Evolutionary Approaches to Walking and Higher-Level Behaviors in 6-Legged Animats

JEAN-ARCADY MEYER
AnimatLab
Ecole Normale Supérieure
46 rue d'Ulm
75230 Paris Cedex 05
France
meyer@wotan.ens.fr

1 Introduction

This article describes the main current research project in evolutionary robotics at the AnimatLab, Paris. It aims at using an artificial selection process to automatically generate neural networks that control walking animats, i.e., simulated insects or real legged-robots. Essentially, it complements an underlying evolutionary process with a developmental procedure - in order to reduce the size of the genotypic space that is explored - and it calls upon an incremental approach - in order to capitalize upon previously found solutions to simpler problems for solving problems of increasing difficulties.

This article will successively outline the historical background of our research, the evolutionary paradigm on which this research relies, and the physical model of artificial insect that we're using. Then it will summarize the main results we have obtained so far and indicate directions for future work.

2 Background

Several research efforts have recently aimed at evolving locomotion controllers for animats with two [9], four [9, 3, 22, 30] or eight [10, 13, 12, 14] legs. Such controllers are traditional programs [10], neural networks [9, 3, 22, 12], classifier systems [6] or Lisp-like programs [30]. Likewise, some of these approaches [9, 3, 30, 6] call upon simulations, while others are performed on real robots [22, 10, 12, 13]. However, all these research efforts are targeted at evolving straight-line walking and problems like direction control, speed control, or rough terrain locomotion remain largely unsolved by the corresponding approaches, not mentioning higher-level tasks like obstacle-avoidance, goal-seeking or pursuit-evasion.

At the AnimatLab, we are trying to extend current results in these directions by drawing inspiration from biology [25] and exploiting the potentialities of a methodology that makes the evolutionary task easier, thus allowing more complex problems to be tackled. Such a methodology employs an indirect genotype-to-phenotype mapping that inserts a developmental process between an animat's genotype - i.e., the information that evolves from generation to generation - and its phenotype - i.e., the animat's nervous system. Therefore, it avoids the main drawbacks incurred by direct mappings, i.e., their lack of scalability - according to which the evolutionary algorithm explores a genotypic space that grows bigger and bigger as the phenotypic solutions sought get more and more complex - and their inaptitude to generate modular architectures - which allow for repeated substructures that help encoding complex controller architectures in compact genotypes.

It has been shown in [17] that indirect encoding schemes currently rely on four different paradigms for modeling development: rewriting rules [4, 11], axonal growth processes [31, 26], genetic regulatory networks [8], and nested directed graphs [29]. Basically, our SGOCE[1] methodology, which entails adding an axonal growth process to the cellular encoding scheme described in [11], is a combination of the two first above-mentioned approaches and is used to evolve developmental rules. Such rules, in turn, are used to develop a neural network module that gets connected to an animat's sensors and actuators in order to control a given behavior and to provide for a given competency. Additional behaviors or competencies can be successively dealt with by evolving new developmental programs, which not only create new modules with new sensori-motor connections, but also generate inter-modular connections that secure an adapted and coherent overall functioning. Eventually, such an incremental methodology generates control architectures that are strongly reminiscent of Brook's subsumption architectures [5].

[1]This name is the acronym of the expression Simple Geometry-Oriented variation of Cellular Encoding.

3 The SGOCE evolutionary paradigm

This paradigm is characterized by an encoding scheme that relates the animat's genotype and phenotype, by syntactic constraints that limit the complexity of the developmental programs generated, by an evolutionary algorithm that generates the developmental programs, and by an incremental strategy that helps producing neural control architectures likely to exhibit increasing adaptive competencies. The neural architectures thus produced are general recurrent neural networks controlling the behavior of the animat and grown from a few initial cells provided by the experimenter.

These neural networks are made of individual neurons each behaving as a leaky integrator [28] - i.e., as a universal dynamics approximator, likely to approximate the trajectory of any smooth dynamic system [2]. Thus, the mean membrane potential m_i of a neuron N_i evolves according to:

$$\tau_i \cdot dm_i/dt = -m_i + \sum w_{i,j} x_j + I_i$$

where $x_j = (1 + e^{-(m_j + B_j)})^{-1}$ is the neuron's short-term average firing frequency, B_j is a uniform random variable whose mean b_j is the neuron's bias, and τ_i is the time constant associated with the passive properties of N_i's membrane. I_i is the input that neuron N_i may receive from a given sensor, and $w_{i,j}$ is the synaptic weight of a connection from neuron N_j to neuron N_i.

3.1 The developmental code

Our encoding scheme calls upon developmental rules that are encoded into artificial tree-like chromosomes [20, 21] that contain two categories of instructions. Some of them specify morphological transformations and apply to specific cells (Table 1), while others are used to generate structured developmental programs.

This scheme also employs a physical two-dimensional substrate in which the experimenter initially arranges a set of sensory cells - that may be connected to the animat's sensors -, a set of motoneurons - that may be connected to the animat's actuators -, and a set of precursor cells - from which the developmental process will be initiated (Figure 1).

Each precursor cell is given a copy of the robot's genotype and, as it executes the corresponding program, divides, grows connections to other cells, differentiates into a functional neuron, or dies (Figures 2 and 3).

DIVIDE α r	create a new cell
GROW α r w	create a connection to another cell
DRAW α r w	create a connection from another cell
SETBIAS b	modify the bias parameter
SETTAU τ	modify the time constant parameter
DIE	trigger cellular death

Table 1: The basic developmental instruction set. α, r, w, b and τ are parameters whose roles are described in the text. Additional developmental instruction will be introduced later in the text.

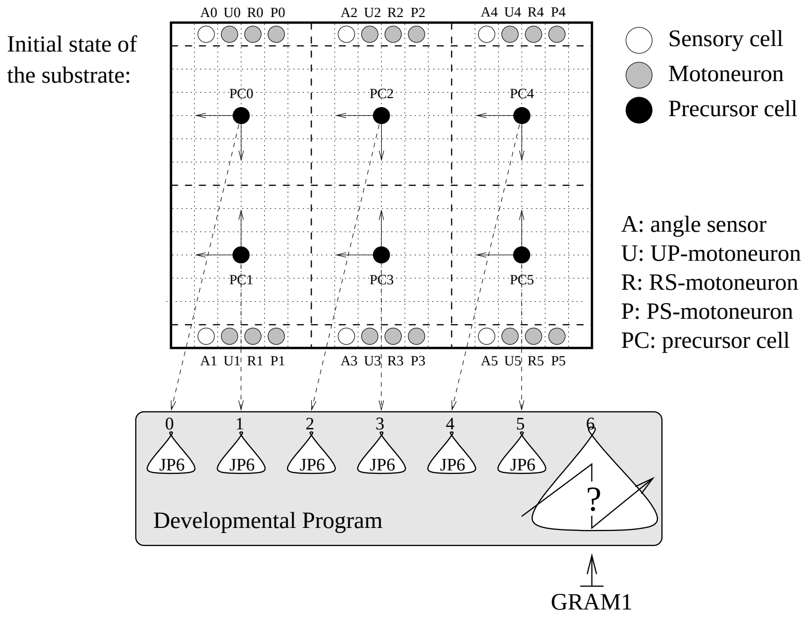


Figure 1: Setup for the evolution of a straight locomotion controller for a six-legged animat. The figure shows the initial positions of the sensory cells, motoneurons and precursor cells provided by the experimenter. Sensory cells and motoneurons are connected to the proprioceptors and muscles that are associated to each leg in the physical model of Figure 8. The figure also specifies the structure of the developmental programs, each calling upon seven subprograms. JP is a call instruction that forces a cell to start reading a new subprogram. Only subprogram 6 needs to be evolved. It's organization is constrained by the GRAM1 grammar, as explained later in the text.

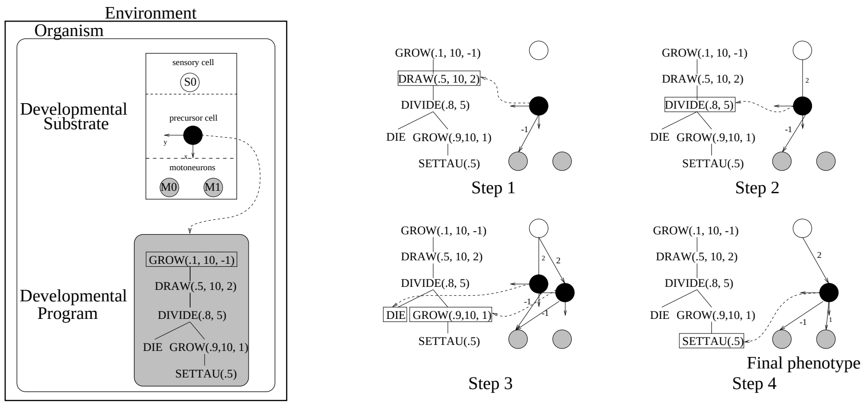


Figure 2: While reading the genotype, a precursor cell generates a neural network after several developmental steps. This network may involve the sensory cells and the motoneurons made available by the experimenter. Each precursor cell is associated with a frame of reference that is inherited by its daughter cell when division occurs.

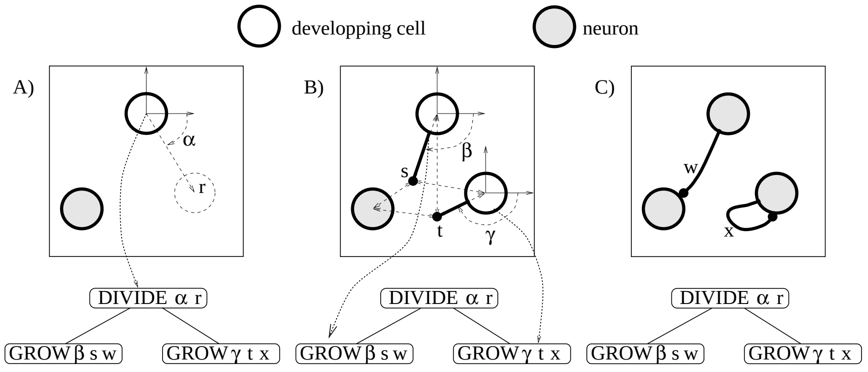


Figure 3: A) When a mother cell divides, a daughter cell is created in a given direction and at a given distance specified by the parameters of the DIVIDE instruction. B) Depending on the values of the arguments of GROW instructions, targets for connections are sought in a given direction and at a given distance in the local framework associated with the acting cell. C) These connections link two different neurons or correspond to self-connections. They can also regress and die when their targets lie outside the developmental substrate.

At the end of such process, a complete neural controller is obtained, whose architecture reflects the geometry and symmetries initially imposed by the experimenter, to a degree that depends on the side-effects of the developmental instructions that have been executed. This controller is connected to the animat's proprioceptive and exteroceptive sensors, as well as to the animat's muscles, through connections to the sensory cells and motoneurons that have been incorporated in its architecture. This, together with the use of an appropriate fitness function, makes it possible to assess the controller's capacity to generate the specific behavior sought by the experimenter.

3.2 Syntactic constraints

In order to reduce the size of the genotypic search-space and the complexity of the generated networks, we constrain the structure of the corresponding developmental programs by requiring that all evolving subprograms be well-formed trees according to given context-free tree-grammars like the GRAM1 grammar of Figure 4.

Additionally, such grammars make it possible to control the nature and size of the program modifications that occur between two successive generations, through the use of genetic operators like mutation or crossing-over. The effect of mutation is to change an instruction, or one of its arguments, into another instruction or argument. The effect of crossing-over is to exchange a sub-tree in a developmental program with a sub-tree in another developmental program.

3.3 Evolutionary algorithm

To slow down convergence by favoring the creation of ecological niches, we use a steady-state evolutionary algorithm that involves a population of N randomly generated well-formed programs distributed over a circle and whose mode of operation is outlined in Figure 5.

The following procedure is repeated until a given number of individuals have been generated and tested:

1. A position P is chosen on the circle.
2. A two-tournament selection scheme is applied in which the better of two programs randomly selected from the neighborhood of P is retained[2].

[2] A program's probability p_s of being selected decreases with the distance d to P: $p_s = max(R - d, 0)/R^2$, with R=4. Programs for which d is greater than or equal to R cannot be selected ($p_s = 0$).

Terminal symbols

DIVIDE, GROW, DRAW, SETBIAS, SETTAU, DIE,

NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.

Variables

Start1, Level1, Level2, Neuron, Bias, Tau, Connex, Link.

Production rules

Start1$\longrightarrow$DIVIDE(Level1, Level1)

Level1$\longrightarrow$DIVIDE(Level2, Level2)

Level2$\longrightarrow$DIVIDE(Neuron, Neuron)

Neuron$\longrightarrow$SIMULT3(Bias, Tau, Connex) | DIE

Bias$\longrightarrow$SETBIAS | DEFBIAS

Tau$\longrightarrow$SETTAU | DEFTAU

Connex$\longrightarrow$SIMULT4(Link, Link, Link, Link)

Link$\longrightarrow$GROW | DRAW | NOLINK

Starting symbol

Start1.

Figure 4: The GRAM1 grammar. The set of terminal symbols consists of the developmental instructions listed in Table 1 and of additional structural instructions that have no side-effect on the developmental process. NOLINK is a “no-operation” instruction. DEFBIAS and DEFTAU leave the default values of parameters b and τ unchanged. SIMULT3 and SIMULT4 are branching instructions that allow the sub-nodes of their corresponding nodes to be executed simultaneously. The introduction of such instructions makes it possible for the recombination operator to act upon whole interneuron descriptions, or upon sets of grouped connections, and thus hopefully to exchange meaningful building blocks.

3. The program selected is allowed to reproduce, and three genetic operators may modify it. The recombination operator is applied with probability p_c. It exchanges two compatible [3] sub-trees between the program to be modified and another program selected from the neighborhood of P. Two types of mutation are used. The first mutation operator is applied with probability p_m. It changes one randomly selected sub-tree into another compatible, randomly generated

[3]Two sub-trees are compatible if they are derived from the same grammatical variable, like Start1, Level1, etc. in Figure 4.

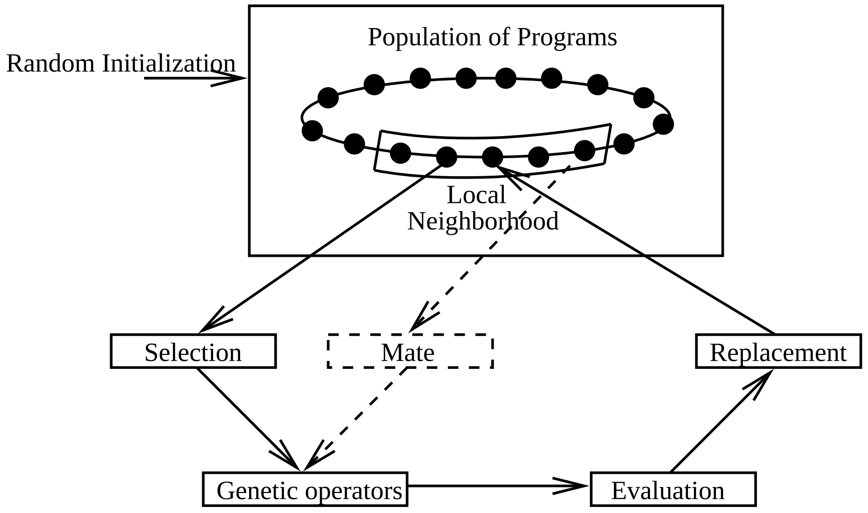


Figure 5: The evolutionary algorithm (See text for explanation).

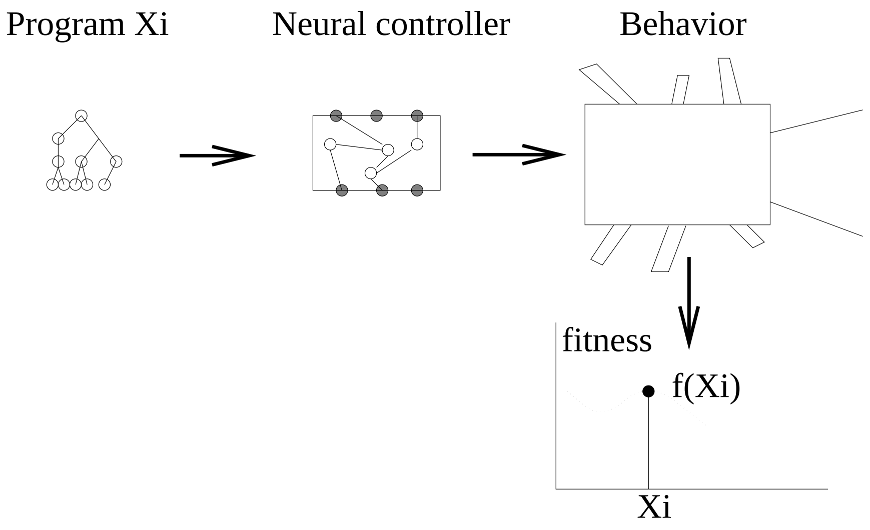


Figure 6: The three stages of the fitness evaluation procedure of an evolved developmental program (X_i). First, the program is executed to yield an artificial neural network. Then the neural network is used to control the behavior of a simulated animat that has to solve a given task in an environment. Finally, the fitness of Program X_i is assessed, according to how well the task has been solved.

one. The second mutation operator is applied with a probability of 1 and modifies the values of a random number of parameters, implementing what Spencer called a *constant perturbation strategy* [30]. The number of parameters to be modified is drawn from a binomial distribution $\mathcal{B}(n, p)$.

4. The fitness of the new program is assessed by collecting statistics while the behavior of the animat controlled by the corresponding artificial neural network is simulated over a given period of time (Figure 6).
5. A two-tournament anti-selection scheme, in which the worst of two randomly chosen programs is selected, is used to decide which individual (in the neighborhood of P) will be replaced by the modified program.

Typical values for the above mentioned parameters are $p_c = 0.6$, $p_m = 0.2$, $n = 6$ and $p = 0.5$.

3.4 Incremental approach

The SGOCE paradigm resorts to an incremental approach that takes advantage of the geometrical nature of the developmental substrate to generate and connect successive neural modules implementing different competencies.

As an example, such a paradigm makes it possible to first evolve a neural network that controls mere locomotion in a 6-legged animat and then to evolve other neural modules that control higher-level behaviors. These modules may influence the locomotion module by creating inter-modular connections. For instance, Figure 7 shows how the successive connection of two additional modules with a locomotion controller was used to first generate a goal-seeking behavior (Subsection 5.4) and then to generate additional obstacle-avoidance capacities (Subsection 5.5).

4 The insect's physical model

The experimental results to be described herein made use of the so-called SWAN[4] model of a simulated insect [15], which is strongly inspired by the work of Beer [1, 3]. According to this model, each of the 6 legs of the animat is equipped with two pairs of muscles that allow both its angular position and the height of its foot to be controlled (Figure 8).

[4]This name is the acronym of the expression Simulated Walking ANimat.

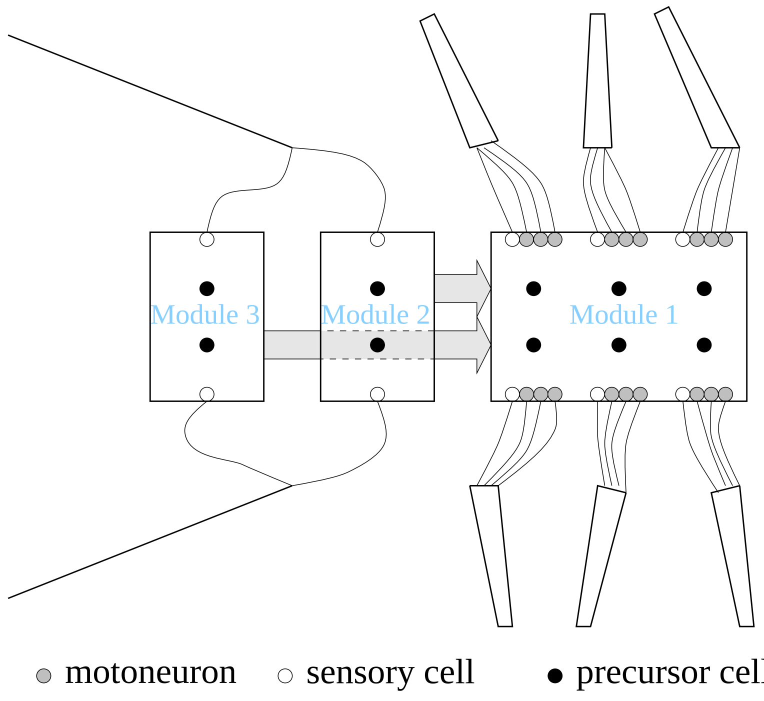


Figure 7: An instantiation of the SGOCE incremental approach. During a first evolutionary stage, Module 1 is evolved. This module receives proprioceptive information through sensory cells and influences actuators through motoneurons. In a second evolutionary stage, Module 2 is evolved. This module receives specific exteroceptive information through dedicated sensory cells and can influence the behavior of the animat by making connections with the neurons of the first module. Finally, in a third evolutionary stage, Module 3 is evolved. Like Module 2, it receives specific exteroceptive informations and it influences Module 1 through inter-incremental connections. In the application to be described later, no connections between Module 2 and Module 3 are allowed but such a constraint could easily be relaxed in other applications.

For three of those muscles, a corresponding motoneuron specifies the value of the resting length parameter in a simple muscle model. Furthermore, each leg is equipped with a sensor that returns the leg's angular position θ. Thus the available motors and sensors correspond to those of Beer and Gallagher's simulated insect [3]. However, a difference to these authors's scheme is that the foot status (up or

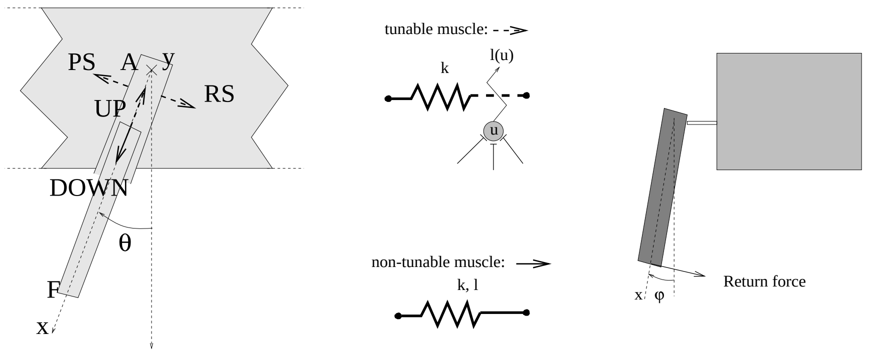


Figure 8: The SWAN model. Left: Each leg has three degrees of freedom. Thanks to the antagonistic PS- (Power Strike) and RS- (Return Strike) muscles the leg can rotate around an axis (Ay) orthogonal to the plane of the figure. The instantaneous value of the θ angle, which measures how far the leg is positioned forward or backward, determines the activity level of the angle sensor in Figure 1. The UP- and DOWN-muscles allow the position of the foot F to be translated along the (Ax) axis. The resting lengths of the tunable PS-, RS- and UP-muscles depend on the activity levels u of the motoneurons of Figure 1. The resting length of the DOWN-muscle is supposed to be non-tunable. Middle: A muscle is modeled as a spring of fixed stiffness k and of (possibly variable) resting length. Right: If a leg deviates from the vertical plane parallel to the body axis, a return force proportional to the deviation tends to bring it back to the vertical plane.

down) is not determined by the state of the corresponding UP-motoneurons only. More realistically, these positions are also influenced by the dynamics of the physical model of the animat.

Additionally, depending upon the activity level of the PS-motoneurons, forces acting on the animat's body can be greater on one side than on the other. This entails leg displacements from the vertical axis and the triggering of return forces that are responsible for rotations.

Finally, the SWAN model allows for monitoring the animat's overall equilibrium. When the animat sets upright after having fallen, the weight of its body opposes the force exerted by the UP-muscles, thus lengthening the return to stability.

5 Experimental results

5.1 Straight-line locomotion

In order to evolve neural controllers for straight-line locomotion and to reduce the size of the search space, we looked for controllers made of six sub-networks grown according to the instructions of a unique developmental subprogram as described in Figure 1. Within the given two-dimensional substrate, six precursor cells called six associated subprograms that, in turn, each called the developmental subprogram. The positions and the local frames of the different precursor cells reflected the bilateral symmetry of the animat's morphology. The motoneurons and sensory cells of each leg had specific coordinates in the local frame associated to the corresponding precursor cell. The execution of the whole developmental program resulted in the creation of a neuro-controller made of six interconnected sub-networks. According to such a logic, only the developmental subprogram had to be evolved.

The fitness function was the distance covered during the evaluation augmented by a term encouraging any leg motion:

$$f = x(T_{max} + \int_{t=0}^{T_{max}} \sum_p |\frac{d\theta_p}{dt}(t)| + \sum_p |\frac{dh_p}{dt}(t)|)dt$$

where $x(t)$ is the position of the animat's center of mass at time t, T_{max} is the evaluation time, and $\theta_p(t)$ and $h_p(t)$ are the angular position and the height of leg p at time t [15]. We did not apply an explicit selection pressure for not falling. However, falls were implicitly penalized because they slowed down locomotion.

We made a series of 5 experiments. In each experiment, 100.000 offspring events were made in a population of 200 programs with different randomly generated developmental subprograms well-formed according to GRAM1 (Figure 4).

Several kinds of walking strategies were obtained. In four experiments, symmetric gaits — in which both sides were moved synchronously — were generated. The corresponding behaviors consisted in making a succession of leaps, using groups of 2, 4 or 6 legs together. None of those gaits was stable because of the high mass of the modeled body. However, in the course of Experiment 2, a stable, non-symmetric tripod gait was obtained (Figure 9). This solution covered the longest distance during the given evaluation time. External feedback provided by the sensors was used only by the controllers that evolved during Experiment 5.

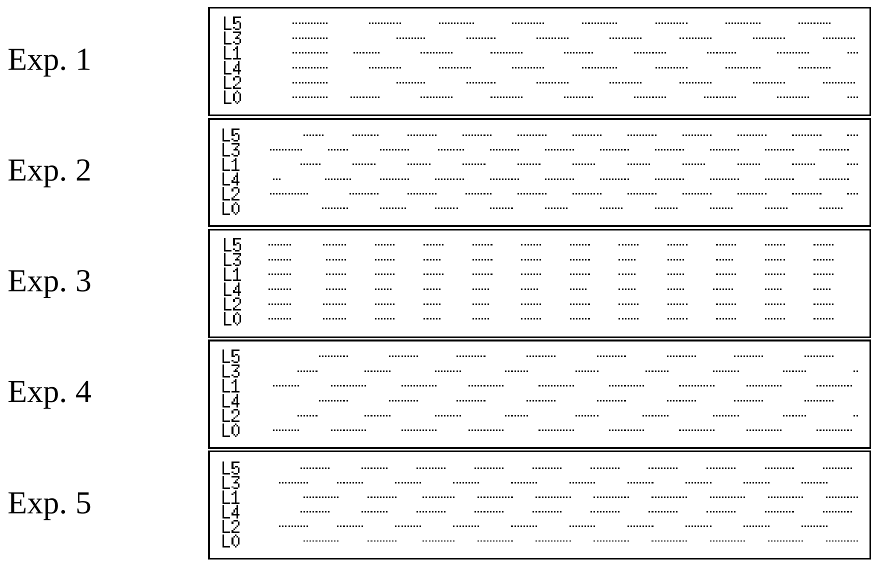
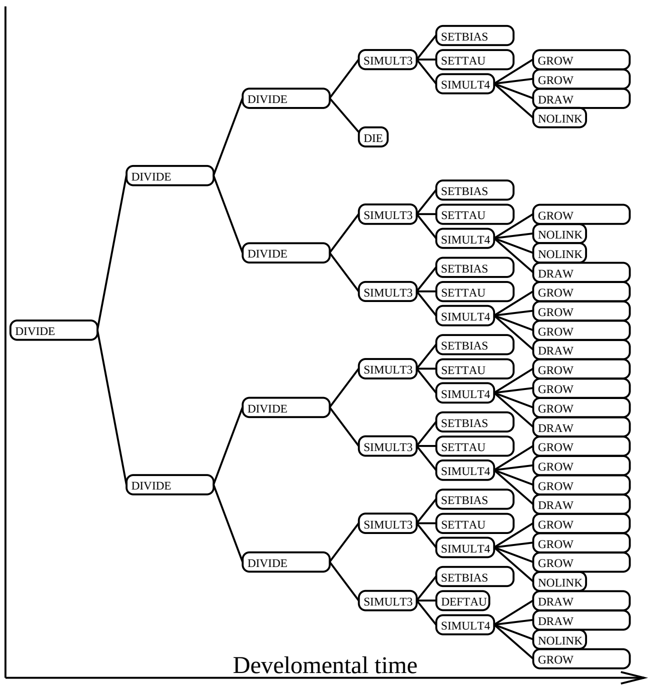


Figure 9: Best gait in the final population for each of the five experiments. The horizontal axis represents time. A dot is plotted when the corresponding leg is raised. Legs are numbered as in Figure 1. Only the results of Experiment 2 correspond to a stable, tripod gait. All other experiments in the series led to unstable, leaping behaviors.

Figure 10 shows the developmental subprogram and the architecture of the corresponding network for the best individual found in Experiment 2. As described in [16], this network implements four central pattern generators that are responsible for the rhythmic movements of the middle and back legs. Suitable connections are responsible for the synchronization of each tripod, according to which the front and back legs on each side of the animat are moved in synchrony with the middle leg of the opposite side. Likewise, other connections are making for phase opposition in the rhythms of the two opposite tripods.

5.2 Rough-terrain locomotion

Similar experiments are actually performed with animats that are committed to walk on rough terrain (Figure 11). Preliminary results indicate that the evolved controllers have a strong tendency to use external feedback to avoid too many falls.



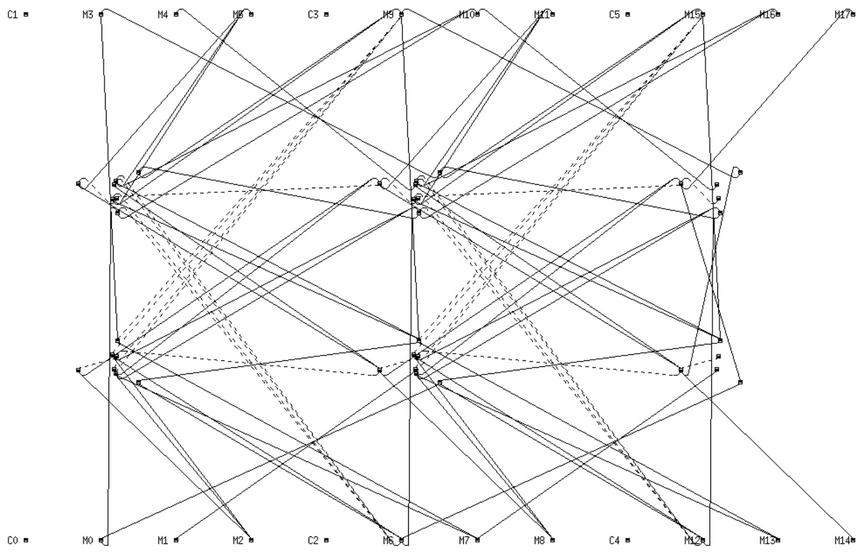


Figure 10: Top: The best developmental subprogram found in Experiment 2 (parameter values are not shown). This subprogram fits the GRAM1 grammar of Figure 4 and generates a tripod gait. It will be called LOCO1 thereafter. Bottom: The corresponding artificial neural network after useless interneurons and connections have been pruned. Solid lines are excitatory connections, dotted lines are inhibitory connections. Fan-in connection arrive at the top and fan-out connections depart from the bottom of each neuron. The network contains 38 interneurons and 100 connections.

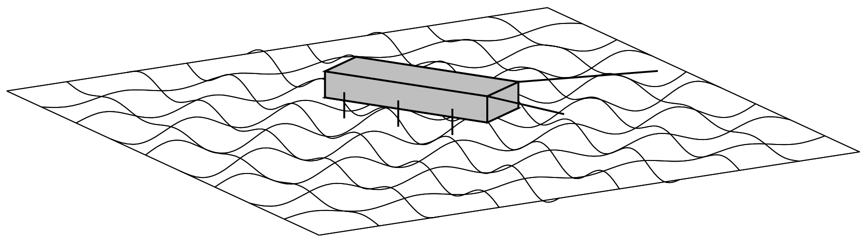


Figure 11: Rough-terrain locomotion.

5.3 Speed control

Having secured mere locomotion by the LOCO1 module described in Figure 10, two experiments were performed that sought to generate a second module capable of activating or stopping walking, and even of controlling the animat's speed.

5.3.1 Boolean stimulus

In the first experiment, a two-module neural network that was able to generate walking or resting according to the value of a boolean command input was obtained. The input value was fixed by the experimenter and was communicated to the system through a specific sensory cell called a control unit. Figure 12 shows the initial conditions for the developmental process and the general structure of the programs.

Assuming that a simple architecture would solve the task, the precursor cells of the second module were connected by default to the control unit at the beginning of the developmental process by *ad hoc* DRAW instructions. These cells were not allowed to divide, to create other intra-modular connections or to modify their bias parameters. Thus, only inter-modular connections toward the first module were allowed. In such conditions, the only task of the evolutionary process was to find a set of connections able to inhibit the locomotion behavior when the value of the command input was maximal (True). Whenever the command input value was zero (False), the neurons of the second module were not activated — because of the specific default value of their bias parameter — and the first module generated the default locomotion behavior.

A new developmental instruction (GROW2) was used to create an afferent connection from a cell in the second module to a cell in the locomotion module. This instruction worked like instruction GROW except that the geometric parameters were

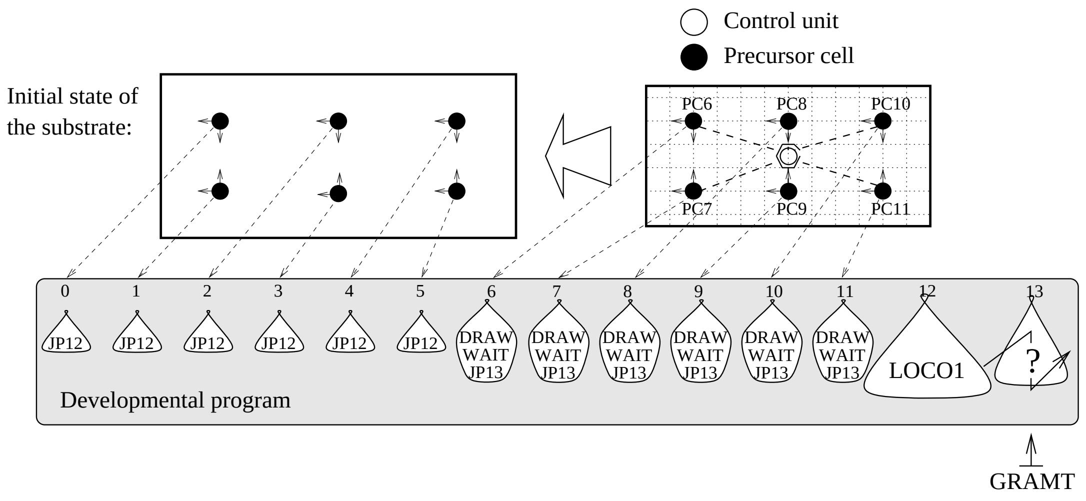


Figure 12: Setup for the evolution of a command network. The figure shows the initial positions of the control unit and the precursor cells, as well as the structure of the developmental programs that call upon 14 subprograms. DRAW instructions (followed by appropriate parameters) are added to create connections (dashed lines) from the control unit to the precursor cells 6 to 11. WAIT is a no-operation instruction used to delay the call of subprogram 13 in order to let time for the three successive divisions of the precursor cells 0 to 5 to occur before instruction GROW2 can be executed by the precursor cells 6 to 11. Subprogram 12 has been evolved in the previous experiments. Only subprogram 13 needs to be evolved.

interpreted in the local frame's projection into the locomotion module. The grammar GRAMT (Figure 13) defined a set of well- formed subprograms liable to create up to 8 connections from a precursor cell of the second module into the locomotion module.

During an evaluation, the value of the command input was successively set to False, True, False, True and False. The fitness function rewarded individuals for not moving and for standing when the command was True:

$$f = \int_{t=0}^{T_{max}} r(t) \cdot dt$$

$r(t) = (k \cdot s(t) - |v(t)|)$ if True and $r(t) = 0$ otherwise;

where $v(t)$ is the speed of the animat's center of mass at time t, k is a weighting coefficient set to 0.01 in the experiments described herein, and $s(t)$ is 1 if the animat

Terminal symbols

GROW2, NOLINK, SIMULT8.

Variables

Start2, Link2.

Production rules

Start2$\longrightarrow$SIMULT8(Link2, Link2, Link2, Link2, Link2, Link2, Link2, Link2)

Link2$\longrightarrow$GROW2 | NOLINK

Starting symbol

Start2.

Figure 13: The grammar GRAMT that is used to evolve a locomotion command module reacting to a boolean stimulus.

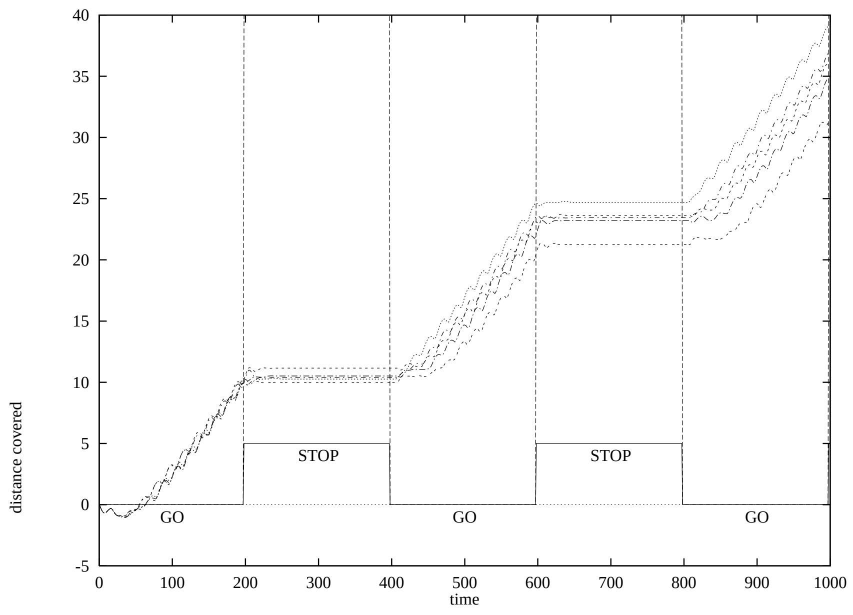


Figure 14: Distance covered as a function of time by the best controllers in the final population for five different experiments. The boolean command input is True between cycles 200 and 400 and between cycles 600 and 800.

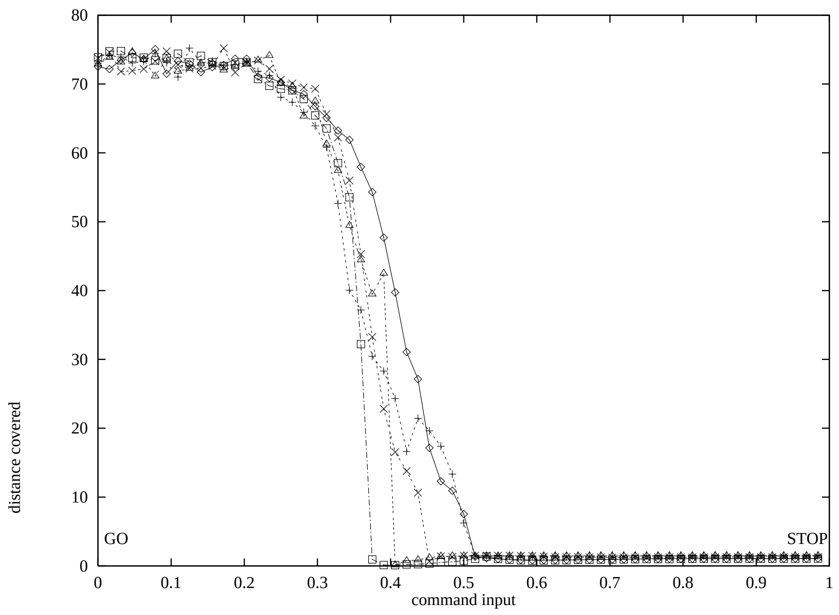


Figure 15: Distance covered during 1000 cycles when a fixed continuous command value is applied. Each curve represents the mean distance covered in ten runs.

is stable at time t and 0 otherwise. No reward was granted while the command was False.

We ran five experiments in which 20.000 offspring events were performed with a population of 200 individuals. In each experiment highly rated controllers were found. Figure 14 illustrates the corresponding behaviors.

To check whether such controllers could generate variable speeds, we submitted them to intermediate command values. Results shown in Figure 15 indicate that, when the control unit is clamped to a value comprised between (about) 0.2 and 0.5, the animat walks at a reduced speed. Beyond 0.5, walking is inhibited. Closer inspection of the inner workings of the controller reveals that this result is due to a decrease in the animat's step size, and not to a change in the rhythm of its basic oscillators, as described in [16].

5.3.2 Transient stimuli

In a second experiment, a two-module neural network able to generate walking or resting according to the values of two transient stimuli, $S1$ and $S2$, has been obtained.

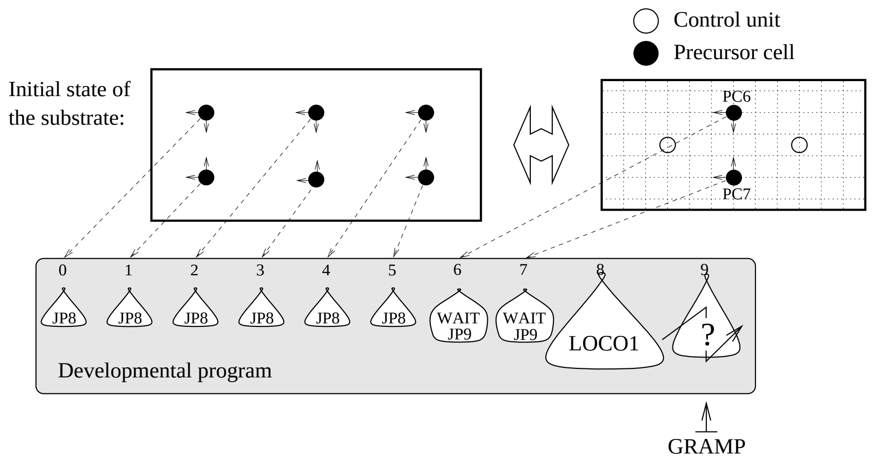


Figure 16: Setup for the evolution of a switching mechanism. The figure shows the initial positions of the control units and the precursor cells, as well as the structure of the developmental programs that call upon 10 subprograms. Subprogram 8 has been evolved in the previous experiments. Only subprogram 9 needs to be evolved.

These stimuli were delivered by the experimenter through two specific control units.

For this task, intra-modular divisions and connections inside the second module were allowed. No connections were grown by default and all the bias were allowed to evolve. A new developmental instruction called DRAW2 was introduced. It caused the creation of an afferent connection from a cell of the first module to the executing cell, and worked like the DRAW instruction, except that it was interpreted within the projection of the local frame into the second module. Furthermore only two precursor cells were placed in the second module. In such conditions, the walking behavior generated by the first module was liable to be perturbed even in the absence of control signals. Figure 16 displays the initial setup for the developmental process.

A new grammar, GRAMP, defined the set of valid subprograms that described the developmental process of a precursor cell of the second module. Such subprograms can create up to 4 neurons and 8 connections (Figure 17).

In order to evolve a switching mechanism, we had to design a conditional evaluation procedure, in which each individual could be evaluated up to three times in different conditions and with different fitness functions. This was necessary to avoid the networks learning to predict the time of occurrence of the stimuli.

In the first evaluation, we checked that the individual had not lost its walking

Terminal symbols

DIVIDE, GROW, DRAW, GROW2, DRAW2, SETBIAS, SETTAU, DIE,

NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.

Variables

Start3, Levelb, Neuronb, Biasb, Taub, Connexb, Linkb.

Production rules

Start3$\longrightarrow$DIVIDE(Levelb, Levelb)

Levelb$\longrightarrow$DIVIDE(Neuronb, Neuronb)

Neuronb$\longrightarrow$SIMULT3(Biasb, Taub, Connexb) | DIE

Biasb$\longrightarrow$SETBIAS | DEFBIAS

Taub$\longrightarrow$SETTAU | DEFTAU

Connexb$\longrightarrow$SIMULT4(Linkb, Linkb, Linkb, Linkb)

Linkb$\longrightarrow$GROW | DRAW | GROW2 | DRAW2 | NOLINK

Starting symbol

Start3.

Figure 17: The grammar GRAMP that is used to evolve a locomotion command module reacting to two transient stimuli.

ability:

$$f = \int_{t=0}^{T_{max}} v(t) \cdot dt = x(T_{max})$$

In case the corresponding individual walked along a minimum distance, he was allowed to go through the next evaluations. In the second evaluation, stimulus $S1$ was presented on the first control unit and the animat had to stop its progression. It was rewarded according to the previously used fitness function:

$$f = \int_{t=T_{S1}}^{T_{max}} (k \cdot s(t) - |v(t)|) \cdot dt$$

Finally, if the individual received a high enough rate, it was allowed to undergo the third evaluation. During this last evaluation, stimulus $S2$ was presented on the second control unit some time after stimulus $S1$ had been presented on the first one and the animat was thereafter rewarded for resuming walking:

$$f = \int_{t=T_{S2}}^{T_{max}} v(t) \cdot dt$$

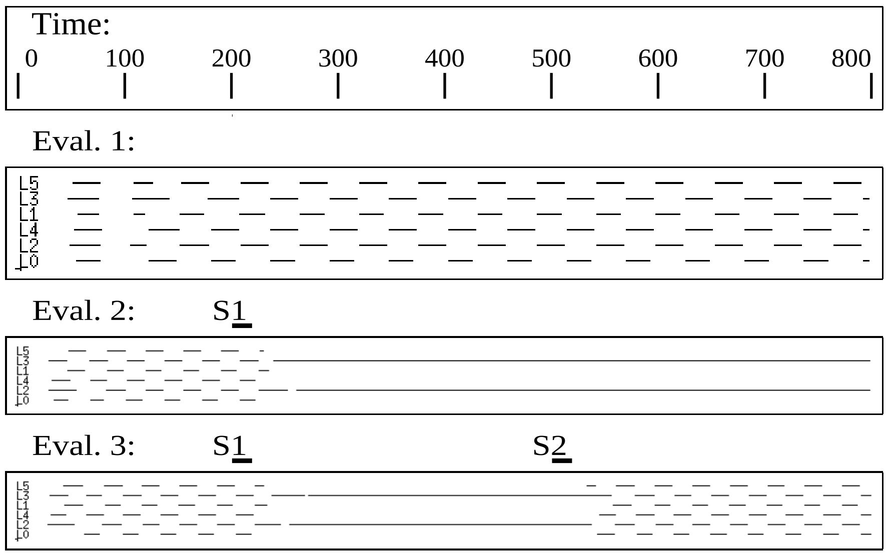


Figure 18: Behavior of a good individual in the three phases of the evaluation. That individual responds correctly to both stimuli. The conditional protocol (described in the text) prevents the animats from just predicting the time of occurence of the stimuli.

At the end of experiments during which 60.000 offspring events were performed in a population of 200 individuals, highly rated controllers have been obtained. The behavior of one such controller is illustrated on Figure 18. As explained in [16], it involves a switching mechanism that implements a rudimentary memory.

5.4 Goal-seeking

In these series of experiments again, a second neural module was used to control an already evolved locomotion module, but the objective was now to solve a goal-seeking task.

To this end, the second module received information from two sensors each returning the intensity of an odor signal perceived at the tip of an antenna. This intensity decreased with proportion to the square of the distance from an odorous source. The goal-seeking module stemmed from two precursor cells that read the same developmental subprogram and executed its instruction in a symmetric way (Figure 19). A new grammar, GRAM2, defined the set of developmental subprograms used for Module 2 and specified that each such subprogram could create at most 4 neurons

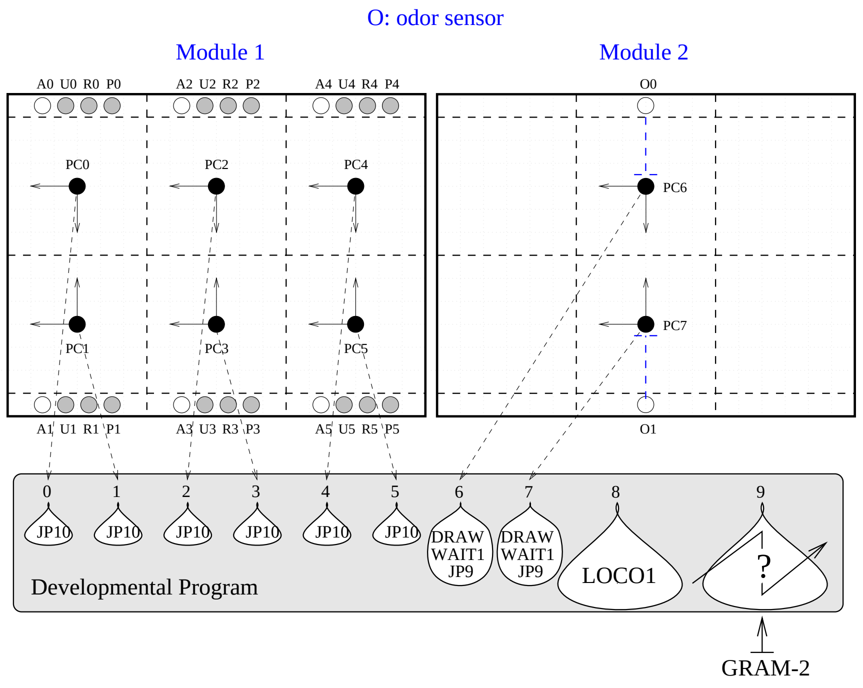


Figure 19: Setup for the evolution of the goal-seeking behavior. DRAW instructions in Subprogram 6 and Subprogram 7 create a default connection between a precursor cell of Module 2 and the associated sensory cell. These connections are copied to any daughter cell the precursor cells may have. WAIT instructions are necessary to synchronize the developments of the two modules because Module 1 goes through 3 division cycles while Module 2 goes through only 2 such cycles. Subprogram 9 is evolved according to the tree-grammar GRAM2 (Figure 20). It will be called GRAD2 hereafter.

and 16 connections. Because the corresponding subprogram was executed by both precursor cells, this resulted in a maximum of 8 neurons and 16 connections in Module 2 (Figure 20).

To evaluate the fitness of each program, a set of $N = 5$ environments env_i with different source positions was used. In each environment, the animat's task was to reach the source of odor, considered as a goal. The animat always started from the same position and was allowed to walk for a given time t_{max}, or until it reached the goal. This event was considered to have occured if the point X situated between the tips of the animat's two antennae came close enough to the source S. The corre-

Terminal symbols

DIVIDE, GROW, DRAW, GROW2, SETBIAS, SETTAU, DIE,

NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.

Variables

Start1, Level1, Neuron, Bias, Tau, Connex, Link.

Production rules

Start1$\longrightarrow$DIVIDE(Level1, Level1)

Level1$\longrightarrow$DIVIDE(Neuron, Neuron)

Neuron$\longrightarrow$SIMULT3(Bias, Tau, Connex) | DIE

Bias$\longrightarrow$SETBIAS | DEFBIAS

Tau$\longrightarrow$SETTAU | DEFTAU

Connex$\longrightarrow$SIMULT4(Link, Link, Link, Link)

Link$\longrightarrow$GROW | DRAW | GROW2 | NOLINK

Starting symbol

Start1.

Figure 20: The GRAM2 grammar that is used to evolve a goal-seeking module.

sponding fitness function was:

$$f(env_i) = t_{max}^{(i)} \cdot min\left\{d(X(t), S(t)), t \in [0, t_{max}^{(i)}]\right\}$$

$$fitness = \frac{\sum_i f(env_i)}{N}$$

This function rewarded animats that quickly approached the source during the evaluation.

Five different experiments have been done, each starting with a different initial population. In each experiment, individuals able to reach the source in each of the five positions of the learning set were obtained after 20.000 selection-replacement events. Such abilities proved to be general enough for allowing the animat to reach the goal in almost any other positions (Figure 21), which sometimes required lengthening the evaluation time t_{max} or changing parameter values in the animat's physical model.

A close inspection of the inner workings of the corresponding controller reveals that the second module's neural circuitry basically serves to compare the activity levels of the two odor sensors. Whenever the odor signal is substantially higher on

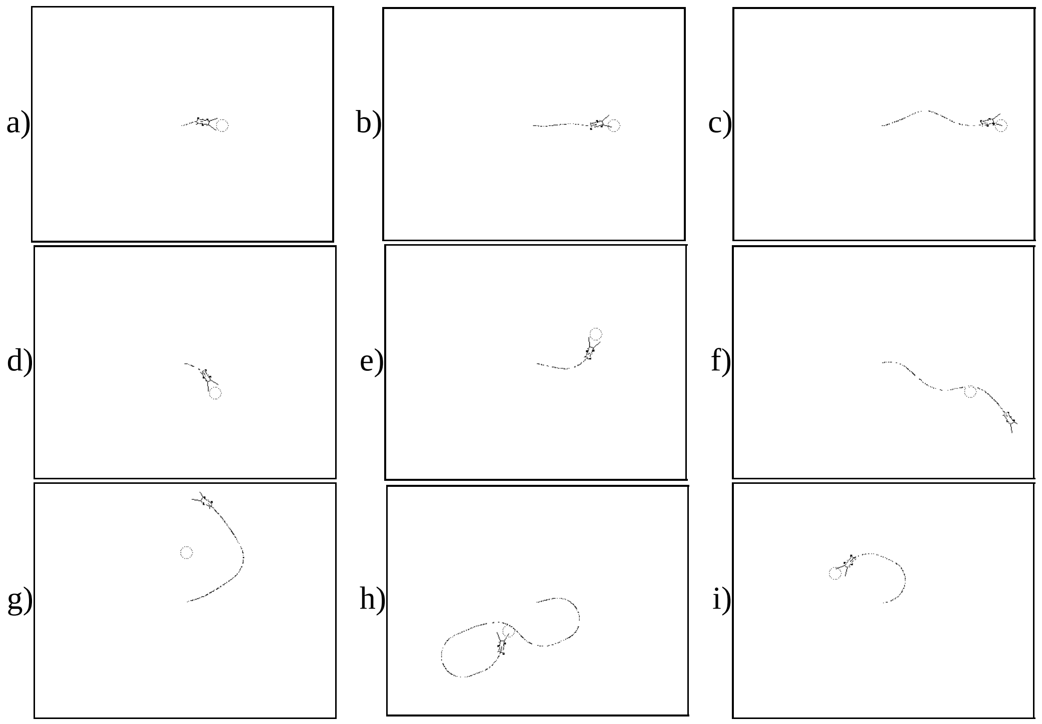


Figure 21: Generalization experiments for the goal-seeking task. An animat, which has been selected to reach a goal in 5 different test positions, is tested against 9 other goal positions. When the animat occasionally misses the goal, it may nevertheless reach it later (as in case h) if the evaluation time is lengthened. Likewise, problems examplified by cases f and g may be solved by simply changing the stiffness of the muscles that bring the legs back to the vertical plane, thus diminishing the animat's turning angle.

one side of the animat, a signal is sent to the hind leg on this side, which prevents that leg from rising. This, in turn, triggers a rotation towards the goal [16].

5.5 Obstacle-avoidance

In order to implement a minimal reactive navigation system, we wanted to combine in the same animat the abilities of seeking a goal and of avoiding obstacles. To this end, we used the two modules that already secured locomotion and goal-seeking, and sought to evolve a third module, Module 3, capable of affording the animat an additional obstacle-avoidance competency.

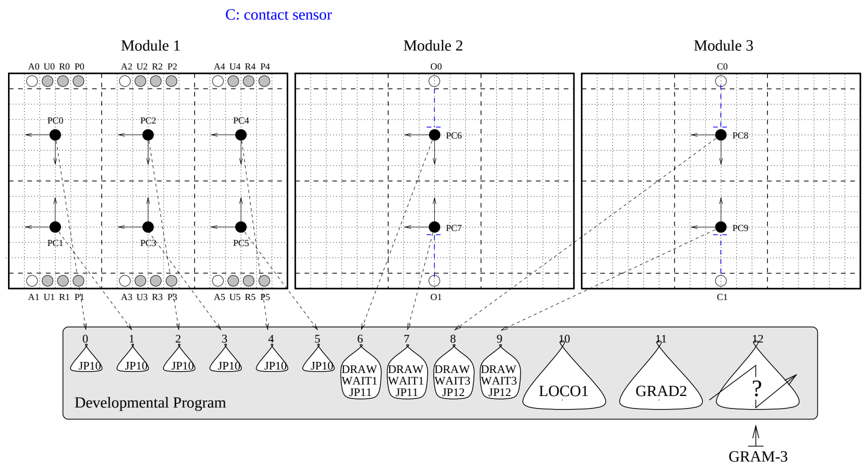


Figure 22: Setup for the evolution of an obstacle-avoidance controller. DRAW instructions in subprograms 6 to 9 create a default connection between a precursor cell of Modules 2 or 3 and the associated sensory cell. These connections are copied to any daughter cell the precursor cells may have. WAIT instructions are added to synchronize the developments of the different modules because Module 1 goes through 3 division cycles, while Module 2 goes through only 2 such cycles, and Module 3 does not lead to any division. Sub-program 12 is evolved according to the GRAM3 tree-grammar specified in Figure 23.

Like Module 2, this module stemmed from two precursor cells that read the same developmental subprogram and executed its instruction in a symmetric way (Figure 22). It was assumed that sensory information would be received from two sensors, each indicating if an antenna got into contact with an obstacle. Although it could influence the behavior of the locomotion module through inter-modular connections created during the evolutionary process, no inter-modular connections were allowed from Module 3 to Module 2 in order to avoid parasitic interferences. A new grammar, GRAM3, defined the set of developmental subprograms used for Module 3 and specified that each precursor cell could grow at most 4 connections to neurons of Module 1 (Figure 23).

To evaluate the fitness of each program, a set of $N = 5$ different environments env_i, each containing an odorous goal and several obstacles, has been used. In each environment, the behavior of the animat was simulated until a final time t_{max} was reached or until the animat reached the goal or hit an obstacle. The corresponding

Terminal symbols

GROW2, SETBIAS, SETTAU,

NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.

Variables

Start1, Bias, Tau, Connex, Link.

Production rules

Start1$\longrightarrow$SIMULT3(Bias, Tau, Connex)

Bias$\longrightarrow$SETBIAS | DEFBIAS

Tau$\longrightarrow$SETTAU | DEFTAU

Connex$\longrightarrow$SIMULT4(Link, Link, Link, Link)

Link$\longrightarrow$GROW2 | NOLINK

Starting symbol

Start1.

Figure 23: The GRAM3 grammar that is used to evolve an obstacle-avoidance module.

fitness function was:

$$\begin{aligned} f(env_i) &= \frac{1}{t_{max}^{(i)}} \cdot \Big(d(X(0), S(0)) - d(X(t_{max}^{(i)}, S(t_{max}^{(i)}) \Big) \\ &+ \int_0^{t_{max}^{(i)}} s(t) \cdot dt \end{aligned}$$

$$fitness = \frac{\sum_i f(env_i)}{N}$$

where $s(t)$ was set to 1 if the animat was stable at time t and to 0 otherwise. The first term in the function rewarded an individual according to the rate of decrease of its distance to the goal during the evaluation. The second term explicitely favored individuals that did not fall, a tendency often exhibited by animats of the first generations.

Again, five different experiments were done, each starting with a different initial population. In each experiment, individuals able to reach the source and to avoid obstacles in each of the five environments of the learning set were obtained after 20.000 offspring events. Besides being surrounded or not by a rectangular wall, these test environments only contained circular obstacles. Generalization experiments, where

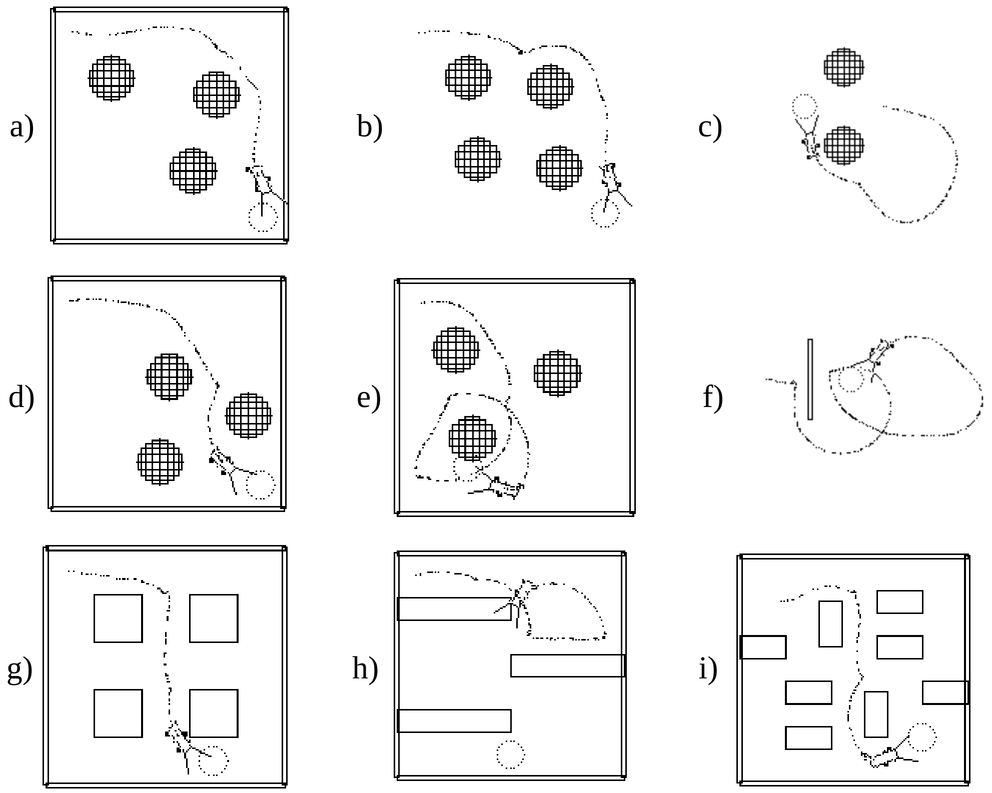


Figure 24: Experimental results obtained when goal-seeking and obstacle-avoidance behaviors are evolved. Cases (a-c) show the animat's trajectory within 3 out of the 5 test environments. Cases (d-i) show results of generalization experiments, in 6 new environments. The animat can deal with obstacle shapes never met during evolution (f-i). However, it cannot always avoid hitting sharp corners (h).

individuals were tested in new environments, were often successfull, although some difficulties avoiding collisions with obstacles exhibiting sharp corners have been noticed (Figure 24).

It appears that the neural circuitry of the third module basically serves to detect the contact of an antenna with an obstacle. Whenever this occurs, a signal is sent to the front leg on the obstacle's opposite side, which prevents that leg from rising. This, in turn, triggers a rotation away from the obstacle [16].

5.6 Real robot application

The SGOCE evolutionary paradigm is currently being used to evolve neural controllers for SECT, a real 6-legged robot manufactured by Applied AI Systems, Inc. (Figure 25).

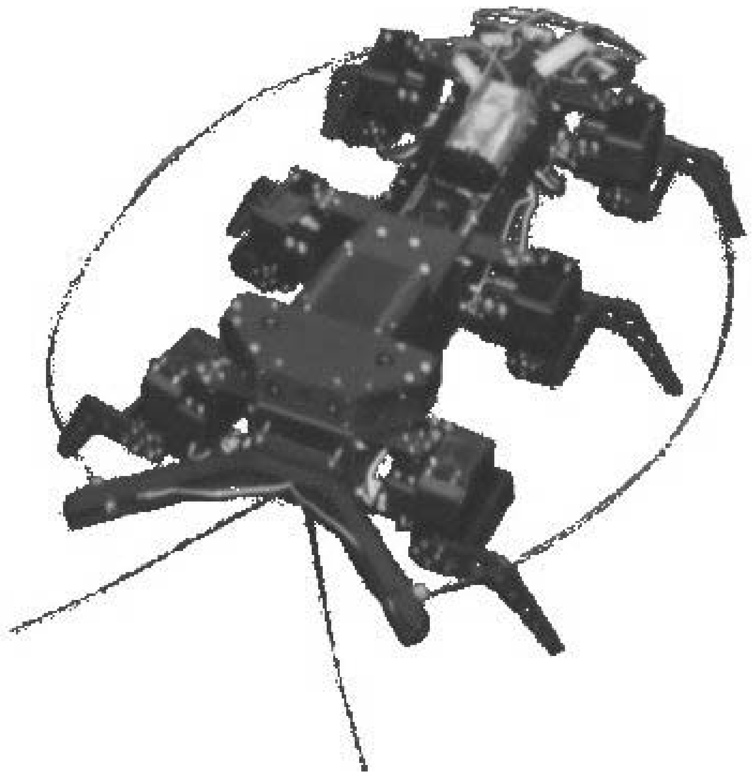


Figure 25: The SECT robot.

Locomotion and obstacle-avoidance controllers are first selected through simulations and then downloaded on the robot. The inputs to the evolved controllers are provided by the robot's infra-red and tactile sensors, while the motoneurons of the controllers set the end positions of vertical and horizontal swings of each leg, as well as its vertical and horizontal speeds. Preliminary encouraging results have been obtained.

6 Conclusions

The results that have been described here and elsewhere [7, 18, 19, 16] inbed the *animat approach* to cognitive science [27] and artificial life [24] into an evolutionary perspective. They prove that the SGOCE paradigm provides a convenient means of generating neural networks that control animat behavior through simple stimulus-response pathways. They also suggest that this paradigm might help automatically discover more cognitive mechanisms that would endow animats with increased adaptive capacities. The animats that we have evolved significantly further previous attempts at automatically designing walking creatures: they are not only capable of straight-line locomotion according to a tripod gait, but also of slowing-down, of stopping or resuming walking, of turning right or left, of aiming towards specific goals, and of avoiding obstacles. Future research will aim at generating higher behavioral competencies in the corresponding controllers, for example through the inclusion of mechanisms that might implement cognitive capacities for memory-based computation, motivation management, and internal simulation [23].

From a general point of view, the efficiency of the SGOCE paradigm is probably due to the compact encoding it affords, which tremendously reduces the size of the genotype space to be explored by the evolutionary algorithm, while offering opportunities for the generation of complex phenotypes. However, it is true that, as far as specific implementation details are concerned, it is presently not possible to assess which are really useful and which are not. The evolved organization of the controllers described above was certainly highly dependent upon several arbitrary choices made by the experimenter concerning, for instance, the developmental instructions, the setup of the initial substrates, the constraining grammars, the fitness functions, the parameters of the evolutionary algorithm, etc. Although such implementation settings were chosen after preliminary trials and errors, it is definitely unclear - in the absence of systematic comparisons that we haven't yet had the opportunity to perform - whether better choices couldn't have been made. It is true, for example, that recourse to grammars constraining the structure of the developmental programs is not mandatory, although it helps to reduce the complexity of the evolved controllers and, thus, to reduce simulation time. In [25] a locomotion controller is presented that evolved in the absence of syntactic constraints, and that generates a tripod gait: it exhibits 192 neurons and 2222 connections. Be that as it may, it is clear that a potentially fruitful direction of future research consists in letting evolve several of the characteristics that have been arbitrarily set by the experimenter up to now, and by devising new developmental instructions that would afford the animats individual learning capacities complementing those of evolution and development.

References

[1] R. D. Beer, *Intelligence as Adaptive Behavior. An Experiment in Computational Neuroethology*. Academic Press. 1990.

[2] R. D. Beer, "On the dynamics of small continuous-time recurrent neural networks," *Adaptive Behavior*, vol. 3, no. 4, pp. 469–510, 1995.

[3] R. D. Beer and J. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 1, no. 1, pp. 91–122, 1992.

[4] E. Boers and H. Kuiper, "Biological metaphors and the design of modular artificial neural networks," Master's thesis, Dept. of CS and Exp. and The. Psy., Leiden University, 1992.

[5] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot." *IEEE Journal of Robotics and Automation*, 2, pp. 14–23, 1986.

[6] L. Bull, T. C. Fogarty and M. Snaith, "Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot," in *Proceedings of the Sixth International Conference on Genetic Algorithms* (L. J. Eshelman, ed.), pp. 382–388, Morgan Kaufmann, San Mateo, CA, 1995.

[7] J. Chavas, C. Corne, P. Horvai, J. Kodjabachian and J.A. Meyer, "Incremental Evolution of Neural Controllers for Robust Obstacle-Avoidance in Khepera," *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98* (P. Husbands and J.A. Meyer, eds.). 1998.

[8] F. Dellaert and R. D. Beer, "Toward an evolvable model of development for autonomous agent synthesis," in *Proceedings of the Fourth International Workshop on Artificial Life* (R. A. Brooks and P. Maes, eds.), The MIT Press/Bradford Books. 1994.

[9] H. de Garis, *Genetic Programming: GenNets, Artificial Nervous Systems, Artificial Embryos*. PhD thesis, Université Libre de Bruxelles. Belgium. 1991.

[10] S. Galt, B. L. Luk and A. A. Collie.“Evolution of Smooth and Efficient Walking Motions for an 8-Legged Robot” in *Proceedings of the Sixth European Workshop on Learning Robots*, (Brighton, England), 1997.

[11] F. Gruau, “Automatic definition of modular neural networks,” *Adaptive Behavior*, vol. 3, no. 2, pp. 151–184, 1994.

[12] F. Gruau and K. Quatramaran, “Cellular Encoding for Interactive Evolutionary Robotics,” *Proceedings of the Fourth European Conference on Artificial Life* (P. Husbands and I. Harvey, eds.). The MIT Press / Bradford Books, Cambridge, MA, 1997.

[13] T. Gomi and K. Ide, “Emergence of Gaits for a Legged Robot by Collaboration through Evolution” *Proceedings of the International Symposium on Artificial Life and Robotics*, (Oita, Japan), 1997.

[14] N. Jakobi, “Running Across the Reality Gap: Octopod Locomotion Evolved in a Minimal Simulation,” *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot’98* (P. Husbands and J.A. Meyer, eds.). 1998.

[15] J. Kodjabachian, “Simulating the dynamics of a six-legged animat,” Tech. Rep., AnimatLab, ENS, Paris, 1996.

[16] J. Kodjabachian, “Développement et évolution de réseaux de neurones artificiels. Application au contrôle d’un animat hexapode.” PhD thesis. Université Paris 6. France. 1998.

[17] J. Kodjabachian and J.-A. Meyer, “Evolution and development of control architectures in animats,” *Robotics and Autonomous Systems*, vol. 16, pp. 161–182, 1995.

[18] J. Kodjabachian and J.-A. Meyer, “Evolution and Development of Modular Control Architectures for 1-D locomotion in Six-Legged Animats.” *Connection Science*. In press.

[19] J. Kodjabachian and J.-A. Meyer, “Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects.” *IEEE Trans. Neural Networks*. In press.

[20] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[21] J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Subprograms*. The MIT Press, 1994.

[22] M. A. Lewis, A. H. Fagg, and A. Solidum, “Genetic programming approach to the construction of a neural network for control of a walking robot,” in *IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[23] B. W. Mel, “Animal Behavior in Four Components,” *Comparative Approaches to Cognitive Science* (H.L. Roitblat and J.A. Meyer, eds.). The MIT Press / Bradford Books, Cambridge, MA, 1995.

[24] J.-A. Meyer, “Artificial life and the animat approach to artificial intelligence,” *Artificial Intelligence*, (M. Boden, ed.). Academic Press. 1996.

[25] J.-A. Meyer, “From natural to artificial life: Biomimetic mechanisms in animat design,” *Robotics and Autonomous Systems*, vol. 22, pp. 3–21, 1995.

[26] S. Nolfi, O. Miglino and D. Parisi. “Phenotypic Plasticity in Evolving Neural Networks,” *From Perception to Action* (P. Gaussier and J.D. Nicoud, eds.). IEEE Computer Society Press. 1994.

[27] H.L. Roitblat and J.-A. Meyer (eds.) *Comparative Approaches to Cognitive Science*. The MIT Press / Bradford Books, Cambridge, MA, 1995.

[28] I. Segev, “Simple neuron models: Oversimple, complex and reduced.” *Trends in Neurosciences*, vol. 15, pp. 414–421, 1992.

[29] K. Sims, “Evolving 3D morphology and behavior by competition,” in *Proceedings of the Fourth International Workshop on Artificial Life* (R. A. Brooks and P. Maes, eds.), The MIT Press/Bradford Books. 1994.

[30] G. Spencer, “Automatic generation of programs for crawling and walking,” in *Advances in Genetic Programming* (K. E. Kinnear, ed.), The MIT Press / Bradford Books, Cambridge, MA, 1994.

[31] J. Vaario, *An Emergent Modeling Method for Artificial Neurol Networks*. PhD thesis, University of Tokyo, 1993.