Control and Cybernetics

vol. **26** (1997) No. 3

Dynamical genomic network applied to artificial neurogenesis

by

Olivier Michel

LAMI-EPFL, Swiss Federal Institute of Technology,
CH-1015 Lausanne, Switzerland,
http://diwww.epfl.ch/lami/team/michel/
E-mail: Olivier.Michel@di.epfl.ch

Abstract: Optimisation of the structure of artificial neural network using evolutionary techniques has been investigated by a number of authors using various approaches. In this paper, we claim that such a process requires the design of a complex neurogenesis model featuring a set of fundamental properties such as modularity and the possibility of complexity adaptation. Developmental and molecular biology might be an interesting source of inspiration for designing such powerful artificial neurogenesis systems allowing the generation of complex modular neural structures. This paper provides a description of a neurogenesis model based on a modelling of a natural genomic network and associated with an evolutionary process. Experimental results demonstrate some basic capabilities of the proposed neurogenesis model to produce multi-layered neural networks. An application to learning in the control of a mobile robot lead to unexpected results, giving hints for continuing the research towards the automatic generation of more complex adaptive neural networks.

Keywords: genomic networks, evolutionary neurogenesis, mobile robotics

1. Introduction

The artificial neural networks designers have to address both problems of designing a suitable topology and of defining an appropriate learning rule in order to obtain artificial neural networks featuring good performances. Such tasks are not easy and may be inspired by neurobiological observations of the learning rules, Hebb (1949), as for the topology, Franceschini and Mura (1994), Burnod (1989), Kohonen (1989). Image processing is a good example where biology gave hints to engineers, Le Cun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1990). But biological data relative to the activities of large groups of neurons are often very complex and specific to a definite part of the nervous system of a given animal under specific conditions, so that it is difficult to draw general conclusions from this kind of data. Although the principles of synaptic strength modifications are now better understood than before, Kandel and Schwartz (1982), most of the functions of biological brains are still unknown. This is mainly due to the huge numbers of interconnected units forming complex structures and leading to very complex dynamics through the constant change of synaptic weights and neural activities.

A possible alternative for copying biological nervous systems is based on modelling another interesting biological process: Evolution. Computer investigations addressing evolution are referred to as Evolutionary Computation (EC), Bäck, Fogel and Michalewicz (1997). 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, Kitano, 1990, Fogel, Wasson, Boughton and Porto, 1997) and evolutionary techniques used to optimise 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.

A review of current research in the area of evolutionary computation for neurogenesis will be followed by the description of an original evolutionary model using genomic network dynamics to model neurogenesis. Preliminary experiments demonstrate that the neurogenesis process is able to produce arbitrarily sized multi-layered neural networks. Other experiments lead to neural network based control of the mobile robot *Khepera*. The genomic network based neurogenesis involves modelling of low level biological entities (i.e., proteins and genes). An evolutionary algorithm associated with this neurogenesis process succeeded in evolving efficient genotypes generating artificial neural networks used as controllers for a navigation task of the mobile robot.

2. Review of evolutionary 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 Kodjabachian and Meyer (1994). 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 used Lindenmayer systems, Lindenmayer (1968), for rewriting groups of neurons, Boers and Kuiper (1992). Gruau applied a complex encoding scheme using a grammar tree as a rewriting rule for neurons Gruau and Whitley (1993). On the other hand, such production rules can be applied to lower level objects, corresponding to chemical components (usually enzymes or other proteins) inside neurons, inducing actions at the neuron level like cell division, cell migration, axon growth, etc. Harvey (1993) proposed such a theoretical framework allowing the modelling of polypeptide chains inside the cells. Vaario and Shimohara (1995) developed such a system that models attractions and repulsions between cells leading to formation of structures. Kitano (1995) observed the emergence of artificial patterns of axon growth similar to those observed in nature. Dellaert and Beer (1994) built a morphogenesis process inspired by Kauffman's genetic regulatory networks, Kauffman (1993), in which a steady state of the genetic network fires a morphogenesis action: a cell division. De Garis (1996) developed a morphogenesis process based upon a cellular automata hardware able to handle a large number of artificial neurons.

Although they do not use production rules, Nolfi and Parisi developed an interesting dynamical neurogenesis model, Nolfi and Parisi (1995), 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. Mobile robotics as an application

Most of the evolutionary neural networks research has been applied to autonomous agents, Kodjabachian and Meyer (1994). This application area was preferred 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, Braitenberg (1984). This expandability ability makes them very well suited for evolutionary computation.
- The recent emergence of scientific interest in the field of Artificial Life, Langton (1988) 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 guideline 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. This research philosophy may help to design a fitness function using something similar to an artificial metabolism to evaluate the individuals.

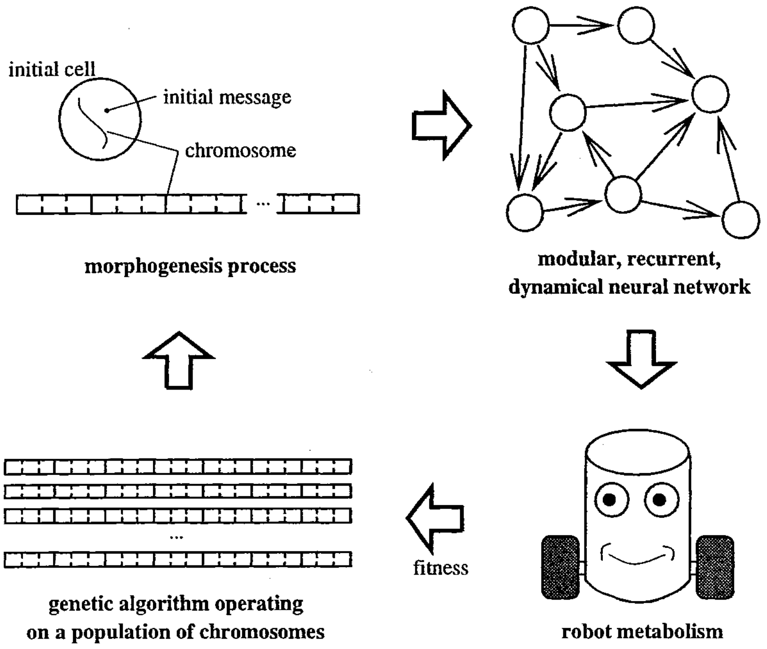

initial cell
initial message
chromosome
morphogenesis process
modular, recurrent, dynamical neural network
genetic algorithm operating on a population of chromosomes
fitness
robot metabolism

Figure 1. The evolutionary loop

The evolutionary loop we propose (see Fig. 1) involves successively an evolutionary algorithm evolving chromosomes, a morphogenesis process allowing to decode a chromosome into a neural network, a dynamic 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 evolutionary algorithm. This methodology was applied in order to observe the emergence of mobile robot behaviours. The evolution occurred in simulation and the resulting neural networks were then embedded on the real robot Michel (1996a).

4. Dynamic neural network model

The terminology "dynamic neural network" refers to a large set of neural networks including recurrent and feedforward networks with a special emphasis on the complexity of their dynamics. Many reasons led us to use dynamic neural networks. This family of networks includes multi-layer perceptrons as well as recurrent networks able of temporal processing. Consequently, it seems to be rather universal. Moreover, the properties of such networks are very interesting for autonomous agents (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. Evolutionary algorithms may be suitable for such a task.

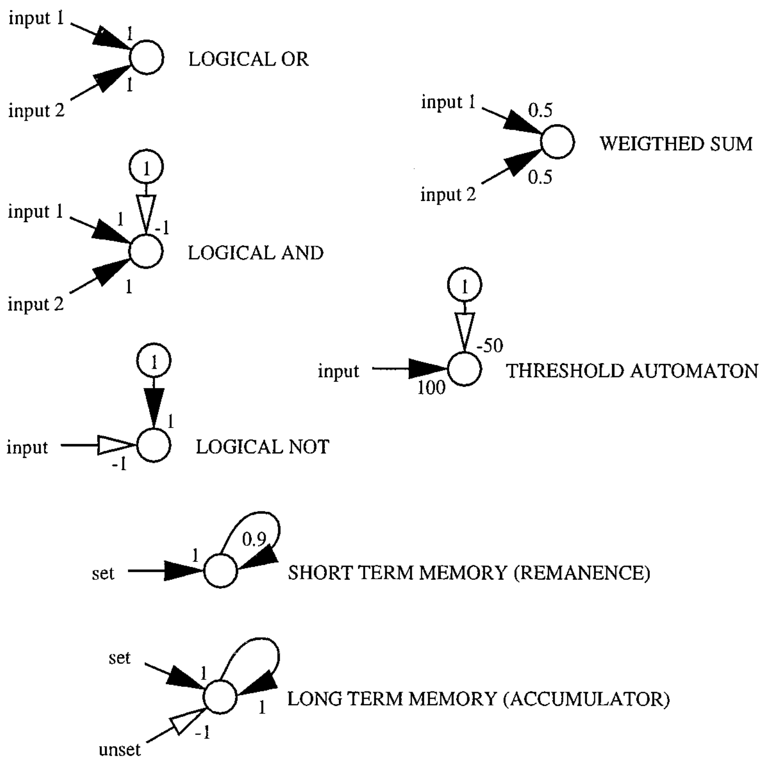


Figure 2. Different neural schemes: excitatory links are represented in black and inhibitory links are represented in white.

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 from the neuron j to the neuron i. The state of neuron i updates as described here:

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

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, Hebb (1949). 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 Fig. 2) that can be seen as building blocks for larger neural networks. The remanence and accumulator units feature a "set" input which allows to add the input value to the current state of the unit. Of course, the resulting state is thresholded to 1. Hence, if the state of the unit is initially zero and the value received by the unit lower than 1, this value is stored as the state of the unit. Then, in the first case (remanence), if the unit receives no more input, this value will slowly decrease according to the weight of the recurrent link (0.9) performing a weak self-excitation until it reaches zero (steady state). In the second case (accumulator), a strong self-excitation will make the state value remain the same until new incoming values from the "set" input are added to the current state or new incoming values from the "unset" input are subtracted from the current state (note that a value of 1 coming from the "unset" input will set the state of the unit to zero in any case).

5. Artificial neurogenesis

The artificial neurogenesis process allows building of dynamic neural networks using a linear chromosome. It takes inspiration from the biological explanation of protein synthesis regulation, Michel and Biondi (1995). This recurrent process allows an easy generation of modular neural networks (where the same substructures 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 border between chaos and order, Kauffman (1993).

5.1. Neurogenesis space

5.1.1. Space structure

The neurogenesis process runs in a two–dimensional space discretized using a hexagonal grid. The choice of hexagons relies on the interesting neighbouring property of such grids: Like the circle, the hexagon has exactly 6 neighbours standing at an equal distance from the central hexagon. Note that in the 8-neighbours model, the distances between neighbouring cells and the central cell are not the same for each neighboring cell. The 4-neighbours model induces a bad ratio of the distance between two cells and the number of cases to be visited between these two cells whereas this ratio is better for the hexagonal neighbouring (see Fig. 3).

Chemical diffusion, allowing cells to communicate with each other, cell migration and axon growth, are implemented within this hexagonal grid. A hexagon usually contains various chemicals (i.e., artificial protein 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 sufficiently large space to contain all necessary neurons for most autonomous agent 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 is 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.

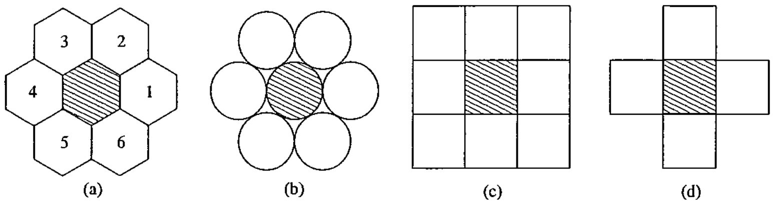


Figure 3. Two dimensional space discretizations: hexagonal neighbouring (a) appears to be the best approximation of circle neighbouring (b) while square grids allows either the 8-neighbours model (c) or the 4-neighbours model (d).

5.1.2. Chemical diffusion

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

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

$C_{ik}(t)$ represents the concentration of the protein k inside the hexagon i at time t. K_{dif} 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} neighbouring hexagon of hexagon i (see Fig. 3a).

5.1.3. Neural cells

Each neural cell lies in a unique hexagon, while an hexagon contains at 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 (one relatively to the other), and hence, they will behave roughly in the same way. The cell identifier is used to produce systematically inside the cell body a protein whose identifier is equal to this cell identifier. A cell moves according to chemical gradients: if the concentration of a protein matches the cell type, the cell moves towards the neighbouring hexagon that contains the highest (or the lowest) concentration of such a protein. This attractive (or repulsive) cell behaviour 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 made of a variable number of genes. As depicted in Fig. 4, each gene contains an identifier part, `Id`, an integer value ranging from 0 to $n-1$, a function part, `Function`, made up of 3 bits (see Table 1) and a data part, `IdData`, also ranging from 0 to $n-1$. The value of n, representing the number of different proteins, depends on the size of the chromosome. It will be explained in the evolutionary algorithm section.

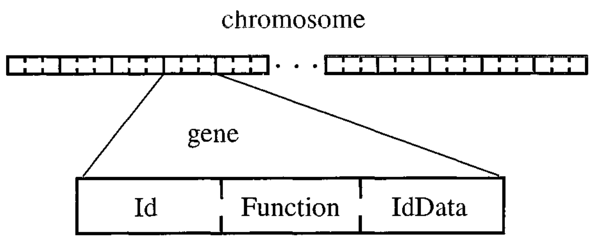


Figure 4. Genotype structure: variable length chromosomes contain genes made up of three parts

5.3. Genomic network

5.3.1. Protein synthesis regulation in biology

The production of chemicals (i.e., protein) inside and outside the cell bodies is under the control of the genomic network: a complex dynamical production system. 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

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.

Table 1. Gene functions

called repressor is present, it can sit on the start codon[1] of a specified gene on the chromosome, so that RNA polymerase cannot read it, and thus the corresponding proteins cannot be synthesised. This system can be recurrent when a synthesised 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 synthesised protein causes the synthesis of other ones.

Such a system is not closed since a protein may also initiate various cell behaviours (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 Fig. 5).

5.3.2. Artificial genomic network

Before the neurogenesis process runs, the chemical contents of all the hexagons is cleaned and a number of initial cells (possibly one single cell) are laid in some hexagons of the neurogenesis 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 occurring 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`

[1] A codon is a specific sequence of three consecutive nucleotides that is a part of the genetic code and that specifies a particular amino acid in a protein or starts or stops protein synthesis.

value to build up such protein. The synthesised 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 neighbouring hexagons according to the diffusion model, and so on, thus bringing chemical messages to neighbouring cells. This extends the notion of genomic network outside the cell body, allowing cells to communicate with each other.

The following equation models 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 positive and inhibited otherwise.

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

5.4. Neurogenesis 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

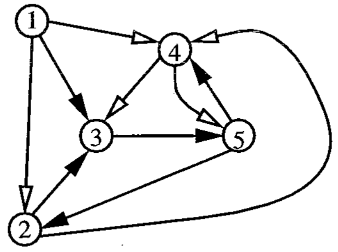
Artificial cells may have several axons connecting them to several other cells and thus allowing different types of synapses to be created. An axon is generated by a gene whose `Function` is `F_LINK_A` (resp. `F_LINK_R`). The axon identifier is set to the `IdData` value of the gene. The newly created attractive (resp. repulsive) axon will grow 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 at 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.)

5.5. Evolutionary algorithm

An evolutionary algorithm was designed to operate on a population of variable length chromosomes. The standard random binary mutation operator was used for the binary values of the chromosomes and another mutation scheme was used to mutate the code values of proteins. These values being comprised between zero and a maximum value, a mutation replaces the current value by a new random value in this interval. In addition, new operators were defined to allow gene addition, gene deletion and crossover between chromosomes of different sizes Michel (1996b). The selection scheme is an elitist model, that is, the best chromosome is always transferred to the next population while the percentage of chance of the other chromosomes to be chosen is proportional to their fitness value. At the beginning of the evolutionary process, each chromosome is randomly initialized.

The maximum values of the fields `Id` and `IdData`, $n-1$, corresponding to the maximum number of different proteins is computed using Kauffman's NK model of dynamical boolean networks, Kauffman (1993).

genomic network



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)

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

Let N be the number of elements (genes) of the genomic network. If each gene depends on K other genes (on the average), the corresponding genomic network will have N links and $N \div K$ vertices (see Fig. 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 the one hand, if $K = N$, the behaviour 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 chains* with very simple dynamics. The most fascinating case corresponds to $K = 2$ where interesting behaviours 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, Langton (1992).

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 increases. Hence, we could consider that, in a certain way, the size of the genes also increases along with the size of chromosomes, because they have to code for an increasing number of proteins. However, in practice, the protein values were stored as fixed size integers (i.e., 8 bits) for simplicity reasons.

6. Preliminary experiments

In order to test the capabilities of the evolutionary neurogenesis system, a fitness function relying only on the structure of the resulting neural networks was designed. The aim was to observe whether the evolutionary process could find an arbitrary neural structure.

6.1. Targeted neural structure

A fully connected two-layered neural network was chosen as a targeted neural structure. Eight input neurons and four output neurons are initially provided to the neurogenesis process. They are laid in two rows on the hexagonal grid. These units are differentiated in the sense that a special initial protein is set in the input units and a different initial protein is set in the output units. As shown in Fig. 6, the resulting neural network should have eight hidden units receiving full connection from the input layer and fully connected to the output layer.

6.2. Fitness function

In order to reach the goal described above, a custom fitness function was designed, depending only on neural structure information. Since the neurogenesis process may generate any recurrent neural architecture (except those where the input neurons receive incoming connections), some constraints have to be defined in the fitness function computation: Initially, the fitness value for a given neural network is set to zero. Then, for each input neuron, this value is incremented by 1 each time the input neuron is connected to a hidden neuron. Moreover it is decremented by 0.5 each time the input neuron is connected to an output neuron, to avoid shortcut connections. Hence the input layer may

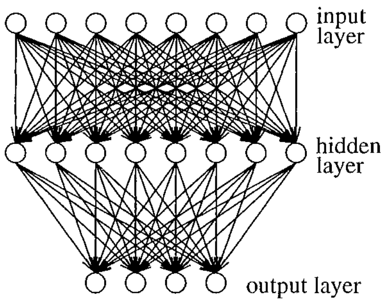


Figure 6. A two-layered, fully connected feedforward neural network as the goal of the evolutionary process

totalize a maximum fitness score of $8 \times 8 = 64$, each of the eight input neurons being connected to each of the eight hidden neurons. For the hidden layer, the same principle is applied: each connection between a hidden neuron and an output neuron is rewarded by 1 fitness point and each connection between two hidden neurons is sanctioned by 0.5 fitness point to avoid recurrent connections in the hidden layer. Hence the maximum fitness value for the hidden layer is $8 \times 4 = 32$. Finally, each connection leaving the output layer results in a decrease of 0.5 of the fitness function to avoid both recurrent connections within the output layer and also recurrent connections between the output layer and the hidden layer. The maximal fitness value for the output layer is then 0. Thus, the maximal overall fitness value for a perfect 8-8-4 fully connected feedforward neural network is $64 + 32 = 96$.

6.3. Results

At the beginning, an initial population of 100 individuals is randomly generated. Each chromosome is initially made up of 16 random genes. The evolutionary process found the solution it was constrained for (i.e, the best individual reaches the fitness of 96 after about 400 generations of the evolutionary algorithm). Fig. 7 illustrates the convergence of the fitness function during more than 1000 generations of the evolutionary algorithm. Note that, in this run, 400 generations were necessary to obtain the solution, but this value may be different if a different initial random population of genotypes is used. The successive fitness steps observed with the best individual correspond to a series of improving structures. Early structures do not have the right number of hidden units (i.e., typically less than 8), then, this number is progressively reached, after which the connections get organised to fit with the targeted neural structure described in Fig. 6.

Such a preliminary experiment demonstrates the capability of the evolutionary process to discover a particular architecture using a custom fitness function. It is likely that some other neural architectures might be achieved this way,

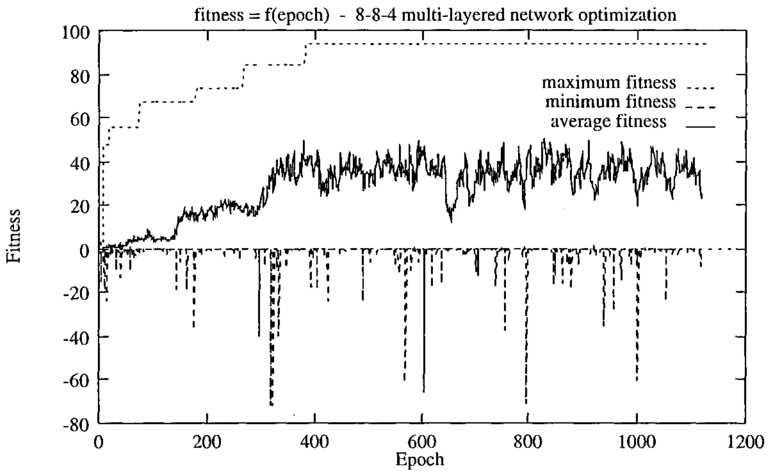


Figure 7. Fitness function evolution during the optimisation of the feedforward neural network structure with two layers (8 input units, 8 hidden units and 4 output units).

but the aim of this research is to generate neural architectures which are optimal for addressing a given problem and which do not necessarily fit structure constraints. Hence a second series of experiments, involving control of a mobile robot, was set up, in which the fitness function does not rely any more on structure information, but on the behaviour of the neural network.

7. Mobile robot control

7.1. Khepera robot and Khepera simulator

Experiments were driven on *Khepera Simulator*, a mobile robot simulator we developed, Michel (1996b), allowing for an easy transfer of the controllers to the real robot *Khepera*, developed by Franzi, Guignard and Mondada (1993) (K-Team).

The mobile robot includes 8 infrared sensors (small rectangles in Fig. 8) allowing it to detect the proximity of objects in front of it, behind it, and to the right and the left 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.

7.2. Interface to the artificial neural network

The neural network will read the sensors through its 8 inputs neurons and control the motors through its four output neurons. In order to connect the

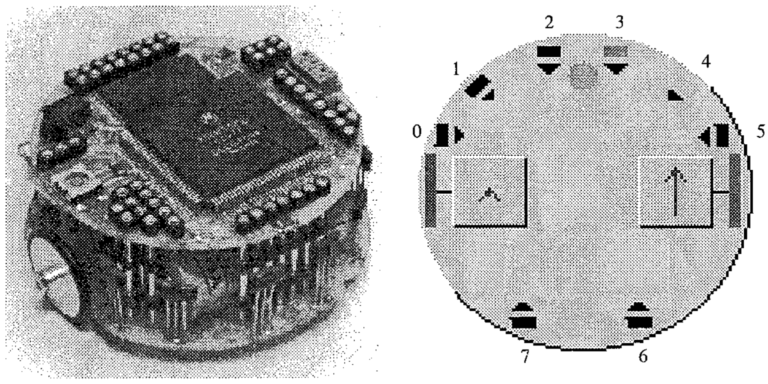


Figure 8. Khepera (5 cm diameter) and its simulated counterpart

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 neurogenesis 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 Fig. 9).

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.

7.3. Fitness function

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 Fig. 10). 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 sent back to the evolutionary algorithm corresponds to the distance between the inital position of the robot and the farthest point reached by the robot. The robot evaluation is stopped in two cases:

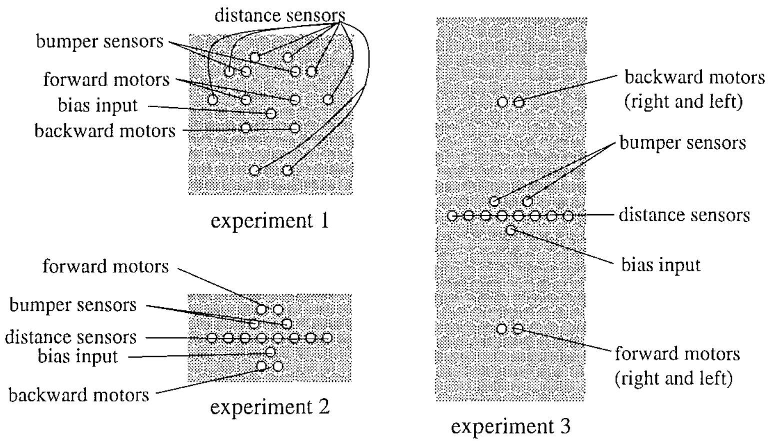


Figure 9. Initial positions of input and output neurons

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

7.4. Early results

Since the input distance neurons are not differentiated (i.e., they have the same numerical identifier), we can expect to observe a similar behaviour (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 us see whether the evolutionary process found a similar structure.

Like in the previous experiments, an initial population of 100 individuals is randomly generated where each chromosome is initially made up of 16 random genes.

After 200 generations of the evolutionary algorithm (corresponding to about 4 hours of running time on a Sun Sparc 20), different artificial neural networks were obtained that exhibited various performances 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 Fig. 11). The values 0.5 and 51 of the

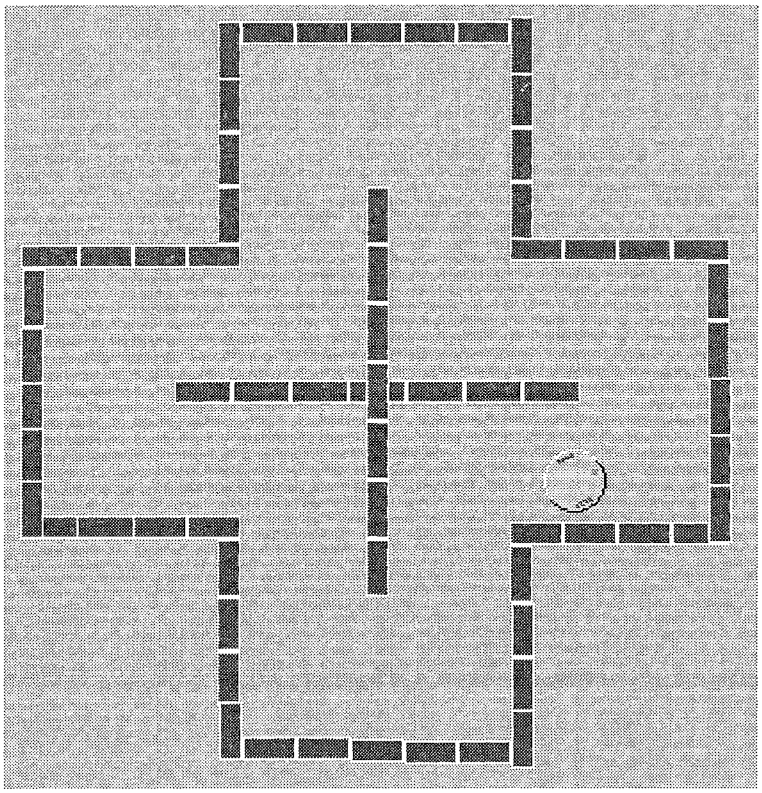


Figure 10. Cross-like maze

synaptic weights were chosen by the evolutionary process within a set of weight values given a priori. We were a bit disappointed since we had not expected that the evolutionary process would succeed in establishing different connections starting from the non-differentiated input distance 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 neighbourhood, leading to different dynamics inside the cell bodies.

In order to try to minimise the difference of behaviour between non-differentiated cells, the input cells were aligned in a layer as described in Fig. 9, 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 behaviour while slight differences in the connection patterns made the overall scheme perform almost as well as in the first experiment: the corresponding robots were able to travel in

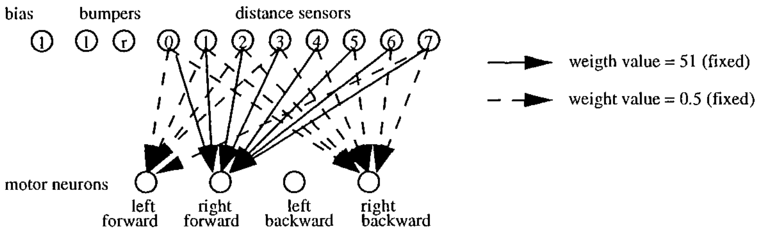


Figure 11. Resulting best neural network during experiment 1

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 which 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.

8. Conclusion

We have proposed an algorithm for evolutionary neurogenesis, applied to the control of a mobile robot, that is characterized by a strong inspiration from biology and the use of dynamical genomic network. Such a proposal leads to interesting early results: First, the evolutionary neurogenesis process proved to be able to generate particular neural structures when properly constrained with a custom fitness function. Second, the evolutionary process turned out to be able to find near-optimal architectures for a simple navigation task involving obstacle avoidance. Indeed, it was impossible for us to design an achitecture having a better fitness function than the one found by the evolutionary process. 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 behaviour, 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 do not need to make such errors since their adapted behaviour is innate. This demonstrates the ability of the neurogenesis 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.

A look back to biology might help to understand better what should happen in artificial system. If we consider primitive animals, like most insects, they 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 mammals 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 behaviours in their genotype. Moreover, for complex tasks (like walking with two legs), adaptive behaviours are far more efficient than hardwired behaviours.

This research should now address a new series of problems for which learning is a really mandatory issue for getting maximum fitness. Hence, the selective pressure will favour individuals able of on-line adaptation during their lifetime (i.e., evaluation time). It is likely that such a methodology might be able to produce more efficient autonomous agent controllers capable of facing more complex environments through the use of learning potentialities.

References

ANDERSON, J.A. and ROSENFELD, E., eds. (1988) *Neurocomputing: Foundations of Research.* MIT Press, Cambridge.

BÄCK, T., FOGEL, D. and MICHALEWICZ, Z. (1997) *Handbook of Evolutionary Computation.* Oxford University Press.

BOERS, E.J.W. and KUIPER, H. (1992) 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.

BRAITENBERG, V. (1984) *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge.

BURNOD, Y. (1989) *An adaptative neural network: the cerebral cortex.* Masson.

DELLAERT, F. and BEER, R.D. (1994) Toward an evolvable model of development for autonomous agents synthesis. In: Brooks, R.A. and Maes, P., eds., *Proceedings of the Fourth International Workshop on Artificial Life*, 246–257, Cambridge, MA. The MIT Press / Bradford Books.

DE GARIS, H. (1996) 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: *Towards Evolvable Hardware*, 76–98. Springer.

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

FOGEL, D.B., WASSON, E.C., BOUGHTON, E.M. AND PORTO, V.W. (1997) Initial results of training neural networks to detect breast cancer using evolutionary programming. Published in this special issue of *Control and Cybernetics.*

GRUAU, F. and WHITLEY, D. (1993) Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect. *Evolutionary Computation*, **1**, 3, 213–234.

HARVEY, I. (1993) *The Artificial Evolution of Adaptive Behavior.* PhD thesis, University of Sussex.

HEBB, D.O. (1949) *The Organization of Behavior.* Wiley, New York.

KAUFFMAN, S.A. (1993) *The Origins of Order: Self-Organisation and Selection in Evolution.* Oxford University Press.

KITANO, H. (1990) Empirical studies on the speed of convergence of neural network training using genetic algorithms. In: *Proceedings AAAI*, 789–795.

KITANO, H. (1995) Cell differentiation and neurogenesis in evolutionary large scale chaos. In: Morán, F., Moreno, A., Merelo, J.J. and Chacón, P., eds., *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada.* Springer.

KODJABACHIAN, J. and MEYER, J.-A. (1994) Development, learning and evolution in animats. In: Gaussier, P. and Nicoud, J.-D., eds., *Perception To Action Conference Proceedings*, 96–109. IEEE Computer Society Press.

KOHONEN, T. (1989) *Self-Organization and Associative Memory.* Springer-Verlag (3rd ed.).

KANDEL, E.R. and SCHWARTZ, J.H. (1982) Molecular biology of an elementary form of learning: modulation of transmitter release through cyclic amp-dependent protein kinase. *Science*, **218**, 433–443.

LANGTON, C.G., ed. (1988) *Artificial Life, proceedings of the First International Conference on Artificial Life.* Addison-Wesley.

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

LE CUN, Y., BOSER, B., DENKER, J.S., HENDERSON, D., HOWARD, R.E., HUBBARD, W. and JACKEL, L.D. (1990) Handwritten digit recognition with a back-propagation network. In: Touretzky, D.S., ed., *Advances in Neural Information Processing Systems II.* Morgan Kaufmann.

LINDENMAYER, A. (1968) Mathematical models for cellular interactions in development, I & II. *Journal of Theoretical Biology*, **18**, 280–315.

MICHEL, O. and BIONDI, J. (1995) Morphogenesis of neural networks. *Neural Processing Letters*, **2**, 1, 9–12, January.

MONDADA, F., FRANZI, E. and IENNE, P. (1994) Mobile robot miniaturisation: A tool for investigation in control algorithms. In: Yoshikawa, T. and Miyazaki, F., eds., *Third International Symposium on Experimental Robotics 1993*, Kyoto, Japan.

MICHEL, O. (1996A) An artificial life approach for the synthesis of autonomous agents. In: Alliot, J.-M., Lutton, E., Ronald, E., Schoenauer, M. and Snyers, D., eds., *Artificial Evolution*, **1063**, *Lecture Notes in Computer Science*, 220–231. Springer Verlag.

MICHEL, O. (1996B) *Expériences en Neuroéthologie Artificielle - EVOTS : une méthodologie évolutionniste appliquée en robotique mobile.* PhD thesis, University of Nice - Sophia Antipolis.

MICHEL, O. (1996C) Khepera simulator package version 2.0. Freeware mobile robot simulator downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`, 1996.

NOLFI, S. and PARISI, D. (1995) Evolving artificial neural networks that develop in time. In: Morán, F., Moreno, A., Merelo, J.J. and Chacón, P., eds., *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada*, 353–367. Springer.

TURNEY, P., WHITLEY, D. and ANDERSON, R. (1997) Evolution, learning and instinct: 100 years of the Baldwin effect. *Evolutionary Computation*, **4**, 3.

VAARIO, J. and SHIMOHARA, K. (1995) On formation of structures. In: Morán, F., Moreno, A., Merelo, J.J. and Chacón, P., eds., *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life, Granada*, 421–435. Springer.