International Journal on Artificial Intelligence Tools
Vol. 6, No. 3-4 (1997) 000—000

ARTIFICIAL NEUROGENESIS: APPLICATIONS TO THE CART-POLE PROBLEM AND TO AN AUTONOMOUS MOBILE ROBOT

OLIVIER MICHEL[1], MANUEL CLERGUE[2], and PHILIPPE COLLARD[2]

[1]*Laboratory of Microcomputing*
Swiss Federal Institute of Technology, IN-F Ecublens, 1015 Lausanne, Switzerland

[2]*I3S laboratory, University of Nice – Sophia Antipolis, CNRS*
bt.4, 250 av. Albert Einstein, 06560 Valbonne, France

Received May 1, 1997
Revised (revised date)

A lot of recent research papers focus on the challenging problem of the combination of genetic algorithms and artificial neural networks. Developmental and molecular biology may be a source of inspiration for designing powerful artificial neurogenesis systems allowing the generation of complex modular structures. This paper describes in details such a neurogenesis model associated with an evolutionary process and its applications to the cart-pole problem and to the control of a mobile robot. Early results demonstrate the surprising efficiency of this methodology and give hints to continue the research towards the generation of more complex adaptive neural networks.

Keywords: Artificial Neurogenesis; Evolutionary Computation; Genomic Networks; Mobile Robotics.

1. Introduction

Artificial neural networks paradigm is a very powerful AI tool since it allows an infinite number of schemes to be build using a very wide variety of learning rules[1]. For practical applications, engineers have to address both problems of designing a suitable topology and defining an appropriate learning rule in order to obtain accurate artificial neural networks. These tasks, and especially the first one, are not easy and may be inspired by neurobiological observations for the learning rules[2] as well as for the topology[3,4,5]. Image processing is a good example where biology gave hints to engineers[6,7]. But biological datas relative to the activities of large groups of neurons are often inextricable and hence difficult to understand. Although the principles of synaptic strengths modifications are better understood[8], most of the functions of biological brains are still unknown. This is mainly due to the big numbers of interconnected units forming complex structures and leading to very complex dynamics.

An alternative to trying to copy biological brains is modeling another interesting biological process: Evolution. Computer investigations addressing evolution are often refered as Evolutionary Computation (EC) or Genetic Algorithms (GA)[9]. The application of Evolutionary Computation to artificial neural networks has been investigated by a number of authors. A distinction must be done between evolutionary techniques used as learning algorithms (i.e., to calculate the synaptic weights of a neural network whose architecture is fixed[10]) and evolutionary techniques used to optimize the topology of a neural network. This paper will focus on the second point since it represents a promising challenge in the search for intelligent artificial neural networks. Two applications of this neurogenesis process will be explained. The first one addresses the classical cart-pole (also known as the inverted pendulum problem). It provide a nice experimental framework for studying the possibilities of the neurogenesis process. The second one addresses autonomous robotics and shows that the same neurogenesis process may be used to produce a neural controller driving a mobile robot.

2. Genetic Algorithms for Neurogenesis

The artificial morphogenesis of neural networks (neurogenesis) is a process that uses informations lying on a chromosome to build up a structure of neurons interconnected via a number of links of different types. A resume of research in developmental neurogenesis can be found in[11]. These early attempts to generate automatically artificial neural networks are usually destinated to produce behavioral controllers for autonomous agents equipped with sensors and actuators.

Most of these researches make an extensive use of production rules at different levels: On one hand, *Boers and Kuiper*[12] use *Lindenmayer* systems[13] for rewriting groups of neurons. *Gruau* applies a complex encoding scheme using a grammar tree as a rewriting rule for neurons [14]. On the other hand, such production rules can be applied to lower level objects, corresponding to chemical elements inside neurons, inducing actions at the neuron level like cell division, cell migration, axon growth, etc. *Harvey*[15] proposed such a theoretical framework allowing the modelization of polypeptide chains inside the cells. *Vaario and Shimohara*[16] developed such a system that modelizes attractions and repulsions between cells leading to formation of structures. *Kitano*[17] observed the emergence of artificial patterns of axon growth similar to those observed in nature. *Dellaert and Beer*[18] built a morphogenesis process inspired from *Kauffman*'s genetic regulatory networks[19] in which a steady state of the genetic network fires a morphogenesis action: a cell division.

Although they do not use production rules, *Nolfi and Parisi* developed an interesting dynamical neurogenesis model[20,21], allowing the environment to influence the morphogenesis process while the agent is interacting with the environment.

The approach presented in this paper features a dynamical genomic network, involving artificial proteins, which is the heart of a neurogenesis process, allowing cell differentiation, cell division, cell migration, axon growth, axon guidance and target recognition in a two dimensional space. The resulting neural networks are embedded in a simulated mobile robot which has to travel across a maze while avoiding obstacles.

3. Application to Mobile Robotics

Most of the evolutionary neural networks research has been applied to autonomous agents[11]. This application area was prefered for three main reasons:

- Other traditional robotics approaches failed in proposing a powerful general framework.
- Simple autonomous agents may involve a relatively simple input to output processing. They are easily expandable and hence may require an increasing structural complexity[22]. This expandability ability makes them very well suited for evolutionary computation.
- The recent emergence of scientific interest in the field of Artificial Life[23] reinforced this research since this way of obtaining artificial neural networks is "biologically plausible" and hence of fundamental interest.

The last point may provide very interesting guidelines for the development of an evolutionary robotics project. Such a framework will give powerful metaphors for the design a self-sufficient, yet powerful, evolutionary system. Such a research philosophy may help to design a fitness function using something similar to an artificial metabolism to make an evaluation of the individuals.

The evolutionary loop we propose (see figure 1) involves successively a genetic algorithm evolving chromosomes, a morphogenesis process allowing to decode a chromosome into a neural network, a dynamical neural network driving a mobile robot and finally an artificial metabolism defining the viability domain of the robots and returning a fitness value to the genetic algorithm. This methodology was applied in order to observe the emergence of mobile robot behaviors. The genetic evolution occured in simulation and the resulting neural networks were then embedded on the real robot[24].

4. Dynamical Neural Network Model

Many reasons led us to use dynamical neural networks. This family of networks includes multi-layer perceptrons as well as recurrent networks. Consequently, it seems to be rather universal. Moreover, the properties of such networks are very interesting for autonomous agents (temporal processing, sequence generation and recognition, models of memory, etc.). It is possible to implement local learning algorithms, cheap in computation time and friendly to parallel computation. Their very complex dynamics make their structural design very difficult. Genetic Algorithms may be suitable for such a task.

All the neurons have the same transfer function (linear thresholded). When the state of a neuron is close to 1, it will be said to be excited. If the state of a neuron is close to 0, it will be said to be inhibited (or at rest). Let $x_i(t)$ be the state of the neuron i at iteration t and ω_{ij}, the weight of the link between neuron j and neuron

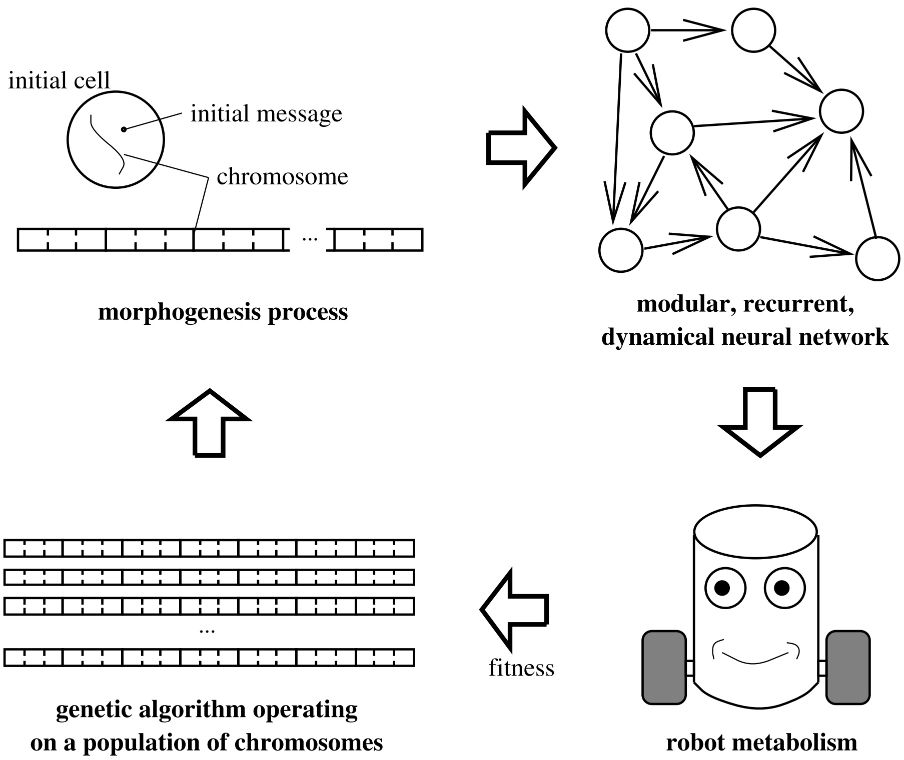


Fig. 1. The evolutionary loop

i. The state of neuron i updates as described in equation 1:

$$x_i(t+1) = \begin{cases} 0 & \text{if } \sum_j \omega_{ij} x_j(t) \leq 0 \\ 1 & \text{if } \sum_j \omega_{ij} x_j(t) \geq 1 \\ \sum_j \omega_{ij} x_j(t) & \text{otherwise} \end{cases} \tag{1}$$

Different kinds of links exist. Some links have a fixed given synaptic weight equal to a positive or a negative real number, while other links have a variable synaptic weight whose value is evolving according to a specific Hebbian learning rule. This system may be easily expandable by adding other learning rules such as different versions of Hebb rule, anti-Hebb rule, etc. A collection of different fixed weight links has been carefully designed, allowing to build various neural schemes (see figure 2) that can be seen as building blocks for larger neural networks.

5. Artificial Morphogenesis

The artificial morphogenesis process allows to build dynamical neural networks using a chromosome organized in a linear structure. It takes inspiration from the biological explanation of protein synthesis regulation[25]. This recurrent process allows an easy generation of modular neural networks (where the same sub-structures

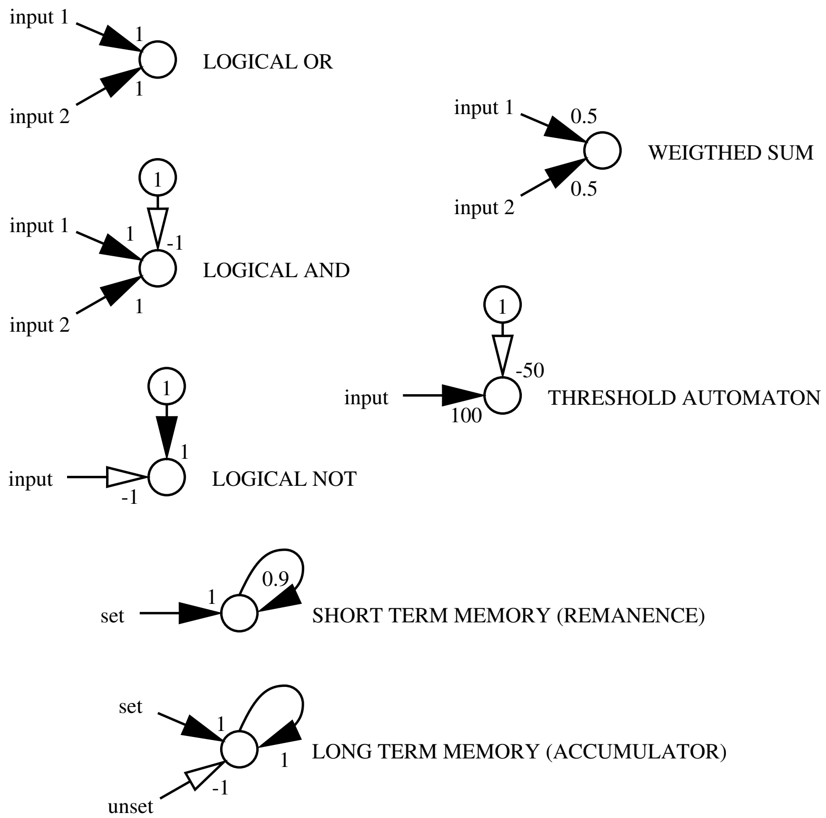


Fig. 2. Different neural schemes: excitatory links are represented in black and inhibitory links are represented in white

may exist at different places in the same overall network). Moreover, due to a strong epistasis, it features some properties of dynamical systems that permit to generate complexity at the edge between chaos and order[19].

5.1. *Morphogenesis Space*

5.1.1. *Space Structure*

The morphogenesis process runs in a two dimensional space discretized using a hexagonal grid. The choice of hexagons relies on the interesting neighboring property of such grids: Like the circle, the hexagon has exactly 6 neighbors. Moreover, it eludes the typical problem of choosing between the 4-neighbors model and the 8-neighbors model induced by a square grid (see figure 3).

Chemicals diffusion, cell migrations and axon growth were implemented within this hexagonal grid. A hexagon usually contains various chemicals (artificial proteins concentrations) ; it may also contain one cell (a single cell per hexagon). An initial rectangular size of $38 \times 28 = 1064$ hexagons was chosen since it represents a big enough space for containing enough artificial neurons necessary for most autonomous agents applications. For systems using a high input data flow like image processing systems, this size should be increased according to the size of the input flow. The hexagon space was configured as a torus (i.e., the upper side communicates with the lower side and the left hand side communicates with the right hand side) to avoid border effects.

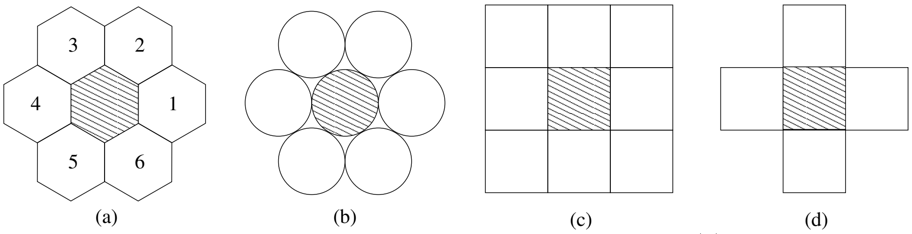


Fig. 3. Two dimensional space discretizations: hexagonal neighboring (a) is the better approximation of circles neighboring (b) while square grids allows either the 8-neighbors model (c) or the 4-neighbors model (d).

5.1.2. *Chemicals Diffusion*

A model of diffusion was designed to allow various chemicals (artificial proteins) to diffuse through this space. Each hexagon contains several concentrations of different artificial proteins. These concentrations diffuse to the neighboring hexagons according to the equation 2 using the preservation of the quantity of proteins associated with a diffusion coefficient:

$$C_{ik}(t+1) = \frac{K_{diff}}{6} \times \sum_{j=1}^{6} [C_{ik}(t) - C_{N_{ij}k}(t)] \tag{2}$$

$C_{ik}(t)$ represents the concentration of protein k in hexagon i at time t. K_{diff} is the diffusion parameter, it must be lower than 0.5 to avoid oscillation effects due to a too coarse discretization (we set it to 0.3 in our experiments). Finally, $N_{ij}, j \in \{1,2,3,4,5,6\}$ represents the j^{th} neighboring hexagon of hexagon i (see figure 3, a).

5.1.3. *Neural cells*

Each neural cell occupies a unique hexagon, while a hexagon contains at the most one cell. Each cell is associated with a non unique identifier (i.e., a numerical value) corresponding to a cell type. Consequently, two different cells may have the same identifier, which means that they are not differentiated (the one relatively to the other), and hence, they will behave roughly the same way. The cell identifier is used to produce systematically inside the cell's body a protein whose identifier is equal to this cell identifier. A cell moves according to chemicals gradients: if the concentration of a protein matches the cell type, the cell moves towards the

Table 1. Gene functions

Value	Name	Description
000	F_INT_A	protein synthesis: create an internal activator protein.
001	F_EXT_A	protein synthesis: create an external activator protein.
010	F_CELL_A	cell split: create an attractive cell.
011	F_LINK_A	axon growth: create an attractive axon.
100	F_INT_R	protein synthesis: create an internal repressor protein.
101	F_EXT_R	protein synthesis: create an external repressor protein.
110	F_CELL_R	cell split: create a repulsive cell.
111	F_LINK_R	axon growth: create a repulsive axon.

neighboring hexagon that contains the highest (or the lowest) concentration of such a protein. This attractive (or repulsive) cell behavior depends upon the cell type and the protein type: (1) Attractive cells are attracted by attractive proteins, (2) Attractive cells are repulsed by repulsive proteins, (3) Repulsive cells are repulsed by attractive proteins and (4) Repulsive cells are attracted by repulsive proteins.

If a cell is already situated on a local maximum (or minimum) of a matching protein concentration, it will not move. Cell division and axon growth will be detailed after describing the chromosome structure and the genomic network.

5.2. *Chromosome Structure*

A chromosome is divided into a variable number of genes. As depicted on figure 4, each gene contains an identifier part, `Id`, made of n bits, a function part, `Function`, made of 3 bits (see table 1) and a data part, `IdData`, also made of n bits. The value of n, depending of the size of the chromosome, will be explained in the genetic algorithm section.

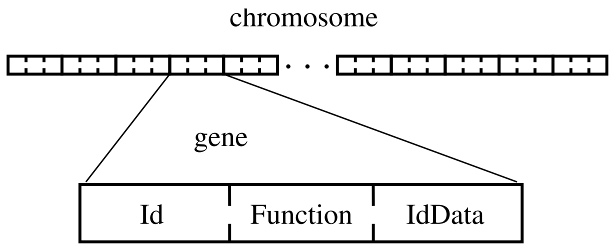


Fig. 4. Genotype structure: variable length chromosomes contain genes made up of three parts

5.3. *Genomic network*

5.3.1. *Protein Synthesis Regulation in Biology*

There are many ways to regulate protein synthesis, here we focused on two interesting points. Of course, we assumed many simplifications in order to be clearer, but the key ideas remain: If a particular protein called repressor is present, it can sit on the start codon of a specified gene on the chromosome, so that RNA polymerase cannot read it, and thus the corresponding proteins cannot be synthesized. This system can be recurrent when a synthesized protein can be a repressor for another gene. The other mechanism can be seen as the positive version of the first one. A molecule called activator is necessary to initiate the process of transcription of the DNA into mRNA at a specific locus, and thus to initiate the proteins synthesis process. Recurrence remains possible when a synthesized protein causes the synthesis of other ones.

Such a system is not closed since a protein may also initiate various cell behaviors (possibly cell division, cell migration, axon growth for neural cells, etc.). This process can be seen as a kind of production system, where proteins (repressors and activators) are conditions to the production of other proteins. If each protein is represented as a vertex of a graph, the genes will be represented as connections between proteins. This is called the genomic network (see figure 5).

5.3.2. *Artificial Genomic Network*

Before the morphogenesis process runs, the chemical contents of all the hexagons are cleaned and a number of initial cells (possibly one single cell) are laid in some hexagons of the morphogenesis space. These cells are of a given type (defined by their identifier). Consequently, they start to produce the corresponding proteins inside their own cell body. These proteins initiate the complex process occuring in the genomic network: They activate some genes that will produce other proteins and so on.

A gene can influence another through the production of a protein. Let assume that a gene is active. If the `Function` of the first gene leads to the synthesis of an activator (resp. repressor) protein, the gene will use its `IdData` value to build up such protein. The synthesized protein will be defined by its type: activator (resp. repressor) and its identifier equal to the value of the `IdData` of the gene. Such a protein will then be able to influence other genes if its identifier matches (i.e., is equal to) `Id` values of other genes.

Functions leading to the production of activator (resp. repressor) proteins include `F_INT_A` (resp. `F_INT_R`) which produces proteins that remain inside the cell body while `F_EXT_A` (resp. `F_EXT_R`) produces activator (resp. repressor) proteins that diffuse through the cell body. Proteins remaining in the cell body cannot move outside of it while diffusion proteins enter the hexagon where the cell lies and diffuse to its neighboring hexagones according to the diffusion model, and so on, thus bringing chemical messages to neighboring cells. This extends the notion of genomic network outside the cell body, allowing cells to communicate with each other.

The equation 3 modelizes the gene activation process where A_k is the activity of gene k, P_k representing the set of proteins matching with A_k, C_i being the concentration of protein i and $T_i \in \{-1, 1\}$ representing whether the protein i is a repressor (-1) or an activator (1). A_k will be said to be active if its value is strictly positive and inhibited otherwise.

$$A_k(t) = \begin{cases} 0 & \text{if } \sum_{i \in P_k}(C_i \times T_i) \leq 0 \\ 1 & \text{if } \sum_{i \in P_k}(C_i \times T_i) \geq 1 \\ \sum_{i \in P_k}(C_i \times T_i) & \text{otherwise} \end{cases} \tag{3}$$

5.4. *Morphogenesis Actions*

5.4.1. *Cell Division*

A gene may initiate a cell division process if its `Function` is `F_CELL_A` (resp. `F_CELL_R`). This will produce a new attractive (resp. repulsive) cell whose identifier is equal to the gene's `IdData` parameter. A copy of the chromosome of the mother cell is given to the child cell, so that all the cells have the same genotype (just like in Nature). The two cells initially occupy the same hexagon, so they will have to move away from each other during next iteration or else, the new cell will be destroyed (since two cells cannot occupy the same hexagon).

5.4.2. *Axon Growth*

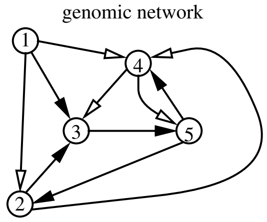
Artificial cells may have several axons connecting them to several other cells and thus allowing different types of synapses to be created. An axon is generated by a gene whose `Function` is `F_LINK_A` (resp. `F_LINK_R`). The axon identifier is set to the `IdData` value of the gene. The newly created attractive (resp. repulsive) axon will growth towards the direction of the positive (resp. negative) chemical gradient corresponding to its identifier using the same principle as cell moving mechanism. Once an axon arrives on a hexagon with a null chemical gradient, it dies, unless the hexagon contains a cell and in this case, it connects to the cell. The resulting synaptic type is corresponding to the axon identifier. It may be a fixed weight synapse (positive or negative) or a synapse associated with a learning law (various forms of Hebbian learning, anti-Hebbian learning, etc.)

6. Genetic Algorithm

A genetic algorithm was designed to operate on a population of variable length chromosomes. New operators were defined to allow gene addition, gene deletion, crossover between individuals of different sizes, etc.

At the beginning, an initial population of 100 individuals is randomly generated. Each chromosome is initially made of 16 genes. The maximum values of the fields `Id` and `IdData`, n, corresponding to the maximum number of different proteins is computed using *Kauffman*'s NK model of dynamical boolean networks[19].

Let N be the number of elements (genes) of the genomic network. If each gene depends on K other genes (on average), the corresponding genomic network will have N links and $N \div K$ vertices (see figure 5). Hence, the maximum number of proteins equals $n = N \div K$. It has been shown by *Kauffman* that the value of K determines the dynamics of the genomic network: on one hand, if $K = N$,



chromosome	
1: repressor(2)	3: activator(5)
1: activator(3)	4: repressor(3)
1: repressor(4)	4: repressor(5)
2: activator(3)	5: activator(2)
2: repressor(4)	5: activator(4)

Fig. 5. The genomic network represented as a graph and as a set of chromosome genes

the behavior of the network is chaotic, that is very sensitive to initial conditions and to perturbations. On the other hand, if $K = 1$, the network is said to be a *set of independent chain* with very simple dynamics. The most fascinating case corresponds to $K = 2$ where interesting behaviors appear: the network becomes resistant to most of the perturbations and a sort of order seems to emerge through complex structures. Such networks feature dynamics at the edge of chaos, which may lead to interesting results since complex structures in life exists at the edge of chaos[26].

We choose to set $K = 2$ in our genomic network, in order to have a chance to observe such interesting dynamics. Consequently, the initial number of proteins available in the system is given by $n = N \div K = 8$ for the initial chromosome. However, since the size of the chromosome may change during the evolutionary process, this value is updated dynamically during evolution. Consequently, the system will create new proteins when the size of the chromosomes increase.

7. The Cart-Pole Problem

In order to validate and analyse this approach, it is interesting to try to address simpler problems than those drawn from autonomous robotics. Indeed, the computational cost of simulations, the sensibility of the fitness function design, make them difficult to get relevant and analysable results. Thus, it might be useful to test artificial neurogenesis on another 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 hence, an objective evaluation is quite easy. Finally, it is well referenced and has been already successfully addressed by many control methods, including artificial neural networks. It might be seen as an adequat benchmark for neural networks and neurogenesis investigations.

7.1. *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 observed here, an optimal solution is found in the case of complete information on the system.

Artificial neural networks have, hardly since the beginning, addressed this problem. The ADALINE model was trained to balance poles using *Widrow-Hoff* LMS algorithm[27]. Reinforcement learning methods have also addressed this problem[28].

The cart-pole system is still a standard research problem for neural networks. There are two recent examples, drawn from *Wieland*[29] and *Withley and al.*[30], where a genetic algorithm is used to evolve neural networks to control the cart-pole system. The *Wieland*'s method is the simpler. This method optimizes the weights of a fixed fully recurrent neural network, using a genetic algorithm. This method leads to very interesting results, but there is a strong limitation to generalization: the number of neurons and the structure of the network is set with *a priori* knowledge on the problem. The *Withley and al.*'s method is close to our approach since growing neural networks are used to solve this problem. They use a morphogenesis method developped by *Gruau*[31], that is called *Cellular Encoding*. Again, the results obtained are very interesting. Efficient networks are obtained faster, and according to the authors, cellular encoding could automatically find small architecture whose structure and complexity fit the specificity of the problem. This is a strong argument for neurogenesis methods.

7.2. *Definition of the Problem*

The system, depicted on figure 6, is composed of a cart moving on a track. A pole is hinged on the cart. The goal of the controller is to avoid the fall of the pole by exerting a force on the left or on the right side of the cart, and this without moving it away from the limits of the track.

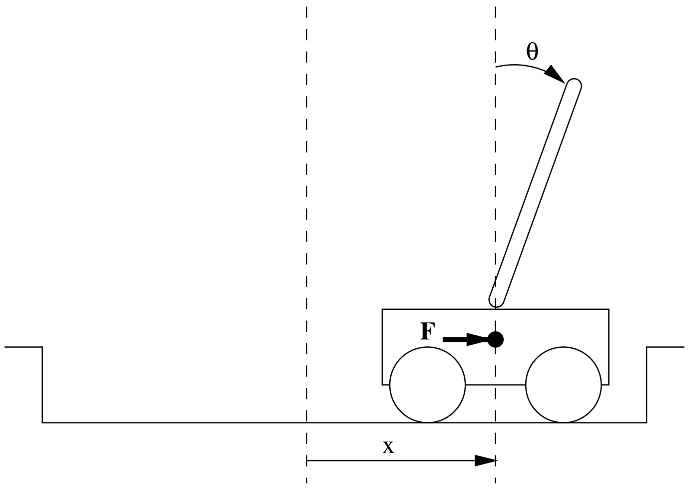


Figure 6: Cart-pole system to be controlled.

The system is simulated by the equations 4 and 5 (drawn from[32]):

Table 2. Meaning and value of used symbols

Symbole	Description	Value
x	Position of cart on track	$[-2.4, 2.4]$ m
θ	Angle of pole from vertical	$[-36, 36]$ deg.
F	Force applied to cart	$[-10, 10]$ N
F_{max}	Maximal Force applied to cart	10 N
g	Gravitational acceleration	-9.8m/s^2
l	Half-length of pole	0.5 m
M	Mass of cart	1.0 kg
m	Mass of pole	0.1 kg
μ_c	Coefficient of friction cart/track	0.0
μ_p	Coefficient of friction of the pole's hinge	0.0
τ	Time step for Euler method	0.002 s

$$\begin{aligned} \ddot{x} &= \frac{F-\mu_c \text{sgn}(\dot{x})+\tilde{F}}{M+m} \\ \ddot{\theta} &= -\frac{3}{4l}(\ddot{x}\cos\theta + g\sin\theta + \frac{\mu_p\dot{\theta}}{ml}) \end{aligned} \tag{4}$$

where $\tilde{F}$ is the effective force of the pole on the cart:

$$\tilde{F} = ml\dot{\theta}^2 \sin\theta + \frac{3}{4} m \cos\theta (\frac{\mu_p\dot{\theta}}{ml} + g\sin\theta) \tag{5}$$

The meaning and the value of each symbols are shown in table 2.

Simulations are done with the Euler's method, with a step of τ. In general, the Euler's method gives poor results in solving differential equations. But this is sufficient in order to validate our approach. Obviously, if the aim was to obtain controllers for real systems, a more accurate method would have been chosen. The advantage of Euler's method 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 both the cart and the pole, there is an optimal solution given by the equation 6:

$$F = F_{max}\text{sgn}(k_1 x + k_2 \dot{x} + k_3 \theta + k_4 \dot{\theta}) \tag{6}$$

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 position ones, i.e., the cart position and the pole angle. Stable control of this system requires the controller to compute the velocities.

7.3. *Experimental Settings*

7.3.1. *Neural Networks*

The neural networks used here differ slightly from those describe above in section 4. They are still dynamical neural networks, but the neuron transfert function is modified to allow output values ranging from -1.0 to 1.0. The major reason for this modification is that the state variables, for example the position of the cart, may have positive or negative values.

To bypass this difficulty, it would also have been possible to assign two input neurons for each state variables, or to scale the input in order to have values ranging from 0 to 1. But obviously, these solutions are not straightforward and, in order to remain as simple as possible, the solution featuring a neuron state ranging from -1 to 1 was prefered.

This involves the new transfert function summarized by equation 7:

$$x_i^{t+1} = \begin{cases} -1 & \text{if } \sum_j w_{ij} x_j^t \leq -1 \\ 1 & \text{if } \sum_j w_{ij} x_j^t \geq 1 \\ \sum_j w_{ij} x_j^t & \text{otherwise} \end{cases} \tag{7}$$

Obviously, the input values are normalized by dividing them by the maximal value they may reach.

Since the cart-pole problem with complete information, that is position and velocity of both the cart and the pole, is too simple, we addressed directly the problem with only position informations, letting the network calculate by itself the velocities. So, only two inputs are necessary: the cart position on the track and the angle of the pole from vertical, divided respectively by 2.4 and 36, to get values in $[-1.0, 1.0]$.

The output value stands in $[-1.0, 1.0]$. This value is multiplied by F_{max} to get the force applied on the cart.

A second modification occurs in the types of synaptic connections. For this problem, a set of fixed given synaptic weights was used for simplicity reasons. Weights have fixed given values drawn from the set $\{1.0, -1.0, 0.1, -0.1, 10, -10, 0.8, -0.8\}$. This set of values allows low, medium and hight inhibitory or excitatory synaptic connections.

7.3.2. *Genetic Algorithm*

For experiments, a classical steady state genetic algorithm was used[9]. With this kind of GA, only a part of the population (the worst individuals) is renewed each generation. In general, this leads to better performances. The part of population being renewed is set to 20%. So, each generation, the 20% worst of the population are removed and replaced by children of selected individuals drawn from the 80% remaining individuals. This strategy gives pretty good results compared to generational ones, where the entire population is renewed each generations.

Table 3. Different initial states used to test the neural network.

cart position	(m)	2.0	0.0	0.0	0.0	0.0	0.0
cart velocity	(m/s)	0.0	0.0	0.0	0.0	0.0	0.0
pole position	(deg.)	1.8	18.0	-9.0	-18.0	0.0	0.0
pole velocity	(deg./sec)	0.0	0.0	-21.5	0.0	43.0	0.0

7.3.3. *Evaluation protocol*

The evalutation protocol was defined accordingly to an artificial life point of view, which claims that systems have to be *abductive*, in the sense that without knowledge on their domain of viability, they remain in this domain during perception-action cycles. In the cart-pole context, this means that neural networks have to keep the cart in the track limits and the pole sufficiently near from the vertical to avoid an irremediable fall. The easier and more direct way to evaluate this abductive ability is to assign a fitness value proportionnal to the time the network successfully controls the cart-pole system, that is, the duration the neural network keeps the system into its viability domain.

In order to get robust networks, several initial positions were tested over which a cumulative fitness value was computed.

First, the fitness value is assigned to zero. The neural network is evaluated with a first initial state, then with a second one and so on, until an end condition raises or the whole set of initial states has been tested. For each initial state, an evaluation lasts at most for MAX_STEP iteration steps, but it stops whenever the neural network fails preventing the cart-pole from falling down. Then, if the number of iterations is greater than a given value THRESH_STEP, the number of iterations is added to the fitness value. Otherwise, an end condition is raised and the evaluation is stopped.

In the following experiments, MAX_STEP was set to 1001 and THRESH_STEP was set to MAX_STEP/10. The initial states used are shown on table 3. They are distributed over the state space. A narrower set of initial states than those used by *Withley and al.* was chosen. It seemed that it would be sufficient to obtain networks with similar generalization capacity. To mesure this property, the generalization test described in[30] was used. The final best neural network is evaluated over 625 different initial states. Generalisation is tested by counting the number of cases in which the network is able to balance the pole for 1000 time step.

7.4. *Results*

Five experiments with the above settings were run. On average, an overall best individual, i.e., with fitness value equal to 6006, is found after only 5544 generations. Since a steady state strategy was used, the learning time is about 119000 fitness evaluations. This might appear to be a very good result comparing to other methods, but the resulting networks seems to be unable of generalizing. This is not a limitation inherent to the neurogenesis process, but a limitation in our experiment protocol due to the poorness of the initial state set. In order to verify this point, a

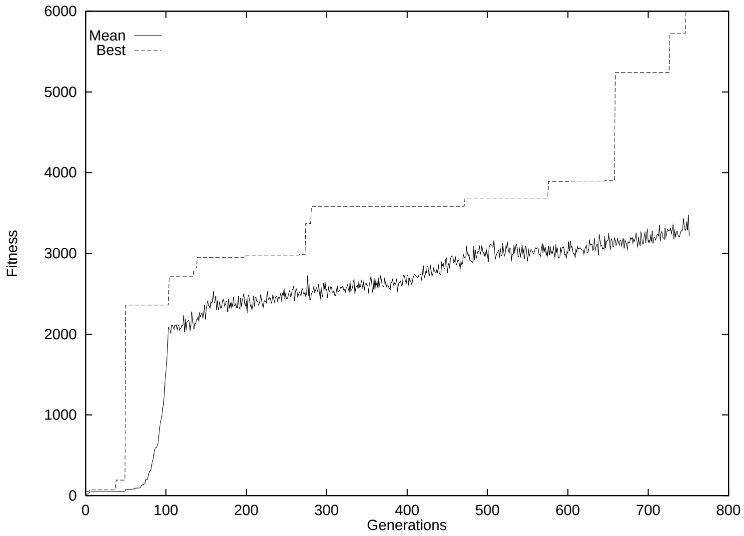


Figure 7: Fitness vs Generations: a typical run.

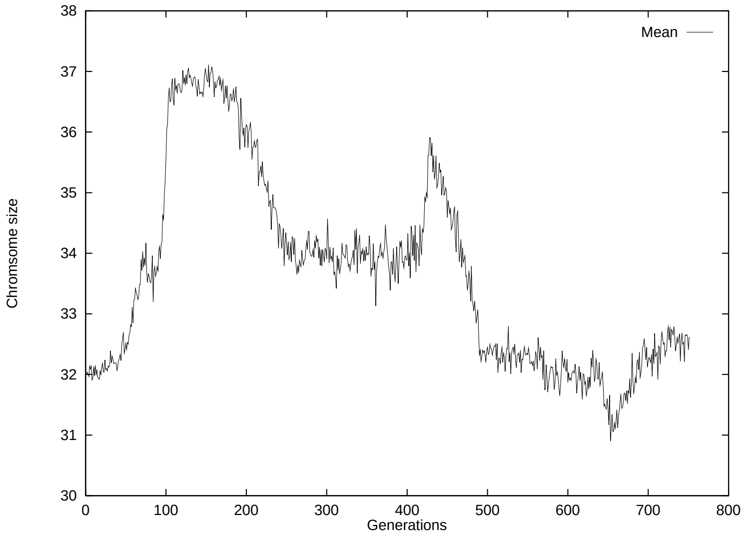


Figure 8: Chromosome size vs Generations: a typical run.

single experience was run with the initial state space drawn from[30]. The GA found the overall best individual in 4690 generations, that is, in the same time than for our initial state. But, the networks obtained have a generalization score of 175, that is close to the scores obtained by other methods.

In figure 7 and 8, a typical run is shown. Figure 7 traces the fitness vs generations. It can be seen that there are, during evolution, some stages of little improvements separated by stages of strong changes. Figure 8 shows the mean chromosome size vs generations. It is worth noticing that there are no correlations between fitness and chromosome size, apart the fact that changes occur simultaneously. A higher fitness does not necessarily involve longer chromosomes. At the end of the experiments, once the individuals became really efficient, the chromosome size come back around its initial value, thus minimizing the quantity of information necessary for coding a solution for this problem.

8. Experimental Robotics Setup

8.1. Khepera ***Robot and*** Khepera Simulator

Robotics experiments were driven on *Khepera Simulator*[33], a mobile robot simulator allowing to transfer easily the controllers to the real robot *Khepera*[34].

The mobile robot includes 8 infrared sensors (small rectangles on figure 9) allowing it to detect the proximity of objects in front of it, behind it, and to the right and the left sides of it. The robot is also equipped with two independent motors able to run forward and backward, thus allowing the robot to turn very efficiently.

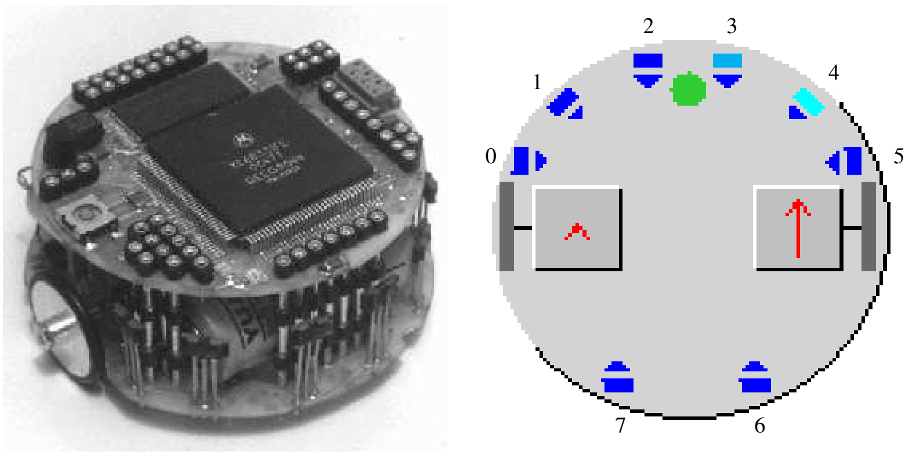


Fig. 9. Khepera (5 cm diameter) and its simulated counterpart

8.2. *Interface to the Artificial Neural Network*

To connect artificial neural networks, resulting from the evolutionary process, to the robot, it is necessary to define how to feed the inputs of the neural network using the robot sensors and how to feed the robot motors using the outputs of the neural network. In order to simplify this process, we chose to set on the initial morphogenesis space all the input and output neurons needed inside different hexagons. Three experiments were conducted using three different initial layouts of input and output neurons made of 8 inputs corresponding to the distance sensors available on the robot, 2 inputs corresponding to bumper sensors (added in the simulator but not available on the real robot) and 4 outputs corresponding to the forward and backward speeds of each motor. Since our neural model needs a bias input, this kind of input was also added (see figure 10). During the first experiment, the input and output cells were initially set accordingly to the real position of the sensors and motors of Khepera. During the second and the third experiment, several layers were formed where similar neurons were aligned. The third experiment features big spaces between input and output layers.

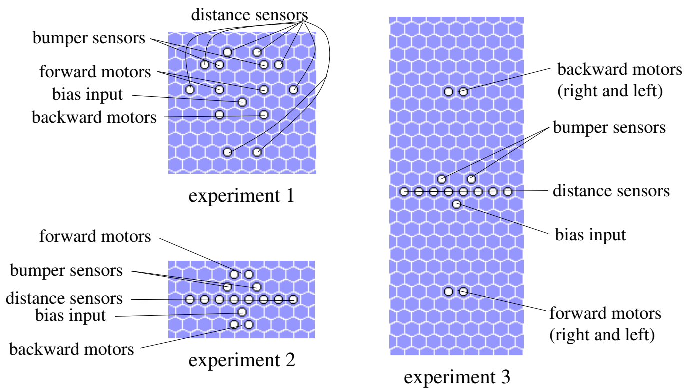


Fig. 10. Initial positions of input and output neurons

8.3. *Robot behavior*

The goal of the experiment is to develop a neural network which would allow a robot to travel, as far as possible, across a maze forming a kind of cross (see figure 11). This shape forces the robot to develop the ability to turn left and right in order progress in the maze while avoiding the walls.

For that purpose a custom fitness function was designed. The fitness value returned to the evolutionary algorithm corresponds to the distance between the inital position of the robot and the farest point reached by the robot. The robot evaluation is stopped in two cases:

- The robot hits an obstacle.
- The evaluation time is over (typically a few seconds).

9. Early Results

Since the distance input neurons are not differentiated (i.e., they have the same nu-

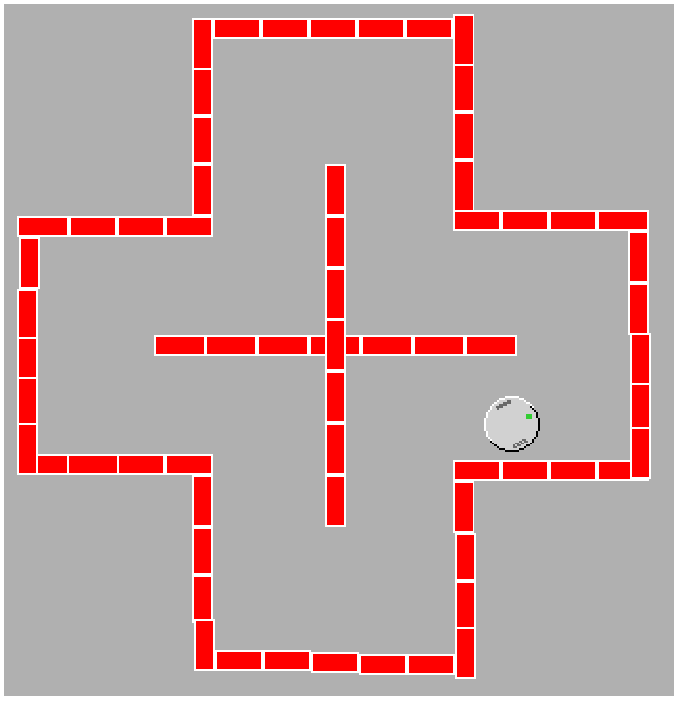


Fig. 11. Cross-like maze

merical identifier), we can expect to observe a similar behavior (connection pattern , migration, etc.) for each of these neurons. The bumper sensors are differentiated, as well as the motor sensors. This should allow the emergence of pre-wired reflexes relying upon these bumper sensors while the neural structures processing the distance informations should be trained using for example the available Hebbian links as a learning law and the bumper sensors as reinforcement signals. We successfully built a handmade neural network that learns to associate the distance sensors with the right motor actions according to the reinforcement signals sent by the bumper sensors. Now let see whether the evolutionary process found a similar structure.

After 200 generations of the genetic algorithm, different artificial neural networks were obtained that exhibited various performance in the maze. The best ones were obtained during experiment 1: the best neural network of the population was able to drive the robot across the maze without touching any wall, as long as we could observe it. The neural network was a single layer feed-forward network connecting the distance sensors inputs to three of the four motor outputs with an appropriate set of fixed weight (see figure 12). The values 0.5 and 51 of the synaptic weights were chosen by the evolutionary process within a set of weight values given a priori. We were a bit disappointed since we didn't expect that the evolutionary process succeed in establishing different connections starting from the non-differentiated distance input neurons. Different connection schemes were achieved by using the fact that the non-differentiated input cells were initially at different geographical locations on the hexagonal grid and hence received different chemical messages from their neighborhood, leading to different dynamics inside the cell bodies.

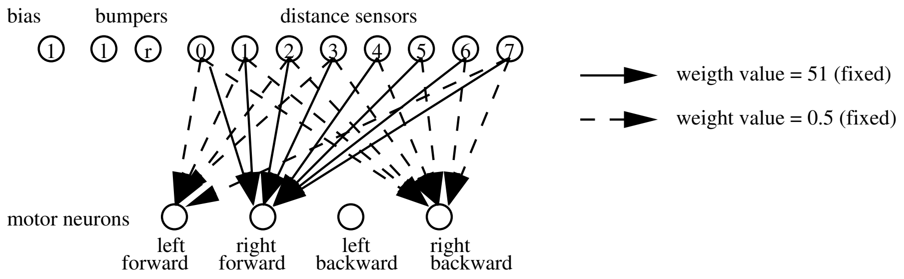


Fig. 12. Resulting best neural network during experiment 1

To try out to minimize the difference of behavior between non-differentiated cells, the input cells were aligned in a layer as described on figure 10, experiment 2. The resulting neural networks were very complex made of lots of hidden neurons and lots of connections especially between inputs and hidden neurons. The non-differentiated cells had a roughly similar behavior while slight differences in the connection patterns made the overall scheme perform almost as good as in the first experiment: the corresponding robots were able to travel in the cross maze even if they sometimes hit a wall.

Finally, we decided to set the output neurons far away from the input neurons so that their chemical influence on the input neurons would be rather similar for all input neurons. The results were similar to those obtained in experiment 2, except that the resulting networks were not so complex.

10. Conclusion

This attempt to design a powerful artificial neurogenesis system inspired from biology and applied to the cart-pole problem and to mobile robotics leads to interesting early results. The evolutionary process turned out to be able to find near-optimal architectures for the cart-pole problem as well as for a simple robotics navigation task involving obstacle avoidance.

The cart-pole problem demonstrated the capability of the neurogenesis process to produce efficient cart controllers. Although the resulting controllers are not able of generalization, they fit the addressed problem and reach a rather good fitness value. Interesting dynamics, especially concerning the evolution of the size of the chromosomes could be observed during this experiment.

In the robotics experiments, such efficient results, achieved with rather simple reactive artificial neural networks, should be compared with our expectations of getting more complex structures involving learning. On one hand, the complex structures we imagined, associated with complex learning behavior, need that the robot learns by trial and error and hence hits some walls to learn to avoid them. On the other hand, simple structures discovered by the evolutionary process don't need to make such errors since their adapted behavior is innate. This demonstrates the ability of the morphogenesis process to be able to connect in a different way non-differentiated cells if necessary and the ability of the overall evolutionary process to find out simple yet near-optimal solutions.

Primitive animals, like most of the insects, are often capable of walking as soon as they are born, without any learning, just like the robots discovered by the evolutionary process. Human beings and other mammalians usually need a learning stage before they are able to walk. This may be explained by the fact that such evolved species developed elaborated learning abilities exempting them from developing and preserving complex hardwired behaviors in their genotype. Moreover, for complex tasks (like walking with two legs), adaptive behaviors are far more efficient than hardwired behaviors.

Further investigations will involve problems in which learning is a mandatory issue for the robot in order to be selected by the evolutionary process. Such a methodology could give birth to a new generation of adaptive autonomous robots able to learn in unknown environments.

References

[1] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Santa Fe Institute, 1991.

[2] D.O. Hebb. *The Organization of Behavior.* Wiley, New York, 1949. Partially reprinted in [35].

[3] N. Franceschini and F. Mura. Visual control of altitude and speed in a flying agent. In D. Cliff, P. Husband, J.-A. Meyer, and W.S. Wilson, editors, *From animals to animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 91–99. MIT Press, 1994.

[4] Y. Burnod. *An adaptative neural network: the cerebral cortex.* Masson, 1989.

[5] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, 1989. (3rd ed.).

[6] K. Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[7] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems II.* Morgan Kaufmann, 1990.

[8] E.R. Kandel and J.H. Schwartz. Molecular biology of an elementary form of learning: modulation of transmitter release through cyclic amp-dependent protein kinase. *Science*, 218:433–443, 1982.

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

[10] H. Kitano. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings AAAI*, pages 789–795, 1990.

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

[12] E.J.W. Boers and H. Kuiper. Biological metaphors and the design of modular artificial neural networks. Master's thesis, Departments of Computer Science and Experimental Psychology at Leiden University, The Netherlands, 1992.

[13] A. Lindenmayer. Mathematical models for cellular interactions in development, I & II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[14] F. Gruau and D. Whitley. Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. *Evolutionary Computation*, 1(3):213–234, 1993.

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

[16] J. Vaario and K. Shimohara. On formation of structures. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada*, pages 421–435. Springer, June 1995.

[17] H. Kitano. Cell differentiation and neurogenesis in evolutionary large scale chaos. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada*. Springer, June 1995.

[18] F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agents synthesis. In R.A. Brooks and P. Maes, editors, *Proceedings of the Fourth International Workshop on Artificial Life*, pages 246–257, Cambridge, MA, 1994. The MIT Press / Bradford Books.

[19] S.A. Kauffman. *The Origins of Order: Self-Organisation and Selection in Evolution*. Oxford University Press, 1993.

[20] S. Nolfi and D. Parisi. Evolving artificial neural networks that develop in time. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada*, pages 353–367. Springer, June 1995.

[21] S. Nolfi, O. Miglino, and D. Parisi. Phenotypic plasticity in evolving neural networks. In P. Gaussier and J.-D. Nicoud, editors, *Perception To Action Conference Proceedings*. IEEE Computer Society Press, 1994.

[22] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, 1984.

[23] C.G. Langton, editor. *Artificial Life, proceedings of the First International Conference on Artificial Life*. Addison-Wesley, 1988.

[24] O. Michel. An artificial life approach for the synthesis of autonomous agents. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 220–231. Springer Verlag, 1996.

[25] O. Michel and J. Biondi. Morphogenesis of neural networks. *Neural Processing Letters*, 2(1):9–12, January 1995.

[26] C.G. Langton. Adaptation to the edge of the chaos. In C.G. Langton, J.D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity*, volume 10, Mass, 1992. Addison-Wesley, Reading.

[27] B. Widrow. The original adaptive neural net broom-balancer. In *Proc IEEE Inter. Symp. on Circuits & Systems*, pages 351–357, 1987.

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

[29] A. Wieland. Evolving neural network controllers for unstable systems. In *Interna-*

tional Joint Conference on Neural Networks, pages 667–673, 1991.

[30] D. Whitley, F. Gruau, and L. Pyeatt. Cellular encoding applied to neurocontrol. In Larrt J. Eshelman, editor, *Proceeding of the sixth International Conference on Genetic Algorithms*, pages 460–467. Morgan Kaufmann, 1995.

[31] F. Gruau and D. Whitley. The cellular developmental of neural networks : the interaction of learning and evolution. Technical report, Ecole Normale Supérieure de Lyon, 46, Allée d'Italie, 69364 Lyon Cedex 07, France, January 1993.

[32] A. Wieland. Evolving controls for unstable systems. In *Connectionist Models: Proc. 19990 Summer school*, pages 91–102. Morgan Kaufmann, 1990.

[33] O. Michel. Khepera simulator package. Freeware mobile robot simulator downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`, 1996.

[34] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Third International Symposium on Experimental Robotics 1993*, pages 501–513, Kyoto, Japan, 1994. Springer Verlag.

[35] J.A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, 1988.