Discovering Genetic Regulatory Network Models in *Pisum sativum*

H.J. Stolk [1, 2, 3] and J. Hanan [2, 3 4]

[1] Dione Complex Systems, Bangor, Gwynedd, United Kingdom
[2] Advanced Computational Modelling Centre, [3] ARC Centre for Complex Systems & [4] ARC Centre of Excellence for Integrative Legume Research, The University of Queensland, Brisbane, Queensland, Australia
(for H.J. Stolk until beginning 2005)
Email: Henk@DioneComplexSystems.com

Keywords: *Emergent models, multi-agent simulation, genetic regulatory network modelling*

EXTENDED ABSTRACT

Living organisms consist of interacting objects such as organs, cells, and genes. Therefore, they can be considered complex systems and we have applied a complex systems simulation methodology to a living organism, focussing on the relationship between interacting genes and phenotype. Different levels can be distinguished in a complex system, from the level with the most fine-grained entities, through intermediate levels of composite entities, to 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. This phenomenon of macro-level properties and behaviour arising out of micro-level properties and behaviour, without being apparent from them, is known as emergence, a fundamental property of complex systems.

The new Emergent Models methodology uses multi-agent simulations to study emergence in complex systems. Agents model micro- and macro-level components of a complex system. Emergent models reveal models of properties and behaviour of higher-level agents as emerging from properties and behaviour of lower-level agents. The present work examines the inverse problem of deriving models of individual agents from higher-level agent properties and behaviour, using genetic programming algorithms to construct interacting individual agents, together composing a genetic regulatory network, from higher-level phenotype observations in the garden pea *Pisum sativum*.

Genetic programming is an extension of genetic algorithms in which the genetic population contains computer programs or mathematical functions to solve problems. 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.

To make clear what 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. 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.

In biology phenotype data are often available, while genetic and biochemical mechanisms remain unknown. Thus, the ability to discover micro-level genetic regulatory network models from macro-level phenotype data is highly desirable. In computational experiments models of genetic regulatory networks regulating branching in *Pisum sativum* were automatically discovered to fit biological data describing effects on the pea phenotype caused by mutations of genes thought to regulate branching. Some of the discovered models explain observed data without assuming two feedback effects appearing in a model formulated by human biologists.

The experiments have demonstrated how the Emergent Models methodology can find models of genetic regulatory networks satisfying specific constraints and optimising fit to observed data. The Emergent Models methodology can assist scientific discovery by discovering models that may not be intuitively obvious to humans.

1. INTRODUCTION

Living organisms consist of interacting objects such as organs, cells, and genes. Therefore, they can be considered complex systems and we have applied a complex systems simulation methodology to a living organism, focussing 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. Therefore, the ability to discover micro-level models from macro-level phenotype data is highly desirable. The present work illustrates how micro-level models can be derived from macro-level data by applying a complex systems methodology to a problem in molecular biology with genetic and biochemical networks.

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. Bar-Yam 1997; Holland 1998).

As argued by Stolk *et al.* (2007) 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 existing so far. Such a methodology is the Emergent Models methodology developed by Stolk (2005) and also described by Stolk *et al.* (2003), which is outlined in Section 2. This methodology can also be applied to the inverse problem of deriving micro-level properties and behaviour from macro-level properties and behaviour.

2. THE EMERGENT MODELS METHODOLOGY

In order to combine the strengths of the mathematical and simulation approaches to scientific discovery, while avoiding their respective limitations, the Emergent Models methodology uses 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 consists of building multi-agent simulations (e.g. Holland 1998; Ferber 1999), with agents at different levels modelling micro-level and macro-level components of a complex system.

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 this problem is addressed, 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 illnesses.

Various methods can be used to discover emergent macro-level models from micro-level simulations, or underlying micro-level models from macro-level models or observed data. If the variables of the model equations are already known from theoretical considerations and only unknown parameters need to be estimated, standard linear or non-linear regression techniques can be used. If the important variables also have to be discovered, more advanced techniques are needed, such as evolutionary algorithms or other machine learning methods.

The present work examines the inverse problem of deriving properties, behaviour, and interactions of individual agents from higher-level agent properties and behaviour, using genetic programming algorithms (see e.g. Koza 1992) to construct interacting individual agents, together composing a genetic regulatory network, from higher-level phenotype observations in *Pisum sativum*.

We use genetic programming to discover micro-level models from observed data, as it is an all-purpose method with sufficient flexibility to be applicable to many interesting cases. 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.

3. 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 1.

Table 1. A genetic algorithm.

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.

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

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 1. 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 function for inclusion in the next generation. *Crossover* operates on two parental functions selected with a probability based on fitness and creates one or two new offspring functions consisting of parts of each parent. In the tree representation crossover means randomly selecting and exchanging subtrees of both parents, as illustrated in Figure 1. The offspring is inserted into the next generation. *Mutation* operates on one parental function selected with a probability based on fitness and creates one new offspring function to be inserted into the next generation. In the mutation operation a point is randomly chosen in the parental function. The subtree rooted at the chosen mutation point is deleted from the function, 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. In the present work the data are observed data from biological experiments and the functions to be discovered describe micro-level models of genetic regulatory networks.

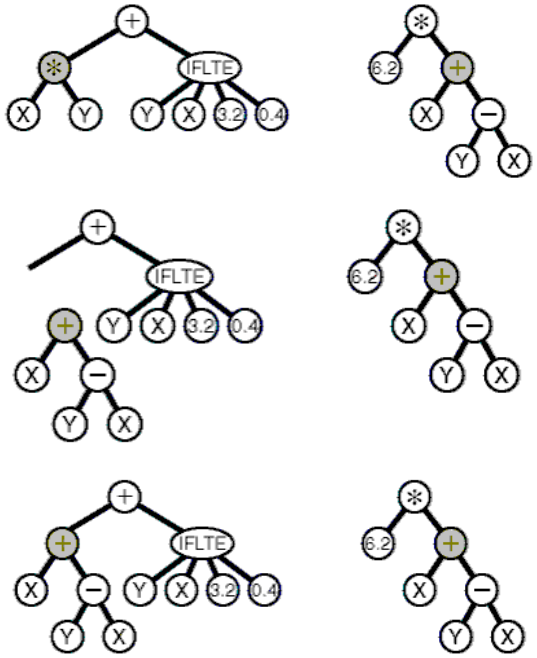


Figure 1. Tree representation of functions and the crossover operation (Luke 2002). Before crossover, the left hand function is $F(x, y) = (x \times y) + (if\ x \leq y\ then\ 3.2\ else\ 0.4)$ and the right hand one $G(x, y) = 6.2 \times (x + (y - x))$. After crossover, the left hand function is $H(x, y) = \big(x + (y - x)\big) + (if\ x \leq y\ then\ 3.2\ else\ 0.4)$.

4. APPLICATION TO GENETIC REGULATORY NETWORK MODELS

To make clear what 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). 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 regulatory 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 1 or $true$ for an active gene (gene products present) and value 0 or $false$ 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 regulatory 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 regulatory 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 regulatory 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.

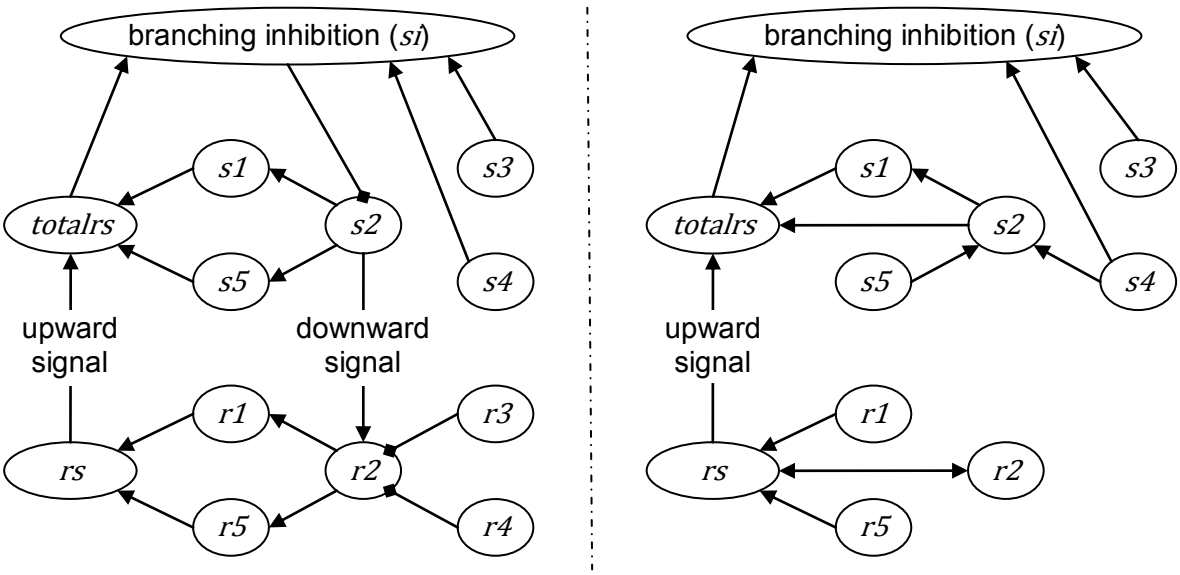
Now, given a set of macro-level data about the phenotype, can we find a genetic regulatory network with a pattern of activity explaining the phenotype data of the whole organism or its organs? This problem was solved by conducting computational experiments with models of genetic regulatory networks regulating branching in garden pea (*Pisum sativum*). Genetic regulatory network models were automatically discovered to fit data collected in biological experiments. The data used described effects on the pea phenotype caused by mutations of the genes thought to regulate branching.

5. EXAMPLE: BRANCHING IN PISUM SATIVUM

A model of a genetic network regulating branching in pea, formulated in the context of a biological research project described by Harding (2003), is schematically presented in Figure 2(a), 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 of the same or a different genotype. The experiments also included grafts of one root with two shoots, one with a different and one with the same genotype as the root. Similarly, two roots could be grafted to a single shoot to get a two-root graft.

In Figure 2 the phenotype of a plant is described by the level of branching inhibition si. A plant with wild type root and scion has by definition a branching inhibition of 1. Smaller values of branching inhibition imply more branching. It has been demonstrated that genes *Rms1*, *Rms2*, *Rms3*, *Rms4* and *Rms5* play a role for the control of branching in *Pisum sativum* (Weller *et al.* 1997; Beveridge *et al.* 2003). Harding's (2003) model describes relationships between variables representing presence or absence of these genes in roots and shoots, as well as levels of gene products regulated by these genes in root and shoot, of intermediate signalling products and of branching inhibition.

It uses algebraic equations with variables $R1$, $R2$, $R3$, $R4$ and $R5$ representing presence or absence of genes *Rms1*, *Rms2*, *Rms3*, *Rms4* and *Rms5* respectively in the root; variables $S1$, $S2$, $S3$, $S4$ and $S5$ represent presence or absence of the same genes in the shoot. The corresponding levels of gene products in the root are denoted in the equations and in Figure 2 by $r1$, $r2$, $r3$, $r4$ and $r5$ and those in the shoot by $s1$, $s2$, $s3$, $s4$ and $s5$. 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.



(a) model as described by Harding (2003) (b) computationally discovered model (Stolk 2005)

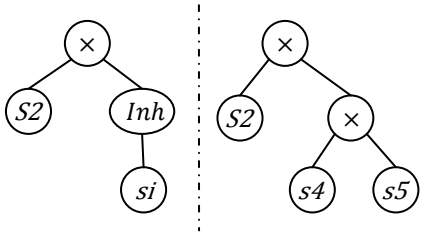
Figure 2. Genetic regulatory network determining branching in *Pisum sativum*.

Intermediate quantities rs and $totalrs$ are computed from gene product levels and can be interpreted as levels of signals. 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. 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. Arrows in Figure 2 represent an activation effect and block arrows an inhibition effect of one gene product on another one.

It was hypothesised that a genetic programming algorithm could automatically discover alternative genetic regulatory network models from the observed data, as presented by Harding (2003). 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.

In computational experiments genetic regulatory networks regulating branching in pea were discovered using genetic programming, starting from observed phenotype data. Inputs for the genetic programming algorithm consisted of variables (gene presence, gene product levels, and signals), constants, and functions relating these. The algorithm constructs possible models by randomly mutating and recombining variables, constants and functions. It selects models from a population of models to minimise a fitness measure reflecting the deviation of model generated outputs from results as measured in biological experiments.

The fitness measure used was the sum of deviations of model results from observed data for the value of branching inhibition, for 74 grafts with different genotypes of roots and shoots and consequently with different observed levels of branching inhibition. Details of the exact fitness measure are given by Stolk (2005). Output consisted of models with a good fit to observed data, with equations for branching inhibition, gene product levels and signals. One model found by the algorithm is represented in Figure 2(b), where the arrows correspond to equations of the model. As an example, the equation for $s2$ is represented by the genetic programming algorithm as in Figure 3. This model is one of many models with a good fit found by genetic programming and is presented here because it has a straightforward interpretation. Automatic discovery as examined here is blind to meaningfulness and selection of meaningful models depends on interpretation by researchers. The discovered model has a fitness value of 0.625, comparable to the fitness value 0.543 of Harding's (2003) model. These fitness values correspond to average deviations of model results from experimental results in the order of 1 %.



(a) (b)

Figure 3. Equations for $s2$ in Figure 2(a) and 2(b). Inhibition function Inh is defined as $Inh(x) = 2/1 + x$

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 interesting differences. For example, in the original model branching inhibition si has an inhibitory feedback effect on gene product $s2$. In the model found here the feedback effect of branching inhibition is replaced by an effect of $s4$ and $s5$ on $s2$. In the original model $s4$ and $s5$ also have an effect on $s2$, but this effect is indirect, through $totalrs$ and/or si. Further, the feedback from shoot to root appearing in the original model is lacking in the discovered model, as $s2$ 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.

6. CONCLUSIONS

In conclusion, using the Emergent Models methodology, in this case implemented by making use of genetic programming, we can find models of genetic regulatory networks satisfying constraints specified by the programmer and optimising fit 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 with predominantly heuristic value. 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. For example, a useful application of automatic discovery of this type could be to point the way to new experiments to discriminate between different models that would otherwise be compatible with available data.

7. REFERENCES

Bar-Yam, Y. (1997), Dynamics of Complex Systems, Perseus Books, Reading, MA.

Beveridge, C.A., J.L. Weller, S.R. Singer, and J.M.I. Hofer (2003), Axillary meristem development: budding relationships between networks controlling flowering, branching, and photoperiod responsiveness, *Plant Physiology*, 131, 927–934.

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

De Jong, H. (2002). Modeling and simulation of genetic regulatory Systems: a literature review, *Journal of Computational Biology* 9(1), 67–103.

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

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

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

Holland, J.H. (1998), Emergence: From Chaos to Order, Oxford University Press, Oxford.

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

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

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

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

Okubo, A., and S.A. Levin (eds) (2001), Diffusion and Ecological Problems: Modern Perspectives, 2nd ed, Springer, New York.

Ptashne, M., and A. Gann (2002), Genes and Signals, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Stolk, H.J. (2005), Emergent Models in Hierarchical and Distributed Simulation of Complex Systems, PhD Thesis, University of Queensland, Brisbane, Australia.

Stolk, H., K. Gates, and J. Hanan (2003), Discovery of emergent natural laws by hierarchical multi-agent systems, paper presented at IEEE/WIC International Conference on Intelligent Agent Technology, Halifax, Canada, October.

Stolk, H.J., M.P. Zalucki, and J. Hanan (2007), Subpopulation agents emerge from individual agents in metapopulation simulations, paper presented at MODSIM07, Christchurch, New Zealand, December.