Dual Network Representation Applied to the Evolution of Neural Controllers

João Carlos Figueira Pujol and Riccardo Poli

School of Computer Science
The University of Birmingham
Birmingham B15 2TT, UK
E-MAIL: {J.Pujol,R.Poli}@cs.bham.ac.uk

Abstract. This paper presents a new approach to the evolution of neural networks. A linear chromosome combined with a grid-based representation of the network and a new crossover operator allow the evolution of the architecture and the weights simultaneously. There is no need for a separate weight optimization procedure and networks with more than one type of activation function can be evolved. This paper describes the representation, the crossover operator, and reports on results of the application of the method to evolve a neural controller for the pole-balancing problem.

1 Introduction

The reliable, general purpose, automatic design of neural networks (NNs) is still a largely unsolved problem. Recently, new promising approaches based on evolutionary algorithms, such as evolutionary programming (EP) [1] and genetic algorithms (GAs) [2], have been applied to the development of artificial neural networks. Approaches based on EP rely exclusively on mutation and operate on the neural network directly [3, 4, 5]. Although it has been advocated that EP is a more suitable approach to the development of neural networks [3], we think that crossover is an inportant mechanism provided by GAs for the exploitation of information on different regions of the search space. Methods based on genetic algorithms usually represent the structure and the weights of NNs as a string of bits or as a combination of bits and real numbers [6, 7], and perform the crossover operation as if the network were a linear structure. However, neural networks cannot naturally be represented as binary vectors. They are oriented graphs, whose nodes are neurons and whose arcs are synaptic connections. Therefore, it is arguable that any efficient approach to evolve NNs should use operators based on this structure.

Some recent work based on genetic programming (GP) [8] is a first step in this direction. For example, in [8, 9] neural networks have been represented as parse trees which are recombined using a crossover operator which swaps subtrees representing subnetworks. However, the graph-like structure of neural networks is not ideally represented directly with parse trees either. Indeed, an alternative approach based on GP, known as cellular encoding [10], has recognized this and used an indirect network representation in which parse trees represent rules which grow a complete network from a tiny neural embryo. Although very compact, this representation enforces a considerable bias on the network architectures that can be achieved.

Recently, a new form of GP, PDGP [11], in which programs are represented as graphs instead of parse trees, has been applied to the evolution of neural networks [12].

The method allows the use of more than one activation function in the neural network but it does not include any operators specialized in handling the connection weights. Nonetheless the results were encouraging and led us to believe that a proper representation, with specialized operators acting on meaningful building blocks, could be used to efficiently evolve neural networks. Indeed, in [13] we improved and specialized PDGP by introducing a dual representation, where a linear description of the network was dynamically associated to a two-dimensional grid. Several specialized genetic operators, some using the linear description, others using the grid, allowed the evolution of the topology and the weights of moderately complex neural networks very efficiently.

In this paper, we propose a new combined crossover operator which allows the determination of the architecture, the activation function, and the weights of a neural network concurrently and very efficiently. In the following sections, our representation and operator are described, and the results of the application of this paradigm to the development of a neural controller for the pole-balancing problem are presented.

2 Representation

In PDGP, instead of the usual parse tree representation used in GP, a graph representation is used, where the functions and terminals are allocated in a two-dimensional grid of fixed size and shape. The grid is particularly useful to solve problems whose solutions are graphs with a natural layered structure, like neural networks.

However, this representation can make inefficient use of the available memory if the grid has the same shape for all individuals in the population. In fact, in this case there is no need to represent the grid explicitly in each individual. Also, in some cases, it is important to be able to abstract from the physical arrangement of neurons into layers and to only consider the topological properties of the net. To do this, it is more natural to use a linear genotype.

These limitations led us to propose a dual representation [13], in which a linear chromosome is converted, when needed, into the grid-based PDGP representation. The dual representation includes every detail necessary to build and use the network: connectivity, weights and the activation function of each neuron. A chromosome is an ordered list of nodes (see Figure 1a). An index indicates the position of the nodes in the chromosome. All chromosomes have the same number of nodes.

Like in standard genetic programming, the nodes are of two kinds: functions and terminals. The functions represent the neurons of the neural network, and the terminals are the variables containing the input to the network (see Figure 1b).

When the node is a neuron, it includes the activation function and the bias, as well as the indexes of other nodes sending signals to the neuron and the weights necessary for the computation of its output. Multiple connections from the same node are allowed. Nodes are evaluated according to their order in the chromosome.

When necessary (see section 3), the linear representation just described is transformed into the two-dimensional representation used in PDGP. A description table defines the number of layers and the number of nodes per layer of the two-dimensional representation (see Figures 1c and 1d). The nodes of the linear chromosome are mapped onto this representation according to the description table. The connections of the network are indicated by links between nodes in the two-dimensional representation. The description table is a characteristic of the population and it is not included in the genotype. The two-dimensional representation may have any number of layers, each of different size. This feature may be used to constrain the geometry of the network (e.g.

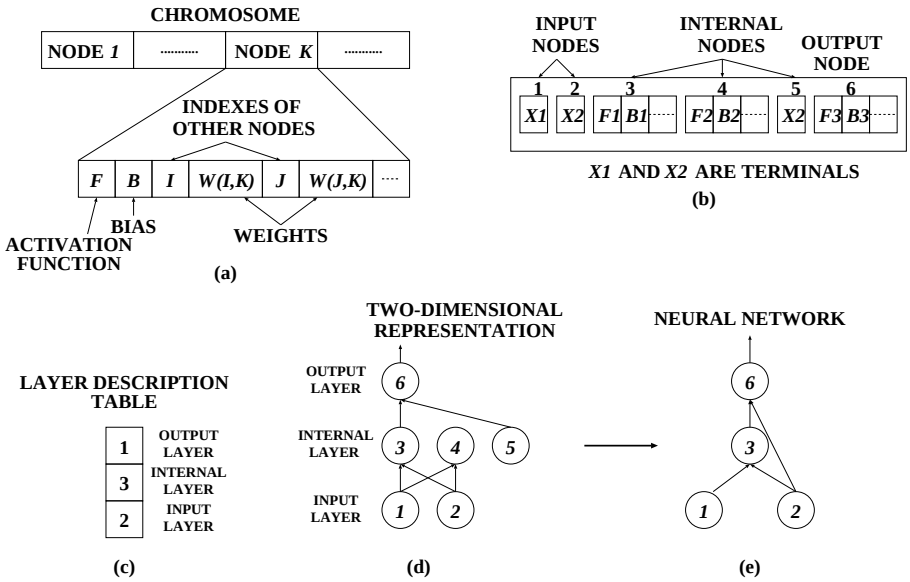

Fig. 1. (a) Chromosome and node description. (b) Example of a chromosome. (c) The table describing the number of nodes per layer of the network. (d) The two-dimensional representation resulting from the mapping of the nodes of the chromosome in (b), according to the description table in (c) (note the intron in the internal layer). (e) Resulting neural network after elimination of introns and transfer of connection from terminals in the internal layer to corresponding terminals in the first layer (node 5 is the same terminal as node 2).

to obtain encoder/decoder nets) and can also be used to guide specialized crossover operators [13].

The nodes in the first layer are terminals, whereas the nodes in the last layer represent the output neurons of the network. The number of input and output nodes depends on the problem to be solved. The remaining nodes, called *internal nodes*, constitute the internal layer(s), and they may be either neurons or terminals.

Although the size of the chromosome is fixed for the entire population, the neural networks represented may have different sizes. This happens because terminals may be present as internal nodes from the beginning, or may be introduced by crossover and mutation (this is discussed in section 3). They are removed from the network during the decoding phase, which is performed before each individual is evaluated. Connections from the removed terminals are replaced with connections from corresponding terminals in the input layer (see Figures 1d and 1e). It must be also pointed out that most methods work with a single pool of hidden neurons without structuring them into more than one layer. For most applications this is sufficient. However, if necessary, our method provides a natural way of structuring the hidden neurons into more than one layer.

Our model allows the use of more than one activation function, so that a suitable combination of activation functions can be evolved to solve a particular problem (this is discussed in the next section).

3 Crossover

The crossover operator proposed in this paper works by randomly selecting a node *a* in the first parent and a node *b* in the second parent (see Figure 2a), and by replacing node *a* in a copy of the first parent (the offspring). Depending on the types of node *a* and node *b*, the replacement of node *a* is carried out as follows:

Both nodes are terminals This is the simplest case, node *b* replaces node *a*, and there is no change either in the topology or in the weights of the network.

Node *b* is a terminal and node *a* is a function In this case, node *b* also replaces node *a*, but the complexity of the network is reduced, because a neuron is removed from the network.

Node *b* is a function and node *a* is a terminal In this situation, the crossover operation increases the complexity of the network. A temporary node, *c*, is created as a copy of node *b*. Before node *c* replaces node *a* in the offspring, each of its connections is analyzed and possibly modified, depending on whether they are connections from terminals or functions:

- If the connection is from a function, the index of the connected node is not modified.
- If the connection is from a terminal, the index is modified to point to another node, as if the connection had been rigidly translated from node *b* to node *a*. For example, the connection between node 10 and node *b* in Figures 2a and 2b is transformed into a connection between node 8 and node *c* in Figures 2c and 2d. In some cases, translating the connection rigidly might lead to point to an nonexistent node outside the limits of the layer. In this case, the index of connected node is modified as if the connection had been wrapped around the layer. For instance, the connection between node 7 and node *b* in Figures 2a and 2b is transformed into a connection between node 1 and node *c* in Figures 2c and 2d. If the rigid translation of the connection requires the presence of a node below the input layer, the connection is deleted.

Sometimes, a connection in node *c*, modified or not, generates feedback loops in the offspring. In this case, as we are interested in feedforward networks, the connection is deleted. For example, the connection between node 13 and node *b* in Figure 2b was deleted when node *c* was created.

This procedure for connection inheritance aims at preserving as much as possible the information present in the connections and weights.

Both nodes are functions This is the most important case. By combining the description of two functions, the topology and the weights of the network can be changed. After creating node *c* as described above, its description and the description of node *a* are combined by selecting two random crossover points, one in each node, and by replacing the connections to the right of the crossover point in node *a* with those to the right of the crossover point in node *c*. This creates a new node to replace node *a* in the offspring. See Figure 3.

This process can easily create multiple connections between the same two nodes. These are very important because their net effect is a fine tuning of the connection strength between two nodes. However, as this may reduce the efficiency of the search, we only allow a prefixed maximum number of multiple connections. If more than the allowed maximum number of multiple connections are created, some of them are deleted before the replacement of node *a* in the offspring.

Modification of the activation function and bias of a node is not performed with our crossover operator. However, this can be indirectly accomplished by replacing a function with a terminal, which can later be replaced with a function with different features.

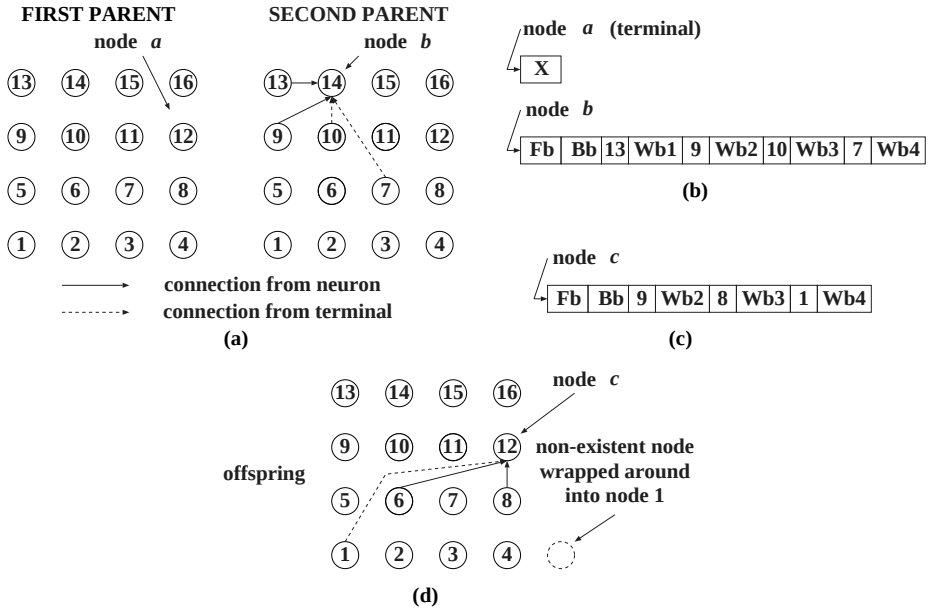

Fig. 2. (a) Two-dimensional representation of the parents. For clarity, only connections relevant to the operation are shown. (b) Nodes *a* and *b*. (c) Node *c* is a copy of node *b* with modified connections. The connections of node *b* whose indexes indicated connections from terminals received new indexes. The connection from node 13 was deleted. (d) Offspring generated by replacing node *a* with node *c*.

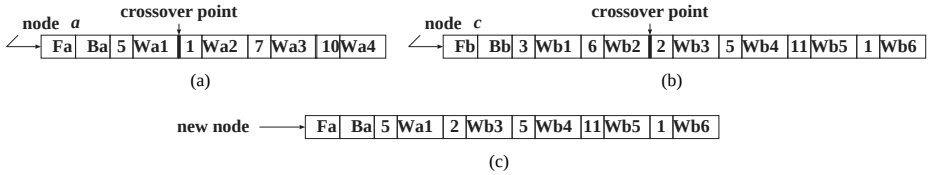

Fig. 3. Combination of two functions. (a) Node *a*. (b) Node *c* . (c) New node created to replace node *a* in the offspring. Note the multiple connection created.

4 Experimental results

To assess the performance of the method proposed, it was applied to the pole-balancing problem. This problem is a well-studied benchmark for control methods [7, 14, 15]. The task consists of balancing a pole hinged in the center of a moving cart, by applying a force to the cart exclusively. The pole is only allowed to move in a vertical plane and the cart moves in a one-dimensional track.

The only forces acting on the system are a control force applied to the cart and gravity. The controller must provide a sequence of left and right pushes to the cart, in order to keep the system within specified limits. Different strategies must be compared exclusively on their net result: failure or success. There is no way of knowing in advance which individual action will contribute to a successful or unsuccessful control strategy in the end.

The state of the cart-pole system is defined by four variables representing the position and velocity of the cart, and the angle of the pole to the vertical and its angular velocity. The equations of motion were numerically integrated to give the new state of the system at each time step (we used Euler's method with a time step of 0.02 s). A failure signal was triggered when the angle of the pole or the position of the cart exceeded specified limits. The limit for the cart position was ± 2.4 m, and the limit for the angle of the pole varied according to the experiment (details are given further on).

At each generation, individuals in the population were evaluated by the number of time steps in which they were able to keep the pole balanced and the cart in the track. At each time step, the neural network received as input the four state variables of the cart-pole system and returned as output the force to be applied to the cart. Before being introduced as input to the network, the state variables were normalized using the same normalization factors reported in [14].

In all experiments a population of 100 individuals was evolved for a maximum of 100 generations using two different topologies: 4-10-1 and 4-5-5-1. The weights and biases were randomly initialized within the range [-1.0, +1.0] and the hyperbolic tangent was used as activation function.

In a first set of experiments, we used the control approach reported in [14, 15], where cellular encoding has been applied to the pole-balancing problem. Two different control actions were investigated: a bang-bang and a continuous control force. In the first case, the force f can only take two discrete values: +10 or -10. In the second case, it can take any value in the range [-10.0,+10.0].

The fitness of an individual is given by the number of time steps in which the system is controlled over different initial states. We used the same initial states reported in [14]. A solution is a network which can control the system over all initial states for 1,000 time steps. The limit for the angle of the pole was set at 36 degrees.

Table 1 shows the results of 50 independent runs. Column 1 indicates the shape of the two-dimensional representation used. Column 2 represents the average number of generations to obtain a solution. Column 3 and 4 show the number of hidden neurons (all internal layers included) and the number of connections of the neural network, respectively. Column 5 shows the computational effort, i.e. the number of fitness evaluations necessary to obtain a solution with 99% probability [8].

There was no noticeable difference between the 4-10-1 and the 4-5-5-1 topologies, although the effort to obtain a solution was slightly smaller with the first one. A typical solution found with the bang-bang control action is shown in Figure 4.

In order to compare our results with those reported in [14, 15], a generalization test was performed for all solutions found, by counting the number of successful control

runs of the cart-pole system for 1,000 time steps. Each solution was required to control the system from 625 different initial states, defined by each normalized state variable assuming the values: ± 0.9, ± 0.5 and 0.

The results are summarized in Tables 1a and b. The second column represents the number of fitness evaluations to achieve a solution. Columns 3, 4 and 5 show the number of successful control runs of the best solution (B), of all solutions (M), and of the worst solution (W), respectively. Column 6 represents the standard deviation (SD) of the generalization test.

In both cases our method considerably outperformed cellular encoding in terms of number of fitness evaluations to achieve a solution. The results are even more impressive if one considers the size of the populations used in the cellular encoding experiments: 2048 in [14] and 4096 in [15]. In [14], the number of individuals in the population is almost of the same size as the number of fitness evaluations to achieve a solution for the bang-bang case, which means that a good approximation was already present in the initial population. In [15], the size of the population is considerably greater than the number of evaluations to achieve a solution in the bang-bang case, which means that the solution was already present in the initial population. It may be also pointed out that with both control actions, the number of fitness evaluations to achieve a solution with our approach are of the same order of magnitude. This suggests that our method scales up better than cellular encoding with the difficulty of the problem.

The generalization power of the solutions we obtained is superior to those reported in [15], but inferior to those reported in [14]. However, we believe that by increasing the size of the population, and by allowing the population to evolve to the maximum number of generations (in the current experiments, the evolution has been interrupted as soon as a solution has been found), better generalization could be achieved.

Table 1. Summary of results obtained in the first set of experiments. Values represent average over 50 random runs.

Topology	Gen	Neurons min/avg/max	Connections min/avg/max	Effort
4-10-1 (bang-bang)	4	0/1.0/5	4/8.0/26	1,100
4-5-5-1 (bang-bang)	4	0/1.3/4	4/9.0/22	1,300
4-10-1 (continuous)	7	0/1.0/3	3/7.7/16	2,600
4-5-5-1 (continuous)	8	0/1.4/6	4/10.0/34	2,700

Table 2. Results obtained with our method and with cellular encoding. (a) Using a bang-bang control action. (b) Using a continuous control action.

Learning stage		Successful runs			
Method	Eval	B	M	W	SD
Cellular[14]	2,234	N/A	430	N/A	N/A
Cellular[15]	1,400	N/A	250	N/A	N/A
4-10-1	367	377	321	189	45
4-5-5-1	392	381	339	228	33

(a)

Learning stage		Successful runs			
Method	Eval	B	M	W	SD
Cellular[14]	19,011	N/A	386	N/A	N/A
Cellular[15]	21,000	N/A	225	N/A	N/A
4-10-1	600	379	325	229	36
4-5-5-1	690	368	328	212	30

(b)

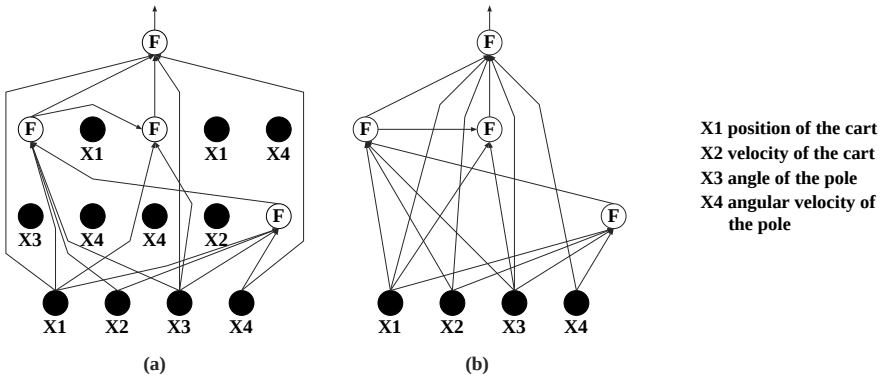

Fig. 4. (a) Two-dimensional representation of a typical solution found using a 4-5-5-1 grid. (b) Corresponding neural network.

In a second set of experiments, we used the approach suggested by Whitley et al. [7], where the cart is initially in the center of the track with zero velocity, the pole is vertically aligned with zero angular velocity, and an individual is considered to be a solution to the problem if it can keep the system within the predefined limits for 120,000 time steps.

A bang-bang control action based on a probabilistic interpretation of the output of the network was used, in which an output of 0.75 does not mean necessarily a push to the right, but rather that this action has a probability of occurrence of 75%. As pointed out in [7] and as we also observed in our experiments, this procedure allows the network to explore more of the state space.

In order to compare our results with those reported in [7], we carried out experiments with a failure signal at two different limits for the angle of the pole: 12 and 35 degrees. The results are shown in Table 3.

As in the first set of experiments, the 4-10-1 topology was slightly better than the 4-5-5-1 one in terms of effort and number of generations to find a solution. A typical solution found for the 12 degrees case is shown in Figure 5.

Table 3. Summary of results obtained in the second set of experiments. In the first column, the specified maximum angle of the pole is indicated in brackets. Values represent average over 50 random runs.

Topology	Gen	Neurons min/avg/max	Connections min/avg/max	Effort
4-10-1 (12)	5	0/1.0/3	4/7.9/16	2,000
4-5-5-1 (12)	7	0/1.0/3	4/7.8/19	2,600
4-10-1 (35)	4	0/0.8/4	4/7.0/14	1,100
4-5-5-1 (35)	4	0/1.1/4	4/8.5/22	1,600

Table 4. Results obtained with our method and those reported by Whitley et al. [8]. (a) 12^o case. (b) 35^o case.

Learning stage		successful runs			
Method	Eval	B	M	W	SD
Whitley et al.	4,097	446	297	24	89
4-10-1	489	524	372	156	90
4-5-5-1	584	508	345	141	94

(a)

Learning stage		successful runs			
Method	Eval	B	M	W	SD
Whitley et al.	4,206	430	304	92	92
4-10-1	359	539	426	235	80
4-5-5-1	353	529	432	204	77

(b)

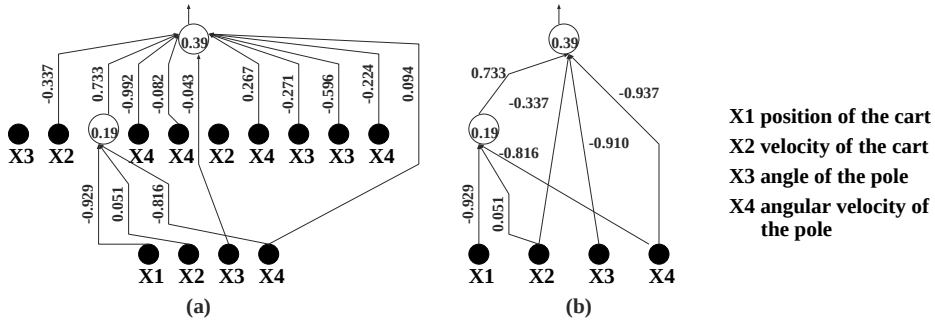

Fig. 5. (a) Two-dimensional representation of a typical solution found using a 4-10-1 topology. (b) Corresponding neural network.

To verify the generalization power of the solutions found in the second set of experiments, each of them was required to balance the system starting from a set of 625 random states (selected within the normalization factors range) for 1,000 time steps. The results of these experiments are summarized in Tables 4a and b.

Our results are substantially better than those reported in [7] in terms of number of fitness evaluations to achieve a solution. The results are even more impressive if one considers that Whitley and collaborators did not evolve the architecture, only the weights. The generalization power of the solutions we attained is also better.

5 Summary

In this paper, a new approach to the automatic design of neural networks has been presented, which makes natural use of their graph structure. The approach is based on a dual representation and a new crossover operator. The method was applied to the design of feedforward networks in a control task showing promising results. In future research, we intend to extend the power of the representation, and to apply it to a wider range of practical problems.

Acknowledgements

The authors wish to thank the members of the EEBIC (Evolutionary and Emergent Behavior Intelligence and Computation) group for useful discussions and comments. This research is partially supported by a grant under the British Council-MURST/CRUI agreement and by CNPq (Brazil).

References

1. D. Fogel. *Evolutionary computation: toward a new philosophy of machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.
2. D. Goldberg. *Genetic algorithm in search, optimization and machine learning.* Addison-Wesley, Reading, Massachusets, 1989.
3. P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1), 1994.
4. D. Fogel. Using evolutionary programming to create neural networks that are capable of playing Tic Tac Toe. In *IEEE International Conference on Neural Networks (ICNN)*, pages 875–880. IEEE Press, 1993.
5. J. McDonnell and D. Waagen. Neural network structure design by evolutionary programming. In D. Fogel and W. Atmar, editors, *Proceedings of the Sec. Annual Conference on Evolutionary Programming*, pages 79–89, La Jolla, CA, USA, Feb. 1993. Evolutionary Programming Society.
6. V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1994.
7. D. Whitley, S. Dominic, R. Das, and C. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284, 1993.
8. J. R. Koza. *Genetic Programming, on the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusets, 1992.
9. B. Zhang and H. Muehlenbein. Genetic programming of minimal neural nets using Occam's razor. In S. Forrest, editor, *Proceedings of the 5th international conference on genetic algorithms (ICGA'93)*, pages 342–349. Morgan Kaufmann, 1993.
10. F. Gruau. *Neural network synthesis using cellular encoding and the genetic algorithm.* PhD thesis, Laboratoire de L'informatique du Parallélisme, Ecole Normale Supériere de Lyon, Lyon, France, 1994.
11. R. Poli. Some steps towards a form of parallel distributed genetic programming. In *Proceedings of the First On-line Workshop on Soft Computing*, pages 290–295, Aug. 1996.
12. R. Poli. Discovery of symbolic, neuron-symbolic and neural networks with parallel distributed genetic programming. In *3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA)*, 1997.
13. J. C. F. Pujol and R. Poli. Evolution of the topology and the weights of neural networks using genetic programming with a dual representation. Technical report CSRP-97-07, The University of Birmingham, School of Computer Science, 1997.
14. D. Whitley, F. Gruau, and L. Pyeatt. Cellular encoding applied to neurocontrol. In *Proceedings of 6th International Conference on Genetic Algorithms*, pages 460–467. Morgan-kaufmann, 1995.
15. F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. Koza, D. Goldberg, D. Fogel, and R. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, Stanford University, CA, USA, Jul. 1996. MIT Press.