Evolutionary Modular Neural Networks for Intelligent Systems

Sung-Bae Cho*
Dept. of Computer Science, Yonsei University, Sudaemoon-ku, Seoul 120-749, Korea

The evolutionary approach to artificial neural networks has been rapidly developing in recent years and shows great potential as a powerful tool. However, most evolutionary neural networks have paid little attention to the fact that they can evolve from modules. This paper presents a hybrid method of modular neural networks and evolutionary algorithm as a promising model for intelligent systems. To build a neural network system that is rich in autonomy and creativity, some ideas of artificial life have been adopted. This paper describes the concepts and methodologies for the evolvable model of modular neural networks, which might not only develop spontaneously new functionality, but also grow and evolve its own structure autonomously. We show the potential of the method by applying it to a visual categorization task with handwritten digits. The evolutionary mechanism has shown a strong potential to generate useful network architectures from an initial set of randomly connected networks. © 1998 John Wiley & Sons, Inc.

1. INTRODUCTION

Intelligent systems adaptively estimate continuous functions from data without specifying mathematically how outputs depend on inputs.[1] System behavior is called *intelligent* if the system emits appropriate, problem-solving responses when faced with problem stimuli. Recently, some researchers have tried to synthesize intelligent systems by using neural networks.

Most of the currently popular network architectures, however, show little structural constraints. Some networks assume total connectivity between all nodes. Others assume a hierarchical, multilayered structure where each node in a layer is connected to all nodes in neighboring layers. However, there is a large body of neuropsychological evidence showing that the human information processing system consists of modules, which are subdivisions in identifiable parts, each with its own purpose or function.

Questions may then be raised about how to design an information processing system with various modules. There has been extensive work to design efficient architectures from the engineering point of view,[2-4] which has pro-

*E-mail: sbcho@csai.yonsei.ac.kr.

duced some success in solving several problems. On the contrary, the concept of evolving neural networks has appeared recently as a subject of intensive scientific discussions from biological, social, and engineering viewpoints.[5–10]

In order to build an artificial neural network that is rich in autonomy and creativity, we have adopted the ideas and methodologies of artificial life. In this paper, we describe the concepts and methodologies for the evolvable model of modularized neural networks, which will be able not only to develop new functionality spontaneously, but also to grow and evolve its own structure autonomously.

2. RELATED WORKS

There is already a body of work wherein researchers both understand and appreciate the importance of incorporating an evolutionary process into the picture. Wilson[11] discussed a general representational framework to set the stage for simulations of evolution. A number of people have implemented some models, which can be roughly categorized into two groups.[12]

Much work involves some kind of growth-model coding for evolving neural networks. Kitano[7] and Whitley and Hanson[6] used grammatical encoding to develop artificial neural networks. Harp[5] tried to evolve the gross anatomy and general operating parameters of a network by encoding areas and projections onto them into the genome. Nolfi and Parisi[13] used an abstraction of axon growth to evolve connectivity architectures. Most of these models do not aspire to be biologically defensible, though. Also, they have not been applied as such in the area of autonomous agents.

In contrast, a number of other researchers have looked at more biologically inspired models of evolutionary process. Some work is based on the grammar based approach first developed by Lyndenmayer, such as deBoer, Fracchia, and Prusinkiewicz.[14] For instance, Mjolsness, Sharp, and Reinitz[15] used grammatical rules, to account for morphological change, coupled to a dynamical neural network to model the internal regulatory dynamics of the cell. Fleischer and Barr[16] combined a hard-coded model for gene expression with a cell simulation program.

It is the combination of a biologically defensible model of development with evolutionary methods that we would like to apply to the design of intelligent systems, something that at this point in time has not yet been addressed in the existing literature. The point of this research is to utilize modules as building blocks for developing intelligent systems by an evolutionary mechanism that is quite similar to genetic programming.

3. EVOLUTIONARY MODULAR NEURAL NETWORKS

3.1. Overview

In order to give autonomy and creativity to a neural processing system, it is vital that the system itself have some mechanism to spontaneously generate

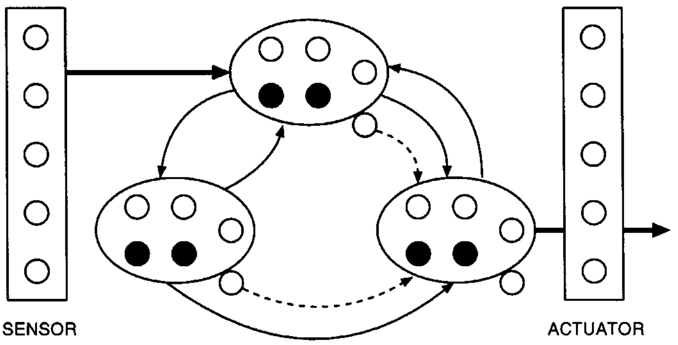


Figure 1. Overall structure of modular neural networks.

change in its function and structure.[17] The overall scheme of the proposed system is presented in Figure 1. In this figure, there are three modules that are connected to each other with two kinds of links: one for information flow (solid lines) and the other for control flow (dashed lines). These modules cooperate to produce appropriate output to the actuator given some sensory inputs. The basic idea is to consider a module as a building block that results in local representations by competition and to develop complex intermodular connections with an evolutionary mechanism

The initial network architecture is encoded as a chromosome. In our work to date, we have used a genetic encoding scheme that stores hierarchically the wiring diagram for the network. The encoding has been developed to be robust with respect to the genetic operators such as mutation and crossover.

3.2. Modular Neural Networks

The basic elements and structure of a module are designed to model neocortical minicolumns, as first proposed by Murre, Phaf, and Wolters.[18] The activation value of each node is calculated as

$$e_i = \sum_j w_{ij} a_j(t), \tag{1}$$

where w_{ij} denotes the weight of a connection from node j to node i. The effective input to node i, e_i, is the weighted sum of the individual activations of all nodes connected to the input side of the node. The input may be either positive (excitatory) or negative (inhibitory).

A module based on such nodes is designed to model neocortical minicolumns.[18] The constraints embodied in the model include:

(1) Dale's principle that individual neurons emit only one type of transmitter.
(2) Learning as a local phenomenon that does not require knowledge of the correct response.
(3) The capacity to differentiate between novel and familiar input and behave differently on that basis.

The internal connections in a module are fixed and the weights of all intramodular connections are nonmodifiable as exemplified in Figure 2.

In a module, the R node represents a particular pattern of input activations to a module, the V node inhibits all other nodes in a module, the A node activates a positive function of the amount of competition in a module, and E node activation is a measure of the level of competition going on in a module. The process in a module goes with the resolution of a winner-take-all competition between all R nodes activated by input. In the first presentation of a pattern to a module, all R nodes are activated equally, which results in a state of maximal competition, which is resolved by the inhibitory V nodes. The most important feature of a module is to autonomously categorize input activation patterns into discrete categories, which is facilitated as the association of an input pattern with a unique R node.

The Hebb rule is used for learning intermodular connections via the equation

$$\Delta w_{ij}(t+1) = \mu_t a_i\left(\left[K - w_{ij}(t)\right]a_j - Lw_{ij}(t)\sum_{f\neq j} w_{if}(t)a_f\right) \qquad \mu_t = d + w_{\mu_E}a_E \tag{2}$$

where a_i, a_j, and a_f are activations of the corresponding R nodes, respectively, $w_{ij}(t)$ is the interweight between R nodes j and i, $w_{if}(t)$ indicates an interweight from a neighboring R node f (of j) to R node i, and $\Delta w_{ij}(t+1)$ is the change in weight from j to i at time $t+1$. Note that L and K are positive constants, and a_E is the activation of the E node. As a mechanism for generating change and integrating the changes into a system, we use module duplication and

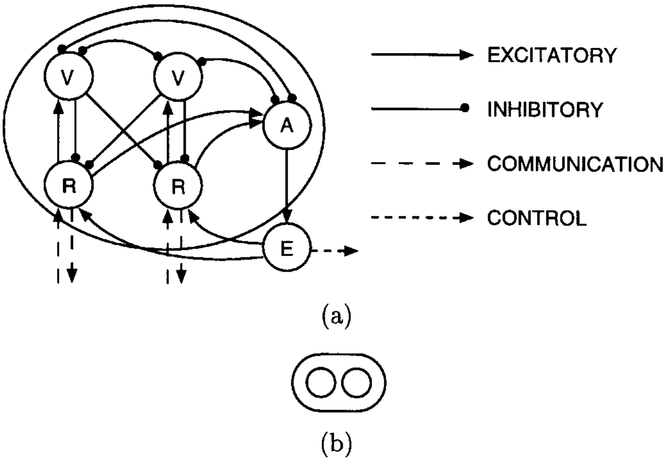


Figure 2. (a) Schematic of the internal structure of a module. (b) Simplified representation of the module in (a).

elimination, and an evolutionary algorithm to determine the parameters in the above learning rule and the structure of intermodular connections.

3.3. Genetic Representation

The chromosome has a tree structure that expresses the intermodular connections in which symbols are replaced by the corresponding modules. A chromosome example is shown in Figure 3. Each node has a number representing a specific module and several parameters on the number of nodes, as well as local settings of the learning and activation rules and the fixed internal weights of intramodular connections. The root of the tree is the input module that

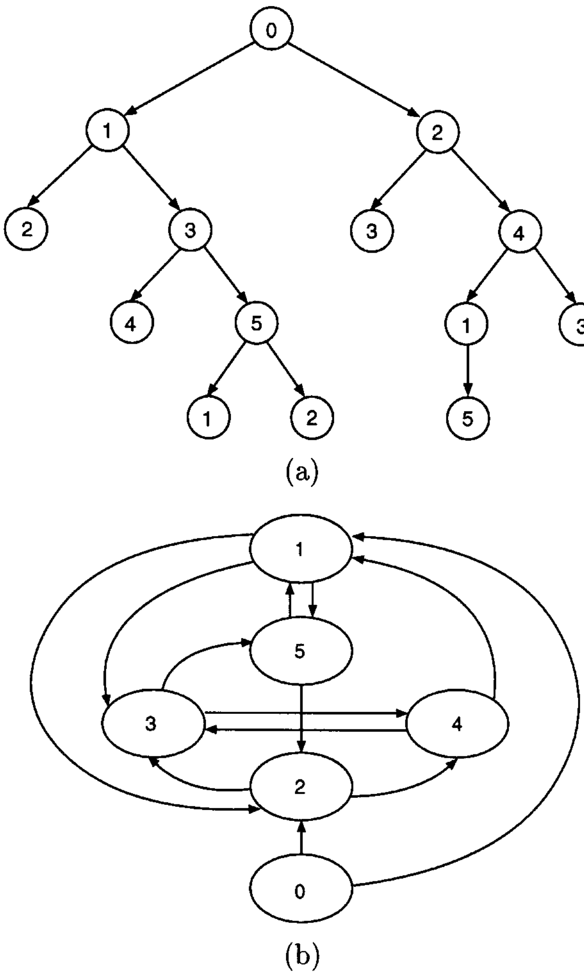


Figure 3. (a) Genetic code encoded by the tree structure. (b) A modular neural network architecture developed by the chromosome of diagram (a).

replaces the start symbol. A child node has a module number to be applied to the symbols that represent the modules connected by its mother module.

By performing the genetic operators in the chromosome pool, the interconnection between modules as well as the number of them are changed. Designing the chromosome to represent the interconnectivity makes it possible to generate a variety of offspring and to evolve them.

3.4. Genetic Operators

The following genetic operators are used in our approach.

- *Selection*. Roulette wheel selection[19] is used. In roulette wheel selection, each individual survives to the next generation in proportion to its performance. Elitist strategy[20] is also applied to the selection. Some of the best individuals in the population are made to remain to the next generation. Elitist strategy prevents all of the best individuals from being eliminated by stochastic genetic drifts.
- *Crossover*. Crossover exchanges subtrees between two chromosomes. It is similar to the operator used in genetic programming.[20] By performing crossover, many useful interconnection parts are gathered, and the intermodular connectivity evolves. An example of the crossover is shown in Figure 4(a).
- *Deletion*. The deletion operator deletes a function block from a chromosome. It is expected to delete useless parts in the chromosome. As a result, a more compact individual with the same function is generated (see Figure 4(b)).
- *Mutation*. Mutation changes each tree node to a new node in proportion to the mutation rate. The mutated node is replaced with the new node and then the subtree below the original node is deleted. Next a new subtree is created below the new node according to some probability. The main roles of the mutation are enforcing local search and making slight modifications to the connectivity parts obtained by crossover and duplication. An example of mutation is shown in Figure 4(c).
- *Insertion*. The insertion operator is similar to duplication except that it inserts a function block from another chromosome (see Figure 4(d)).
- *Duplication*. Duplication imitates the gene duplication[21] in living creatures and makes it possible to evolve the chromosome from a simple structure to a complex one. It also increases the complexity of functions and expands the network size. Duplication inserts a copy of a function block within the same chromosome. The function block is the part of the chromosome where the nodes with the same category appear in a list. Duplication leads to a functionally correct interconnection. Just after duplication is performed, the inserted function block does not affect the individual's behavior and is neutral. Therefore, duplication does not change the functionality for the individual. The inserted block may be modified by mutations and a new function might emerge. The other parts of the chromosome also change and the inserted block is incorporated into the whole behavior. Figure 4(e) shows an example.

4. SIMULATION RESULTS

In order to illustrate the potential of the proposed model, a data set of handwritten digits was used as a source of both training and test samples. Handwritten digits were inputted to the computer (SUN workstation) by a Photron FIOS-6440 LCD tablet, which samples at the rate of 80 dots per

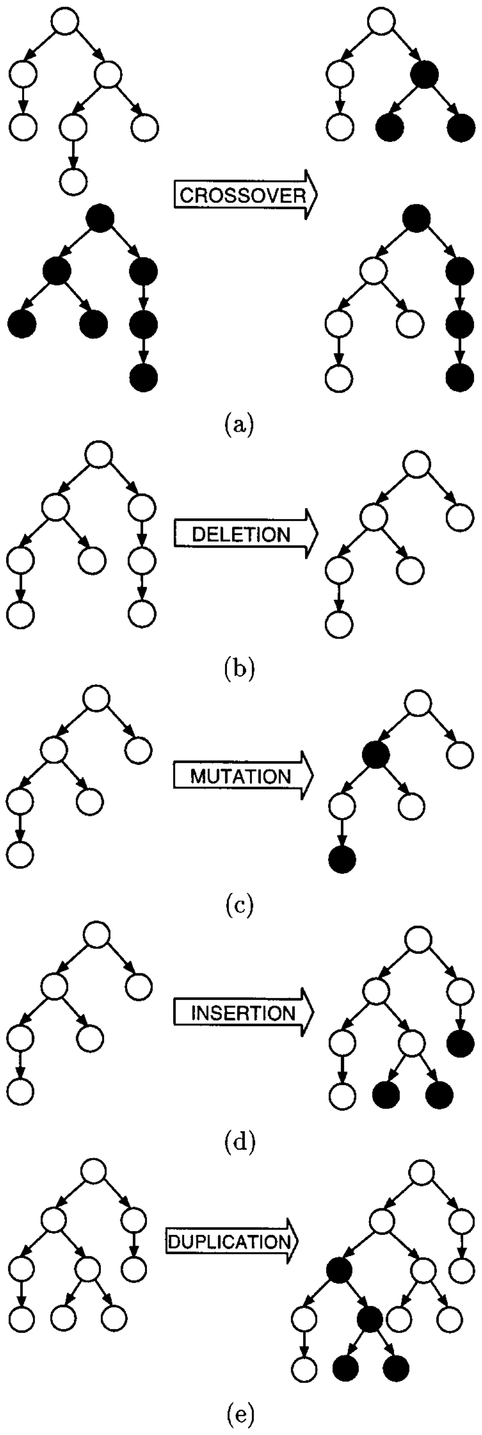


Figure 4. Graphical explanation of the genetic operators used.

second. A sample of the task was initially obtained by having different writers draw 200 digits within prepared square boxes in order to facilitate segmentation.

The size of a pattern was normalized by fitting a coarse 10×10 grid over each digit. The proportion of blackness in each square of the grid provided 100 continuous activation values for each pattern. Network architectures generated by the evolutionary mechanism were trained with 100 patterns in two rounds of subsequent presentations. A single presentation lasted for 60 cycles (i.e., iterative updates of all activations and learning weights). A fitness value was assigned to a solution by testing the generalization performance of a trained network with the untrained half of the 200 digits.

The initial population consisted of 100 neural networks having random connections. Each network contains one input module of size 100, one output module of size 10, and different numbers of hidden modules. The size in a module means the number of R-V pairs, and every module can be connected to every other module. The evolution parameters used in this experiment are as follows: crossover is 0.5, mutation is 0.02, and insertion and deletion are 0.001, respectively.

After making the population evolve, 10 networks were selected, and all of the networks produced a recognition rate higher than 92%. Figure 5 shows the best and average fitness changes in the course of evolution. As the figure depicts, clearly the performance increases with increased generations, and after the initial radical improvements the overall fitness stabilizes. Nearly all the best networks present in the population after a few number of generations scored

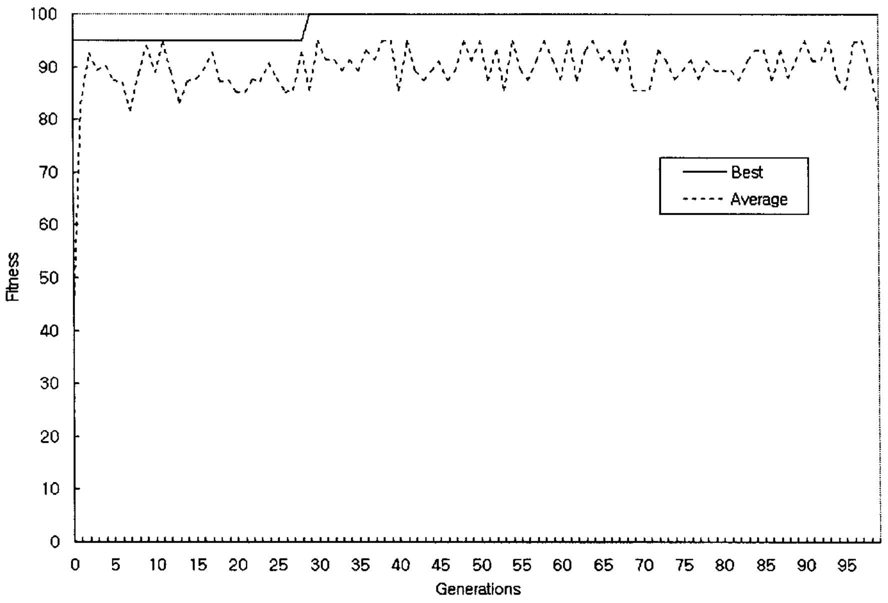
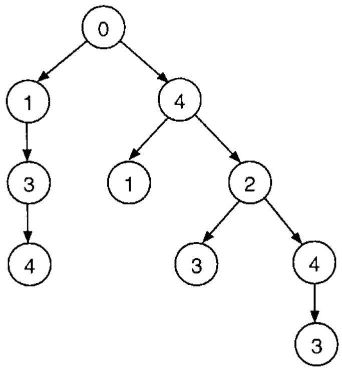


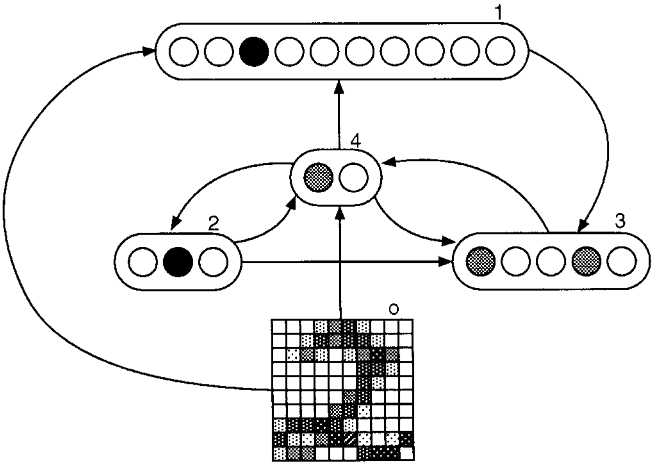
Figure 5. Best and average fitness changes with increased generations.

100% correct recognition rate for the training set and over 90% for the test set. However, the generalization performance was gradually improved with additional training.

Figure 6(b) shows a final network architecture producing the best result, and Figure 6(a) depicts the corresponding genetic code, which contains three hidden modules of sizes 2, 3, and 5, implementing different subsystems that cooperatively process input at different resolutions. The direct connection from the input module to the output module forms the most fine-grained processing stream. It is supplemented by a sophisticated modular structure that is globally connected with the input and occupies two coarser processing streams as well as local feedback projections.



(a)



(b)

Figure 6. (a) Genetic code encoded by the network producing the best result. (b) The corresponding modular neural network architecture developed by the chromosome of diagram (a).

Table I. A series of snapshots of the internal activations. The numbers in each column represent the activated nodes and the asterisk ($*$) means that there is no node activated in the module.

Step	Module 1	Module 2	Module 3	Module 4
1	0123456789	*	*	01
2	0123456789	012	01234	01
3	0234567	012	034	01
4	023467	01	034	01
5	0237	1	034	01
6	27	1	03	01
7	2	1	03	0

In the course of simulation, for the trained patterns and the patterns that are similar to the trained, the network produced direct activation through a specific pathway. On the contrary, the network oscillated among several pathways to obtain a consensus for strange patterns. The basic processing pathways in this case complemented each other to result in an improved overall categorization. Furthermore, the recurrent connections utilized bottom-up and top-down information that interactively influenced categorization in both directions. The oscillation stops when the whole network stabilizes as only one R node at the output module remains to be activated.

Table I shows a series of snapshots of the internal activations of the network (Fig. 6(b)) with an input pattern of class 2. This is not comparable to a practical pattern recognizer, but we can assure that the proposed evolutionary neural network works. Especially, we can appreciate that the effectiveness of evolution for designing complex structures with some sophisticated network architectures resulted from the evolution process.

5. CONCLUDING REMARKS

We have presented a preliminary design of a modular neural network built by an evolutionary algorithm, rich in flexibility, adaptability to environmental changes, and creativity. It has a modular structure with intramodular competition and intermodular excitatory connections. This sort of network will also assume an important part in several engineering tasks exhibiting adaptive behavior. We are currently endeavoring to apply this method to design a control system for behavior based robots, especially the Khepera-type of robots.[22]

The author thanks Dr. Y. Tohkura and Dr. K. Shimohara at ATR HIP Laboratories for continuous encouragement. This work was supported in part by Grant 961-0901-009-2 from the Korea Science and Engineering Foundation and by Grant SC-13 from the Ministry of Science and Technology in Korea.

References

1. B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
2. S.-B. Cho and J.H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. Syst. Man, Cybernetics*, **25**, 380–384 (1995).
3. R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, **3**, 79–87 (1991).
4. J.B. Hampshire, II, and A. Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Trans. Pattern Anal. Machine Intell.* **14**, 751–769 (1992).
5. S.A. Harp, "Towards the genetic synthesis of neural networks," *Proc. Genetic Algorithms*, 1989, pp. 360–369.
6. D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search," *Proc. Genetic Algorithms*, 1989, pp. 391–396.
7. H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Syst.*, **4**, 461–476 (1990).
8. D.T. Cliff, I. Harvey, and P. Husbands, *Incremental Evolution of Neural Network Architectures for Adaptive Behavior*, Tech. Rep. CSRP 256, School of Cognitive and Computing Science, Univ. of Sussex, 1992.
9. S. Nolfi, O. Miglino, and D. Parisi, *Phenotypic Plasticity in Evolving Neural Networks: Evolving the Control System for an Autonomous Agent*, Tech. Rep. PCIA-94-04, Institute of Psychology, C.N.R., Rome, 1994.
10. S.-B. Cho and K. Shimohara, "Toward evolvable model of modularized neural networks," *Proc. Fifth Annual Conf. Japanese Neural Network Society*, Tsukuba, November 1994, pp. 117–118.
11. S.W. Wilson, "The genetic algorithm and simulated evolution," in *Artificial Life*, C.G. Langton, Ed., Addison-Wesley, Reading, MA, 1989, pp. 157–166.
12. F. Dellaert and R.D. Beer, "Toward an evolvable model of development for autonomous agent synthesis," in *Artificial Life*, P. Maes and R. Brooks, Eds., MIT Press, Cambridge, MA, 1994, pp. 246–257.
13. S. Nolfi and D. Parisi, *Growing Neural Networks*, Tech. Rep. PCIA-91-15, Institute of Psychology, C.N.R., Rome, 1991.
14. M.J.M. deBoer, F.D. Fracchia, and P. Prusinkiewicz, "Analysis and simulation of the development of cellular layers," in *Artificial Life II*, C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, Reading, MA, 1992, pp. 465–483.
15. E. Mjolsness, D.H. Sharp, and J. Reinitz, "A connectionist model of development," *J. Theor. Biol.*, **152**, 429–453 (1991).
16. K. Fleischer and A.H. Barr, "A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis," in *Artificial Life III*, C.G. Langton, Ed., Addison-Wesley, Reading, MA, 1994, pp. 389–416.
17. K. Shimohara, "Evolutionary systems for brain communications—Towards an artificial brain," in *Artificial Life IV*, R. Brooks and P. Maes, Eds., MIT Press, Cambridge, MA, 1994.
18. J.M.J. Murre, R.H. Phaf, and G. Wolters, "CALM: Categorizing and learning module," *Neural Networks*, **5**, 55–82 (1992).
19. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
20. J.R. Koza, *Genetic Programming on the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.
21. K. Wada and S. Tanaka, "Can GAs survive?," *J. Soc. Instrum. Control Eng.*, **32**, 17–23 (1993).
22. K-Team, *Khepera Users Manual*, Laboratoire de Microinformatique, Swiss Federal Institute of Technology (EPFL), Applied AI Systems, 1993.