Evolution and Development of Neural Networks Controlling Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects

Jérôme Kodjabachian and Jean-Arcady Meyer
AnimatLab. Ecole Normale Supérieure. France.

Abstract— **This paper describes how the SGOCE paradigm has been used to evolve developmental programs capable of generating neural networks that control the behavior of simulated insects. This paradigm is characterized by an encoding scheme, by an evolutionary algorithm, and by an incremental strategy that are described in turn. The additional use of an insect model equipped with 6 legs and two antennae made it possible to generate control modules that allowed to successively add gradient-following and obstacle-avoidance capacities to walking behavior. The advantages of this evolutionary approach, together with directions for future work, are discussed.**

Keywords— **SGOCE Paradigm, Recurrent Neural Networks, Leaky Integrators, Genetic Programming, Animats.**

I. Introduction

Since the pioneering attempts of a few researchers [1], [2], [3], [4], [5] in the late 80's, the automatic design of artificial neural networks using some variety of evolutionary algorithm is a common occurrence (reviews in [6], [7], [8], [9]), in particular in the application domains of evolutionary robotics (for a review, see [10]) and of animat design (for a review, see [11]). However, such an approach is not without raising specific problems [12], notably that of choosing how to genetically encode the neural networks produced by the evolutionary algorithm.

Indeed, it turns out that numerous encoding schemes that are currently used in such application domains, because they implement a direct so-called genotype-to-phenotype mapping, are hampered by a lack of scalability, according to which the size of the genetic description of a neural network grows as the square of the network's size. As a consequence, the evolutionary algorithm explores a genotypic space that grows bigger and bigger as the phenotypic solutions sought get more and more complex. Moreover, it also turns out that such encoding schemes are usually not able to generate modular architectures, i.e. that they do not allow for repeated substructures that would help to encode complex control architectures in compact genotypes.

In [11], we have argued that it might be wise to tackle these problems in the same way that nature does, i.e., by using an indirect genotype-to-phenotype mapping that would insert a developmental process between the genotype and the phenotype of an animat interacting with its environment. In [13], we have implemented such a developmental process within the framework of an incremental evolutionary approach that made it possible to evolve 1-D locomotion controllers in 6-legged animats.

This paper reports on the extension of this approach to the automatic generation of neural networks controlling 2-D locomotion and higher-level behaviors in simulated insects. It aims to contribute to the *animat approach* to cognitive science [14] and artificial life [15]. As such, it is heavily inspired by the work of Beer [16] - who designed the nervous system of an artificial coackroach capable of walking, of avoiding obstacles and of getting to an odorous food source — although the controllers that are used in the present work are evolved instead of being hand-coded. It is also heavily inspired by the work of Beer and Gallagher [17] — who let evolve the nervous system of a walking insect — although the neural networks that are evolved here are capable of controlling more than mere locomotion. The paper starts with a description of the SGOCE evolutionary paradigm and of the SWAN model of a hexapod animat that we are using. Experimental results on the evolution of artificial insects exhibiting a tripod walking rhythm and capable of both following an odor gradient and avoiding obstacles are then described. The paper ends with a discussion of the results and proposes directions for future work.

II. 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 capacities.

A. The encoding scheme

SGOCE is a simple geometry-oriented variation of Gruau's cellular encoding scheme [18], [19], [20] that is used to evolve simple developmental programs capable of generating neural networks of arbitrary complexity. According to the SGOCE scheme, each cell in a developing network occupies a given position in a 2D metric substrate and can get connected to other cells through efferent or afferent connections. In particular, such cells can get connected to sensory or motor neurons that have been positioned by the experimenter at initialization time in specific locations within the substrate. Moreover, during development, such cells may divide and produce new cells and new connections that expand the network. Ultimately, they may become fully functional neurons that participate to the behavioral control of a given animat, although they also may occasionally die and reduce the network's size.

SGOCE developmental programs call upon subprograms that have a tree-like structure like those of genetic programming [21], [22]. Therefore, a population of such subprograms can evolve from generation to generation, a process during which individual instructions can be mutated within a given subprogram and sub-trees belonging to two different subprograms can be exchanged.

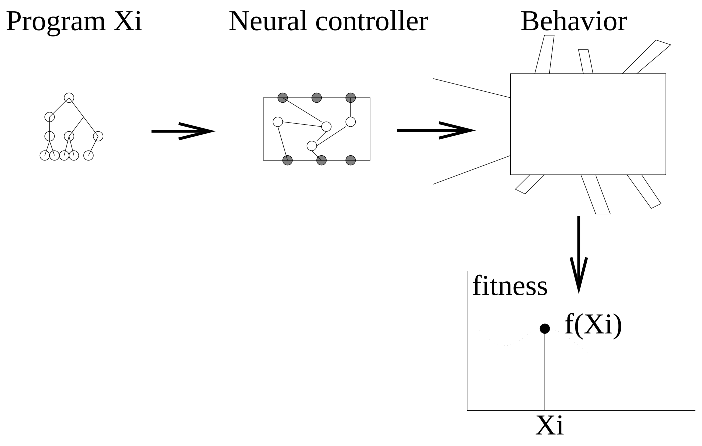


Fig. 1. The three stages of the fitness evaluation procedure of a developmental program (X_i). First, the program is executed to yield an artificial neural network. Then the neural network is used to control the behavior of a simulated animat that has to solve a given task in an environment. Finally, the fitness of Program X_i is assessed, according to how well the task has been solved.

At each generation, the fitness of each developmental program is assessed (Figure 1). To this end, the experimenter must provide and position within the substrate a set of precursor cells, each characterized by a local frame that will be inherited by each neuron of its lineage and according to which the geometrical specifications of the developmental instructions will be interpreted. Likewise, the experimenter must provide and position a set of sensory cells and motoneurons that will be used by the animat to interact with its environment. Lastly, an overall description of the structure of the developmental program that will generate the animat's control architecture, together with a specification of the grammar that will constrain its evolvable subprograms as described further, must be supplied (Figure 2).

Each cell within the animat's control architecture is assumed to hold a copy of this developmental program. Therefore, the program's evaluation starts with the sequential execution of its instructions by each precursor cell and by each new cell occasionally created during the course of development (Figure 3). At the end of this stage, a complete neural network is obtained, whose architecture will reflect 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. Through its sensory cells and its motoneurons, this neural network is then connected to the sensors and actuators of the insect model to be described later. This, together with the use of an appropriate fitness function, makes it possible to assess the network's capacity to generate a specific behavior. Thus, from generation to generation, the reproduction of good controllers – and hence of good developmental programs – can be favored to the detriment of the reproduction of bad controllers and bad programs, according to standard genetic algorithm practice [23].

In the present application, a small set of developmental instructions can be included in evolvable subprograms (Table I). A cell division instruction (DIVIDE) makes it possible for a given cell to generate a copy of itself. A direction parameter (α) and a distance parameter (r) associated with that instruction specify

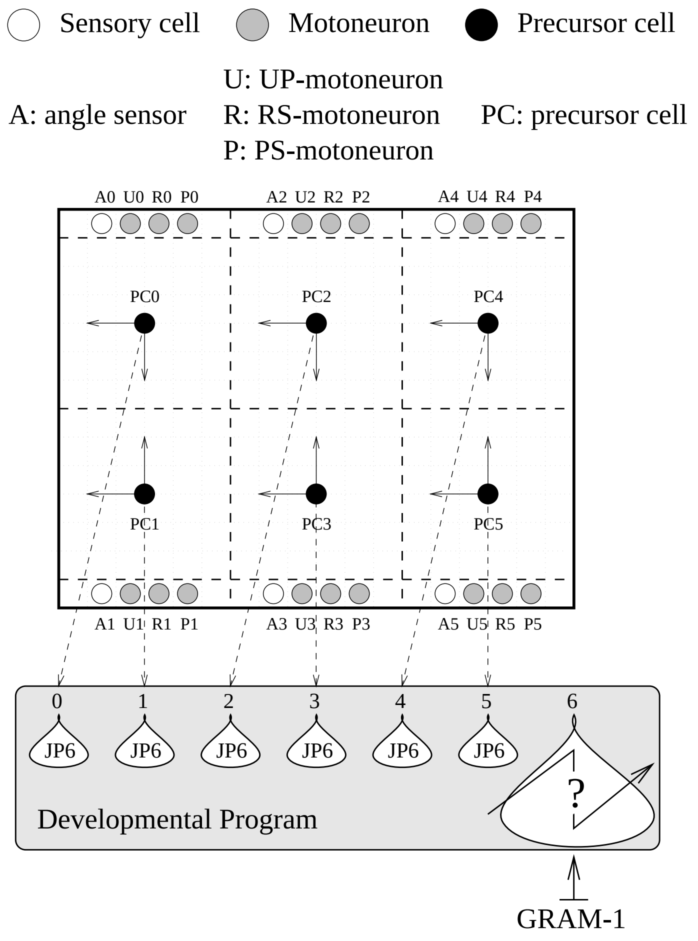


Fig. 2. Setup for the evolution of a neural network that will be used in section III-A to control locomotion in a six-legged animat. The figure shows the initial positions of the sensory cells, motoneurons and precursor cells within the substrate, as well as the structure of the developmental programs that call upon 7 subprograms. JP is a call instruction that forces a cell to start reading a new subprogram. Only subprogram 6 needs to be evolved. Its structure is constrained by the GRAM-1 tree-grammar (to be described in section II-B). Additional details are to be found in the text.

the position of the daughter cell to be created in the coordinates of the local frame attached to the mother cell. Then, the local frame associated to the daughter cell is centered on this cell's position and is oriented as the mother cell's frame (Figure 4). Two instructions (GROW and DRAW) respectively create one new efferent and one new afferent connection. The cell to be connected to the current one is the closest to a target position that is specified by the instruction parameters, provided that the target position lays on the substrate (Figure 4). No connection is created if the target is outside of the substrate's limits. Another instruction called GROW2 will be used in sections III-B and III-C below. It is similar to instruction GROW but creates a connection from a cell in a given neural module to another cell in another module. The synaptic weight of a new connection is given by the parameter w. Two additional instructions (SETTAU and SETBIAS) specify the values of a neuron's time constant τ and bias b. Finally the instruction DIE causes a cell to die.

Neurons of intermediate complexity between abstract binary neurons and detailed compartmental models are used in the

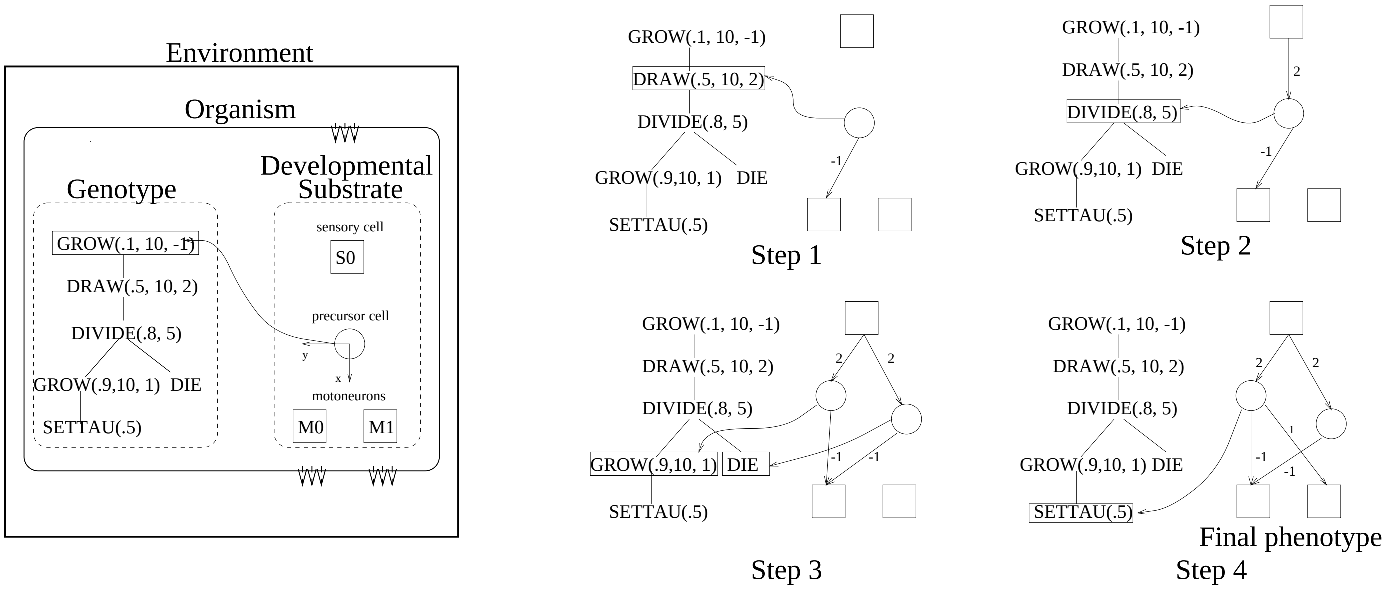


Fig. 3. The developmental encoding scheme of SGOCE. The genotype that specifies the animat's nervous system is encoded as a grammar tree whose nodes are specific developmental instructions. Within such chromosomes, mutations change one branch into another, and crossovers swap branches. Each cell in the developping network reads the chromosome at a different position. More or less developmental steps are required to generate a phenotype, depending upon the length of the corresponding genotype.

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 I

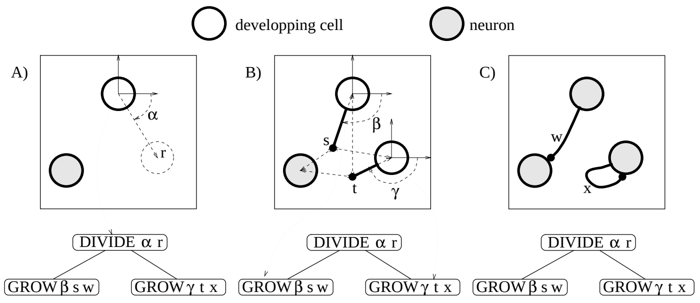


Fig. 4. The effect of a sample developmental code. A) When the upper cell executes the DIVIDE instruction, it divides. The position of the daughter cell in the mother cell's local frame is given by the parameters α and r of the DIVIDE instruction, which respectively set the angle and the distance at which the daughter cell is positioned. B) Next, the mother cell reads the left sub-node of the DIVIDE instruction while the daughter cell reads the right sub-node. C) As a consequence, a connection is grown from each of both cells. The two first parameters of a GROW instruction determine a target point in the local frame of the corresponding cell. The connection is realized with the cell closest to the target point — a developing cell, an interneuron, a motoneuron or a sensory cell — and its synaptic weight is given by the third parameter of the GROW instruction. Note that, in this specific example, the daughter cell being closest to its own target point, a recurrent connection is created on that cell. Finally, the two cells stop developing and become interneurons.

present application. Contrary to neurons used in traditional PDP applications [24], [25], such neurons exhibit an internal dynamics. However, instead of simulating each activity spike of a real neuron, the *leaky-integrator model* used here only monitors each neuron's average firing frequency. According to this model, the mean membrane potential m_i of a neuron N_i is governed by the equation:

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

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 firing threshold, and τ is a time constant associated with the passive properties of the neuron'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. This model has already been used in several applications involving continuous-time recurrent neural network controllers [17], [26], [19], [27]. It has the advantage of being a universal dynamics approximator [28], i.e., of being likely to approximate the trajectory of any smooth dynamic system.

B. *Syntactic restrictions*

In order to reduce the size of the genetic search-space and the complexity of the generated networks, a context-free tree-grammar is used to impose each evolvable subprogram to have the structure of a well-formed tree. For instance, Figure 5 shows the GRAM-1 grammar used in Section III-A to constrain the structure of the subprograms that participate in the developmental process of locomotion controllers.

The set of terminal symbols consists of the developmental instructions listed in Table I and of additional *structural instructions* that have no side-effect on the developmental process. NO-LINK is a "no-operation" instruction. DEFBIAS and DEFTAU

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.

Fig. 5. The GRAM-1 grammar.

leave the default value of the parameter b and τ unchanged. These instructions label nodes of arity 0. 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 to hopefully exchange meaningful building blocks. Those instructions are associated to nodes of arity 3 and 4, respectively.

As a consequence of the use of syntactic constraints that predefine the overall structure of a developmental program, the timing of the corresponding developmental process is constrained. First divisions occur, then cells die or parameters are set, and finally connections are grown. No more than three successive divisions can occur and the number of connections created by any cell is limited to four. Thus, the final numbers of interneurons and connections created by a subprogram well-formed according to GRAM-1 cannot be greater than 8 and 32 respectively.

C. *Evolutionary algorithm*

To slow down convergence and to favor the apparition of ecological niches, the SGOCE evolutionary paradigm resorts to a steady state genetic algorithm that involves a population of N randomly generated well-formed programs distributed over a circle and whose functioning is sketched in Figure 6.

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 2-tournament selection scheme is applied, in which the best of two programs randomly selected from the neighborhood of P is kept[1].
3. The selected program is allowed to reproduce and three genetic operators possibly modify it. The recombination operator is applied with probability p_c. It exchanges two *compatible*[2] 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 a randomly selected sub-tree into another compatible, randomly generated one. The second mutation operator is applied with probability 1. It modifies the values of a random number of parameters, implementing a *constant perturbation* strategy [29]. 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.
5. A 2-tournament anti-selection scheme, in which the worse 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.

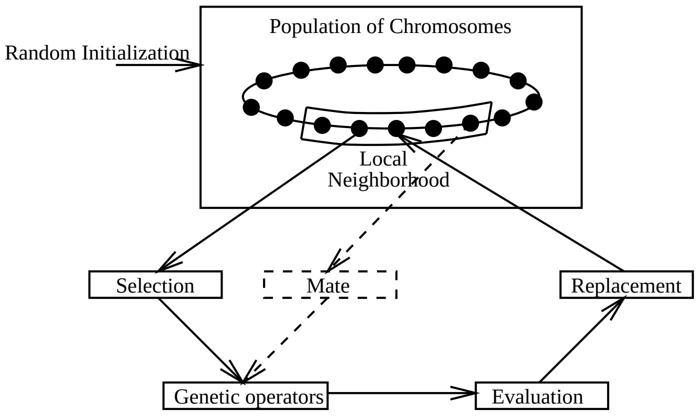


Fig. 6. The evolutionary algorithm. See text for explanation.

In all the experiments reported in this paper, $p_c = 0.6$, $p_m = 0.2$, $n = 6$ and $p = 0.5$.

D. *Incremental methodology*

The SGOCE paradigm resorts to an incremental approach that takes advantage of the geometrical nature of the developmental model. In a first evolutionary stage, locomotion controllers for a simulated insect are evolved. At the end of that stage, the developmental program corresponding to the best evolved controller is selected to be the locomotion module used thereafter (section III-A). During further evolutionary stages, other neural modules are evolved that control higher-level behaviors. These modules may influence the locomotion module by creating intermodular connections. For instance, Figure 7 shows how the successive connection of two additional modules with a locomotion controller has been used in the experiments reported below to first generate a gradient-following behavior (section III-B) and then to generate additional obstacle-avoidance capacities (section III-C).

E. *The SWAN model*

The experimental results to be described herein made use of the SWAN model of a simulated walking animat [30] that is

[1]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 (ps=0)

[2]Two sub-trees are compatible if they are derived from the same grammatical variable, like Start1, Level1, etc., in Figure 5.

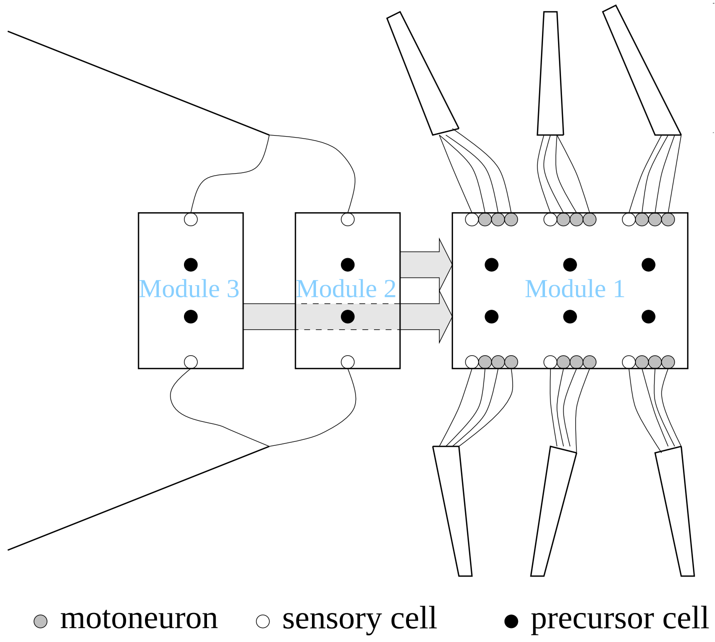


Fig. 7. The SGOCE incremental approach. During a first evolutionary stage, Module 1 is evolved. This module receives proprioceptive information through sensory cells and influences actuators through motoneurons. In a second evolutionary stage, Module 2 is evolved. This module receives specific exteroceptive 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 present work, no connections between Module 2 and Module 3 are allowed.

inspired by the work of Beer and Gallagher [17]. Each of the 6 legs of the animat is equipped with two pairs of muscles that allow control of its angular position and of the height of its foot (Figure 8, Left).

For three of those muscles, a corresponding motoneuron specifies the value of the resting length parameter in a simple muscle model (Figure 8, Middle). Furthermore, each leg is equipped with a sensor that measures the leg's angular position θ. Thus the available motors and sensors correspond to those of Beer and Gallagher's simulated insect [17]. However, a difference with Beer and Gallagher's scheme is that the foot status (up or down) is not determined by the state of the corresponding UP-motoneurons only. More realistically, instantaneous foot positions are determined by the dynamics of the physical model of the animat.

Additionally, depending upon the activity level of the PS- and RS-motoneurons, forces acting on the animat's body can be greater on one side than on the other. This entails leg displacements from a vertical plane and the triggering of return forces proportional to the angular displacement φ that are responsible for the animat's rotations (Figure 8, Right).

Lastly, the SWAN model allows for the monitoring of 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.

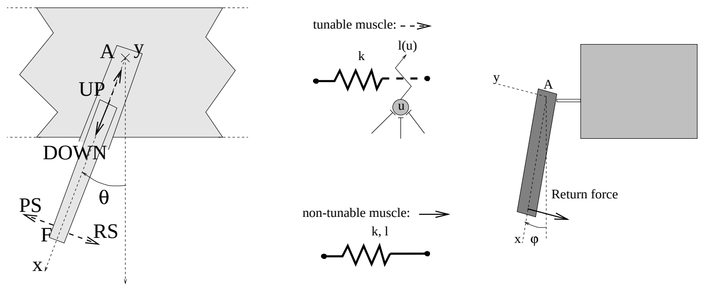


Fig. 8. The SWAN model. Left: According to the former version of this model [13], each leg has two degrees of freedom. Thanks to the antagonistic PS- (Power Strike) and RS- (Return Strike) muscles a leg can rotate around an axis (Ay) orthogonal to the plane of the figure. The UP- and DOWN-muscles allow 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 corresponding motoneurons. 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 l (after [31]). Right: In the extended model of the hexapod animat that has been used herein, each leg is afforded a third degree of freedom and can accordingly deviate from the vertical plane parallel to the body axis. In this case, a return force proportional to the deviation tends to bring the leg back to the vertical plane.

III. Experimental results

The SGOCE methodology and the SWAN model have been used to evolve the developmental programs of neural networks that are able to control 2D-locomotion, gradient following and obstacle avoidance in a 6-legged animat. This has been possible thanks to a 3-stage incremental approach, according to which an efficient locomotion controller was first generated and then connected to two additional controllers. The first one permitted the simulated insect to reach a given odor source, while the second provided the added capacity of avoiding obstacles while walking towards the odorous goal.

A. 2D-locomotion

Results concerning the automatic production of 1D-locomotion controllers have been reported at length elsewhere [13]. In particular, it has been shown that some of the neural networks that have been obtained were capable of generating a *tripod gait* because they called upon 4 *central pattern generators* that were responsible for the rhythmic movements of the middle and back legs. Moreover, suitable connections were responsible for the synchronization of each tripod, according to which the front and back legs on each side of the animat were moved in synchrony with the middle leg of the opposite side. Likewise, other connections were making for phase opposition in the rhythms of the two opposite tripods.

Such results have been obtained with a 1D version of the SWAN model in which only 2 degrees of freedom were afforded to each animat's leg. 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 $x(t)$ was the position of the animat's center of mass at time t, t_{max} was the evaluation time, and $\theta_p(t)$ and $h_p(t)$ were

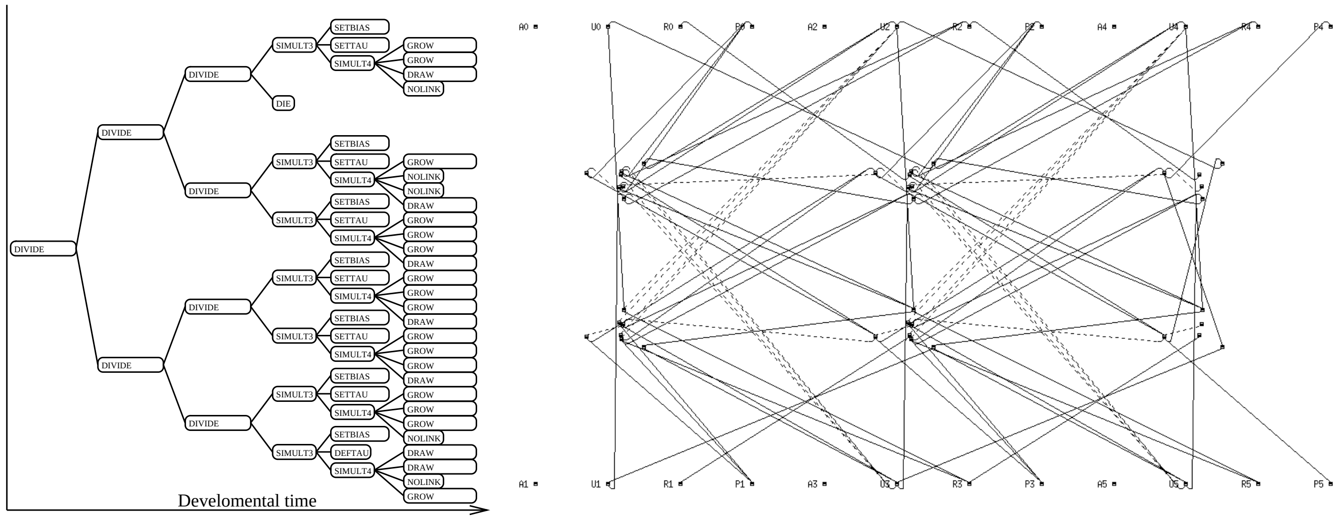


Fig. 9. Left: A developmental subprogram obtained for the locomotion task after 100.000 selection-replacement events. Instructions parameters are not shown. Right: The corresponding developped locomotion network. Filled lines are excitatory connections, dotted lines are inhibitory connections. Fan-in connection arrive at the top and fan-out connections depart from the bottom of each neuron [13].

the angular position and the height of leg p at time t [30]. Although no explicit selection pressure was introduced for not falling, falls were implicitly penalized because the stepping upright process slowed down locomotion.

The extension of this approach to 2D-locomotion has been straightforward because it only entailed adding a third degree of freedom to the SWAN model, as described in section II-E above. In other words, simply modifying the SWAN model made it possible for a previously generated 1D-locomotion controller to generate tripod walking in a 2D-environment. Figure 9 shows the best developmental subprogram (subsequently called LOCO1) that has been obtained after 100,000 selection-replacement events had been made by the genetic algorithm in a population of 200 programs. It also shows the corresponding neural network, which included 38 interneurons and 100 connections — after useless interneurons and connections had been pruned — and which will serve as Module 1 in the subsequent experiments described herein. Several 2D-trajectories generated by Module 1, together with an illustration of the tripod gait obtained, are shown on Figure 10: the turning direction of the animat depends upon initial conditions but, after a transitory period, straight locomotion resumes.

It thus appears that using Module 1 and the extended SWAN model together affords the simulated insect the possibility of walking straight ahead and of changing direction, provided that some dissymetry is imposed to the activity levels of motoneurons controlling leg and foot movements on each side of the animat. In the following sections, such a dissymetry will be generated through new connections brought by additional neural controllers.

B. Gradient-following

In this section, a new neural module is used to control the already evolved locomotion module in order to solve a goal-seeking task.

To this end, we evolved a gradient following module that received information from two sensors, each measuring 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.

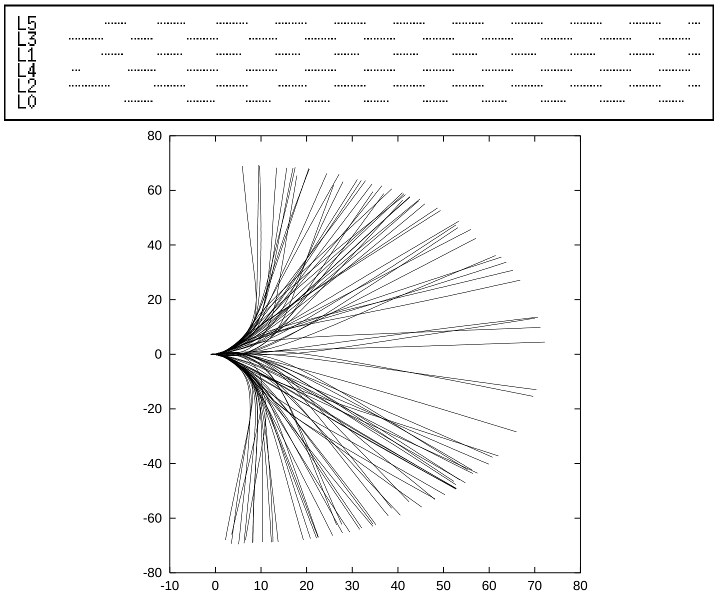


Fig. 10. Top: Tripod gait produced by an animat controlled by the controller shown in figure 9. The horizontal axis represents time. A dot is plotted when the corresponding leg is raised. Legs are numbered as in Figure 2. Bottom: 100 trajectories generated by the controller when the extended SWAN model was used. All the trajectories start from the same point $(0, 0)$. Differences in the random number sequences used to implement neuronal noise lead to differences in the initial turning directions.

The gradient following module stemmed from two precursor cells that read the same developmental subprogram and executed its instructions in a symetric way (Figure 11). It had the possibility of influencing the behavior of the locomotion module through inter-incremental connections created during development. A new developmental instruction (GROW2) was used to create a connection from a cell in the second module to a cell in the locomotion module. This instruction worked like the

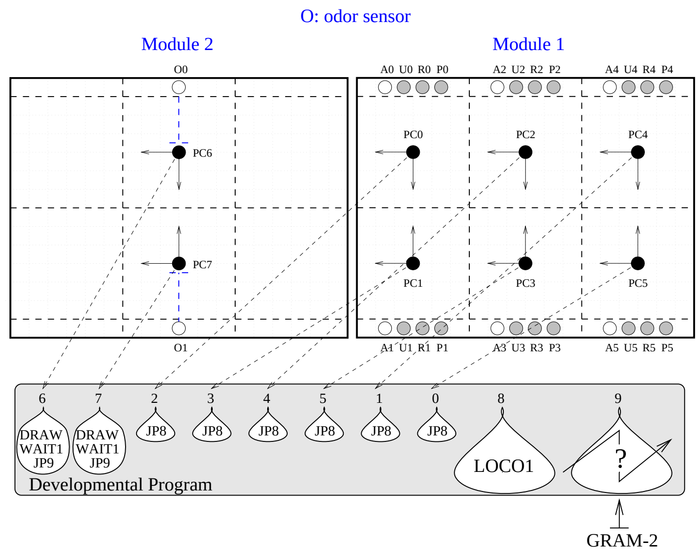


Fig. 11. Setup for the evolution of a goal-seeking controller. DRAW instructions in subprograms 6 and 7 create a default connection between a precursor cell of Module 2 and the associated sensory cell. These connections are copied to any daughter cell the precursor cells may have. WAIT instructions are 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. Sub-Program 9 is evolved according to the GRAM-2 tree-grammar (specified in Figure 12 below).

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.

Fig. 12. The GRAM-2 grammar.

instruction GROW except that the geometric parameters were interpreted in the local frame's orthogonal projection into the locomotion module[3].

The GRAM-2 Grammar (Figure 12) defines the set of developmental subprograms that were used for Module 2. The corresponding subprograms could create at most 4 neurons and 16 connections. Because such subprograms were executed by both precursor cells, this resulted in a maximum of 8 neurons and 32 connections in Module 2.

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 corresponding fitness function was:

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

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

where $t^{(i)}_{end}$ was the time at which the evaluation in Environment env_i stopped.

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, which sometimes required lengthening the evaluation time t_{max} (Figure 13).

Figure 14 describes how the gradient following capacities are implemented in the control architecture of the animat whose behavior is shown in Figure 13. This animat's Module 2 contains 6 interneurons and 22 connections. Subsequently, the developmental program of this module will be called GRAD1.

C. *Obstacle-avoidance*

In this section, we seek to implement a minimal reactive navigation system that allows an animat to both follow an odor-gradient and to avoid obstacles.

To this end, capitalizing on the developmental subprograms LOCO1 and GRAD1 previously generated, we let evolve a Module 3 that added obstacle-avoidance capacities to those of walking and gradient-following already secured. This module stemmed from two precursor cells that read the same developmental subprogram and executed its instruction in a symetric way (Figure 15). It was assumed to receive information from two sensors, each indicating if an antenna got into contact with an obstacle, and it could influence the behavior of the locomotion module through inter-incremental connections created during the evolutionary process. To avoid parasitic interferences, no inter-incremental connections from Module 3 to Module 2 were allowed. The GRAM-3 grammar that was used is described in Figure 16. It only permitted each precursor cell to grow at most four connections to neurons of Module 1.

To evaluate the fitness of each program, a set of $N = 5$ different environments env_i, each containing a source of odor (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. When an obstacle was hit, the animat could no longer move until the end of the trial. The corresponding fitness function was:

$$f(env_i) = \frac{1}{t^{(i)}_{end}} \cdot \left(d(X(0), S(0)) - d(X(t^{(i)}_{end}), S(t^{(i)}_{end})) \right) + \int_0^{t^{(i)}_{end}} s(t) \cdot dt$$

[3] The second module had the same dimensions as the first and was considered to be positioned above it.

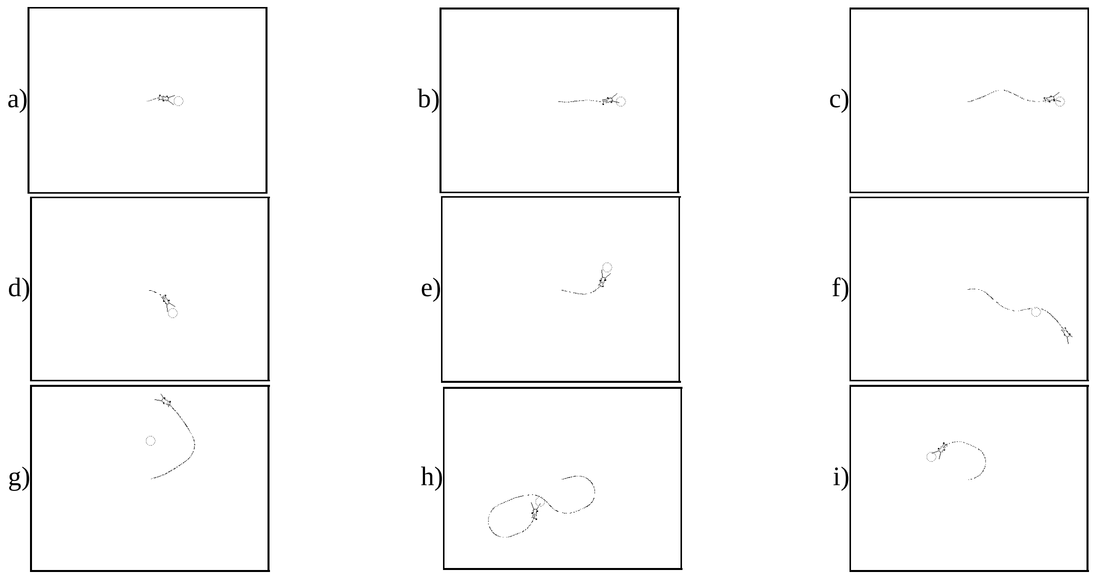


Fig. 13. Generalization experiments for the gradient-following task. An animat, which has been selected to reach a goal in 5 different test positions, is tested against 9 other goal positions. When the animat occasionally misses the goal (as in cases f and g), it may nevertheless reach it later (as in case h) if the evaluation time is lengthened.

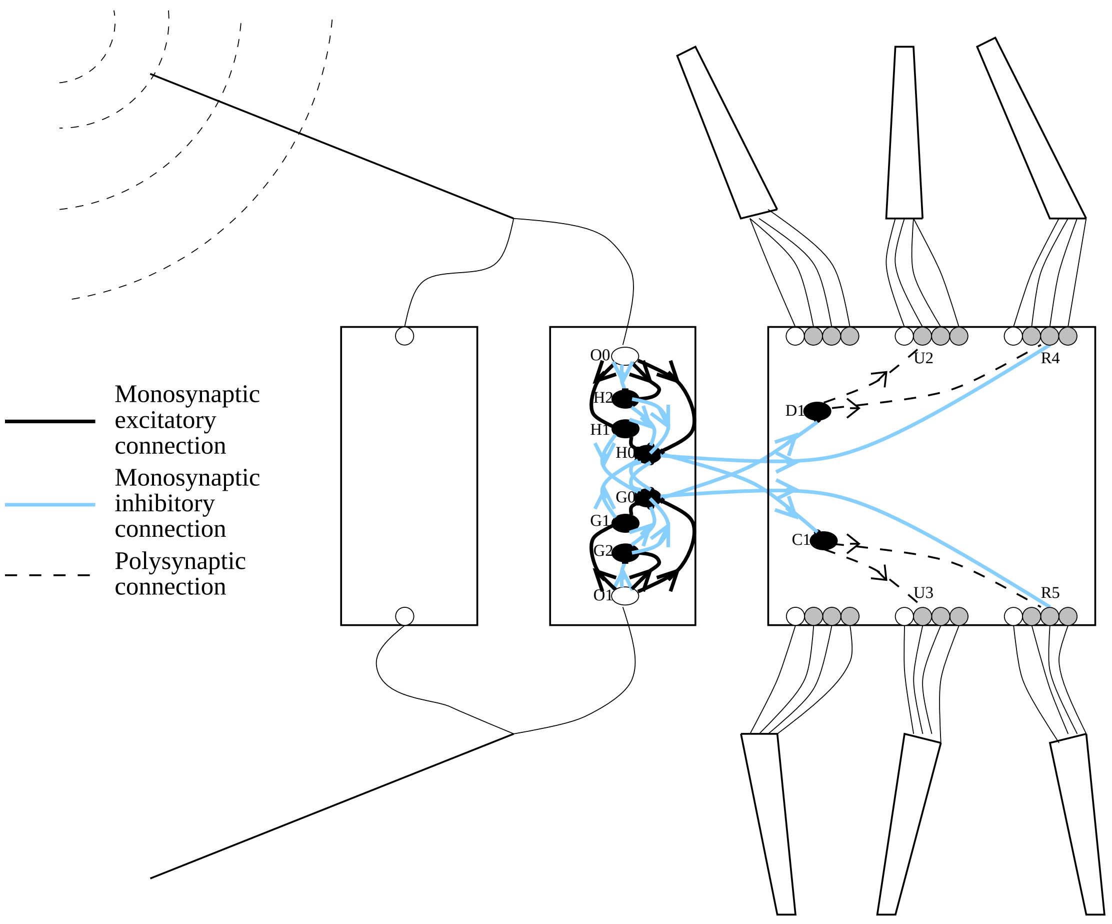

Monosynaptic excitatory connection
Monosynaptic inhibitory connection
Polysynaptic connection
O0
H2
H1
H0
G0
G1
G2
O1
D1
C1
U2
R4
U3
R5

Fig. 14. Gradient following mechanism for the animat of Figure 13. The activity of the two odor sensors $O0$ and $O1$ are compared thanks to the reciprocal inhibitory connection between neurons $G0$ and $H0$. When the goal is on the right of the animat, Interneuron $H0$ wins the competition against $G0$. If such is the case, a rotation towards the goal is instigated by the inhibition of Motoneuron $R4$, that prevents the right hind leg from rising. As soon as the difference between the signals received by both antennae becomes small enough, straight locomotion resumes. The inhibitory connection from Interneuron $H0$ to Interneuron $C1$ in Module 1 seems to play no meaningful functional role.

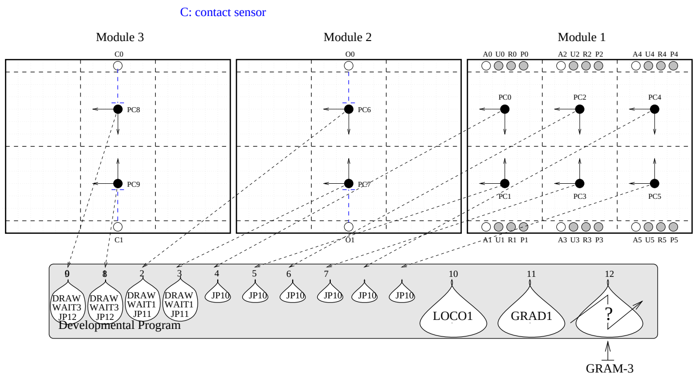


Fig. 15. Setup for the evolution of an obstacle-avoidance controller. DRAW instructions in subprograms 6 to 9 create a default connection between a precursor cell of Modules 2 or 3 and the associated sensory cell. These connections are copied to any daughter cell the precursor cells may have. WAIT instructions 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 GRAM-3 tree-grammar specified in Figure 16 below.

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.

Fig. 16. The GRAM-3 grammar.

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

where $t_{end}^{(i)}$ was the time at which the evaluation in Environment env_i stopped, and $s(t)$ 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 and will be discussed later on.

Again, five different experiments have been 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 selection-replacement events. Besides being surrounded or not by a rectangular wall, these test environments only contained circular obstacles. Generalization experiments, where individuals were tested in new environments, were often successfull, although some difficulties avoiding collisions with obstacles exhibiting sharp corners have been noticed (Figure 17).

Figure 18 describes how the obstacle avoidance capacities are implemented in the control architecture of the animat whose behavior is shown in Figure 17. The corresponding Module 3 contains 2 interneurons and 6 connections.

IV. Discussion

Insofar as the use of the SGOCE paradigm only requires that the experimenter provides a means of connecting the network's input and output neurons to the problem domain, together with a suitable fitness function, this paradigm should prove useful for the automatic design of neural networks in many application area. Thanks to the possibility of using appropriate grammars, it is likely to reduce the complexity of the networks it generates. However, it should be stressed that recourse to developmental programs to evolve neural networks has probably numerous consequences that are yet to be fully understood and assessed. In particular, it will probably be very difficult and counter-intuitive to understand the role that mutations and crossovers may have depending upon where they occur within tree-like developmental programs. It is, for instance, well known that a mutation occurring in the part of a genotype that is expressed in an early developmental phase will have more extensive consequences on the final phenotype than a mutation occurring in a late phase. Because it is already very difficult to adapt the genetic operators of traditional evolutionary algorithms so that they can select and favor useful building blocks, it is possible that the acquisition of the corresponding empirical or theoretical knowledge will be much more difficult and lengthy for applications resorting to a developmental process.

Be that as it may, results that have been obtained here prove that the SGOCE evolutionary paradigm provides a convenient means of generating neural networks capable of controlling the behavior of an animat. In particular, such results go further than any previous attempt [32], [17], [33], [29], [19] at automatically designing the control architecture of simulated insects or real 6-legged robots, attempts that have been limited to the evolution of mere straight locomotion controllers. As compared to the way these attempts were conducted, the efficiency of the SGOCE paradigm is probably due to the compact encoding it affords, thus tremendously reducing the size of the search space that other approaches are committed to explore. This paper also demonstrates that the incremental approach on which the SGOCE paradigm relies makes it possible to progressively enrich an animat's behavioral repertoire by capitalizing upon already functional neural networks whose inner workings are modulated by additional control modules. Such capacities are afforded by the geometry oriented processes of axonal growth that have been added to Gruau's basic encoding scheme [19]. SGOCE's incremental approach is similar to that used by Brooks [34] for the hand design of so-called *subsumption architectures*. It is also similar to the methodology advocated by de Garis [32] and by Harvey et al. [35]. Lastly, it is commonly used by nature to build control hierarchies [36] that are responsible for the adaptive behavior of animals.

Recourse to grammars constraining the structure of the developmental programs is not mandatory, although it helped in the present application to reduce the complexity of the evolved controllers and, thus, to reduce simulation time. In [37] a locomotion controller that has been evolved in the absence of syntactic constraints is presented: it exhibits 192 neurons and 2222 con-

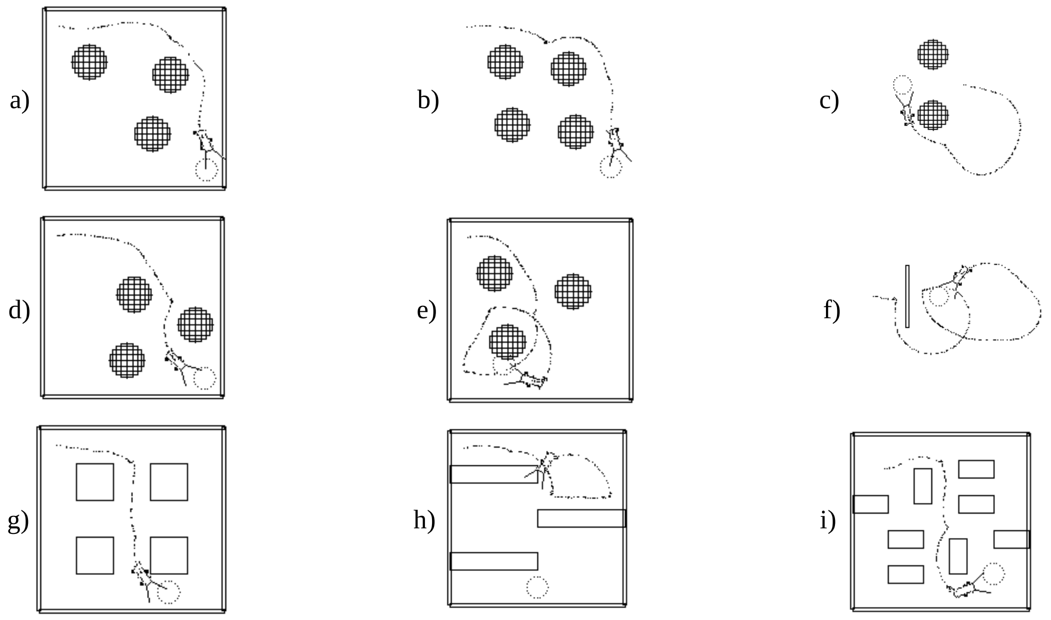


Fig. 17. Experimental results obtained when gradient-following and obstacle-avoidance behaviors are evolved. Cases (a-c) show the animat's trajectory within 3 out of the 5 test environments. Cases (d-i) show results of generalization experiments, in 6 new environments. The animat can deal with obstacle shapes never met during evolution (f-i). However, it cannot always avoid hitting sharp corners (h).

nections. However, it should be stressed that a potential drawback of using grammars is that the controllers thus generated may be too constrained. For instance, in the absence of additional experiments, one cannot dismiss the possibility that a less stringent grammar than GRAM-3 might have permitted the inclusion of more neurons and connections into Module 3, which would have improved obstacle-avoidance behavior and helped to deal more efficiently with sharp corners. Such a remark suggests future interesting research directions, which would let the grammars co-evolve with the developmental programs.

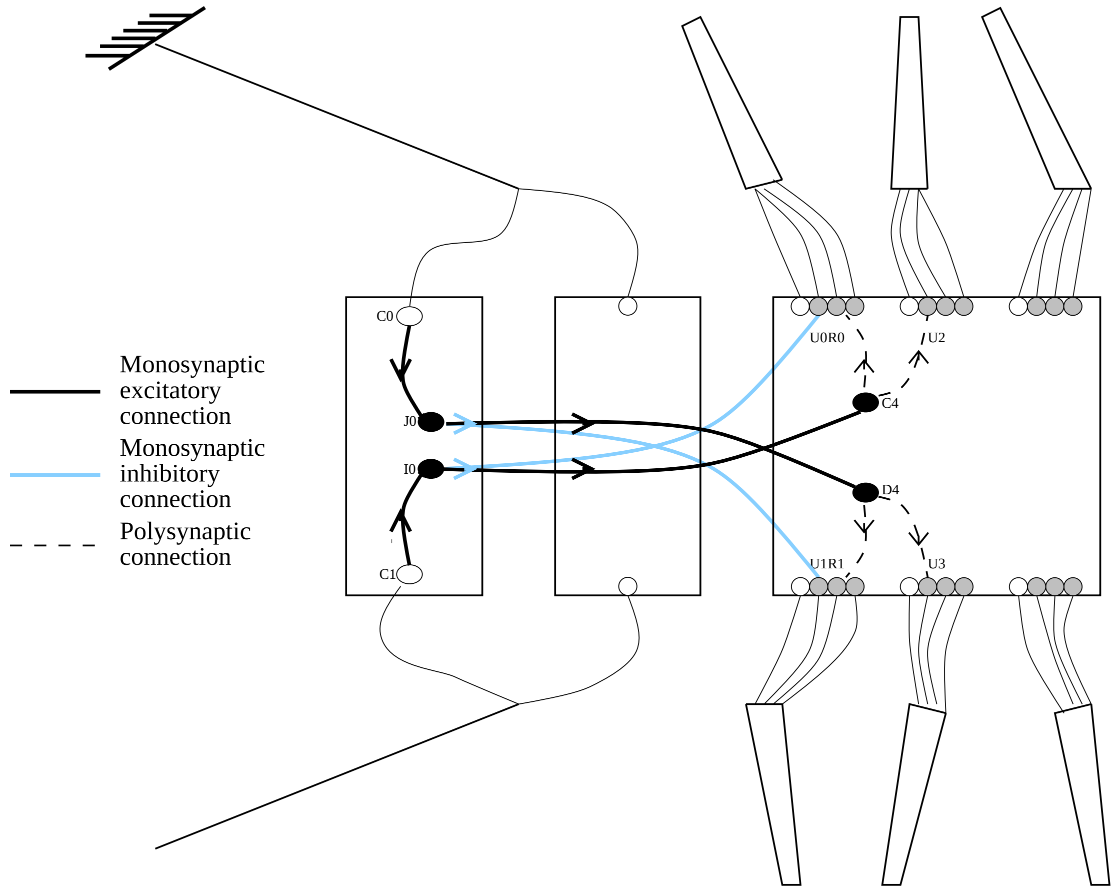
Likewise, although efficient overall behaviors have been obtained here while forbidding interconnections between Modules 2 and 3, it would certainly be interesting to seek how to optimize the animat's control architectures, for instance thanks to using co-evolving modules. In particular, in the absence of additional experiments, one may wonder whether the absence of suitable interconnections was not responsible for some antagonistic effects that Modules 2 and 3 had with respect of Module 1, antagonistic effects that were responsible for the animat's high falling rate in preliminary experiments. Although such effects have been cured by adding a second term penalizing falls in the fitness function of section III-C, future work might reveal that proper interconnections — like those that have been hand-coded by Beer [16] — are more suited to this end.

In the same manner, the solutions described in this paper were certainly heavily determined by the initial setups that have been used. In particular, according to their respective positions in the substrate, some cells had a better chance of getting connected to each other than did others. This, in particular, was the case with sensors, motoneurons and precursor cells whose initial positions were set by the experimenter, and which were, therefore, more or less likely to be incorporated into the final, developed neural network. Here again, a possible way of combating the negative consequences of an experimenter's arbitrary choice is to let the morphology of the animat co-evolve with its control architecture, an approach already explored by others [38], [39].

Finally, co-evolution could be used to let the learning set co-evolve with the animat population, so as to propose the most challenging environmental situations according to current population abilities. Such a possibility has been first proposed by Hillis [40] and further explored in [41], [42], [43].

The present work also demonstrates that, among the different paradigms that have been used to evolve the control architecture of an animat —- e.g., Lisp functions [21], [44], logic trees [45], [46], classifier systems [47] — recurrent artificial neural networks exhibit several specific and attractive features. Besides being universal dynamics approximators as already mentioned, it turns out that they are low-level, non specific primitives that can be combined to give rise to several mechanisms known to be at work in the control architectures of animals [48]. Thus, inside the controllers that have been evolved in this work, it is possible to identify reflexes and feedback mechanisms, together with oscillators and central pattern generators. Other mechanisms, like chronometers and rudimentary memories, have also been observed in a previous work [13]. Furthermore, incremental architectures like those that have been sought and generated herein are also known to be exhibited by the nervous systems of animals. Simple connections between such modules proved to be sufficient to control behaviors of increasing complexity, but the generality of this finding remains to be assessed. In particular, it will be enlightening to see how far such an approach could lead towards the discovery of more cognitive abilities than the simple stimulus-response pathways that have been evolved so far.

There are several research directions to be investigated in order to improve the behavior of the insects that have been synthetized here. In particular, it turns out that the locomotion controller that has been evolved as Module 1 is perfectly capable of



Monosynaptic excitatory connection
Monosynaptic inhibitory connection
Polysynaptic connection
C0
J0
I0
C1
U0R0
U2
C4
D4
U1R1
U3

Fig. 18. Obstacle avoidance mechanism for the animat of Figure 17. When the right antenna contacts an obstacle, the corresponding sensory cell $C0$ sends an excitatory signal to Interneuron $J0$. If the intensity of this signal is sufficient, the interneuron modulates the locomotion behavior so as to turn left, mainly by inhibiting motoneuron $U1$, thus preventing the left front leg from rising. As soon as the right antenna does no longer detect the obstacle, straight locomotion resumes. The effect of the excitatory connection from Interneuron $J0$ to Interneuron $D4$ of Module 1 is more difficult to elucidate, although it has been observed that the presence of this connection enhances the obstacle avoidance behavior.

generating backward locomotion. Therefore, recourse to appropriate fitness functions would probably lead to the generation of animats exhibiting improved obstacle-avoidance behavior that would, in particular, be able to escape from dead-ends.

It may likewise be hoped that the extension of the results described in [13], which lead to the automatic discovery of a switch device, will make it possible to implement more complex memory mechanisms in the animat's control architecture. This, together with the use of additional sensors that would afford minimal visual capacities, might help improving the animat's navigation behavior if it could detect and memorize specific landmarks in its environment.

Lastly, additional developmental instructions could be devised that would allow the synaptic weights of some connections to be changed during an animat's lifetime, thanks to an individual learning process. In a similar manner, other developmental instructions could be devised that would allow the developmental pathway to be dynamically changed depending upon the specific interactions that the animat experiences with its environment. Besides its operational value, every step in such directions would contribute to theoretical biology and enable to better understand the interactions between development, learning and evolution, i.e., the three main adaptive processes exhibited by natural systems [11].

V. Conclusion

It has been shown here that the current implementation of the SGOCE evolutionary paradigm makes it possible to automatically design the control architecture of a simulated insect that is capable not only of quickly walking according to an efficient tripod gait, but also of following an odor gradient while avoiding obstacles. Such results provide marked improvements over current state-of-the-art in the automatic design of straight-locomotion controllers for artificial insects or real 6-legged robots. They rely upon specific mechanisms implementing the developmental process of a recurrent dynamic neural network and upon an incremental strategy that amounts to fixing the architecture of functional sub-networks in a still evolving higher-level control system. There are several ways of improving the corresponding mechanisms, in particular by letting evolve several characteristics that have been arbitrarily set here by the experimenter, or by devising new developmental instructions that would add individual learning capacities to the processes of development and evolution. Such research efforts might be as usefull in an engineering perspective as in a contribution to a better understanding of the mechanisms underlying adaptation and cognition in natural systems.

References

[1] W. B. Dress, "Darwinian optimization of synthetic neural systems," in *Proceedings of the IEEE First International Conference on Neural Networks*, SOS Printing, San Diego, CA, 1987.

[2] A. Guha, S. Harp, and T. Samad, "Genetic synthesis of neural networks," Tech. Rep. CSDD-88-I4852-CC-1, Honeywell Corporate Systems Development Division, 1988.

[3] D. Whitley, "Applying genetic algorithms to neural net learning," Tech. Rep. CS-88-128, Department of Computer Science, Colorado State University, 1988.

[4] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: Using the genetic algorithm with connectionist learning," Tech. Rep. CS90-174, CSE, UCSD, CA, June 1990.

[5] G. F. Miller, P. M. Todd, and S. U. Hedge, "Designing neural networks using genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[6] J. Schaffer, D. Whitley, and L. Eschelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Combinations of Genetic Algorithms and Neural Networks* (D. Whitley and J. Schaffer, eds.), IEEE Computer Society Press, 1992.

[7] I. Kuscu and C. Thorton, "Design of artificial neural networks using genetic algorithms: review and prospect," Cognitive Science Research Paper 319, ISSN 1350-3162, University of Sussex, Brighton, UK, 1994.

[8] K. Balakrishnan and V. Honavar, "Evolutionary design of neural architectures — preliminary taxonomy and guide to literature," Tech. Rep. CS TR#95-01, Artificial Intelligence Group, Iowa State University, 1995.

[9] J. Branke, "Evolutionary algorithms for neural network design and training," in *Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications* (Talander, ed.), (Vaasa), 1995.

[10] T. Gomi and A. Griffith, "Evolutionary Robotics — An overview," in *Proceedings of the IEEE Third International Conference on Evolutionary Computation*, IEEE Press, 1996.

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

[12] M. Matarić and D. Cliff, "Challenges in evolving controllers for physical robots," *Robotics and Autonomous Systems*, vol. 19, pp. 67–83, 1996.

[13] J. Kodjabachian and J.-A. Meyer, "Evolution and development of modular control architectures for 1-d locomotion in six-legged animats," 1997. Submitted for publication.

[14] J.-A. Meyer, "The animat approach to cognitive science," in *Comparative Approaches to Cognitive Science* (H. Roitblat and J.-A. Meyer, eds.), The MIT Press / Bradford Books, 1995.

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

[16] R. Beer, *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*. Academic Press, San Diego, CA, 1990.

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

[18] F. Gruau, *Synthèse de Réseaux de Neurones par Codage Cellulaire et Algorithmes Génétiques*. Thèse d'université, ENS Lyon, Université Lyon I, January 1994.

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

[20] F. Gruau, "Artificial cellular development in optimization and compilation," in *Evolvable Hardware'95* (E. Sanchez and Tomassini, eds.), Lecture Notes in Computer Science, Springer Verlag, 1996.

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

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

[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[24] J. L. McClelland and D. E. Rumelhart, eds., *Parallel Distributed Processing*, vol. 1. The MIT Press/Bradford Books, Cambridge, MA, 1986.

[25] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing*, vol. 2. The MIT Press/Bradford Books, Cambridge, MA, 1986.

[26] D. Cliff, I. Harvey, and P. Husbands, "Explorations in evolutionary robotics," *Adaptive Behavior*, vol. 2, no. 1, pp. 73–110, 1993.

[27] D. Cliff and G. F. Miller, "Co-evolution of pursuit and evasion ii: Simulation methods and results," in *From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson, eds.), The MIT Press/Bradford Books, Cambridge, MA, 1996. Submitted.

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

[29] G. Spencer, "Automatic generation of programs for crawling and walking," in *Advances in Genetic Programming* (K. E. K. Jr., ed.), pp. 335–353, The MIT Press / Bradford Books, Cambridge, MA, 1994.

[30] J. Kodjabachian, "Simulating the dynamics of a six-legged animat," tech. rep., AnimatLab, ENS, Paris, 1996.

[31] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, ch. 36: Muscles, Effectors of the Motor Systems. Prentice Hall International Inc., third ed., 1991.

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

[33] 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), pp. 2618–2623, 1992.

[34] R. A. Brooks, "A robot that walk: Emergent behavior form a carefully evolved network," *Neural Computation*, vol. 1, no. 2, pp. 253–262, 1989.

[35] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: Artificial evolution, real vision," in *From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior* (D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, eds.), pp. 392–401, The MIT Press/Bradford Books, Cambridge, MA, 1994.

[36] Dawkins, *The blind watchmaker*. Longman Scientific & Technical, Essex, England, 1986.

[37] J.-A. Meyer, "From natural to artificial life: Biomimetic mechanisms in animat design," *Robotics and Autonomous Systems*, 1997. In press.

[38] 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, Cambridge, MA, 1994.

[39] F. Dellaert and R. D. Beer, "A developmental model for the evolution of complete autonomous agents," in *From Animats to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson, eds.), The MIT Press/Bradford Books, Cambridge, MA, 1996. Submitted.

[40] W. D. Hillis, "Coevolving parasites improve simulated evolution as an optimization procedure," in *Artificial Life II* (C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, eds.), pp. 313–324, Addison-Wesley, 1992.

[41] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–376, 1995.

[42] C. D. Rosin and R. K. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proceedings of the Sixth International Conference on Genetic Algorithms* (L. J. Eshelman, ed.), pp. 373–380, Morgan Kaufmann, San Mateo, CA, 1995.

[43] H. Juillé and J. B. Pollack, "Dynamics of co-evolutionary learning," in *From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson, eds.), pp. 526–534, The MIT Press/Bradford Books, Cambridge, MA, 1996.

[44] C. W. Reynolds, "Evolution of corridor following behavior in a noisy world," in *From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior* (D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, eds.), pp. 402–410, The MIT Press/Bradford Books, Cambridge, MA, 1994.

[45] W.-P. Lee, J. Hallam, and H. H. Lund, "A hybrid GA/GP approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks," in *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, 1996.

[46] W.-P. Lee, J. Hallam, and H. H. Lund, "Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots," 1997. To appear in *Proceedings of the Fourth IEEE International Conference on Evolutionary Computation*.

[47] L. B. Booker, "Classifier systems that learn internal world models," *Machine Learning*, vol. 3, pp. 161–192, 1988.

[48] C. R. Gallistel, *The Organization of Action: A New Synthesis*. Laurence Erlbaum Associates, Hillsdale, New Jersey, 1980.