Genetic Algorithm for Artificial Neurogenesis

Manuel Clergue, Philippe Collard

Abstract— **This paper presents a neurogenesis process based on the protein regulation system. The novelty consists in applying some genetic algorithm work with neurogenesis, in particular work on genetic operators and work on fitness landscapes. In order to get relevant tests with reduced simulation costs, neurogenesis is applied to a well know problem, the cart-pole system, then results are commented.**

I. Introduction

For years, evolutionary methods have been used in combination with neural networks. In older approaches, the network structure remains fixed and the connection weights are just evolved, with a genetic algorithm e.g., performing a kind of evolutionary learning ([1], [2]). There is a strong limitation to generalize as the number of neurons and the structure of the network are set with an *a priori* knowledge on the problem. Moreover, there is no general method for determining the structure of a neural network fitted to a particular problem. There are only problem dependent heuristics. To fill this gap, some proposed encoding the whole network in the chromosome. Two approaches have been considered. The first one is direct encoding. The chromosome contains information about the network structure, such as the number of neurons and the way they are connected. There is a strong limitation in this case too. Indeed, this kind of method exhibits a lack of modularity, thus difficulties increase as the size of target networks increases. With the second approach, the chromosome contains instructions to generate the network. Now, the network is initialized with an initiator, and is constructed step by step with the instructions contained within the chromosome. These approaches recall the way living bodies grow from the original cell and so, are often called morphogenesis, and sometime neurogenesis, since this is neural material that is considered.

Apart from the fact that they are inspired by Nature, these approaches are characterized by some interesting features. There is an obvious modularity. So, the information needed to grow neural networks do not need to increase with the size of target neural networks. Moreover, any kind of network may be created by this way. Obviously, this depends upon the expressiveness of the morphogenesis process, but there is no theoretical reason that a network could not be generated by this way.

I3S Laboratory, 250 av. A. Einstein, F-06560 VALBONNE, FRANCE, E-mails : {clerguem,pc}@alto.unice.fr

And finally, a level is added where auto-organization is introduced. The process that makes a network from a chromosome is no longer a simple mapping. This is a dynamic process, where chaos can introduce itself.

This approach is now well known, and much work has been done (see [3] for a resume of research in neurogenesis). The morphogenesis process may make use of production rules at the neuron level, such as *Gruau* who uses a grammar tree as rewriting rules for neurons ([4]). It may be implemented on a cellular automata ([5]). Finally, it can make use of production rules for cellular elements such as proteins ([6], [7], [8]). It is with this approach that we address the evolution of neural network.

II. Artificial Neurogenesis

The method we developed is inspired by the protein synthesis regulation process ([9]). The process is modeled this way : a protein is created by a gene; this protein can excite the same or another gene, creating the same or another protein; the protein can inhibit a gene, preventing the creation of another protein; finally, the protein can perform different operations on the host cell, such as splitting or connection. Obviously, this model is somewhat far removed from reality. But it is sufficient for the purpose, that is to get a process allowing growth of neural networks.

In this section we explain the mechanism of neural networks growth. The first point of the section explains how the mechanism of neural networks growth by protein synthesis works. The second point talks about the cellular topology. Finally, we show how information are coded on the chromosome.

A. Protein Synthesis

In our system, there are 128 different possible proteins. Some of them have a particular function in the cell. Table I lists them. There are two sorts of proteins, apart those without significance. The action ones perform an action on the cell, such as splitting it or connecting it to another cell. Some of these actions need parameters. There are directional parameters, which indicate in which direction the action should be executed. Note that there is a directional parameter for the cell itself, in order to allow self connected neurons. In the case of the connect-

TABLE I

Different proteins and their function in the cell

action proteins			
#	name	function	parameters
26	F_SPLIT	cell splitting	direction
27	F_CONNECT	cell connection	direction & strength
28	F_PERMUT	cell permutation	direction
29	F_DIFFUSE	protein diffusion	direction
30	F_DIE	cell death	none

parameter proteins		
#	name	value
16	F_LEFT	to the left
17	F_RIGHT	to the right
18	F_MYSELF	to the cell itself
19	F_FI	-10.0
20	F_fI	-1.0
21	F_sI	-0.1
22	F_ZE	0.0
23	F_sA	0.1
25	F_fA	1.0
24	F_FA	10.0

ing action, there is also a strength parameter, which indicates the value of the connection. These parameters are set by parameter proteins. When an action protein is present, the action is performed with parameters corresponding to parameter proteins present at the time in the cell. For example, if the proteins `F_PERMUT` and `F_LEFT` are present at the same time in the cell, this one permutes with its left neighbor. The proteins {`F_CONNECT`,`F_LEFT`,`F_MYSELF`,`F_sA`,`F_fA`} imply that the cell starts a connection of weight $1.0+0.1 = 1.1$ with the left cell and itself.
We will not consider protein concentration. The only thing of interest is the presence or the absence of the protein. Once all the actions concerned with a protein are performed, the protein is removed from the cell.

B. Cellular topology

The cellular topology is modeled by a double linked list structure. Each cell has two neighbors, except the final ones which have only one neighbor. Connections between cells do not respect this topology since neighbor cells establishing a connection may be separated by a split or a permuting action.
The morphogenesis is started with a set of initial cells (corresponding to the different input and output neurons of the network). Each cell is initialized with an initial protein, which may be different according to the type of the cell. In the experiments we ran, there are five different initial proteins, one for the output cell, and four for the different types of input cell. Each cell contains a copy of the chromosome. At each time step, the actions for each cell,including protein synthesis activation or inhibition, is calculated according to the present proteins. These actions are executed in a parallel way and ambiguities are resolved by considering cells from left to right and actions the way they appear in table I.
At the end of the morphogenesis process, cells become neurons, and links between cells become synapses.

C. Genomic Network

Genomic networks are oriented graphs, where a node represents a protein and an arc represents the influence of a protein over the synthesis of another one. Figure 1 represents such a graph.

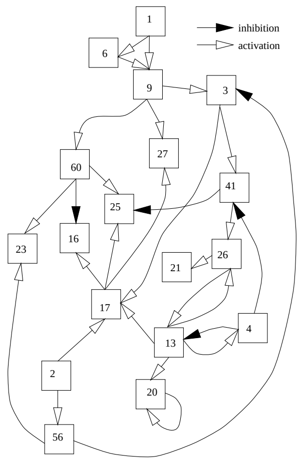

Fig. 1. A genomic graph

Genomic graphs are coded in chromosomes by a direct encoding. Chromosomes only contain information about link. A link is coded by a gene which look like `X-(A)>Y`, where `X` is the source of the link, `Y` is the destination and `A` expresses whether the link is an activation or an inhibition. For example, if a chromosome contains the gene `9-(1)>27`, so the corresponding genomic graph has an activation link from the protein 9 to the protein 27.
For example, the graph shown in figure 1 is coded by:

```
13-1>17 17-1>27 60-0>16 60-1>25 60-1>23 17-1>16
90-1>7   4-0>13  9-1>3  13-1>4   1-1>9   9-1>60
56-1>23 13-1>26 17-1>25  3-1>41 26-1>21 20-1>20
41-1>26 13-1>20  2-1>17  1-1>6  41-0>25 26-1>13
 9-1>27 56-0>3   2-1>56  6-1>9   3-1>17  4-0>41
```

A chromosome may be of any length, that is, it may have any number of genes. The position of a gene in the chromosome does not matter. Since there are 128 possible source proteins, 128 possible destination proteins, and 2 possible link types, the number of rules is $2 \times 128 \times 128 = 2^{15}$, so the number of different chromosomes, i.e. the size of the searched space, is $2^{2^{15}}$.
Since length is variable, the chromosomes defined here are very different from those used with classic genetic algorithms. For this reason we needed to define new genetic operators to match their structure.
The crossover operators need to have a property defined by *Radcliff* and termed *respect* ([10]). This prop-

erty says that all children must inherit all the common genes of their parents. Figure 2 shows the way a crossover verifying this property for variable length chromosomes works : the common genes (here `R1`) are first copied in the children; then, a site is randomly chosen on chromosomes; finally, the parts after the sites are just swapped (here `R127` and `R58` are swapped).

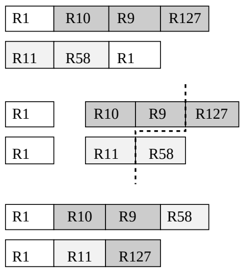

Fig. 2. *Respectful* crossover for variable length chromosomes.

Mutations are performed by adding random genes, deleting genes or modifying them. This mutation operator respects another *Radcliff* property, the *ergodicity*. Indeed, from any chromosome, any chromosome may be reached by a finite sequence of mutations.
The neural network produced from the genomic graph of figure 1 is drawn in figure 3. Its structure is quite complex. There are recurrent connections and a relative diversity in weight values.

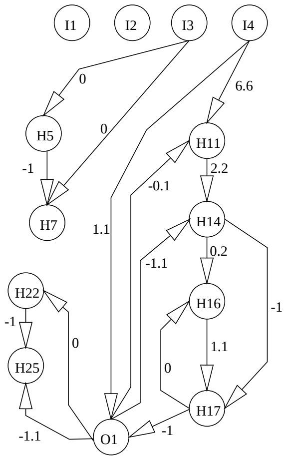

Fig. 3. Neural network

III. The Cart-Pole Problem

In order to validate and analyze our approach, we apply artificial neurogenesis to a well studied class of control problems: the cart-pole problem (also known as the inverted pendulum problem). This problem offers great possibilities with a reduced simulation cost. Moreover, the task the controller has to execute is simple and so easy to objectively evaluate. Finally, it is well referenced and has already been successfully addressed by many control methods, including neural networks.

A. Past Work

The cart-pole problem is a well known problem in control theory. Many exhaustive studies have been done in this context, and as it will be seen, an optimal solution is found in the case of complete information on the system. Neural Networks have, virtually since the beginning, addressed this problem. The ADALINE model was trained to balance poles using Widrow-Hoff LMS algorithm ([11]). Reinforcement learning methods have also addressed this problem ([12]).
The cart-pole system is still a standard research problem for neural networks. There are two recent examples, drawn from *Wieland* ([2]) and *Whitley and al.* ([13]), where a Genetic Algorithm is used to evolve Neural Networks to control the cart-pole system. The Wieland's method optimizes the weights of a fixed fully recurrent Neural Network, using a Genetic Algorithm. This method leads to very interesting results but is not entirely satisfying since the number of neurons and the structure of the network are fixed. *Whitley and al.* use the *Gruau*'s morphogenesis, that is *Cellular Encoding*, to generate neural networks. Again, the results obtained are very interesting. Efficient networks are obtained faster, and as the authors say, cellular encoding could automatically find small architectures whose structure and complexity fit the specificity of the problem. This is a strong argument for Neurogenesis methods.

B. Definition of the Problem

The system (see figure 4) is composed of a cart moving on a track. A pole is hinged on the cart. The controller's goal is to avoid the pole falling by exerting force on the left or on right of the cart, without moving it away from the limits of the track.

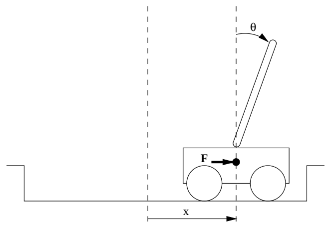

Fig. 4. Cart-pole system to be controlled

The system is easily simulated by differential equations (found in [14]). Simulations are done with the Euler's method. The Euler's method gives in general poor results in solving differential equations. But this is sufficient for our purposes, which is to validate our

approach. Obviously, if we had aimed to obtain controllers for real systems, we would have chosen a more accurate method. The advantage of Euler's method is its low computational cost.
The problem is quite simple to solve. When the controller is provided with four inputs, the position and the velocity of the cart and the pole, there is an optimal solution given by:

$$F = F_{max}\text{sgn}(k_1 x + k_2 \dot{x} + k_3 \theta + k_4 \dot{\theta})$$

where the coefficients k_i depend on the parameters of the system such as the masses and the friction coefficients. This is the equation of a neuron with four inputs. So, with a given learning method, it is possible to solve the problem with a single neuron neural network.
To increase the difficulty, the available state variables are reduced to positional ones, i.e. the cart position and the pole angle. Stable control of this system requires the controller to compute the velocities.
Note that contrary to *Wieland*, rather than a bang-bang control, we use a continuous control, where the force may vary in $[-F_{max}, F_{max}]$. This solution is proposed by *Whitley and al.*, and said to be more difficult.

IV. Experimentations

A. Experimental Settings

A.1 Neural Network

We directly addressed the problem with only position information, letting the network calculate by itself the velocities. So we need only two inputs (neurons `I3` and `I4` in figure 3): the cart position on the track and the angle of the pole from vertical, normalized to get values in $[-1.0, 1.0]$. There are two more inputs. The first one (neuron `I1`) is always set to 1.0 to allow biases on neurons. The second one (neuron `I2`) is always set to 0.0. As output, we get also a value in $[-1.0, 1.0]$. We multiply this value by F_{max} to get the force to be applied on the cart.

A.2 Genetic Algorithm

We use a classical steady state genetic algorithm ([15]), with a population size of 100 individuals. With this GA, only a part of the population, the worst individuals, is renewed each generation. In general, this leads to better results. The part of population being renewed is set to 80.0%. So, each generation, the worst 80.0% of the population are removed and replaced by children of selected individuals drawn from the 20.0% remaining. This strategy gives pretty good results compared to generational ones, where the entire population is renewed each generation. The selection method is rank-based with fitness sharing.

A.3 Evaluation protocol

The easier and more direct way to evaluate the network performances is to assign a fitness value proportional to the time it successfully controls the cart-pole system, that is, the time it keeps the cart in the track limits and the pole sufficiently near to the vertical to avoid an irremediable fall.
In order to get robust networks, we test them from 11 initial positions, shown on table II. This test bed was first proposed by *Whitley and al.*, and modified in order to have no stable initial states.

TABLE II
Initial states to test the cart-pole system

x	(m)	2.0	2.16	0.0	0.0	2.16	0.0
$\dot{x}$	(m/s)	0.0	0.0	0.0	0.0	0.0	0.0
θ	(deg.)	1.8	-18.0	9.0	18.0	-1.8	0.0
$\dot{\theta}$	(deg./sec)	0.0	0.0	21.5	0.0	-43.0	0.0
x	(m)	-2.0	-2.16	0.0	0.0	-2.16	0.0
$\dot{x}$	(m/s)	0.0	0.0	0.0	0.0	0.0	0.0
θ	(deg.)	-1.8	18.0	-9.0	-18.0	1.8	0.0
$\dot{\theta}$	(deg./sec)	0.0	0.0	-21.5	0.0	43.0	0.0

The fitness of an individual is the time the network successfully controls the cart-pole over all initial states. For each initial state, simulation stops after MAX_STEP steps of simulation or when the cart-pole leaves its viability domain. For the next experiments MAX_STEP is set to 1000.

B. Results

We ran four experiments with the above settings. On average, an overall best individual, i.e. with fitness equal to 11000, is found after 2507 generations. As we used a steady state strategy, the learning time is about 200560 fitness evaluations. These results are very similar to those obtained by *Whitley and al.* with the *Gruau*'s method ([13]). In figures 5 and 6, a typical run is shown. The first figure traces fitness vs generations. It can be seen that at the beginning of the evolution, there are a lot of discoveries. At the end, the improvements are few and far between. Figure 6 shows the mean chromosome size vs generations. It is worth noticing that there are absolutely no correlation between fitness and chromosome size. Higher fitnesses do not involve longer chromosomes. At the end of the experiments, where the individuals are the most efficient, the chromosome size comes back to about its initial value. Moreover the best individual at the end of the run has a chromosome size of 30 genes, while having a fitness of 11000.
The corresponding genomic graph and neural network are shown figures in 1 and 3. The functional part of the neural network is the neurons {`I4`,`H11`,`H14`,`H16`,`H17`,`O1`}. There is no control on the cart position (the neuron `I3` is not connected to the

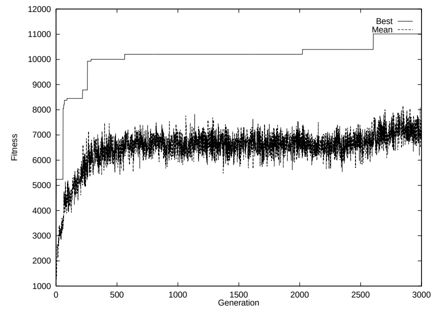

Fig. 5. Fitness vs Generations. A typical run

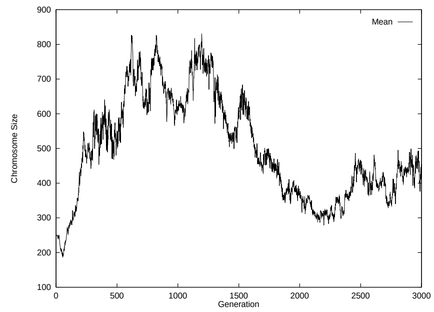

Fig. 6. Chromosome size vs Generations. A typical run

functional subnetwork). This may be implied by the choice of initial states, that allows individuals without such control to carry on all of the tests. The network operates a kind of derivation on the angular position of the pole, thus calculating the angular velocity of the pole.

V. Statistical Properties

When using optimization methods, especially with evolutionary methods, one wants to know why a particular method works or not. The underlying question is what is the aspect of the fitness landscape. In order to obtain a kind of answer, some have tried to make statistical measures on the fitness landscape. In this section we adapt their development for the morphogenesis.

A. Topology of the Search Space

Before talking about fitness landscapes, we should define a neighborhood on the search space. In the case of the optimization of binary strings, the search space topology is quite obvious. This is a boolean hypercube, with as many dimensions as the length of the string. The distance between 2 individuals is the *Hamming* distance between the strings.
In our case, the topology is more complex. Nevertheless, it is still possible to define a distance. Indeed, consider a mapping between a chromosome and a binary string of length 2^{15}, such that each position of the string corresponds to a gene; a 1 at a position implies that the corresponding gene is present in the chromosome; a 0 indicates that the gene is not present. There exist such mappings which are bijective as far as the gene positions in the chromosomes have no importance. We define the distance between 2 chromosomes as the *hamming* distance between the 2 binary strings obtained by this mapping.
The mutation operator nearly respects this distance, that is, its application moves the chromosome at distance 1. In fact, this applies to the gene deletion, which flips the bit corresponding to the deleted gene from 1 to 0; it applies too for the adding mutation, which flips a bit from 0 to 1; it is not valid for the modification mutation, which in fact has the same effect that the combination of a deletion and an adding mutation has on the binary string, so moving the chromosome away at distance 2.

B. Fitness Distance Correlation

Knowing the correlation between the fitness of an individual and its distance to the optimum, allows one to qualify the difficulty of the search ([16]). Strong correlation (in fact anti-correlation, since fitness should decrease as the distance to optimum increases) may induce an easy search, while in the case of no correlation, a random search should perform at least the same as other optimization methods.
In our case it is unthinkable to calculate this correlation over the entire search space or even, over a representative sample. Moreover, the space is highly redundant, since the same network may have many possible chromosomes, and in general, the fitness landscape is multimodal, since the same problem may be solved by different networks. All this complicates the calculus of the fitness distance correlation and makes irrelevant the results obtained.
To overcome these difficulties, we choose to calculate a local fitness distance correlation. From an optimal individual, 50 sequences of 100 successive mutations are applied, in order to get a sample. We calculate the correlation on this sample. Results are shown on table III, for 4 different optima. They exhibit a negative fitness distance correlation, below -0.3. This implies that near an optimum, the search should be easy for a local optimization method, such as a GA.

TABLE III
Fitness Distance Correlation

Run	1	2	3	4
Mean Fitn.	1849.72	2208.37	1526.32	3348.01
Mean Dist.	33.4593	58.0808	56.5947	55.7019
Std Dev Fitn.	19.4795	30.2869	24.6771	37.1812
Std Dev Dist.	0.2032	0.4672	0.4491	0.4437
Correlation	-0.5510	-0.2937	-0.3102	-0.4142

On figure 7 a scatter plot of points distance vs fitness corresponding to the run 2 is shown. Figure 8 is a scatter plot of chromosome length vs. fitness. Since the plot is nearly symmetrical, there is no correlation between the length and the fitness. This illustrates the assumption made earlier. However, if relative length, i.e. the absolute value of the difference between the chromosome length and the one of the optimal, are considered, we meet again the distance vs. fitness plot, since there is an obvious correlation between relative length and distance. In fact, the length vs. fitness plot indicates that the two mutations, adding or deleting a gene, play the same role.

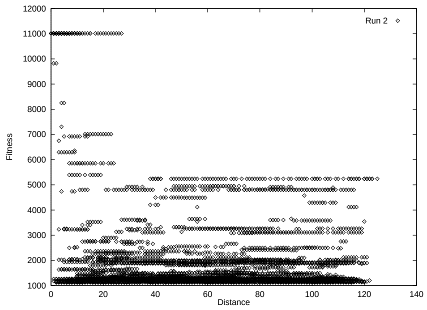

Fig. 7. distance to optimal vs fitness

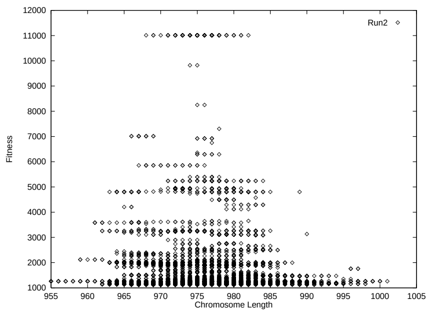

Fig. 8. length vs. fitness

VI. Conclusion

In this paper, we propose a new method of neural network growth based on the protein synthesis regulation process. Much work has been already done on this subject. However, few of them proposed theoretical frameworks to analyze and to improve performances.

We aim to make a better connection between the morphogenesis process and genetic algorithm theory. This is shown by two points.

First, we develop a chromosome structure that allows genetic operators to respect some GA properties, such as *respect* and *ergodicity*. In our sense, this is important to have such operators, because of the search space size.

Second, we introduce some tools for qualifying the search, such as the fitness distance correlation. These tools do not give a precise and exact view of the fitness landscape. However, they give some indications on the way the search occurs, and the utility of a particular method over a random search.

These are the first steps toward a more complete study of neurogenesis. The two points, structure respectful operators and accurate analysis, will be the major topics of our future work.

References

[1] H. Kitano, "Empirical studies on the speed of convergence of neural network training using genetic algorithms", in *Proceedings AAAI*, 1990, pp. 789–795.

[2] A. Wieland, "Evolving neural network controllers for unstable systems", in *International Joint Conference on Neural Networks*, 1991, pp. 667–673.

[3] J. Kodjabachian and J.-A. Meyer, "Development, learning and evolution in animats", in *Perception To Action Conference Proceedings*, P. Gaussier and J.-D. Nicoud, Eds. 1994, pp. 96–109, IEEE Computer Society Press.

[4] F. Gruau and D. Whitley, "Adding learning to the cellular development of neural networks: Evolution and the baldwin effect", *Evolutionary Computation*, vol. 1, no. 3, pp. 213–234, 1993.

[5] H. De Garis, "Cam-brain: The evolutionary engineering of a billion neuron artificial brain by 2001 which grows/evolves at electronic speeds inside a cellular automata machine (cam)", in *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'95)*, D. W. Pearson, N. C. Steele, and R. F. Albrecht, Eds. 1995, Springer-Verlag Wien New York.

[6] O. Michel and J. Biondi, "Morphogenesis of neural networks", *Neural Processing Letters*, vol. 2, no. 1, pp. 9–12, January 1995.

[7] O. Michel and Ph. Collard, "Artificial neurogenesis: An application to autonomous robotics", in *Proceedings of the 8th International Conference on Tools with Artificial Intelligence (TAI'96)*, M.G. Radle, Ed. 1996, pp. 207–214, IEEE Computer Society Press.

[8] I. Harvey, *The Artificial Evolution of Adaptive Behavior*, PhD thesis, University of Sussex, 1993.

[9] J. Biondi, O. Michel, and M. Clergue, "Biological metaphors for artificial cognitive systems evolution", in *Proceedings of the 14th International Congres on Cybernetics, Namur, August 21-25*, 1996, pp. 108–112.

[10] N. J. Radcliffe, "Equivalence class analysis of genetic algorithms", in *Complex Systems*, vol. 5, pp. 183–205. 1991.

[11] B. Widrow, "The original adaptive neural net broom-balancer", in *Proc IEEE Inter. Symp. on Circuits & Systems*, 1987, pp. 351–357.

[12] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning problems", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, pp. 834–846, 1983.

[13] D. Whitley, F. Gruau, and L. Pyeatt, "Cellular encoding applied to neurocontrol", in *Proceeding of the sixth International Conference on Genetic Algorithms*, Larrt J. Eshelman, Ed. 1995, pp. 460–467, Morgan Kaufmann.

[14] A. Wieland, "Evolving controls for unstable systems", in *Connectionist Models: Proc. 19990 Summer school.* 1990, pp. 91–102, Morgan Kaufmann.

[15] D.E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, Massachussets, 1989.

[16] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms", in *Sixth Intenational Conference on Genetic Algorithms*, L.J. Eschelman, Ed., San Francisco, California, July 1995, Morgan Keaufmann.