Incremental Evolution of Neural Controllers for Navigation in a 6-legged Robot

D. Filliat J. Kodjabachian J.-A. Meyer
AnimatLab / OASIS - LIP6
Paris. France.

Abstract

This paper describes how the SGOCE paradigm has been used within the context of a "minimal simulation" strategy to evolve neural networks controlling locomotion and obstacle-avoidance in a 6-legged robot. Such controllers have been first evolved through simulation and then successfully downloaded on the real robot.

1 Introduction

In two previous articles (Kodjabachian and Meyer [5], Kodjabachian and Meyer [6]), it has been shown how the so-called SGOCE evolutionary paradigm could be used to generate neural networks capable of controlling the navigation of an artificial insect. More specifically, developmental programs generating controllers for locomotion, obstacle-avoidance and gradient-following have been automatically generated, thus endowing the insect with navigation abilities through a simple guidance strategy (Trullier and Meyer [10], Trullier et al. [11]). In this paper, we show how the SGOCE paradigm has been used to generate neural controllers for locomotion and obstacle-avoidance in a real 6-legged SECT robot manufactured by Applied AI Systems (figure 1). Results obtained on gradient-following will be published elsewhere. A review of similar approaches involving a variety of neural controllers and robots is available in Meyer [8].

2 The SGOCE evolutionary paradigm

The SGOCE evolutionary paradigm is characterized by an encoding scheme, by an evolutionary algorithm, by an incremental strategy, and by a fitness evaluation procedure that will be sketched in turn. More detailed descriptions can be found in Chavas et al. [1], Filliat [2], Kodjabachian and Meyer [5], Kodjabachian and Meyer [6].

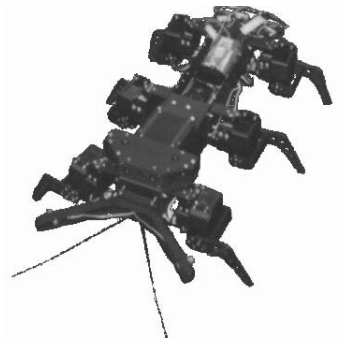


Figure 1: The SECT robot. It is equipped with infrared sensors that can be used to detect obstacles, and with light sensors that can be used for light-following. Each leg is controlled by two servo-motors that react to angular-position commands, e.g., one for horizontal moves and one for vertical moves.

2.1 Encoding scheme

The encoding scheme of SGOCE (figure 2) is a geometry-oriented variation of Gruau's cellular encoding (Gruau [3]). The developmental programs that are evolved have a tree-like structure and call upon developmental instructions that cause a set of precursor cells positioned by the experimenter in a 2D metric substrate to divide, die, or grow efferent or afferent connections. In particular, such cells can get connected to each other, or to sensory cells or motoneurons that have also been positioned in the substrate. Thus, a possibly short and compact genotype may ultimately produce a complex phenotype, i.e., a fully recurrent neural network made of individual leaky-integrator neurons and able to control the behavior of the robot through its sensors and actuators.

2.2 Evolutionary algorithm

The evolutionary algorithm of SGOCE is a steady-state genetic algorithm that involves a population of well-formed developmental programs whose structure

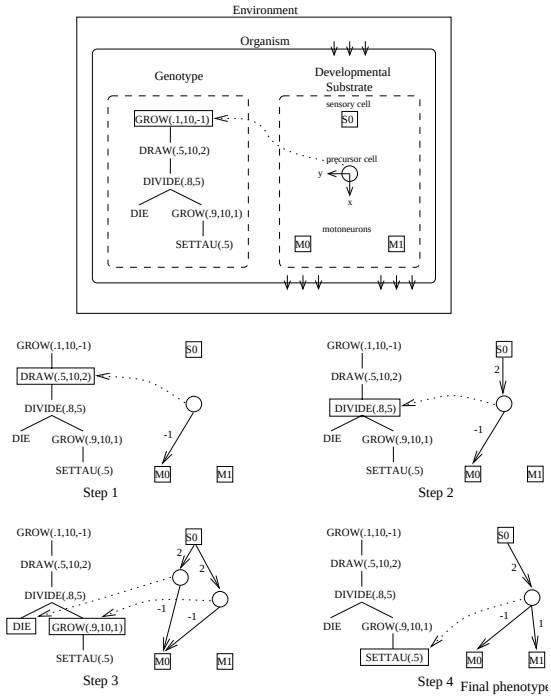


Figure 2: The developmental encoding scheme of SGOCE.The genotype that specifies the robot's nervous system is encoded as a program whose nodes are specific developmental instructions. This developmental program is read at a different position by each cell in the substrate. The precursor cell first makes connections with the motoneuron M0 and the sensory cell S0 (Steps 1 and 2). Then it divides, giving birth to a new cell that gets the same connections than those of the mother cell (Step 3). Finally, the mother cell dies, while the daughter cell makes a connection with the motoneuron M1 and changes the value of its time constant (Step 4).

is constrained by a grammar provided by the experimenter (figure 3). The use of such a grammar makes it possible to reduce the size of the genotypic space explored by the algorithm and to limit the complexity of the neural networks that are evolved.

2.3 Incremental strategy

Finally, the SGOCE paradigm resorts to an incremental strategy that takes advantage of the geometrical nature of the developmental process. In particular, it makes it possible to automatically generate appropriate controllers and behaviors through successive stages, in which good solutions to a simpler version of a given problem are iteratively used to seed the initial

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 3: This grammar defines a set of developmental programs, i.e., those that can be generated from it, starting with the Start1 symbol. When this grammar is used, a cell that executes such a program undergoes two division cycles, yielding four daughter cells, which can either die or modify internal parameters (e.g., time constant or bias) that will influence their behavior within the final neural controller. Finally, each surviving cell establishes a number of connections, either with another cell, or with the sensory cells or motoneurons that have been positioned by the experimenter in the developmental substrate. According to this grammar, no more than three successive divisions can occur and the number of connections created by any cell is limited to four. Thus, the final number of interneurons and connections created by a program well-formed according to this grammar cannot be greater than 8 and 32, respectively.

population of solutions likely to solve a harder version of the same problem.

Thus, in a first stage of the present work, the SGOCE paradigm was used to generate a recurrent neural network controlling straight locomotion in the SECT robot. At the end of this stage, this network was frozen, in the sense that the number of its neurons, their individual parameters, and their intra-modular connections were not allowed to evolve anymore. However, during a second evolutionary stage, an additional recurrent neural network was evolved and its neurons were allowed to grow, not only intra-modular connections between themselves, but also inter-modular connections to neurons in the locomotion controller. This additional controller was expected to modulate the leg movements secured by the first controller, so as to make it possible for the robot to turn in the presence of an obstacle in order to avoid it.

Figure 4 describes the two substrates that have been used to generate the two modules of the present application.

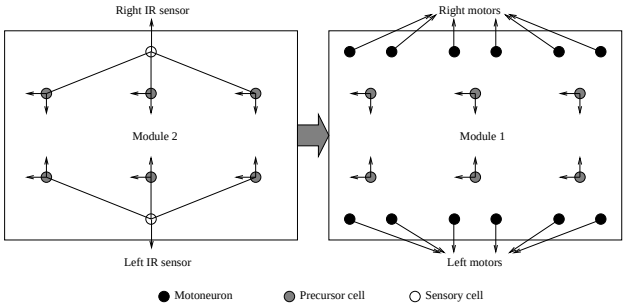


Figure 4: The substrates and the modular approach that have been used in the present work. Module 1 produces straight walking, while Module 2 modifies the behavior of Module 1 in order to avoid obstacles.

2.4 Fitness evaluation

Fitness evaluation is one of the main difficulties of the evolutionary design of controllers in real robots (Meyer et al. [9], Mataric and Cliff [7]). Such difficulties are enhanced in the case of legged robots, because they tend to be more brittle than their wheeled counterparts, and because fitness evaluations require a lot of time and cannot be easily automated. We therefore chose to use simulations to assess the fitness of our controllers, taking advantage of an argument put forth by Jakobi (Jakobi [4]), namely that what really matters is to accurately simulate the efficient behaviors that will be used by the real robot. Less efficient behaviors, which an efficient robot won't exhibit in reality, do not need to be minutely simulated, as long as we are sure that their fitnesses will be lower than the fitnesses of the behaviors that are sought. Pushing such reasoning to the extreme, Jakobi evolved neural controllers for an octopod robot capable of walking, of avoiding obstacles using its infra-red sensors, and backing away from objects that were hit with its bumpers. Jakobi's approach didn't resort to any simulation of the robot's behavior in its environment, and only relied on the specification that legs on the floor should move backwards as fast as possible, and that legs in the air should move forward as fast as possible. Given such specification, the simulation only rewarded controllers that did generate these movements. However, despite the practical success of Jakobi's approach, our aim was to provide less constraints on the target behavior. Therefore, we only specified that the robot should go ahead as far as possible while avoiding obstacles, and we didn't provide any hints about leg movments. To this end, we had to design a simulation of the behavior of the robot in its environment.

As Jakobi points out, the difficulty in devising a legged robot simulation is to manage the cases when some leg slippage occurs. However, because such events are only involved in poorly efficient behaviors, they are not expected to occur with a fast-walking gait. As a consequence, controllers producing leg slippage will never be used by an efficient real robot, and therefore leg slippage does not need to be accurately simulated, thus tremendously simplifying the simulation problem.

According to such considerations, our simulation assumed that all the legs were characterized by the same constant slippage factor, and simply calculated the movement of the robot body that minimized the slippage of any leg touching the floor between two time steps. As a consequence, if the real movement did not involve any slippage, the calculated movement was exact and, conversely, if the real movement did involve slippage, the calculated movement was a good approximation of the real one. This computation entailed the zeroing of the partial derivatives, with respect to translation and rotation, of the sum of the squared distances covered by each leg touching the floor. Technically, it only required a linear system inversion, which could be performed very efficiently (Filliat [2]).

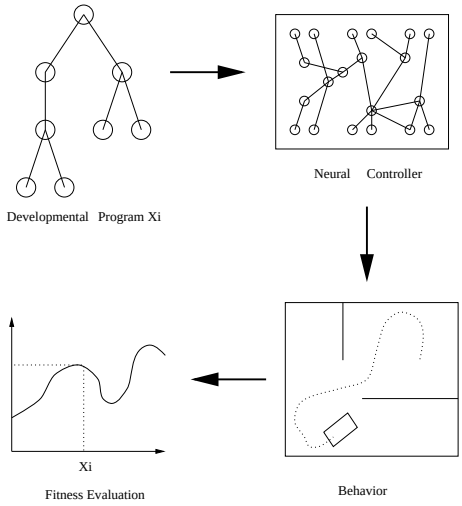


Figure 5: The three stages of the fitness evaluation procedure. An evolved developmental program is executed to yield an artificial neural network. Then, the neural network is used to control the behavior of a simulated robot. Finally, the fitness of the program is assessed according to the result of the simulation.

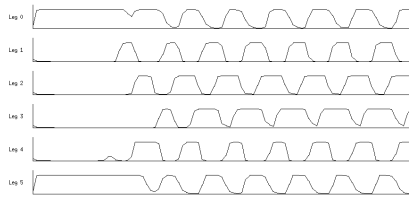
Such simulations have been used to assess the fitness of each developmental program produced by the evolutionary process (figure 5). Once an efficient con-

troller was thus obtained, it was downloaded on the SECT robot, where its ability to generate the target behaviors in reality was assessed again.

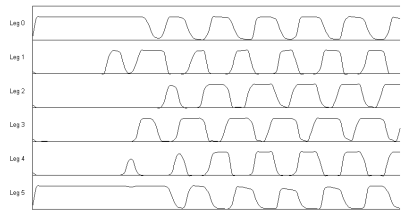
3 Experimental results

3.1 Locomotion

The 2D substrate that was used in this experiment is the Module 1 shown in figure 4. It contained 12 motoneurons that were connected to the 12 motors of the robot, in the sense that the activity level of a given motoneuron determined the target angular position that was sent to the corresponding servo-motor. The six precursor cells executed the same evolved developmental program in order to impose symmetrical constraints to the growing neural network. The corresponding fitness was the distance covered in an obstacle-free environment during a fixed amount of time, increased by a slight bonus encouraging any leg motion (Kodjabachian and Meyer [5], Kodjabachian and Meyer [6]). Finally, the size of the population was of 100 individuals that evolved during 500 generations (taking 24 hours on a SUN Ultra 1).



Simulated robot.



Real robot.

Figure 6: Comparison of leg movments in the simulated and the real robot. Both graphs show commands sent to leg swing motors. Althought the controller's outputs might have been altered by the procedure preventing over currents in the real robot, the commands actually sent to the motors, and hence the robot's overall behaviors, are qualitatively the same in simulation and in reality.

All the evolved controllers could be classified into two categories: those generating tripod gaits and those generating symmetrical gaits (i.e., moving the corresponding legs on both sides of the robot simultaneously). Such controllers were as efficient in reality as they were in simulation (figure 6). The main differences between both situations were due to the fact that, on the real robot, a continuous monitoring of the motor currents might entail modifying the motor commands independently of the neural network when such current were too high. This security procedure, which was implemented to avoid motor breaks in leg-blocking situations, was triggered in a few occasions when symmetric gaits were used, because such gaits occasionally provoked jumps producing very high motor currents.

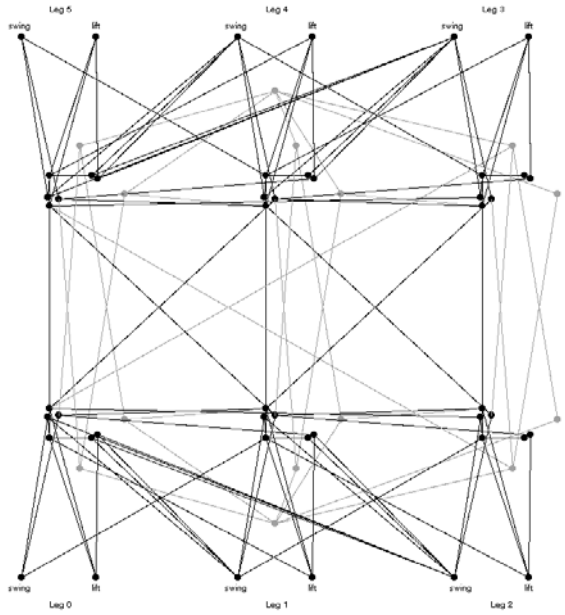


Figure 7: An example of an evolved controller. Black neurons belongs to the module controlling locomotion, gray neurons belong to the module controlling obstacle avoidance. The first controller contains 48 neurons and 82 connections; the second one contains 14 neurons and 36 connections.

Figure 7 shows the best tripod-gait controller that has been obtained. We analyzed its behavior in order to understand how this gait was generated. The mechanism responsible for leg lift is straightforward. It calls upon 6 pattern generators, one for each leg, made up of only 3 neurons. These pattern generators are synchronized by two connections between the legs of the same side of the body, and by two connections linking 2 symmetric legs on each side of the body. The

mechanism producing leg swing is far more intricate. In fact, the activation of a given leg's swing neuron depends on neurons involved in the control of all the other legs, and it cannot be decomposed in six similar units as is the case for the lift neurons.

3.2 Obstacle-avoidance

Obstacle-avoidance was sought using a second module whose neurons could be connected to those of the tripod-gait controller of figure 7. The substrate of this second module is shown on figure 4. It contained two input neurons linked to the IR sensors of the robot, and six precursor cells as in the first module. These six cells have forced connections to the input neurons. Each IR sensor was binary, i.e., it returned 0 when it detected no obstacle, and it returned 1 when an obstacle was detected within a 30 cm-range. The robot evolved in a closed environment containing some obstacles (figure 8). Instead of specifying what leg movments should be favored in each possible sensory circumstance, as Jakobi [4] did, the fitness was the distance covered by the robot until it touched an obstacle or until the simulation time was over. The population was made of 100 individuals and evolution lasted 200 generations.

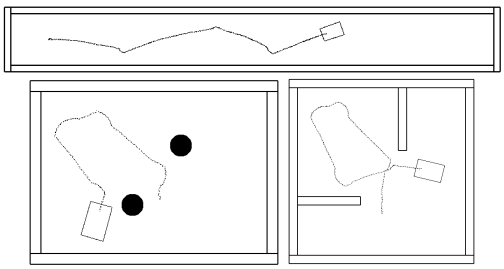
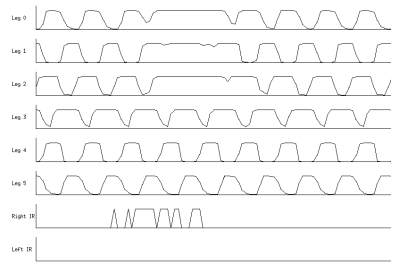
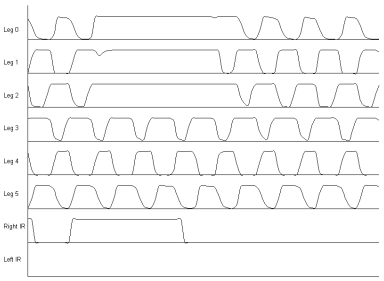


Figure 8: Examples of the behavior of the simulated robot in different environments.

As shown in figure 9, the robot's simulated and actual behaviors were very similar and quite simple: legs on the opposite side of a detected obstacle were blocked, thus causing the robot to change its direction and to avoid the obstacle. Analyzing the inner workings of the corresponding controller (figure 7), it turned out that such behavior was possible due to a strong inhibitory connection between a given sensor and the swing neurons of the legs on the opposite side of the robot.



Simulated robot.



Real robot.

Figure 9: Leg swings in the simulated and the real robot during obstacle avoidance. When an obstacle is detected at the right (Right IR) of the robot, the three legs on its left (Leg0, Leg1, Leg2) are temporarily blocked.

4 Discussion

Results that have been shown here demonstrate that the "mimimal simulation" methodology advocated by Jakobi (Jakobi [4]) may be effectively used to evolve non trivial behaviors in a real robot. However, our implementation of this methodology makes it possible to avoid specifying as many details about the dynamics of each effector involved in the production of the sought behavior as Jakobi was committed to do. Actually, we succeeded to evolve locomotion and obstacle-avoidance in a legged robot simply rewarding movement and punishing obstacle-hitting, and such high-level specification is more in the spirit of the automatic generation of behavioral controllers that evolutionary approaches afford than Jakobi's solution is. Nevertheless, it must be emphasized that a cost is associated to such a benefit, namely that of entailing a more detailed simulation than that of Jakobi. In other words, what is gained at the level of the simulation, may be lost at the level of the fitness evaluation and, for obvious lack of hindsight reasons, it is definitely unclear where it is worth devoting more resources in order to evolve any given behavior on a real robot.

Results that have been shown here also demonstrate the effectiveness of the SGOCE evolutionary paradigm. Its potentialities to evolve guidance navigation capacities in a simulated insect have already been exemplified elsewhere (Kodjabachian and Meyer [5], Kodjabachian and Meyer [6]). In the present work such potentialities have been extended to a real robot, although evolving a light-following behavior is still a matter of current research. For the same lack of hindsight reasons as above, it is however still unclear whether each aspect of this paradigm is mandatory, useful, or without any effect at all. One may wonder, for instance, if another evolutionary procedure than a genetic algorithm wouldn't lead to better results, if the developmental instructions or the constraining grammars that have been used here might not have been replaced by others, and if freezing one controller and letting a second one evolve is a better strategy than evolving both controllers at the same time. Concerning the latter point, however, it has been shown on a specific application involving obstacle-avoidance in a Khepera robot that this was not the case (Chavas et al. [1]) but, again, any generalization to other behaviors and other robots would be definitely premature.

5 Conclusions

Assuming that an on-board evolution of neural controllers wouldn't have been feasible on a SECT robot because of the excessively high demands that would have been placed on its motors, we called upon a "minimal simulation" approach and upon the SGOCE evolutionary paradigm to evolve tripod-gait locomotion and obstacle-avoidance. Successful neural controllers have been obtained, both in simulation and in reality, and their inner workings have been deciphered. However, it is yet unclear whether every implementation detail that has been used here was mandatory to the present success, nor whether it would be useful in any other application.

Acknowledgements

The authors express their gratitude to Dr. Takashi Gomi and to Applied AI Systems for having kindly lent us the SECT robot.

References

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

[2] Filliat, D. "Evolution de réseaux de neurones pour le contrôle d'un robot hexapode." *Lip6 Technical Report*, 1998.

[3] Gruau, F. "Automatic definition of modular neural networks.", *Adaptive Behavior*, 3, 1994.

[4] Jakobi, N. "Running across the reality gap : octopod locomotion evolved in minimal simulation." In Husbands, P. and Meyer, J.A. (Eds.). *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98*, Springer Verlag, 1998.

[5] Kodjabachian, J. and Meyer, J.A. "Evolution and Development of Modular Control Architectures for 1-D Locomotion in Six-legged Animats." *Connection Science*, 10, 1998.

[6] Kodjabachian, J. and Meyer, J.A. "Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects." *IEEE Transactions on Neural Networks*, 9:5, 1998.

[7] Mataric, M. and Cliff, D. "Challenges in evolving controllers for physical robots.", *Robotics and Autonomous Systems*, 19(1) , 1996.

[8] Meyer, J.A. "Evolutionary approaches to neural control in mobile robots." *In Proceedings of the IEEE International Conference on Systems, and Cybernetics*, San Diego, 1998.

[9] Meyer, J.A., Husbands, P. and Harvey, I. "Evolutionary Robotics: a Survey of Applications and Problems." In Husbands, P. and Meyer, J.A. (Eds.).*Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98*, Springer Verlag, 1998.

[10] Trullier, O. and Meyer, J.A. "Biomimetic Navigation Models and Strategies in Animats." *AI Communications*, 10, 1997.

[11] Trullier, O., Wiener, S., Berthoz, A. and Meyer, J.A. "Biologically- based artificial navigation systems: Review and Prospects." *Progress in Neurobiology*, 51, 1997.