(2.1), (4.1)
$r1 = R1 \times r2$	(3.7)
$r5 = R5 \times r2$	(3.8)

Table 7.1. Branching model for pea (adapted from Harding 2003; Harding's numbering of equations is used).

$rr3 = RR3$	
$rr4 = RR4$	
$rr2 = TwoRoot \times (\frac{2}{1 + rr3 \times rr4} \times RR2 + sr)$	(2.2), (4.2)
$rr1 = RR1 \times rr2$	(3.9)
$rr5 = RR5 \times rr2$	(3.10)
$rs = r1 \times r5$	(3.11)
$rrs = rr1 \times rr5$	(3.12)
$totalrs = s1 \times s5 + splitU \times u_1 \times rs + TwoRoot \times splitU \times u_1 \times rss$	(2.5)
$totalrsc = c1 \times c5 + splitU \times u_1 \times rs$	(2.6)
$si = s3 \times s4 \times totalrss$	(2.7)
$ci = c3 \times c4 \times totalrsc$	(2.8)

where symbols are explained as follows:	
$R1$, $R2$, $R3$, $R4$, $R5$	genes in root A
$RR1$, $RR2$, $RR3$, $RR4$, $RR5$	genes in root B
$S1$, $S2$, $S3$, $S4$, $S5$	genes in the scion
$C1$, $C2$, $C3$, $C4$, $C5$	genes in the cotyledonary shoot
$r1$, $r2$, $r3$, $r4$, $r5$	gene products in root A

Table 7.1 (continued).

rr1 , *rr2* , *rr3* , *rr4* , *rr5*	gene products in root B
rs	root to shoot signal in root A
rrs	root to shoot signal in root B
totalrs	total root to shoot signal in the scion
totalrsc	total root to shoot signal in the cotyledonary shoot
si	branching inhibition in the scion
ci	branching inhibition in the cotyledonary shoot
splitD	fraction of signals moving from shoot to each of two roots
splitU	fraction of signals moving from root to each of two shoots
d_2	proportion of RMS2 controlled downward signal not metabolised
u_1	proportion of RMS1/RMS5 controlled upward signal not metabolised
YGraft	variable indicating if the plant is a Y graft (= 0 if no, = 1 if yes)
TwoRoot	variable indicating if the plant is a two root graft (= 0 if no, = 1 if yes);

Table 7.1 (continued).

Genes are represented by discrete values, only taking values 0 (gene absence) or 1 (gene presence). Gene product levels are represented by continuous variables with any value greater than or equal to 0. Gene product levels are determined by the presence or absence of the corresponding producing gene, as well as activation or inhibition by other gene product levels. For example, the equation $s3 = S3$ explicitly states that gene product level $s3 = 1$ if gene *Rms3* is present and $s3 = 0$ otherwise. In Harding's model, $S3$ is directly used in equations (2.1) and (4.1), which is of course a simplification expressing the same relationship. Having made this conceptual observation, similar simplifications will be used without hesitation in the following.

Using the simplified versions amounts to imposing constraints on the possible models. For example, in the case of *Rms3* the constraint is imposed that gene product level $s3$ only depends on gene activity $S3$ and not on any other gene products. If it is deemed desirable to relax this constraint, the equation for $s3$ can be assumed unknown and left to be discovered.

In addition, gene product levels in one organ (root, shoot, or cotyledonary shoot) depend directly only on gene product levels in the same organ and are not directly influenced by gene product levels in different organs. Any influence from other organs occurs only indirectly through signals.

Equations for I-grafts	
$s2=\frac{2}{1+si}\times S2$	(3.1)
$r2=\frac{2}{1+R3\times R4}\times R2+splitD\times d_2\times s2$	(2.1) (4.1)
$rs=(R1\times r2)\times(R5\times r2)$	(3.11)
$totalrs=(S1\times s2)\times(S5\times s2)+splitU\times u_1\times rs$	(2.5)
$si=S3\times S4\times totalrs$	(2.7)
Equations for Y-grafts	
$s2=\frac{2}{1+si}\times S2$	(3.1)
$c2=\frac{2}{1+ci}\times C2$	(3.2)
$r2=\frac{2}{1+R3\times R4}\times R2+splitD\times d_2\times s2+splitD\times d_2\times c2$	(2.1) (4.1)
$rs=(R1\times r2)\times(R5\times r2)$	(3.11)
$totalrs=(S1\times s2)\times(S5\times s2)+splitU\times u_1\times rs$	(2.5)
$totalrsc=(C1\times c2)\times(C5\times c2)+splitU\times u_1\times rs$	(2.6)
$si=S3\times S4\times totalrs$	(2.7)
$ci=C3\times C4\times totalrsc$	(2.8)

Table 7.2. Simplified branching model for pea (adapted from Harding 2003).

Equations for two-root-grafts	
$s2=\frac{2}{1+si}\times S2$	(3.1)
$r2=\frac{2}{1+R3\times R4}\times R2+splitD\times d_2\times s2$	(2.1) (4.1)
$rr2=\frac{2}{1+RR3\times RR4}\times RR2+splitD\times d_2\times s2$	(2.2) (4.2)
$rs=(R1\times r2)\times(R5\times r2)$	(3.11)
$rrs=(RR1\times rr2)\times(RR5\times rr2)$	(3.12)
$totalrs=(S1\times s2)\times(S5\times s2)+splitU\times u_1\times rs+splitU\times u_1\times rrs$	(2.5)
$si=S3\times S4\times totalrs$	(2.7)

Table 7.2 (continued).

Again, shortcuts can be made to make the model more concise. A simplified model, expressing the same relationships in more succinct form and substantially the same as Harding's multi-cycle model, is derived from the model in Table 7.1 by variable substitutions and is described in Table 7.2.

It was hypothesised that alternative genetic network models could be automatically discovered from the observed data by a genetic programming algorithm. To the extent that the automatically discovered model would be different from the original model, conclusions could be drawn about uniqueness of and possible alternatives to the original model.

7.1.1 Computational Experiments

By carrying out computational experiments I showed that genetic networks regulating branching in pea can be discovered using genetic programming, starting from observed phenotype data.

Inputs for the genetic programming algorithm consisted of variables (gene presence, gene product levels), constants, and functions relating these. Output consisted of equations for branching inhibition si and ci, gene product level $r2$, $s2$, and/or $rr2$, as well as signals rs, $totalrs$, $totalrsc$, and/or rrs. All other equations were assumed to be known, as in Table 7.2.

The genetic programming algorithm was set up to minimise a fitness measure reflecting the deviation of model generated outputs from results as measured in biological experiments. Parameters of the genetic programming algorithm were set in a pragmatic way to obtain results with a good fit to observed data, as the purpose of the experiments was to demonstrate the feasibility of using genetic programming for model discovery rather than to build fully automated discovery software.

Each experiment tended to reveal strengths as well as weaknesses of the genetic programming algorithm, and one experiment's results led to the design of further experiments to build on the strengths and overcome the weaknesses.

Branching Network Experiment 1

Method

In this experiment the algorithm was set up to find expressions for determining the variable (gene product or signal level) in the first column of Table 7.3, using the variables (gene product or signal levels), constants, and functions in the second column. Genes only determined the initial levels of gene products: 0 for absent genes, and 1 for present genes.

The genetic network model was calculated for 25 iterations. Model results were gene product and signal levels at the 25th iteration. The fitness measure was the sum of deviations of model results from observed data for the quantities si and ci: the smaller this sum, the better the fitness. For observed data $si=1$ or $ci=1$, model results greater than 1 were considered to fit the data (i.e. the fitness contribution of these results was set to 0).

Results

A run with 5000 individuals for 101 generations (in total 505000 individuals) of the genetic programming algorithm produced the set of equations in Table 7.4. The found model had a fitness of 3.0, i.e. the sum of deviations of the model output from the observed data was equal to 3.0, a good fit. The number of hits was 54. i.e. a very nearly exact fit was obtained for 54 out of a possible 58 data points.

Gene products	*Building blocks of expressions*
si	$s3$, $s4$, $totalrs$, ERC , $+$, $-$, $*$
$r2$	$r2$, $s2$, $c2$, ERC , $+$, $-$, $*$
rs	$r1$, $r2$, $r5$, ERC , $+$, $-$, $*$
$totalrs$	$s1$, $s2$, $s5$, rs , ERC , $+$, $-$, $*$
ci	$c3$, $c4$, $totalrsc$, ERC , $+$, $-$, $*$
$totalrsc$	$c1$, $c2$, $c5$, rs , ERC , $+$, $-$, $*$

Table 7.3. Branching Network Experiment 1 building blocks for genetic programming. Symbols for variables and operators are as in Table 7.1. ERC stands for *ephemeral random constant*, meaning a constant that is randomly varied by the algorithm to find a good value.

For I-grafts:
$si = s3 * s4 * s4 * totalrs$
$r2 = s2 - 0.629$
$rs = r1 * r5$
$totalrs = s1 * s5 * s2 + (2 * s2 + si) * rs$
For Y-grafts:
$ci = totalrsc$
$r2 = c2$
$totalrsc = (c5 + 0.109) * c2 * rs$
(equations for si, rs, and $totalrs$ are as for I-grafts)

Table 7.4. Branching Network Experiment 1: Model found by genetic programming.

Discussion

These results show already that genetic programming is able to find a model with good fit to given data. However, the model was not deemed biologically realistic, as arithmetic operators such as subtraction were allowed to be used indiscriminately, without any biological justification. Also, only levels of gene products and signals at the 25th iteration were considered as results, which is arbitrary: some of the levels exhibited important fluctuations, and, depending on the exact iteration considered as model result, these levels could be quite different. A further unrealistic assumption was that genes only determine initial levels of their gene products.

Branching Network Experiment 2

Method

In this experiment the genetic programming system looked for expressions for determining the variable (gene product or signal level) in the first column of Table 7.5, using the variables (gene, gene product or signal levels), constants, and functions in the second column. The operator of inhibition, considered more biologically realistic than subtraction, was introduced.

Again, 25 iterations of the genetic network were calculated. Genes were now considered at each iteration, being assigned a value of 0 if not present, and 1 if present.

Gene products	*Building blocks of expressions*
si	$S3$, $S4$, $totalrs$, ERC , $+$, $-$, $*$
r2	$R2$, $R3$, $R4$ $s2$, $c2$, ERC , Inh , $+$, $-$, $*$
rs	$R1$, $R5$, $r2$, ERC , $+$, $-$, $*$
totalrs	$S1$, $S5$, $s2$, rs , ERC , $+$, $-$, $*$
ci	$C3$, $C4$, $totalrsc$, ERC , $+$, $-$, $*$
totalrsc	$C1$, $C5$, $c2$, rs , ERC , $+$, $-$, $*$

Table 7.5. Branching Network Experiment 2 building blocks. Inhibition function Inh is defined as:
$Inh(x, y)=2.0*x/(1.0+y)$.

For I-grafts:
$si = -0.171-S3-0.270*S4+0.270*totalrs +2.0*S3*totalrs+totalrs*(S4-totalrs)$
$r2 = Inh(s2, R2)*(0.130*R4-Inh(R2, R3))+R2$
$rs=(r2+R1)*(R1-r2+2*R5)$
$totalrs=rs+S1*S5*s2$
For Y-grafts:
$ci = totalrsc-C4$
$r2 = R2$
$totalrsc = rs$
(equations for si , rs , and $totalrs$ are as for I-grafts)

Table 7.6. Branching Network Experiment 2: Model found by genetic programming.

Results

A run with 5000 individuals for 201 generations (a total of 1005000 individuals) of the genetic programming algorithm produced the set of equations in Table 7.6, with a fitness of 0.013 and 58 out of a possible maximum of 58 hits.

Discussion

In this result the inhibition operator was used, but, without further constraints, it could be inserted at any place in the equations by the genetic programming algorithm, making them biologically hard to interpret. Genes had a continuous effect on their gene products. Otherwise, the results still had many of the weaknesses of those of Experiment 1.

Branching Network Experiment 3

Method

As in Experiment 2, the algorithm was set up to find expressions for determining the variable (gene product or signal level) in the first column of Table 7.7, using the variables (gene, gene product or signal levels), constants, and functions in the second column.

Genes were again considered at each iteration, being assigned a value of 0 if not present, and 1 if present. In the previous experiments, periodic behaviour of model results was sometimes observed. Therefore, 50 iterations were calculated in

this experiment and the fitness contributions of model results were averaged over the last 10 iterations, so essentially only steady state solutions were obtained.

The subtraction operator was no longer used. Instead, a probably more realistic inhibition function was used. In contrast to Experiment 2, it was constrained to occur only at the top level of a function tree in the genetic programming algorithm. In other words, any equation of the system could model a total inhibitory effect of all variables at the right hand side on the variable at the left hand side. Specifically, it was assumed that the branching inhibition could have an inhibitory effect on the RMS2 gene product, as in Harding's (2003) modified multi-cycle model.

The effects of genes on their own gene products were no longer variables to be used by the genetic programming algorithm, but instead hard-coded in the equations. This is an example of a constraint imposed by a priori biological knowledge.

Random constants were not used in this experiment.

Gene products	*Building blocks of expressions*
s2	*si* , + , * , *Inh*
r2	*R3* , *R4* , *s2* , *c2* , + , * , *Inh*
rs	*R1* , *R5* , *r2* , + , *
totalrs	*S1* , *S5* , *s2* , *rs* , + , *
si	*S3* , *S4* , *totalrs* , + , *
c2	*ci* , + , * , *Inh*
totalrsc	*C1* , *C5* , *c2* , *rs* , + , *
ci	*C3* , *C4* , *totalrsc* , + , *

Table 7.7. Branching Network Experiment 3 building blocks. Inhibition function Inh is defined as $Inh(y)=2.0/(1.0+y)$.

For I-grafts:
$s2 = S2 * si$
$r2 = R2 + s2 * R3$
$rs = r2 * R1 * R5 * R5$
$totalrs = rs + s2 * S1 * S1 * S1 * S5$
$si = totalrs * S3 * S4 * S4$
For Y-grafts:
$c2 = C2 * Inh(ci)$
$r2 = R2 + s2$
$totalrsc = c2 * C1 * C5$
$ci = 4 * totalrsc * C4 * C4$
(equations for $s2$, rs , $totalrs$, and si are as for I-grafts)

Table 7.8. Branching Network Experiment 3: A model found by genetic programming.

For I-grafts:
$s2 = S2 * Inh(si)$
$r2 = R2 + s2 * R3$
$rs = r2 * R1^7 * R5^7$
$totalrs = rs + (S5 * rs^2 + S5 * s2 * S1^3) * S1^2$
$si = S3 * totalrs * S4^4$
For Y-grafts:
$c2 = C2 * Inh(ci)$
$r2 = R2 + c2$
$totalrsc = c2 * C1 * C5$
$ci = 4 * totalrsc * C4^2 + C3 * totalrsc^3 + C4 * totalrsc^3$
(equations for $s2$, rs, $totalrs$, and si are as for I-grafts)

Table 7.9. Branching Network Experiment 3: Another model found by genetic programming.

Results

A run with 10000 individuals for 101 generations (1001000 individuals evaluated) of the genetic programming algorithm produced the set of equations in Table 7.8, with a fitness of 0.600 and 56 hits.

In the 25th generation of this run (after evaluating 250000 individuals), the set of equations in Table 7.9 was found, with a fitness of 1.00 and 56 hits. Output of the model is compared with experimental data in Appendix 2, Table 2.

Discussion

The fitness of the model in Table 7.9 was slightly worse than that of the final model, but the model seems more realistic, as the inhibition function is used for both si and ci. This is an example of biological constraints to be imposed on automatic discovery: functional relationships between corresponding variables should be the same in shoot and cotyledon (see Experiment 5).

The fitness value of about 1.0 was essentially obtained by a nearly exact fit of all model results to the data, apart from two results (for root/shoot grafts rms2/rms1 and rms2/rms5) with a rather large deviation of 0.5. If models with many small deviations are preferred to models with a few larger deviations, this could be a further constraint to be imposed. This is done in Experiment 4.

Branching Network Experiment 4

Method

A different fitness measure was used in this experiment. Because of inherent inaccuracy of data, many small deviations were preferred to a few large ones. Therefore, any model result with a deviation less than 0.25 from the experimental result was considered to have a fitness contribution of 0.0. Building blocks were the same as in Experiment 3.

Results

A run with 10000 individuals for 101 generations of the genetic programming algorithm produced the set of equations in Table 7.10, with a fitness of 0.250 and 57 hits (any model result within a deviation of 0.25 from the experimental result was considered a hit, with a fitness contribution of 0). Model output is compared with experimental data in Appendix 2, Table 3.

For I-grafts:
$s2 = S2 * S1 * (S5 + s2 * si)$
$r2 = R2 + 0.125 + s2 * (R4 + 2.0 * R3)$
$rs = r2 * R1 * R5$
$totalrs = 0.211 * S3 * S4 + S4 * S5 * s2 + rs * S4 * S4 * (S4 + s2) * (0.215 * S4 + 0.755 * rs * S4)$
$si = S3 * totalrs$
For Y-grafts:
$c2 = C2 * C1 * (C3 + C4)$
$r2 = R2 + s2$
$totalrsc = C3 * rs^2 + C3^2$
$ci = c2 * C5^2$
(equations for $s2$, rs, $totalrs$, and si are as for I-grafts)

Table 7.10. Branching Network Experiment 4: Model found by genetic programming.

Discussion

The genetic programming algorithm searched for functional relationships for both the scion and the cotyledonary shoot. As the algorithm can use any function to obtain a good fit to the data, corresponding equations for the scion and for the cotyledonary shoot can be quite different. To obtain more realistic solutions, further constraints have to be imposed.

Branching Network Experiment 5

Method

The same fitness measure was used in this experiment as in Experiment 4. In addition, functional relationships between corresponding variables were constrained to be the same in scion and cotyledonary shoot. Building blocks were the same as in Experiment 3.

Results

A run with 10000 individuals for 100 generations of the genetic programming algorithm produced the model in Table 7.11, with a fitness of 0.750 and 56 hits out of a possible maximum of 58 (again any model result within a deviation of 0.25 from the experimental result was considered a hit, with a fitness contribution of 0). Output of the model is compared with experimental data in Appendix 2, Table 4.

For I-grafts:
$s2 = S2 * S1$
$r2 = R2 + 0.125$
$rs = r2 * R1 * R5$
$totalrs = S3 * S4 * (rs + 1.622 * s2 * S5)$
$si = totalrs$
For Y-grafts:
$c2 = C2 * C1$
$r2 = R2 + 0.250$
$totalrsc = C3 * C4 * (rs + 1.622 * c2 * C5)$
$ci = totalrsc$
(equations for $s2$, rs, $totalrs$, and si are as for I-grafts).

Table 7.11. Branching Network Experiment 5: Model found by genetic programming.

Discussion

Finally we have found a rather simple model satisfying constraints imposed to reflect intuitions regarding biological realism. In contrast to the original model, no inhibition is used in this model. As in the original model, root-shoot signals are used, but no signalling from shoot to root takes place. Therefore, it appears that inhibition and root-shoot signalling are not necessary to explain the experimental data on I-grafts and Y-grafts.

Branching Network Experiment 6

Method

In this experiment the two root data were taken into account, adding 16 branching inhibition data to be considered, bringing the total number to 74. Functional relationships between corresponding variables were constrained to be the same in scion and cotyledonary shoot, and in the two roots, if present. Building blocks were the same as in Experiment 3. The same fitness measure was used as in Experiments 4 and 5.

Results

A run with 10000 individuals for 100 generations of the genetic programming algorithm produced (after only 26 generations) the model in Table 7.12, with a fitness of 0.000 and 74 hits out of a possible maximum of 74 (again model results within a deviation of 0.25 from experimental results were considered hits, with 0 fitness contribution). Output of the model is compared with experimental data in Appendix 2, Table 5.

Discussion

Although the found model seems ideal in the sense of excellent fit to observed data, it is rather ugly, as genes or gene products can appear any number of times in each of the equations, producing sometimes very convoluted and non-intuitive expressions.

For I-grafts:
$s2 = S2 * S4$
$r2 = R2 + Inh(R4 + R1)$
$rs = R1 * R5 * (r2 + R1) * r2^2$
$totalrs = 8*(S3+S4)*(S4+S5)*(S4+s2+2*S3) * S1^2*S3^4*S4^3*S5^4*s2^4 + 0.226*rs$
$si = S3*S4*totalrs*(S1+s2*(2*s2+S1*totalrs))$
For Y-grafts:
$c2 = C2 * C4$
$r2 = R2 + 0.5 * Inh(R4 + R1) + 0.5 * Inh(R4 + R1)$
$totalrs = 8*(S3+S4)*(S4+S5)*(S4+s2+2*S3) * S1^2*S3^4*S4^3*S5^4*s2^4 + 0.5 * 0.226*rs$
$totalrsc = 8*(C3+C4)*(C4+C5)*(c2+C4+2*C3) * c2^4*C1^2*C3^4*C4^3*C5^4 + 0.5 * 0.226*rs$
$ci = C3*C4*totalrsc*(C1+c2*(2*c2+C1*totalrsc))$
For two root grafts:
$r2 = R2 + 0.5 * Inh(R4 + R1)$
$rr2 = RR2 + 0.5 * Inh(RR4 + RR1)$
$rrs = RR1 * RR5 * (rr2 + RR1) * rr2^2$
$totalrs = 8*(S3+S4)*(S4+S5)*(S4+s2+2*S3)*S1^2*S3^4 * S4^3*S5^4*s2^4 + 0.5 * 0.226*rs + 0.5 * 0.226*rrs$
(equations for $s2$, rs, and si for Y-grafts and for two root grafts are the same as for I-grafts).

Table 7.12. Branching Network Experiment 6: Model found by genetic programming.

Branching Network Experiment 7

Method

This experiment was similar to Experiment 6, except that each variable was constrained to appear only once in each of the equations, or parts of equations, to be discovered by the genetic programming algorithm. There were again 74 branching inhibition data to be considered in the optimisation process carried out by the algorithm. Building blocks were the same as in Experiment 3.

Results

Two runs with 10000 individuals for 100 generations of the genetic programming algorithm produced the models in Table 7.13 and Table 7.14, both with a fitness of 0.625 and 72 hits out of a possible maximum of 74 (again any model result within a deviation of 0.25 from the experimental result was considered a hit, with a fitness contribution of 0). Output of the models is compared with experimental data in Appendix 2, Tables 6 and 7.

Model 1 for I-grafts:
$s2 = S2 * S4 * S5$
$r2 = R2 + rs$
$rs = R1 * R5 * r2$
$totalrs = S1 * s2 + 0.750 * rs$
$si = S3 * S4 * totalrs$
Model 1 for Y-grafts:
$c2 = C2 * C4 * C5$
$r2 = R2 + 0.5 * rs + 0.5 * rs$
$totalrs = S1 * s2 + 0.5 * 0.750 * rs$
$totalrsc = C1 * c2 + 0.5 * 0.750 * rs$
$ci = C3 * C4 * totalrsc$
Model 1 for two root grafts:
$r2 = R2 + 0.5 * rs$
$rr2 = RR2 + 0.5 * rrs$
$rrs = RR1 * RR5 * rr2$
$totalrs = S1 * s2 + 0.5 * 0.750 * rs + 0.5 * 0.750 * rrs$
(equations for $s2$, rs, and si are the same for Y-grafts and two root grafts as for I-grafts).

Table 7.13. Branching Network Experiment 7: Model 1 found by genetic programming.

Model 2 for I-grafts:
$s2 = S2 * Inh(S5)$
$r2 = R2 + rs$
$rs = R1 * R5 * r2$
$totalrs = S1 * S5 * s2 + 0.750 * rs$
$si = S3 * S4 * totalrs$
Model 2 for Y-grafts:
$c2 = C2 * Inh(C5)$
$r2 = R2 + 0.5 * rs + 0.5 * rs$
$totalrs = S1 * S5 * s2 + 0.5 * 0.750 * rs$
$totalrsc = C1 * C5 * c2 + 0.5 * 0.750 * rs$
$ci = C3 * C4 * totalrsc$
Model 2 for two root grafts:
$r2 = R2 + 0.5 * rs$
$rr2 = RR2 + 0.5 * rrs$
$rrs = RR1 * RR5 * rr2$
$totalrs = S1 * S5 * s2 + 0.5 * 0.750 * rs + 0.5 * 0.750 * rrs$
(equations for $s2$, rs, and si for Y-grafts and for two root grafts are the same as for I-grafts).

Table 7.14. Branching Network Experiment 7: Model 2 found by genetic programming.

Discussion

The models discovered by genetic programming in this experiment are comparable to Harding's original model in complexity. Some features of the original model appear in the discovered models, but there are also some interesting differences. For example, in the original model branching inhibition si has an inhibitory effect on gene product $s2$ (and similarly ci on $c2$). In the models found here the feedback effect of branching inhibition is replaced by an effect of $S4$ and $S5$ ($C4$ and $C5$) in the first model, and an inhibitory effect of $S5$ ($C5$) in the second model. In the original model $S5$ also has an effect on $s2$, but this effect is indirect, through $totalrs$ and si. Further, the feedback from shoot to root appearing in the original model is lacking in the discovered models, as $s2$ (or $c2$) no longer has an effect on $r2$.

Thus, models are possible that approximately explain the observed data without assuming two feedback effects appearing in the original model, and by relying on direct effects of genes and their products. It would be interesting to compare predictions of both the original and the newly discovered models for not yet conducted biological experiments, and to carry out those experiments for which different models predict different branching inhibition values.

7.1.2 Lessons from Branching Network Experiments

The models found by the genetic programming algorithm are similar to, yet in several details different from those found by a human biologist. The genetic programming algorithm has no sense of biological realism, unless constraints are explicitly built in by the programmer. In Experiment 1, a model was found using usual arithmetic operations, without paying attention to physical or biological realism, for example using the subtraction operator. It mainly serves to illustrate the fact that it is often possible to find a mathematical or computational model producing results with a very good fit to observed data, without necessarily being realistic. This is a mistake that can also be made by human model builders, but they often circumvent the problem by 'having a feeling for what makes sense'. This amounts to imposing constraints, without making these constraints explicit. When using an automated discovery tool such as genetic programming, the researcher has to make all assumptions explicit.

On the positive side, a genetic programming algorithm is able to find any solution that fits the data, within the specified constraints. Thus, an advantage can be that solutions are found that do not make immediate intuitive sense, but may be good alternatives solutions to those found by a researcher, who may have a biased judgement with a priori assumptions about characteristics of a 'good' solution.

7.2 Flowering in Pea: Genes, Modules, and Phenotype

Flowering in pea is determined by genes and mobile signals, mediating the influence of, for example, photoperiod (Weller et al. 1997; Beveridge et al. 2003; Bell et al. 2003). Table 8 in Appendix 2, taken from (Bell et al. 2003), summarises experimental data on gene expression and flowering time of wild type and several mutant pea plants. It is assumed there is a linear relationship between time and plant growth, expressed as the number of nodes produced. When growth starts, vegetative nodes are produced before the first floral node is produced, so flowering time can be estimated by the number of the first floral node, or node of floral initiation (NFI). The first three rows of the table contain gene expression data. In the first column gene expression of genes Gigas (GI), Sterile Nodes (SN), and Late Flower (LF) is assigned a value 1 by definition. The other columns give expression values relative to the wild type for six mutants. The last two rows contain NFI values for each plant, for a photoperiod of 24 and 8 hours respectively.

In Arabidopsis it has been demonstrated that the circadian clock also plays a role in determining flowering time (Mouradov et al. 2002). Possible molecular mechanisms of circadian clocks have been elucidated in Drosophila (see Goldbeter 1996).

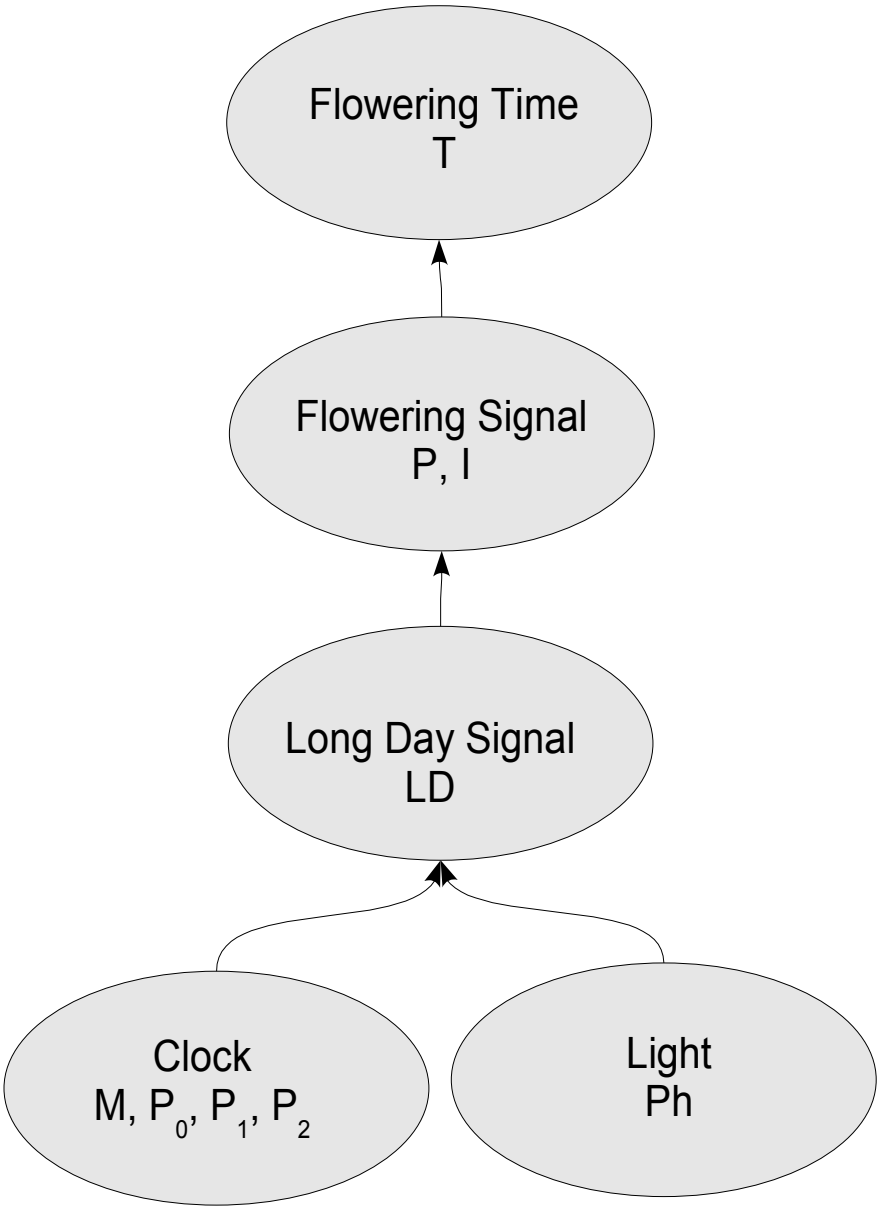


Figure 7.2. Modular model of flowering in pea.
Variables are as in sections 7.2.1 and 7.2.2.

It was hypothesised that a model describing the genetic and biochemical mechanisms explaining flowering time of pea could incorporate several modules, such as a circadian clock, the effect of photoperiodism, and a switching mechanism (see Figure 7.2). Each of these modules should function by a mechanism of chemical reactions. Thus, three levels of reality are involved:

the top level of the whole plant (phenotype);

a middle level of modules;

a bottom level of chemical reactions.

It was also hypothesised that, given qualitative assumptions about the overall structure of the model, details of models describing each level could be automatically discovered by genetic programming, starting from the next higher level. The discovered models should predict the experimental results in Appendix 2, Table 8.

7.2.1 Computational Experiments: Circadian Clock

In three experiments it was attempted to reconstruct a network of chemical reactions leading to a circadian clock, i.e. a time evolution of at least one of the chemical substances in the network with a periodicity of about 24 hours. Nodes of possible networks were concentrations of chemical substances, and connections were reactions between these substances. Possible reactions included activation, inhibition, synthesis,

transformation, and degradation. The genetic programming system was given as input variables the concentrations of substances involved, as well as the possible reaction functions. Using these inputs, the algorithm was run to find a network of chemical reactions producing a time evolution of one of the substances approximating a given sine function with a periodicity of 24 hours.

The circadian clock was assumed to be described by a model similar to the period protein (PER) model for Drosophila (Goldbeter 1996). The PER model consists of the following equations (for clarity, all concentration variables are written in uppercase and all parameters in lowercase):

$$\frac{dM}{dt} = v_s \frac{k_I^n}{k_I^n + P_N^n} - v_m \frac{M}{k_m + M}$$

$$\frac{dP_0}{dt} = k_s M - v_1 \frac{P_0}{k_1 + P_0} + v_2 \frac{P_1}{k_2 + P_1}$$

$$\frac{dP_1}{dt} = v_1 \frac{P_0}{k_1 + P_0} - v_2 \frac{P_1}{k_2 + P_1} - v_3 \frac{P_1}{k_3 + P_1} + v_4 \frac{P_2}{k_4 + P_2}$$

$$\frac{dP_2}{dt} = v_3 \frac{P_1}{k_3 + P_1} - v_4 \frac{P_2}{k_4 + P_2} - k_5 P_2 + k_6 P_N - v_d \frac{P_2}{k_d + P_2}$$

$$\frac{dP_N}{dt} = k_5 P_2 + k_6 P_N$$

where:

M is the concentration of messenger RNA;

P_0, P_1, and P_2 are concentrations of three proteins;

P_N is concentration of P_2 transported to the cell nucleus;

t is time;

and all other lowercase symbols are parameters of the model.

With appropriate values of parameters this model produces oscillations of M, P_0, P_1, P_2, and P_N with a period of about 24 hours.

The model was first formulated in a slightly different way to bring out the different kinds of chemical reactions, to be used as functions (building blocks) by the genetic programming algorithm. Reaction functions were defined as follows.

$$inhibition(v_s, k_I, P) = v_s \frac{k_I^n}{k_I^n + P^n}$$

$$synthesis(k_s, M) = k_s M$$

$$transformation(v_1, k_1, v_2, k_2, P_1, P_2) = -v_1 \frac{P_1}{k_1 + P_1} + v_2 \frac{P_2}{k_2 + P_2}$$

$$degradation(v_d, k_d, P) = v_d \frac{P}{K_d + P}$$

Using these reaction functions, the original PER model can be written as follows.

$$\frac{dM}{dt} = inhibition(v_s, K_I, P_N) - degradation(v_m, k_m, M)$$

$$\frac{dP_0}{dt} = synthesis(k_s, M) + transformation(v_1, k_1, v_2, k_2, P_0, P_1)$$

$$\frac{dP_1}{dt} = transformation(v_2, k_2, v_1, k_1, P_1, P_0) + transformation(v_3, k_3, v_4, k_4, P_1, P_2)$$

$$\frac{dP_2}{dt} = transformation(v_4, k_4, v_3, k_3, P_2, P_1) - k_5 P_2 + k_6 P_N - degradation(v_d, k_d, P_2)$$

$$\frac{dP_N}{dt} = k_5 P_2 + k_6 P_N$$

Inputs of the genetic programming algorithm were the reaction functions $inhibition$, $synthesis$, $transformation$, and $degradation$, variables M, P_0, P_1, and P_2, and all parameters except K_I, assumed to be a constant. No function was defined for transport of protein into the nucleus, as the transport process does not significantly affect the dynamics of the network, which can still produce a circadian periodic behaviour without it. Fitness was defined as the sum of deviations of the target protein concentration from a sine function with a period of 24 hours.

Circadian Clock Experiment 1

Method

In the first experiment the equations of the original PER model, without the transport equations, were given as input to the genetic programming system, which had to find optimal parameters to fit the time evolution of P_0 to the given sine function with 24 hour period. The algorithm only had to find parameters and basically worked like an evolution strategy.

Results

The following set of equations was obtained in a run of 1024 individuals during 51 generations (a total of 52224 individuals was evaluated).

$$\frac{dM}{dt} = inhibition(0.2477, P2) - degradation(0.4869, 2.415, M)$$

$$\frac{dP_0}{dt} = synthesis(0.7828, M) - transformation(2.632, 1.550, 1.360, 1.664, P_{0,} P_1)$$

$$\frac{dP_1}{dt} = transformation(2.632, 1.550, 1.360, 1.664, P_{0,} P_1) - transformation(4.962, 1.665, 1.602, 1.753, P_{1,} P_2)$$

$$\frac{dP_2}{dt} = transformation(4.962, 1.665, 1.602, 1.753, P_{1,} P_2) - degradation(0.943, 0.0845, P_2)$$

Fitness, or the total deviation of modelled P_0 concentration over 300 time steps, was 10.72, with 263 hits (modelled results practically equal to the objective function) out of a possible 300.

The time evolution of P_0 produced by these equations was thus very close to the sine objective function, as shown in Figure 7.3.

Discussion

Assuming the chemical reaction functions involved in the circadian clock to be known apart from their parameters, genetic programming found a set of parameters leading to a good clock model, closely approximating the objective periodic function, and in particular exhibiting a periodicity close to 24 hours. Given a set of equations for which it is known that periodic solutions exist for certain parameter combinations, it is a rather trivial task to find these combinations, and the genetic programming algorithm was indeed able to find a good solution with rather small numbers of individuals and generations. In the following experiments the assumption that the equations are already known apart from their parameters, is relaxed.

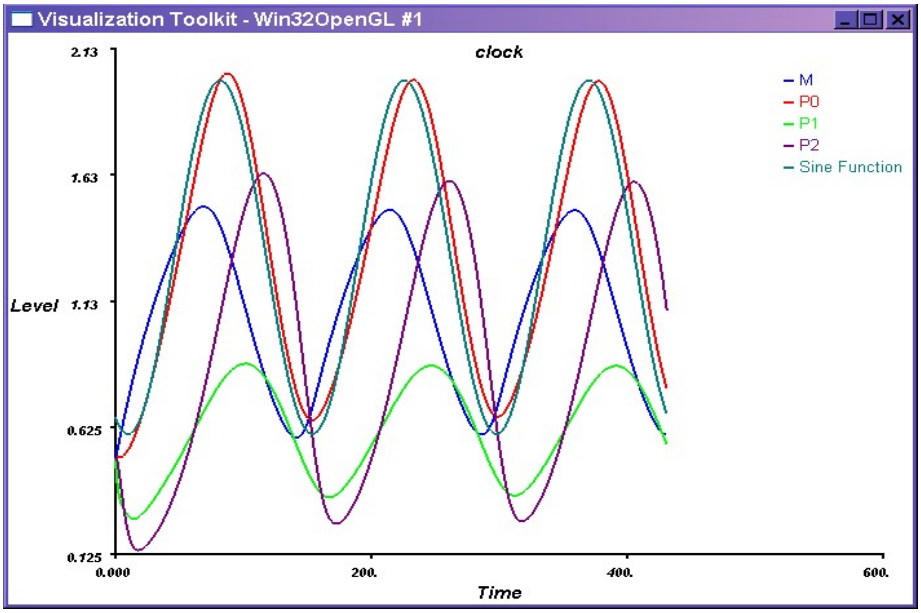


Figure 7.3. Circadian clock found in Experiment 1.
The two curves with the highest peaks represent the time evolution of P_0 and the sine function respectively.

Circadian Clock Experiment 2

Method

This experiment was similar to Experiment 1, apart from the space of possible models searched by the genetic programming system, which now could also rearrange the concentration variables and reaction functions.

Results

The following set of equations was obtained in a run of 1024 individuals during 1000 generations (a total of 1024000 individuals was evaluated).

$$\frac{dM}{dt} = inhibition(0.2166, P_2) - inhibition(0.6028, P_0)$$

$$\frac{dP_0}{dt} = inhibition(1.142, P_1) - transformation(2.245, 2.995, 0.2622, 1.046, P_{0,} P_1)$$

$$\frac{dP_1}{dt} = transformation(2.245, 2.995, 0.262, 1.046, P_{0,} P_1) - inhibition(1.280, M)$$

$$\frac{dP_2}{dt} = inhibition(1.280, M) - transformation(2.286, 2.866, 0.06645, 1.034, P_{2,} P_1)$$

Fitness, or the total deviation of modelled P_0 concentration over 500 time steps, was 28.27, with 171 hits (modelled results practically equal to the objective function) out of a possible 500. The time evolution of P_0 produced by these equations was again close to the sine objective function, as shown in Figure 7.4.

Discussion

When allowed to use different combinations of reactions functions, the set of reactions found by the genetic programming algorithm was rather different from the original PER model. Using only combinations of inhibition and transformation, and not synthesis or degradation, a time evolution of P_0 with a period of approximately 24 hours was obtained, with a good fit to the sine target function.

However, no constraints were imposed on possible concentration values, and, the genetic programming algorithm being oblivious to physical reality, it took advantage of this to come up with a solution exhibiting physically unrealistic negative concentration values for M. Also, transformation functions could be used without constraints, so, for example, P_1 can be transformed into P_2 without decreasing its own concentration, and transformation of P_1 into P_0 increases P_1. Suitable constraints should be introduced on the use of transformation functions in the equations to avoid such effects.

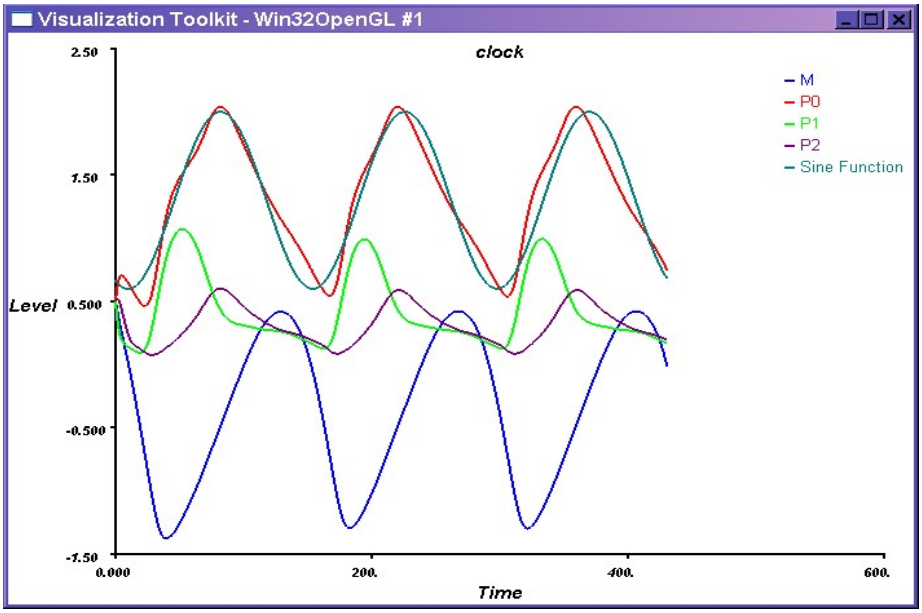


Figure 7.4. A reverse engineered circadian chemical clock.

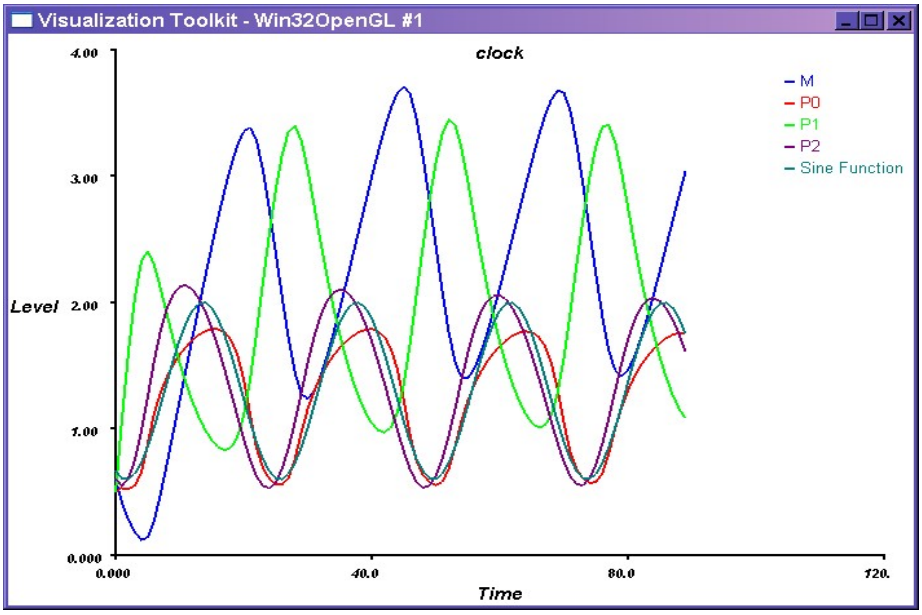


Figure 7.5. A reverse engineered circadian chemical clock with only positive concentrations.

Circadian Clock Experiment 3

Method

This experiment was similar to Experiment 2, apart from an extra constraint ensuring that solutions with negative concentrations of any of the substances in the model were discarded, by imposing a prohibitive fitness penalty on individuals representing such solutions in the genetic programming algorithm.

Results

The following set of equations was obtained in a run of 10000 individuals during 100 generations (a total of 1000000 individuals was evaluated).

$$\frac{dM}{dt} = transformation(3.611, 0.5842, 0.9187, 1.534, P_{0,}, M) - transformation(4.538, 2.823, 0.6177, 1.422, P_{0,}, P_0)$$

$$\frac{dP_0}{dt} = inhibition(0.7160, P_0) - inhibition(0.7279, P_2)$$

$$\frac{dP_1}{dt} = inhibition(0.7279, P_2) - synthesis(0.1301, P_1)$$

$$\frac{dP_2}{dt} = synthesis(0.1301, P_1) - degradation(0.4869, 1.9375, M)$$

Fitness, or the total deviation of modelled P_0 concentration over 300 time steps, was 33.26, with 157 hits (modelled results practically equal to the objective function) out of a possible 300. The time evolution of P_0 produced by these equations was again close to the sine objective function, as show in Figure 7.5.

Discussion

Without assuming how to use the possible reaction functions, a model with a good fit was found. Concentration values were constrained to be positive, and a solution was found with only positive values for all substances. However, the use of chemical reaction functions was still too little constrained, and the model allows strange phenomena to occur, for example transformation of a substance (P_0 in the first equation) into itself, and a substance (P_1 in the third equation) decreasing as a result of promoting its own synthesis.

To obtain realistic solutions, further constraints have to be introduced to exclude such phenomena. In Experiment 1 the equations were constrained to allow flexibility only in the choice of parameters. In experiments 2 and 3 there were not enough constraints to ensure realistic solutions. It is obvious that genetic programming can successfully reverse engineer a circadian chemical clock, but that the exact solution found depends on the flexibility allowed by imposed constraints.

7.2.2 Computational Experiments: Modules

Method

In this experiment it was assumed that flowering time of the plant was determined by a network of modules, schematically shown in Figure 7.2. Nodes of possible networks were modules, each consisting of a set of chemical reactions performing a well-defined function. The modules were connected by reactions between chemical substances representing output and input signals. One module's input could be activated or inhibited by another module's output. Possible modules included:

- a circadian clock;
- a light sensing module (in this case there were no explicitly modelled chemical reactions, but just an external light input; module output can be assumed to be a concentration of a light sensitive substance such as phytochrome);
- a long day signal module combining the clock and light signals to determine day length;
- a flowering signal module with flowering promotion and inhibition substances;
- a flowering switch module determining flowering time.

It was attempted to derive parameters of the connections between these modules, for which different types of activation and inhibition functions were used, by genetic programming. Activation and inhibition functions used were defined as follows.

$$activation\,(k_A, S) = k_A\left(1.0 - \frac{1.0}{1.0 + S^4}\right)$$

$$inhibition\,(k_I, S) = k_I \frac{1.0}{1.0 + S^4}$$

$$timedactivation\,(k_A, k_H, S, T) = k_A\left(1.0 - \frac{\left(\frac{k_H}{S}\right)^4}{\left(\frac{k_H}{S}\right)^4 + T^4}\right)$$

$$timedinhibition\,(k_I, k_H, S, T) = k_I \frac{\left(\frac{k_H}{S}\right)^4}{\left(\frac{k_H}{S}\right)^4 + T^4}$$

where:

k_A, k_I, and k_H are constants;

S is the concentration of an activating or inhibiting substance;

T is time.

Results

The following expressions for the signals between the different modules were obtained in a run of 100 individuals during 51 generations (a total of 5100 individuals was evaluated).

$$fromClock = activation(1.702, C)$$

$$fromLight = activation(1.736, Ph)$$

$$fromLongDay = activation(1.455, LD)$$

$$fromFloweringSignalP = timedactivation(0.7208, 804.4, P, T)$$

$$fromFloweringSignalI = timedinhibition(1.488, 947.9, I, T)$$

where:

C is the concentration of one of the substances of the chemical clock;

Ph is the concentration of a light-sensitive substance;

LD is the concentration of a long days indicating signal;

P is the concentration of a substance promoting flowering;

I is the concentration of a substance inhibiting flowering;

T is time.

These signals were used by the modules, in addition to an internally used degradation function defined as

$$degradation(k_D, k_M, S) = k_D \frac{S}{k_M + S}$$

where:

k_D and k_M are constants;

S is the concentration of the degrading substance.

Modules updated concentrations of substances as follows.

Long Day Signal module:

$$\begin{aligned} \frac{dLD}{dt} &= 0.76 * fromClock \\ &\quad * fromLight \\ &\quad - degradation(1.0, 0.5, LD) \end{aligned}$$

Flowering Signal module:

$$\begin{aligned} \frac{dP}{dt} &= fromLongDay \\ &\quad + inhibition(1.0, I) \\ &\quad - degradation(1.0, 0.5, P) \end{aligned}$$

Flowering Time module:

$$FS = fromFloweringSignalP * fromFloweringSignalI$$

where:

FS is the flowering signal determining flowering time.

Flowering time according to genotype and photoperiod predicted by this model was close to the experimental data, as shown in Appendix 2, Table 9.

Fitness, or the total deviation of modelled flowering time from observed data, was 25.82, with 9 hits (modelled results practically equal to the objective function) out of a possible 14.

Discussion

When the genetic programming system was given as inputs possible modules and their connecting activation and inhibition functions, it produced parameters for these functions leading to a model with a good fit to the experimental data on flowering time of different pea mutants in different photoperiod conditions.

In this section it has been shown how genetic programming can be used in a modular system at several levels. At the lowest level, possible mechanisms of a module can be inferred from the module's function. At a higher level, knowledge about the way modules fit together can be obtained from the behaviour of the whole system.

7.3 Conclusion

In conclusion, genetic programming can find models of genetic regulatory networks satisfying constraints specified by the programmer and optimising fit to an objective function or to observed data.

Any constraint could be imposed in principle, and it is important to justify constraints to be used by other arguments than purely empirical ones. Realistic constraints should be developed from scientific knowledge. Computer-assisted model discovery as used in the present work does not replace the difficult job of building an accurate scientific model. It simply makes the job easier by using a genetic programming algorithm to find appropriate solutions based on the imposed constraints. Arbitrarily defined constraints will only lead to meaningless results. Realistically defined constraints will possibly lead to usable results.

Thus, physical, chemical and biological realism has to be built in by specifying suitable constraints. Genetic programming is not a panacea for automatically finding solutions, but a tool for computer assisted discovery. It has the advantage that it forces a scientist to make assumptions explicit. Given explicit assumptions about the set of allowed solutions, incorporating physical, chemical and biological realism, as well as information from other sources, intuition, etc., genetic programming can search 'allowed solutions space' to find good solutions, given

the imposed constraints. This makes it a powerful and flexible tool to assist in scientific discovery.

The present chapter has dealt with the inverse problem of deriving micro-level models from macro-level data. Using a slightly different approach, one can specify constraints, form a forward system and see if it explains the data. If not, one can specify multiple forward models until a good one is found.

8 Conclusions and Future for Emergent Models in Scientific Discovery

In this thesis I have demonstrated that complex systems and emergence can be modelled and elucidated using computer simulation. I have done this by developing a general methodology, the *Emergent Models methodology*, making it possible to discover emergent models in computer simulations. The computer simulations used for this purpose were multi-agent simulations, combined with evolutionary algorithms, including genetic programming and evolution strategies, as model discovery tools.

I have used computational experiments to explore how, given emergent properties of higher-level agents, their behaviour and interactions can be derived from properties, behaviour, and interactions of lower-level agents. In these experiments I have used genetic programming algorithms to construct emergent models of higher-level agents' behaviour, described as interactions between these agents' properties, corresponding to given properties and behaviour of interacting lower-level agents.

In other computational experiments I have explored how behaviour of micro-level agents can be inferred from data on properties and behaviour of macro-level entities composed by these micro-level agents. In these experiments I have used

genetic programming algorithms to construct lower-level agents from higher-level emergent agents.

8.1 Overview of Results

We have seen in Chapter 2 that emergence is a fundamental characteristic of complex systems, and that much attention has been devoted in the philosophy of science to elucidate the concept of emergence. In the present thesis I have taken the concept of emergence to refer to the phenomenon observed in complex systems that properties and behaviour of macro-level entities depend on properties and behaviour of their composing macro-level entities, yet cannot be easily predicted from these.

Using the Emergent Models methodology I have developed, it is in principle possible to derive macro-level behaviour from micro-level properties and behaviour using a general search method such as genetic programming, even when mathematical derivation is impossible, and assuming that macro-level properties have been defined. It seems that *unpredictability in principle* of macro-level phenomena from micro-level phenomena, which is often claimed as a characteristic of emergent phenomena, is difficult if not impossible to establish, as general search methods can always exploit regularities in the relationship between micro-level and macro-level, at least to the extent that these regularities can be

captured by the modelling formalism used to describe behaviour.

As discussed in Chapter 3, the concepts of complex systems and of emergence have also been used, at least implicitly, in scientific theory and practice to study relationships between macroscopic and microscopic levels in reality, where macroscopic properties and behaviour emerge from microscopic properties and behaviour. In this thesis I have examined two widely used methods for studying these relationships. The first method was that of mathematical reasoning, with the advantage of generality of results, but the disadvantage of being limited to often highly simplified situations. The other method was that of computer simulation, enabling the study of less simplified models, but with doubtful generality of results. In particular I have examined examples of mathematical analysis of relationships between levels in mathematical physics and in mathematical ecology, and examples of computer simulation in ecology. I have thus demonstrated the desirability of a methodology combining the advantages of potential generality of results with the ability to study less simplified systems.

In Chapter 4 I have discussed design and implementation issues related to developing an Emergent Models simulation environment. A number of relevant issues could only be discussed briefly within the scope of the present work, and to implement a simulation environment at a realistic scale much more work needs to be done. For example, agent models have

to be combined with environment models. This will often involve combination of models of different types, e.g. a discrete Lindenmayer system and a numerical model of an environmental process. Methods to do this should be developed. As another example, visualisation of agents should be combined with visualisation of the environment. Some issues to be addressed include combination of surface and volume rendering, visualising the internal environment of agents and signals, visualising the environment (light, soil, air, water, ...). Simulation and visualisation of agents at multiple space and time scales (e.g. plant - leaves - parts of leaves – cells) also needs to be addressed.

Taking account of the findings in the previous chapters, I have formulated in Chapter 5 the Emergent Models methodology as a methodology of computer simulation of complex systems. It uses multi-agent systems for simulating complex systems. Further, entities in complex systems at different hierarchical levels are simulated by agents at different levels. Finally, genetic programming is used as a general method to discover models at the macro-level, given information about the micro-level, or models at the micro-level, given information about the macro-level. However, it should be stressed that the idea of emergent models and the Emergent Models methodology are not limited to the use of the particular technologies used in the present thesis. Multi-agent systems and genetic programming serve well to implement application of

the methodology, but these technologies can be usefully complemented or even substituted by other technologies.

I have illustrated the general applicability of the methodology in chapters 6 and 7, by studying some typical applications in ecological modelling and in genetic regulatory networks. The aim of these applications was to provide examples of applying the Emergent Models methodology to sufficiently different real world problems.

In the ecological modelling applications in Chapter 6 I have shown that, starting with a model of microscopic entities such droplets, insects and/or plants, it is possible to derive an emergent model of macroscopic entities such as a cloud of droplets, a population of insects and/or a patch of plants.

In the genetic network applications in Chapter 7, I have shown that, starting with macroscopic data on phenotypic results of genetic regulation, it is possible to derive microscopic models of genetic networks able to produce the observed data.

In both cases I have concluded that care has to be taken to obtain reasonable results, consistent with physical, chemical, and biological knowledge and intuition. This prior knowledge can be expressed as constraints on possible solutions. Specifying enough constraints will also ensure that unique solutions are produced by a genetic programming algorithm. Any system without a unique solution is considered unsolved and requires more work.

As a general conclusion, the Emergent Models methodology is not a substitute for reasoning and creativity in the scientific discovery process, but can be a valuable aid to assist in discovering relationships between levels, in the form of emergent macroscopic models consistent with microscopic models of the same complex system, or, inversely, microscopic models consistent with macroscopic models or observed macroscopic data.

8.2 Some Examples of Possible Applications

The Emergent Models methodology is a general methodology applicable to any complex system. It offers a way to discover relationships between levels in complex systems and can be applied in any situation where one is interested in such relationships. Potential applications are therefore very numerous and this section is only meant to give a rough idea of the spectrum of possibilities.

8.2.1 Molecular Dynamics: Atoms and Molecules

In molecular simulations of polymeric materials it is important to take into account different time and length scales in order to predict macroscopic properties such as viscosity from local interactions on the atomic level (Abrams et al. 2002, p. 144-145). Simulations including all atoms in a material have important complications. Quantum simulations are confined to

very small systems, so an empirical force field has to be used. But no single force field can be used at significantly different temperatures or compositions. In addition, simulations of all atoms would provide an enormous amount of data, almost all of which would be irrelevant for the questions under consideration (Abrams et al. 2002, p. 145). Therefore, mesoscopic models of macromolecules represent parts of the molecule as beads, connected together with strings (Abrams et al. 2002, p. 148-153). To relate such models to microscopic models, mapping schemes are needed for coarse-graining (Abrams et al. 2002, p. 153-158). Automatic coarse-graining schemes have been developed, assuming a certain structure of the mesoscopic model and using a simplex scheme to optimise its parameters (Abrams et al. 2002, p. 159-162).

The application of Emergent Models could be investigated as a general methodology to derive mesoscopic molecular models from microscopic all atom models, and to perform mesoscopic simulations to predict macroscopic properties of materials.

8.2.2 Medicine and Botany: Cells and Tissues

A multi-agent simulation model relating human cell properties and behaviour with tissue level properties and behaviour has been developed by (Walker et al. 2004) for cells in epithelial tissue. In this model individual epithelial cells are modelled as agents with a rule-based behaviour. Simple rules

are assigned to cells depending on their differentiation status and executed depending on the cell's internal state and external environment (Walker et al. 2004, p. 90). However, the authors state their intention to include modules representing the mechanisms behind these rules, such as intracellular signalling pathways involved in cell-cycle control and expression of genes regulating these pathways (Walker et al. 2004, p. 90-91).

Computational cell level microscopic models have also been developed for other organisms, such as plants, to predict macroscopic plant tissue properties and behaviour (see e.g. Hogeweg 2002; Rudge 2003).

Emergent Models could be useful for extending this kind of models of cells, tissues, and organs, for example by providing a method to derive cell behaviour rules from intracellular mechanisms. Alternatively, if the cell behaviour is known, an multi-agent model of intracellular processes could be discovered.

8.2.3 Biological Psychology: Neural Networks and Brain

The brain of vertebrate animals can be described as consisting of functional modules or resources, each specialised for performing a certain task (see e.g. Minsky 1986; Gardner 1993). For example, there are sensors to recognise situations, modules with knowledge of ways to react to them, and motors to execute actions (Minsky 2004, §1-4). There are resources for

making comparisons, for formulating goals or plans, for containing knowledge that other resources can use, and for turning other resources off or on (Minsky 2004, §1-6).

The brain and its resources can naturally be modelled as agents. At the highest level we have the brain itself, observing the world through its sensory modules, formulating its goals and working out how to achieve them, and acting on the world through its motor modules. Each of these modules in turn can be described as an agent with its specific task and goals to achieve, observing its world, including the other agents, and acting on it. A module is composed of still lower level component modules, and in principle this decomposition in lower level modules could be continued down to the level of individual neurones.

A neural network is composed of units with inhibitory and excitatory interactions. As an example, O'Reilly & Munakata model object recognition by the visual form pathway in the brain using a hierarchical neural network building spatially invariant representations (O'Reilly & Munakata, 2000, p. 241-257). By performing successive transformations across layers in the network, shapes of objects can be recognised independently from their position in the visual input space and from their sizes. Each layer builds on the outputs of the underlying layer, and the model learns a hierarchical series of transformations producing increasingly more complex and spatially invariant representations.

Applying emergent modelling to such a network, we would regard the units of the neural network as agents with a behaviour at the micro-level of individual neurones and at various intermediate levels. Emergent Models could be discovered for the agents composing each intermediate level. The macro-level behaviour consists of a description of, for example, spatially invariant object recognition.

8.3 The Future

Emergent Models is a general methodology for studying emergent properties and behaviour in complex systems in a systematic way. Possible applications can be imagined in any area of science and engineering where complex systems are relevant. In this thesis I have only provided illustrative examples of systems that can be studied and the kind of results that can be obtained. The full potential of the methodology is to be realised in large scale simulations of complex systems.

To realise the full potential of the Emergent Models methodology, it also needs to be further developed. Within the scope of the present work it has been assumed that emergent macro-level properties of the studied systems were already defined in some way, and models of emergent behaviour have been discovered using these predefined emergent properties. It would be desirable to develop a systematic method for discovering emergent properties of a system from first

principles, without having recourse to only rather vague notions about intuition and cognitive processes, as is presently the case. Developing such a method is an important topic for a future research programme.

The models studied in the present work have also deliberately been kept simple, in order to demonstrate the possibility of the approach. In more elaborate simulations emergent models to be discovered would not be limited to just simple functions representing an aspect of macro-level agent behaviour, as has been the case in the examples studied in Chapter 6. An agent is essentially defined as a computer program consisting of building blocks, so the methodology can equally well be applied to these building blocks to construct complete macro-level agents. The same is true with regard to the inverse problem of discovering micro-level agents from macro-level descriptions studied in Chapter 7.

Proposals for such more elaborate applications of the Emergent Models methodology have at present been formulated in the area of emergency response systems. One of these proposals has already led to a study of the possible contribution of complex systems approaches and in particular the Emergent Models methodology to flood event management in the United Kingdom.

Bibliography

Abbott C.A., Berry M.W., E.J., Gross L.J. & Luh H.-K. 1997. Parallel Individual-Based Modeling of Everglades Deer Ecology. *Computational Science & Engineering* 4 (4), p. 60-72, IEEE Computer Society, Los Alamitos, CA, USA.

Abrams C., Delle Site L. & Kremer K. 2002. Multiscale Computer Simulations for Polymeric Materials in Bulk and near Surfaces. *Lecture Notes in Physics* 605, p. 143-164, Springer, Berlin, Germany.

Adami C. 1998. *Introduction to Artificial Life.* Springer, New York, NY, USA.

Akers R.L., Kant E., Randall C.J., Steinberg S. & Young R.L. 1997. SciNapse: A Problem-Solving Environment for Partial Differential Equations*. Computational Science & Engineering* 4 (3), p. 32-42, IEEE Computer Society, Los Alamitos, CA, USA.

Aklecha V. 1999. *Object-Oriented Frameworks Using C++ and CORBA.* The Coriolis Group, Scottsdale, AZ, USA.

Allen T.F.H. & Hoekstra T.W. 1992. *Toward a Unified Ecology*. Columbia University Press, New York, NY, USA.

Baas N.A. 1994. Emergence, Hierarchies, and Hyperstructures. In Langton C.G. (ed), *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity* XVII, p. 515-537, Addison-Wesley, Redwood City, CA, USA.

Baas N.A. 1997. Self-organization and Higher Order Structures. In Schweitzer F. (ed), *Self-organization of Complex Structures: From Individual to Collective Dynamics*, p. 71-82, Gordon and Breach, New York, NY, USA.

Bäck T. 2000. Introduction to Evolutionary Algorithms. In Bäck T., Fogel D.B. & Michalewicz Z. (eds), *Evolutionary Computation 1: Basic Algorithms and Operators*, p. 59-63, Institute of Physics Publishing, Bristol, UK.

Bäck T., Fogel D.B. & Michalewicz Z. (eds) 2000. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK.

Bar-Yam Y. 1997. *Dynamics of Complex Systems*. Perseus Books, Reading, MA, USA.

Batterman R.W. 2001. *The Devil in the Details: Asymptotic Reasoning in Explanation, Reduction, and Emergence*. Oxford University Press, Oxford, UK.

Bedau M.A. 1997. Weak Emergence. In Tomberlin J. (ed), *Philosophical Perspectives* 11*, (Mind, Causation, and World)*, p. 375-399, Blackwell, Malden, MA, USA.

Beer R.D., Ritzmann R.E. & McKenna T. (eds) 1993. *Biological Neural Networks in Invertebrate Neuroethology and Robotics.* Academic Press, San Diego, CA, USA.

Bell P., Parmenter K., Beveridge C. & Hanan J. 2003. *Hypothesis Driven Mathematical Modelling of the Flowering Network in Pea*. Project Report, University of Queensland, Brisbane, Australia.

Beveridge C.A., Weller J.L., Singer S.R. & Hofer J.M.I. 2003. Axillary Meristem Development: Budding Relationships between Networks controlling Flowering, Branching, and Photoperiod Responsiveness. *Plant Physiology* 131, p. 927-934.

Bonabeau E., Dessalles J.-L. & Grumbach A. 1995a. Characterizing Emergent Phenomena (1): A Critical Review. *Revue Internationale de Systémique* 9, p. 327-346.

Bonabeau E., Dessalles J.-L. & Grumbach A. 1995b. Characterizing Emergent Phenomena (2): A Conceptual Framework. *Revue Internationale de Systémique* 9, p. 347-371.

Booch G., Rumbaugh J. & Jacobson I. 1999. *The Unified Modeling Language User Guide.* Addison-Wesley, Reading, MA, USA.

Bower J.M. & Bolouri H. (eds) 2001. *Computational Modeling of Genetic and Biochemical Networks.* MIT Press, Cambridge, MA, USA.

Breton L., Zucker J.-D. & Clément E. 2001. A Multi-Agent Based Simulation of Sand Piles in a Static Equilibrium. *Lecture Notes in Artificial Intelligence* 1979, p. 108-118, Springer, Berlin, Germany.

Buschmann F., Meunier R., Rohnert H., Sommerlad P. & Stal M. 1996. *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons, Chichester, UK.

Cariani P. 1989. *On the Design of Devices with Emergent Semantic Functions*. PhD Dissertation, State University of New York, Binghamton, NY, USA.

Cariani P. 1991. Emergence and Artificial Life. In Langton C.G., Taylor C., Farmer J.D. & Rasmussen S. (eds), *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity* X, p. 775-797, Addison-Wesley, Redwood City, CA, USA.

CERN 2003. *Particles, Fields and the Big Bang*. European Organization for Nuclear Research. Retrieved 6 October 2005, from http://public.web.cern.ch/Public/Objects/Chapters/Education/OnlineResources/CERN_Pos3.pdf.

Chelle M. & Andrieu B. 1998. The Nested Radiosity Model for the Distribution of Light within Plant Canopies. *Ecological Modelling* 111, Elsevier, Amsterdam, The Netherlands.

Chelle M. 1997. *Développement d'un modèle de radiosité mixte pour simuler la distribution du rayonnement dans les couverts végétaux.* PhD Thesis, Université de Rennes I, Rennes, France.

Ciancarini, P., and Wooldridge, M.J. (eds) 2001. *Agent-Oriented Software Engineering*, Lecture Notes in Computer Science 1957, Springer, Berlin, Germany.

Cleary A., Kohn S., Smith S.G. & Smolinski B. 1999. Language Interoperability Mechanisms for High-Performance Scientific Applications. In *Object oriented methods for interoperable scientific and engineering computing*, Society for Industrial and Applied Mathematics, Philadelphia, MA, USA.

Cohen M.F. & Wallace J.R. (with Hanrahan P.) 1993. *Radiosity and Realistic Image Synthesis.* Academic Press, London, UK.

Cunningham B. 2001. The Reemergence of 'Emergence'. *Philosophy of Science* 68 (3), p. 62-69.

Davidsson P. 2001. Multi Agent Based Simulation: Beyond Social Simulation. *Lecture Notes in Artificial Intelligence* 1979, p. 97-107, Springer, Berlin, Germany.

DeAngelis D.L. & Gross L.J. (eds) 1992. *Individual-Based Models and Approaches in Ecology: Populations, Communities and Ecosystems.* Chapmann & Hall, New York, NY, USA.

De Jong H. 2002. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology* 9 (1), p. 67-103.

Deutschman D.H., Levin S.A., Devine C. &, Buttel L.A. 1997. Scaling from Trees to Forests: Analysis of a Complex Simulation Model. *Science Online* 277 (5332), p. 1684, American Association for the Advancement of Science.

Di Cola G., Gilioli G. & Baumgärtner J. 1999. Mathematical Models for Age-structured Population Dynamics. In Huffaker C.B. & Gutierrez A.P. (eds), *Ecological Entomology*, 2nd edn, p. 503-534, John Wiley & Sons, New York, NY, USA.

D'Inverno M., Luck M., Fisher M. & Preist, C. 2002. *Foundations and Applications of Multi-Agent Systems.* Lecture Notes in Artificial Intelligence 2403, Springer, Berlin, Germany.

Ehleringer J.R. & Field C.B. (eds) 1993. *Scaling Physiological Processes.* Academic Press, San Diego, CA, USA.

Emmeche C. 1994. *The Garden in the Machine: The Emerging Science of Artificial Life.* Princeton University Press, Princeton, NJ, USA.

Emmerich W. 2000. *Engineering Distributed Objects.* John Wiley & Sons, Chichester, UK.

Eriksson H.-E. & Magnus Penker M. 1998. *UML Toolkit.* John Wiley & Sons, New York, NY, USA.

Ferber J. 1999. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Harlow, UK.

Ferber J., Gutknecht O. & Michel F. 2003. *MadKit Development Guide*, Version 3.1. Retrieved 2 January 2005, from http://www.madkit.org/madkit/doc/devguide/devguide.html.

Ferree P.M., Frydman H.M., Li J.M., Cao J., Wieschaus E. & Sullivan W. 2005. Wolbachia Utilizes Host Microtubules and Dynein for Anterior Localization in the Drosophila Oocyte. *PLoS Pathogens* 1(2). Retrieved 19 November 2005, from http://pathogens.plosjournals.org.

Fogel D.B. 2000. Introduction to Evolutionary Computation. In Bäck T., Fogel D.B. & Michalewicz Z. (eds), *Evolutionary Computation 1: Basic Algorithms and Operators*, p. 1-3, Institute of Physics Publishing, Bristol, UK.

Forster M.R. & Kryukov A. 2003. The Emergence of the Macro-World: A Study of Intertheory Relations in Classical and Quantum Mechanics. *Philosophy of Science* 70, p. 1039-1051.

Fournier C. & Andrieu B 1999. ADEL-Maize: An L-System Based Model for the Integration of Growth Processes from the Organ to the Canopy: Application to Regulation of Morphogenesis by Light Availability. *Agronomie* 19, p.313-327.

Friedlander K.F. 2000. *Smoke, Dust, and Haze - Fundamentals of Aerosol Dynamics,* 2nd edn. Oxford University Press, New York, NY, Oxford, UK.

Gardner H. 1993. *Frames of Mind*, 2nd edn. Fontana Press, London, UK.

Gimblett H.R. 2002. *Integrating Geographic Information Systems and Agent-based Modeling Techniques.* Oxford University Press, Oxford, UK.

Goldbeter A. 1996. *Biochemical Oscillations and Cellular Rhythms – The molecular bases of periodic and chaotic behaviour*. Cambridge University Press, Cambridge, UK.

Greengard L.F. 1987. *The Rapid Evaluation of Potential Fields in Particle Systems.* PhD Thesis, Yale University, New Haven, CT, USA.

Grünbaum D. 1998. Using Spatially Explicit Models to Characterize Foraging Performance in Heterogeneous Landscapes. *The American Naturalist* 151, p. 97-115.

Hanan, J., Room, P., Geitz, G. & Battaglia, R. 2000. Pesticide Targeting: Simulating Effects of Plant Architecture on Pesticide Deposition. In *Insect Pest Management in Sweet Corn*, (Proceedings of Workshop no. 4), p. 63-65, Queensland Department of Primary Industries, Brisbane, Australia.

Hand D., Mannila H. & Smyth P. 2001. *Principles of Data Mining.* The MIT Press, Cambridge, MA, USA.

Harding E.A. 2003. *Computational Analysis and Molecular Physiology of the Branching Regulatory Network in Pea*. Bsc Honours Thesis, University of Queensland, Brisbane, Australia.

Hibbard B. 2002. Building 3D User Interface Components Using a Visualization Library, *Computer Graphics* 36 (1), p. 4-7, Association for Computing Machinery, New York, NY, USA.

Hogeweg P. 2002. Computing an Organism: On the Interface between Informatic and Dynamic Processes. *Biosystems* 64, p. 97-109.

Holland J.H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Holland J.H. 1998. *Emergence: From Chaos to Order*. Oxford University Press, Oxford, UK.

Houstis E., Gallopoulos E., Bramley R. & Rice J. 1997. Problem-Solving Environments for Computational Science. *Computational Science & Engineering* 4 (3), p. 18-21, IEEE Computer Society, Los Alamitos, CA, USA.

Huffaker C.B. & Gutierrez A.P. (eds) 1999. *Ecological Entomology*, 2nd edn. John Wiley & Sons, New York, NY, USA.

Kadanoff L.P. 1999. *From Order to Chaos II*. World Scientific, Singapore, Singapore.

Kareiva P. & Odell G. 1987. Swarms of Predators Exhibit "Preytaxis" if Individual Predators Use Area-restricted Search. *The American Naturalist* 130 (2), p. 233-270.

Kauffman S.A. 1993. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, NY, USA.

Keller E.F. & Segel L.A. 1971. Model for Chemotaxis. *Journal of Theoretical Biology* 30, p. 225-234.

Kendrick R.E. & Kronenberg G.H.M. (ed) 1994. *Photomorphogenesis in Plants*, 2nd edn. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kim J. 1999. Making Sense of Emergence. *Philosophical Studies* 95 (1-2), p. 3-36.

Knox R.G., Kalb V.L., Levine E.R. & Kendig D.J. 1997. A Problem-Solving Workbench for Interactive Simulation of Ecosystems. *Computational Science and Engineering* 4 (3), p. 52-60, IEEE Computer Society, Los Alamitos, CA, USA.

Koza J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Koza J.R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, USA.

Koza J.R., Bennet F.H. III, Andre D. & Keane M.A. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann, San Francisco, CA, USA.

Koza J.R., Keane M.A., Streeter M.J., Mydlowec W., Yu J. & Lanza G. 2003. *Genetic Programming IV : Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Boston, MA, USA.

Koza J.R., Mydlowec W., Lanza G., Yu J. & Keane M.A. 2000. *Reverse Engineering and Automatic Synthesis of Metabolic Pathways from Observed Data using Genetic Programming*. Stanford Technical Report SMI-2000-0851, Stanford, CA, USA.

Kubik A. 2003. Toward a Formalization of Emergence. *Artificial Life* 9 (1), p. 41-65.

Kumar V. (ed) 2002. *Biological Rhythms*. Narosa Publishing House, New Delhi, India.

Kvasnička V. & Pospíchal J. 2002. Emergence of Modularity in Genotype-Phenotype Mappings. *Artificial Life* 8 (4), p.295-310.

Lam L. (ed) 1997. *Introduction to Nonlinear Physics.* Springer, New York, NY, USA.

Lanner R.M. 1996. *Made for Each Other, A Symbiosis of Birds and Pines.* Oxford University Press, Oxford, UK.

Lavender R.G. & Schmidt D.C. 1999. Active Object: An Object Behavioral Pattern for Concurrent Programming. In *Proceedings of the Second Pattern Languages of Programs Conference*, Monicello, Illinois, Sep. 6-8, retrieved 2 January 2005, from http://www.cs.wustl.edu/~schmidt/PDF/.

Lloyd Morgan C. 1927. *Emergent Evolution*. Williams & Norgate, London, UK.

Luke S. 2002. *ECJ: An Evolutionary Computation and Genetic Programming System*. Retrieved 2 January 2005, from http://cs.gmu.edu/~eclab/projects/ecj/docs/.

Lumsden P.J. & Millar A.J. (ed) 1998. *Biological Rhythms and Photoperiodism in Plants*. Bios Scientific Publishers, Oxford, UK.

Markus S., Weerawarana S., Houstis E.N. & Rice J.R. 1997. Scientific Computing via the Web: The Net Pellpack PSE Server. *Computational Science & Engineering* 4 (3), p. 43-51, IEEE Computer Society, Los Alamitos, CA, USA.

Mech et al. 1998. *CPFG User's Manual.* Included in L-Studio software distribution. University of Regina, Canada.

Mech R. & Prusinkiewicz P. 1996. Visual Models of Plants Interacting with their Environment. In Proceedings of SIGGRAPH 96, New Orleans, LA, August 4-9, *Computer Graphics* Proceedings, Annual Conference Series, p. 397-410, ACM SIGGRAPH.

Michalewicz M.T. (ed) 1997. *Plants to Ecosystems.* CSIRO Publishing, Collingwood, VIC, Australia.

Mill J.S. 1843. *A System of Logic: Ratiocinative and Inductive*. Longmans Green, London, UK.

Miller J.R. & Miller T.A. (eds) 1986. *Insect-Plant Interactions.* Springer, New York, NY, USA.

Minsky M.L. 1986. *The Society of Mind*. Simon and Schuster, New York, NY, USA.

Minsky M. 2004. *The Emotion Machine.* Retrieved 11 January 2005, from http://web.media.mit.edu/~minsky/.

Mitchell S.D. 2003. *Biological Complexity and Integrative Pluralism*. Cambridge University Press, Cambridge, UK.

Mouradov A., Cremer F. & Coupland G. 2002. Control of Flowering Time: Interacting Pathways as a Basis for Diversity. *The Plant Cell*, supplement 2002, p. S111-S130, American Society of Plant Biologists.

Okubo A. & Levin S.A. 2001. *Diffusion and Ecological Problems: Modern Perspectives*, 2nd edn. Springer, New York, NY, USA.

Okubo A. & Grünbaum D. 2001. Mathematical Treatment of Biological Diffusion. In Okubo A. & Levin S.A., *Diffusion and Ecological Problems: Modern Perspectives*, 2nd edn, p. 127-169, Springer, New York, NY, USA.

O'Neill R.V. et al. 1986. *A Hierarchical Concept of Ecosystems*. Princeton University Press, Princeton, NJ, USA.

Oosawa F. & Nakaoka Y. 1977. Behavior of Micro-Organisms as Particles with Internal State Variables. *Journal of Theoretical Biology* 66, p. 747-761.

O'Reilly R.C. & Munakata Y. 2000. *Computational Explorations in Cognitive Neuroscience*. MIT Press, Cambridge, MA, USA.

ORNL 2005. *Genomics Image Gallery*. U.S. Department of Energy Human Genome Program, Oak Ridge National Laboratory. Retrieved 6 October 2005, from http://www.ornl.gov/sci/techresources/Human_Genome/graphics/slides/images1.shtml

Palmer 1992. Hierarchical and Concurrent Individual-Based Modeling. In DeAngelis D.L. & Gross L.J. (eds), *Individual-Based Models and Approaches in Ecology: Populations, Communities and Ecosystems*, p. 188-212, Chapmann & Hall, New York, NY, USA.

Parker S.G., Christopher R. Johnson C.R. & David Beazley D. 1997. Computational Steering Software Systems and Strategies. *Computational Science & Engineering* 4 (4), p. 50-59, IEEE Computer Society, Los Alamitos, CA, USA.

Patlak C.S. 1953. Random Walk with Persistence and External Bias. *Bulletin of Mathematical Biophysics* 15, p. 311-338.

Pavé A. 1994. *Modélisation en biologie et en écologie*. Aléas, Lyon, France.

Payne T.L., Birch M.C. & Kennedy C.E.J. (eds) 1986. *Mechanisms in Insect Olfaction.* Clarendon Press, Oxford, UK.

Picot J.J.C. & Kristmanson D.D. 1997. *Forestry Pesticide Aerial Spraying.* Kluwer, Dordrecht, The Netherlands.

Prusinkiewicz P. & Hanan J. 1989. *Lindenmayer Systems, Fractals, and Plants.* Lecture Notes in Biomathematics 79, Springer, New York, NY, USA.

Prusinkiewicz P., Hanan J. & Mech R. 2000. An L-System-Based Plant Modeling Language. In *Applications of Graph Transformations with Industrial Relevance*: International Workshop, Springer, Berlin, Germany.

Prusinkiewicz P. & Lindenmayer A., with Hanan J.S. et al. 1990. *The Algorithmic Beauty of Plants.* Springer, New York, NY, USA.

Ptashne M. & Gann A. 2002. *Genes and Signals*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, USA.

Rasmussen S., Baas N.A., Mayer B., Nilsson M. & Olesen M.W. 2001. Ansatz for Dynamical Hierarchies. *Artificial Life* 7 (4), p.329-353.

Reynolds J.F., Hilbert D.W. & Kemp P.R. 1993. Scaling Ecophysiology from the Plant to the Ecosystem: A Conceptual Framework. In Ehleringer J.R. & Field C.B. (eds), *Scaling Physiological Processes,* Academic Press, San Diego, CA, USA.

Robinson S., Nance R.E., Paul R.J., Pidd M. & Taylor S.J.E. 2004. Simulation Model Reuse: Definitions, Benefits and Obstacles. *Simulation Modelling Practice and Theory* 12 (7-8) , p. 479-494.

Rudge T. 2003. *A Computational System for Modelling Cellular Morphogenesis in Plants*. Unpublished manuscript, University of Queensland, Brisbane, Australia.

Rumbaugh J., Jacobson I. & Booch G. 1999. *The Unified Modeling Language Reference Manual.* Addison-Wesley, Reading, MA, USA.

Salthe S.N. 1985. *Evolving Hierarchical Systems: Their Structure and Representation*. Columbia University Press, New York, NY, USA.

Samet H. 1990a, *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, USA.

Samet H. 1990b, *Applications of Spatial Data Structures*. Addison-Wesley, Reading, MA, USA.

Schmidt D., Stal M., Rohnert H. & Buschmann F. 2000. *Pattern-Oriented Software Architecture, vol. 2: Patterns for Concurrent and Networked Objects.* John Wiley & Sons, Chichester, UK.

Schowalter T.D. 2000. *Insect Ecology, An Ecosystem Approach.* Academic Press, San Diego, CA, USA.

Schroeder W., Martin K. & Lorensen B. 1998. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics,* 2nd edn. Prentice Hall, Upper Saddle River, NJ, USA.

Schroeder W.J., Avila L.S. & Hoffman W. 2000. Visualizing with VTK: A Tutorial. *IEEE Computer Graphics and Applications* 20 (5), p. 20-27.

Schwefel, H.-P. 1995. *Evolution and Optimum Seeking.* John Wiley & Sons, New York, NY, USA.

Servat D., Perrier E., Treuil J.P. & Drogoul A. 1998a. Towards Virtual Experiment Laboratories: How Multi-Agent Simulations Can Cope with Multiple Scales of Analysis and Viewpoints*. Lecture Notes in Artificial Intelligence* 1434, p. 205-217, Springer, Berlin, Germany.

Servat D., Perrier E., Treuil J.P. & Drogoul A. 1998b. When Agents Emerge from Agents: Introducing Multi-Scale Viewpoints in Multi-Agent Simulations. *Lecture Notes in Artificial Intelligence* 1534, p. 183-198, Springer, Berlin, Germany.

Shorey H.H. & McKelvey Jr J.J. (eds) 1977. *Chemical Control of Insect Behavior.* John Wiley & Sons, New York, NY, USA.

Siegel J. 2000. *CORBA 3 Fundamentals and Programming.* John Wiley & Sons, New York, NY, USA.

Silberstein 2002. Reduction, Emergence and Explanation. In *The Blackwell Guide to the Philosophy of Science*, p. 80-107, Blackwell Publishers, Malden, MA, USA.

Sirignano W.A. 1999. *Fluid Dynamics and Transport of Droplets and Sprays.* Cambridge University Press, Cambridge, UK.

Skellam J.G. 1972. Some Philosophical Aspects of Mathematical Modelling in Empirical Science with Special Reference to Ecology. In Jeffers J.N.R. (ed), *Mathematical Models in Ecology*, p. 13-28, Blackwell, London, UK.

Skellam 1973. The Formulation and Interpretation of Mathematical Models of Diffusionary Processes in Population Biology. In Bartlett M.S. & Hiorns R.W. (eds), *The Mathematical Theory of the Dynamics of Biological Populations*, p. 63-85, Academic Press, New York, NY, USA.

Spirtes P., Glymour C. & Scheines R. 2000. *Causation, Prediction, and Search.* The MIT Press, Cambridge, MA, USA.

Stolk H.J. 1992. *Parameter Optimization of Control Systems using Learning Algorithms.* MSc Thesis (in Dutch), Free University of Brussels, Brussels, Belgium.

Tari Z. & Bukhres O. 2001. *Fundamentals of Distributed Object Systems.* John Wiley & Sons, New York, NY, USA.

Ter Beek M.H. 1999. Simple Eco-Grammar Systems with Prescribed Teams. In Pawn G.L. & Salomaa A. (eds), *Grammatical Models of Multi-Agent Systems*, Gordon & Breach, London, UK.

Thomas B. & Vince-Prue D. 1997. *Photoperiodism in Plants*. Academic Press, San Diego, CA, USA.

Thompson J.M.T. & Stewart H.B. 1986. *Nonlinear Dynamics and Chaos – Geometrical Methods for Engineers and Scientists*. John Wiley & Sons, Chichester, UK.

Turchin P. 1991. Translating Foraging Movements in Heterogeneous Environments into the Spatial Distribution of Foragers. *Ecology* 72 (4), p. 1253-1266.

Turchin P. 1998. *Quantitative Analysis of Movement: Measuring and Modeling Population Redistribution in Animals and Plants*. Sinauer Associates, Sunderland, MA, USA.

Turchin P. & Omland K.S. 1999. Quantitative Analysis of Insect Movement. In Huffaker C.B. & Gutierrez A.P. (eds), *Ecological Entomology*, 2nd edn, p. 463-502, John Wiley & Sons, New York, NY, USA.

Unwin D.M., & Corbet S.A. 1991. *Insects, Plants and Microclimate.* Richmond Publishing, Slough, UK.

Varlet-Granchier C., Bonhomme R. & Sinoquet H. 1993. *Crop Structure and Light Microclimate*, INRA, Paris, France.

Walker D.C., Southgate J., Hill G., Holcombe M., Hose D.R., Wood S.M., Mac Neil S. & Smallwood R.H. 2004. The Epitheliome: Agent-based Modelling of the Social Behaviour of Cells. *BioSystems* 76, p. 89–100.

Weiß G. (ed) 1997. *Distributed Artificial Intelligence Meets Machine Learning.* Lecture Notes in Artificial Intelligence 1221, Springer, Berlin, Germany.

Weiß G. & Sen S. (eds) 1996. *Adaption and Learning in Multi-Agent Systems.* Lecture Notes in Artificial Intelligence 1042, Springer, Berlin, Germany.

Weller J.L., Reid J.B., Taylor S.A. & Murfet I.C. 1997. The Genetic Control of Flowering in Pea. *Trends in Plant Science* 2, p. 412-418.

Whitehead A.N. 1978. *Process and Reality* (corrected edition). The Free Press, New York, NY, USA.

Woiwod I.P., Reynolds D.R. & Thomas C.D. (eds) 2001. *Insect Movement: Mechanisms and Consequences.* CAB International, Wallingford, UK.

Wooldridge M.J. 2002. *An Introduction to Multiagent Systems*. John Wiley & Sons, Chichester, UK.

Wooldridge M.J. & Jennings N. 1995. *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Amsterdam, The Netherlands, August 8-9, 1994. Lecture Notes in Computer Science 890, Springer, Berlin, Germany.

Wooldridge M.J., Weiß G. & Ciancarini P. (eds) 2002. *Agent-Oriented Software Engineering II.* Lecture Notes in Computer Science 2222, Springer, Berlin, Germany.

Ye Y.-Q. 1986. *Theory of Limit Cycles*. American Mathematical Society, Providence, RI, USA.

Zalucki M.P. 1983. Simulation of Movement and Egglaying in Danaus Plexippus (Lepidoptera: Nymphalidae). *Resources in Population Ecology* 25, p. 353-365.

Zapryanov Z. & Tabakova S. 1999. *Dynamics of Bubbles, Drops, and Rigid Particles.* Kluwer, Dordrecht, The Netherlands.

Zeigler B.P. 1997. *Objects and Systems.* Springer, New York, USA.

Zeigler B.P., Moon Y., Kim D. & Ball G. 1997. The DEVS Environment for High-Performance Modeling and Simulation. *Computational Science & Engineering* 4 (3), p. 61-71, IEEE Computer Society, Los Alamitos, CA, USA.

Zeigler B.P., Praehofer H. & Kim T.G. 2000. *Theory of Modeling and Simulation.* Academic Press, San Diego, CA, USA.

Appendix 1: Ecological Simulation Data

	Non-milkweed area					Patch 1					Patch 2				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
1	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	77	0	15	1	8	2	0	0	2	0	0	0	0	0	0
3	69	0	8	5	5	2	0	0	0	0	0	0	0	0	0
4	60	0	10	5	4	2	0	0	1	1	0	0	0	0	0
5	50	0	7	3	6	3	0	0	1	0	2	0	1	3	0
6	44	0	6	4	4	4	1	0	2	2	2	0	0	1	1
7	40	0	7	5	2	18	15	0	0	1	1	0	0	0	1
8	34	2	8	3	3	31	12	0	1	0	0	1	1	0	1
9	39	3	2	10	6	43	13	1	4	4	11	13	0	0	2
10	49	3	3	18	8	69	30	2	5	7	22	15	0	1	5
11	52	4	10	29	20	116	57	8	7	9	35	20	4	5	8
12	62	4	10	34	18	173	68	8	3	6	48	20	0	6	13

Table 1. Data produced by a micro-level simulation of monarch butterflies: numbers of non-milkweed area and patch subpopulations (columns Pop), births (columns Born), deaths (columns Died), immigration (columns Im), and emigration (columns Em).

	Non-milkweed area					Patch 1					Patch 2				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
13	77	5	13	46	23	269	111	7	5	13	66	25	3	9	13
14	76	5	23	37	20	368	118	10	4	13	84	25	3	8	12
15	87	5	23	52	23	484	140	16	5	13	93	25	6	4	14
16	90	4	29	61	33	569	123	27	7	18	106	20	4	6	9
17	122	5	21	77	29	717	170	15	7	14	117	26	2	6	19
18	129	4	35	81	43	819	147	39	13	19	126	22	7	4	10
19	136	7	57	95	38	994	272	78	11	31	136	39	18	4	15
20	177	4	21	102	44	1084	137	29	11	29	152	31	4	5	16
21	212	3	27	105	43	1150	118	31	10	30	159	24	5	7	19
22	248	2	25	107	48	1206	113	38	10	29	169	21	4	8	15
23	270	2	38	101	43	1259	99	31	12	28	180	20	4	7	12
24	308	2	35	117	46	1309	99	32	14	31	191	23	4	6	14

Table 1 (continued).

	Patch 3					Patch 4				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
1	1	0	0	0	0	0	0	0	0	0
2	6	0	0	6	1	0	0	0	0	0
3	5	0	0	2	3	1	0	0	1	0
4	4	0	0	0	1	0	0	0	0	1
5	7	3	0	2	2	0	0	0	0	0
6	13	6	0	1	1	0	0	0	0	0
7	26	16	2	1	2	2	2	0	0	0
8	35	12	3	1	1	4	2	0	0	0
9	42	12	2	1	4	11	6	0	1	0
10	57	16	0	0	1	14	6	3	1	1
11	75	26	4	5	9	26	13	0	0	1
12	101	29	0	6	9	42	18	2	2	2

Table 1 (continued).

	Patch 3					Patch 4				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
13	146	54	4	3	8	64	25	1	2	4
14	194	59	9	4	6	85	25	5	2	1
15	240	64	13	8	13	101	25	5	4	8
16	272	54	16	6	12	111	20	6	8	12
17	325	70	10	4	11	129	25	2	9	14
18	358	60	17	7	17	129	20	12	11	19
19	433	114	33	8	16	151	35	7	10	17
20	471	60	13	6	15	156	19	7	5	12
21	495	48	14	6	16	157	15	6	7	15
22	515	46	16	5	15	164	15	2	10	16
23	520	38	19	7	21	162	12	3	2	13
24	529	35	15	4	15	161	13	2	6	18

Table 1 (continued).

	Patch 5					Patch 6				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	2	2
4	0	0	0	0	0	1	0	0	3	2
5	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	1	1	0	0	0
7	0	0	0	0	0	5	4	0	1	1
8	1	0	0	1	0	7	3	0	0	1
9	1	0	0	0	0	10	3	0	0	0
10	1	0	0	0	0	10	3	0	1	4
11	2	1	0	0	0	17	6	0	3	2
12	6	5	0	0	1	25	12	2	1	3

Table 1 (continued).

	Patch 5					Patch 6				
Hour	Pop	Born	Died	Im	Em	Pop	Born	Died	Im	Em
13	16	10	0	0	0	36	15	0	4	8
14	30	14	0	0	0	49	18	2	2	5
15	44	15	1	1	1	65	20	2	1	3
16	54	12	0	0	2	78	16	1	6	8
17	57	15	4	0	8	92	24	2	3	11
18	60	13	5	3	8	103	20	6	5	8
19	68	21	6	3	10	125	36	11	2	6
20	76	11	1	5	7	130	18	2	12	23
21	75	9	5	5	10	135	15	3	8	15
22	73	9	2	4	13	137	13	3	11	19
23	72	7	5	5	8	135	14	7	10	19
24	63	8	3	1	15	135	10	1	15	24

Table 1 (continued).

Appendix 2: Genetic Network Data

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	2.33	1	2.08	1	0.90	0	0.00	0	0.00	1	2.08
rms1	1	1.23	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	1.49	0.5	0.75	0	0.00	0	0.00	0	0.00	0.5	0.79
rms3	1	2.33	1	2.08	1	0.90	0	0.00	0	0.00	1	2.08
rms4	1	2.33	1	2.08	1	0.90	0	0.00	0	0.00	1	2.08
rms5	1	1.23	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 1. Experimental data (E) and model results (M). Experimental data are from (Harding 2003).
- represents a graft not yet conducted experimentally. Model results have been calculated with Harding's model as stable values after 50 iterations (averages of last 10 iterations) and correspond roughly to model results at the 10^{th} iteration, as in (Harding 2003). Fitness of Harding's model would be 0.543 as calculated in Computational Experiments 4 - 7 hereafter (deviations in bold of model results from experimental data contribute to the fitness value).

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot.:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	1.69	1	1.69	1	1.23	1	1.75	1	0.79	1	1.45	1	1.23	1	1.75
rms1	1	1.09	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	-	0.00	-	0.00
rms2	1	1.05	0	0.08	-	0.20	-	0.20	0	0.00	0	0.00	-	0.20	-	0.20
rms5	1	1.21	0	0.00	-	0.00	-	0.00	-	0.00	-	0.00	0	0.00	0	0.00

Two root grafts

	Shoot:	WT		rms1		rms2		rms5	
Root 1:	Root 2:	E	M	E	M	E	M	E	M
WT	WT	1	1.93	1	1.55	1	0.90	1	1.55
WT	rms1	1	1.45	0.75	0.91	-	0.45	-	0.91
WT	rms2	1	1.48	-	0.97	0.5	0.45	-	0.97
WT	rms5	1	1.45	-	0.91	-	0.45	1	0.91
rms1	rms1	1	1.02	0	0.00	-	0.00	-	0.00
rms2	rms2	1	1.16	-	0.33	0	0.00	-	0.33
rms5	rms5	1	1.01	-	0.00	-	0.00	0	0.00

Table 1 (continued).

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	3.83	1	1.73	1	2.00	0	0.00	0	0.00	1	1.73
rms1	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	1.88	0.5	**1.00**	0	0.00	0	0.00	0	0.00	0.5	**1.00**
rms3	1	2.56	1	1.00	1	2.00	0	0.00	0	0.00	1	1.00
rms4	1	3.83	1	1.73	1	2.00	0	0.00	0	0.00	1	1.73
rms5	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 2. Experiment 3: model output and experimental data for I-grafts.
Deviations in bold give a fitness value of 1.001.

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	100	1	1.41	1	1.41	0	0.00	0	0.00	1	1.41
rms1	1	100	0	0.21	0	0.21	0	0.00	0	0.00	0	0.21
rms2	1	100	0.5	**0.25**	0	**0.25**	0	0.00	0	0.00	0.5	**0.25**
rms3	1	100	1	1.41	1	1.41	0	0.00	0	0.00	1	1.41
rms4	1	100	1	1.41	1	1.41	0	0.00	0	0.00	1	1.41
rms5	1	100	0	0.21	0	0.21	0	0.00	0	0.00	0	0.21

Table 3. Experiment 4: model output and experimental data.
Deviations in bold give a fitness value of 0.250 (the cut-off point for counting a deviation as > 0 was 0.25; two of the bold values are just below 0.25).

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot.:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	100	1	2.00	1	1.18	1	2.00	1	1.18	1	2.00	1	1.18	1	2.00
rms1	1	100	0	0.00	0	0.21	0	0.00	0	0.21	0	0.00	-1	0.21	-1	0.00
rms2	1	100	0	0.00	-1	0.21	-1	0.00	0	0.21	0	0.00	-1	0.21	-1	0.00
rms5	1	100	0	0.00	-1	0.21	-1	0.00	-1	0.21	-1	0.00	0	0.21	0	0.00

Table 3 (continued).

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	2.87	1	1.25	1	1.25	0	0.00	0	0.00	1	1.25
rms1	1	1.62	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	1.87	0.5	**0.25**	0	**0.25**	0	0.00	0	0.00	0.5	**0.25**
rms3	1	2.87	1	1.25	1	1.25	0	0.00	0	0.00	1	1.25
rms4	1	2.87	1	1.25	1	1.25	0	0.00	0	0.00	1	1.25
rms5	1	1.62	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 4. Experiment 5: model output and experimental data.
Deviations in bold give a fitness value of 0.750.

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot.:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	2.75	1	2.75	1	1.13	1	2.75	1	1.13	1	2.75	1	1.13	1	2.75
rms1	1	1.62	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	-1	0.00	-1	0.00
rms2	1	1.75	0	0.13	-1	0.13	-1	0.13	0	0.13	0	0.13	-1	0.13	-1	0.13
rms5	1	1.62	0	0.00	-1	0.00	-1	0.00	-1	0.00	-1	0.00	0	0.00	0	0.00

Table 4 (continued).

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	100	1	3.35	1	1.67	0	0.00	0	0.00	1	7.83
rms1	1	100	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	100	0.5	0.33	0	0.17	0	0.00	0	0.00	0.5	0.53
rms3	1	100	1	3.35	1	1.67	0	0.00	0	0.00	1	7.83
rms4	1	100	1	5.43	1	2.71	0	0.00	0	0.00	1	15.50
rms5	1	100	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 5. Experiment 6: model output and experimental data.
The fitness of the model is 0.0

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot.:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	100	1	100	1	1.67	1	100	1	0.84	1	100	1	3.21	1	100
rms1	1	100	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	-1	0.00	-1	0.00
rms2	1	100	0	0.08	-1	0.17	-1	0.08	0	0.08	0	0.08	-1	0.26	-1	0.08
rms5	1	100	0	0.00	-1	0.00	-1	0.00	-1	0.00	-1	0.00	0	0.00	0	0.00

Two root grafts

Shoot:		WT		rms1		rms2		rms5	
Root 1:	Root 2:	E	M	E	M	E	M	E	M
WT	WT	1	100	1	1.88	1	0.94	1	3.69
WT	rms1	1	100	0.75	0.94	-1	0.47	-1	1.63
WT	rms2	1	100	-1	0.97	0.5	0.49	-1	1.69
WT	rms5	1	100	-1	0.94	-1	0.47	1	1.63
rms1	rms1	1	100	0	0.00	-1	0.00	-1	0.00
rms2	rms2	1	100	-1	0.07	0	0.03	-1	0.10
rms5	rms5	1	100	-1	0.00	-1	0.00	0	0.00

Table 5 (continued).

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	17.1	1	34.9	1	53.6	0	0.00	0	0.00	1	75.0
rms1	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	1.37	0.5	0.37	0	0.37	0	0.00	0	0.00	0.5	0.37
rms3	1	17.5	1	35.3	1	54.0	0	0.00	0	0.00	1	75.0
rms4	1	76.0	1	75.0	1	75.0	0	0.00	0	0.00	1	75.0
rms5	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 6. Experiment 7: model 1 output and experimental data.
The fitness of the model is 0.625.

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot.:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	9.25	1	9.25	1	17.6	1	18.6	1	27.0	1	28.0	1	37.5	1	38.5
rms1	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	-1	0.00	-1	0.00
rms2	1	1.19	0	0.19	-1	0.19	-1	0.19	0	0.19	0	0.19	-1	0.19	-1	0.19
rms5	1	1.00	0	0.00	-1	0.00	-1	0.00	-1	0.00	-1	0.00	0	0.00	0	0.00

Two root grafts

Shoot:		WT		rms1		rms2		rms5	
Root 1:	Root 2:	E	M	E	M	E	M	E	M
WT	WT	1	2.50	1	1.50	1	1.50	1	1.50
WT	rms1	1	1.75	0.75	0.75	-1	0.75	-1	0.75
WT	rms2	1	1.75	-1	0.75	0.5	0.75	-1	0.75
WT	rms5	1	1.75	-1	0.75	-1	0.75	1	0.75
rms1	rms1	1	1.00	0	0.00	-1	0.00	-1	0.00
rms2	rms2	1	1.00	-1	0.00	0	0.00	-1	0.00
rms5	rms5	1	1.00	-1	0.00	-1	0.00	0	0.00

Table 6 (continued).

I-grafts

Shoot:	WT		rms1		rms2		rms3		rms4		rms5	
Root:	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	17.1	1	34.9	1	53.6	0	0.00	0	0.00	1	75.0
rms1	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
rms2	1	1.37	0.5	0.37	0	0.37	0	0.00	0	0.00	0.5	0.37
rms3	1	17.5	1	35.3	1	54.0	0	0.00	0	0.00	1	75.0
rms4	1	76.0	1	75.0	1	75.0	0	0.00	0	0.00	1	75.0
rms5	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

Table 7. Experiment 7: model 2 output and experimental data.
The fitness of the model is = 0.625

Y-grafts

Shoot:	WT				rms1				rms2				rms5			
	Scion		Cot.		Scion		Cot.		Scion		Cot.		Scion		Cot.	
Root / Cot:	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M
WT	1	9.25	1	9.25	1	17.6	1	18.6	1	27.0	1	28.0	1	37.5	1	38.5
rms1	1	1.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	-1	0.00	-1	0.00
rms2	1	1.19	0	0.19	-1	0.19	-1	0.19	0	0.19	0	0.19	-1	0.19	-1	0.19
rms5	1	1.00	0	0.00	-1	0.00	-1	0.00	-1	0.00	-1	0.00	0	0.00	0	0.00

Two root grafts

Shoot:		WT		rms1		rms2		rms5	
Root 1:	Root 2:	E	M	E	M	E	M	E	M
WT	WT	1	2.50	1	1.50	1	1.50	1	1.50
WT	rms1	1	1.75	0.75	0.75	-1	0.75	-1	0.75
WT	rms2	1	1.75	-1	0.75	0.5	0.75	-1	0.75
WT	rms5	1	1.75	-1	0.75	-1	0.75	1	0.75
rms1	rms1	1	1.00	0	0.00	-1	0.00	-1	0.00
rms2	rms2	1	1.00	-1	0.00	0	0.00	-1	0.00
rms5	rms5	1	1.00	-1	0.00	-1	0.00	0	0.00

Table 7 (continued).

Mutant	*L107*	*L7*	*L60*	*L66*	*M2/176*	*L59*	*L158*
Gene							
GI	1	1	1	1	1	1	0.1 – 0.7
SN	1	1	1	1	0.1 – 0.6	0.1 – 0.6	1
LF	1	0 – 0.4	0.4 – 0.6	1.4 - 9	1	0.4 – 0.6	1
Flowering							
NFI (24)	15	7	11	24	11	9	29
NFI (8)	26	7	11	50	11	9	70

Table 8. Gene expression data, photoperiod and flowering time of pea. Expression levels of genes *Gigas* (GI), *Sterile Nodes* (SN), and *Late Flower* (LF) are shown in the first three rows of data, with levels of wild type L107 set to 1, and levels of 6 mutants set to values relative to wild type expression. The fourth and fifth rows show the *node of floral initiation* (NFI) of each mutant under 24 and 8 hours photoperiodic exposure respectively (Bell 2003).