SNAKE-LIKE BEHAVIORS USING MACROEVOLUTIONARY ALGORITHMS AND MODULATION BASED ARCHITECTURES

J. A. BECERRA, F. BELLAS, AND R. J. DURO[*]
Grupo de Sistemas Autónomos, Universidade da Coruña, Spain

J. DE LOPE
Grupo de Percepción Computacional y Robótica, Universidad Politécnica de Madrid, Spain

In this paper we describe a methodology for obtaining modular artificial neural network based control architectures for snake-like robots automatically. This approach is based on the use of behavior modulation structures that are incrementally evolved using macroevolutionary algorithms. The method is suited for problems that can be solved through a progressive increase of the complexity of the controllers and for which the fitness landscapes are mostly flat with sparse peaks. This is usually the case when robot controllers are evolved using life simulation for evaluating their fitness.

1. Introduction

The main goal in the design of snake-like robots is to imitate the body configuration and, therefore, the motion of biological snakes [1]. Snakes are able to crawl on almost any surface, including slopes, slippery ground or both. With more or less difficulty, the snake will arrive at the goal zone.

This kind of robots are mainly indicated when the objective is to reach zones that are difficult to access, or where the workspace and the environment do not allow the operation of conventional, wheeled or legged, robots. For a snake-like robot, a small space slightly larger than its body section is enough for moving forward. A classical application example of this kind of robots is motion in channels or pipes. From the energy consumption point of view, snake-like robots represent very efficient solutions. The total balance of consumed energy is comparable to other animals of similar mass [2].

Probably, the main contributions to snake-like robots are due to Hirose and his group [1] and more recently Chen et al. [3]. Basically they describe the implementation of several prototypes that emulate the behavior and gaits of bio-

[*] This work was supported by the MEC of Spain through project CIT-370300-2005-24 and Xunta de Galicia through project PGIDIT03TIC16601PR.

logical snakes. In order to define the motion as realistically as possible, Hirose formulated the *serpenoid* function, which is a sinusoidal function that associates parameters such as amplitude, frequency or phase to the curvature in the spine of the snake. Another approach is to define pattern generators whose outputs are applied consecutively and cyclically to the robot for producing the desired motion. Some of the most representative papers in this area are those by Shan and Koren [4], in which a set of patterns are established for controlling the robot and make it move forward and turn, and Nilsson [5] with his pattern for climbing.

The rest of the paper is organized as follows. In the next section we explain the structure and construction of the compound artificial network based behavior structure. Then we provide a description of the macroevolutionary algorithm used for the evolution of the controllers. After this we provide some examples of results obtained through the application of this approach and finally we provide some conclusions.

2. Constructing Control Architectures with Modulation

Evolving artificial neural network based control structures is commonplace in behavior based robotics research. The problem has become how to obtain complex multi-ANN structures that allow for the seamless combination of multiple ANNs operating together in an autonomous robot. A very promising approach is to contemplate the construction of the control system as an inter-modulating structure where the very basic behaviors are evolved individually and then, through the introduction of modulating structures, they are modified on line in order to adapt them to slightly changing situations. Thus, in the case of the construction of a control system for snake-like robots, one can obtain controllers for the generation of different types of snaking strategies when moving in a straight line and then use a modulating architecture in order to generate turns or intermediate strategies through their mutual modulation. If this process is carried out carefully, most of the structures thus obtained will be reusable for other tasks.

We consider two types of modulating structures: sensor modulators and actuator modulators. In our case, both types of modulators are constructed using ANNs and are obtained through evolution. Formally:

- A module X is an ancestor of a module Y if there is a path from X to Y.
- X is a descendant of Y if there is a path from Y to X.
- X is a direct descendant if there is a path of length 1 from Y to X.
- X will be called a Root node (denoted as R) if it has no ancestors.
- X is an actuator node (A) if its outputs establish values for the actuators.
- X is a selector node (S) if its output selects one of its direct descendants as the branch to follow, short-circuiting the others.

- X is an actuator modulating node (AM) if its outputs modify (multiplying by a value between 0 and 2) the outputs of its descendant nodes of type A. The modulations propagate through the controller hierarchy. If between R and A there is more than one AM that modulates one output of A, the resulting modulating value will be the product of the individual modulations in the path. Assuming that AM modulates the values of *n* actuators, its number of outputs must necessarily be *n*number of direct descendants*, as the modulation propagated to each descendant is usually different. When more than one node A provides values for the same actuator, the actuator receives the sum of these values. An AM does not necessarily modulate all the actuators over which the set of nodes acts, just any subset of them.
- X is a sensor modulating node (SM) if its outputs modify (multiplying by a value between 0 and 2) the inputs of its descendant nodes. The modulations propagate through the controller hierarchy to the actuator nodes. If between R and Y there is more than one SM that modulates one input of Y, the resulting modulating value will be the product of the individual modulations in the path. Assuming that a SM modulates the values of *n* sensors, its number of outputs must necessarily be *n*number of direct descendants*, as the modulation propagated to each descendant is usually different. A SM does not necessarily modulate all the sensors over which the nodes act, just any subset.

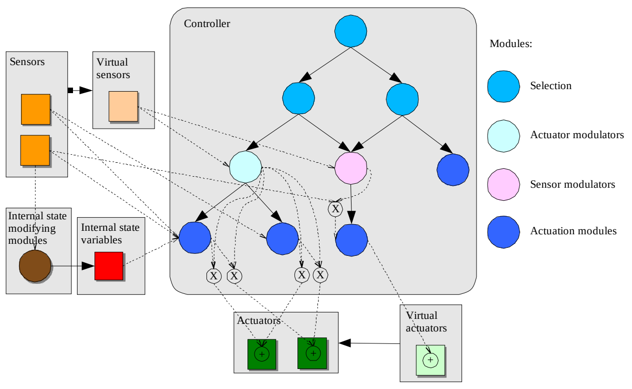

Fig. 1. Example of a controller with all the elements of the architecture.

The use of actuator modulators leads to a continuous range of behaviors for the transitions between those determined by the individual controllers. This is due to the fact that actuator values can now be a linear combination of those produced by every low level module.

Sensor modulators permit changing how a module reacts under a given input pattern transforming it to a different one. This way, it is very easy to make changes in the reaction to already learnt input patterns.

In addition to increasing the architecture's possibilities, modulation results in a very interesting secondary effect: there can be more than one sub-tree being executed simultaneously in the controller. So, the architecture is not really different from a distributed architecture where modules are categorized into different groups, because actuator modulators can be put together in the same level and sensor modulators can be set aside of the hierarchy and attached to the appropriate inputs where necessary. Figure 2 displays an alternative representation for a controller taking this equivalence into account.

Due to the large computational requirements for calculating the fitness of each solution (possible robot controller that must live its life out in a real or simulated environment), and in order to make computing times bearable, most processes in evolutionary robotics imply relatively small populations. The usual fitness landscapes with these types of life fitness functions imply large areas of mostly flat fitness values and some very sparse peaks where all the action takes place. This makes it very difficult for traditional genetic or evolutionary algorithms, which tend to converge to suboptimal solutions after under-exploring these very large solution spaces.

In this work we address this issue through the application of macroevolutionary algorithms which were proposed in [7]. The authors consider a new temporal scale, the "macroevolutionary" scale, in which the extinctions and diversification of species are modeled. The population is interpreted as a set of species that model an ecological system. The species may become extinct if their survival ratio with respect to the others is lower than a "survival coefficient". This ratio measures the fitness of a species with respect to the fitness of the others. When species become extinct, a diversification operator colonizes the vacancies with species derived from those which survived or with completely new ones.

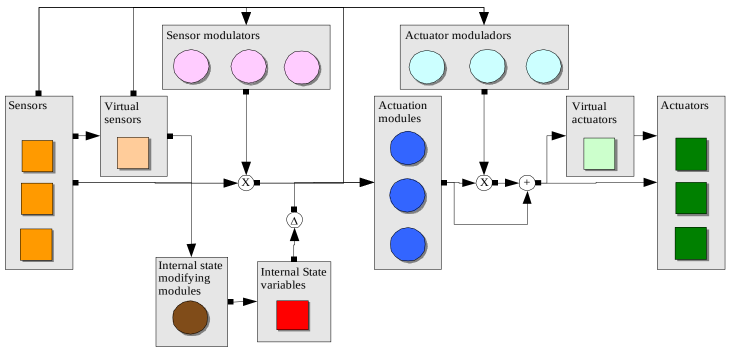

Fig. 2. Alternative representation for the architecture

3. Experiments and Results

To test the approach, we designed a very simple testing environment where, as in [4], we have a 7 segment snake with inter-segment joints that can turn in angles between -120 and 120 degrees and a set of actuators (solenoids) that allow the snake to establish friction with the surface (similar to thrusting sticks into the ground). These actuators can either be on or off. The snake operates in an environment with a light source it must reach.

To obtain an unbiased fitness function for the evolutionary process, during evolution we lay pieces of food on the ground in the direction of the light and every time the snake goes over a piece of food its energy is increased. When it moves, its energy is decreased. This type of fitness function favours reaching the light as fast as possible. All of the controllers evolved are artificial neural networks whose outputs provide the angles for the joints and the state of the solenoids and their inputs are the output angles and solenoids in the previous instant of time. In the case of modulators, the input is the sensor value for the position of the light and the outputs are the modulation values for the angles.

We have run multiple evolution processes under different physical and energetic constraints where, initially, the snake had to reach the light following a straight path in order to obtain the basic snaking behavior. After the behavior was obtained, we ran evolutionary processes to produce the modulators that would allow the snake to reach a light even if it had to turn.

Figures 3a and 3b display the results of two evolutions of a snaking behavior under different constraints. The populations used in the tests were between 800 and 16000 individuals. Each chromosome was 320 genes long containing real numbers. The controller obtained by the process was a synaptic delay based type network with two 6 node hidden layers, whose 13 inputs were the outputs of the controller in the previous instant of time in a recursive topology. The gait the snake obtained in the first case is very similar to the typical sidewinder mo-

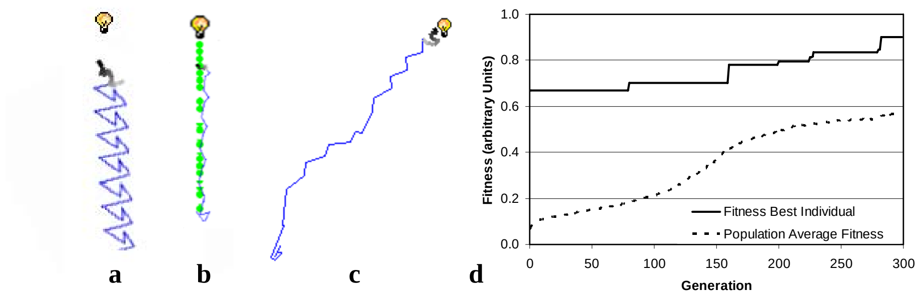

Fig 3: Best individual in basic straight line sidewinding motion behavior and in a regular snaking behavior (a, b). Turning behavior through the modulation of the previous straight line motion behavior (c). Evolution of fitness when evolving the modulator (d)

tion found in some real snakes. In the second case (figure 3b) the motion is more similar to a traditional snaking behavior. Figure 3c displays the behavior obtained through the modulation of the robot's main behavior controller (shown in 3b) in order to turn towards the objective. This modulating controller is also a synaptic delay based neural network with 2 input nodes, one 6 node hidden layer and 6 output nodes (modulating the angles of the basic snaking behavior). It was evolved for 300 generations using a 72 gene chromosome and 16000 individuals and, as shown in Figure 3c provides a successful modulated behavior. Figure 3d shows the fitness for the best individual and the average of the population when evolving the modulator where, as we can see, the fitness increases to a stable level in a few generations due to the intrinsic simplicity of the modulators that arise using this architecture.

4. Conclusions

In this paper we propose an approach for constructing multiple module behavior architectures for snake-like robots through a modulation structure using macro-evolutionary algorithms. The application of this method implies first the generation of the basic behavior controller in a simplified environment and the consequent generation of the necessary modulators in order to achieve the complex behavior by gradually increasing the complexity of the environment. The results show that this way it is quite simple to generate all types of intermediate behaviors in a very natural manner.

5. References

1. S. Hirose, Biologically Inspired Robots. Snake-like Locomotors and Manipulators. Oxford University Press (1993).
2. M. Walton and B.C. Jayne, "The Energetic Cost of Limbless Locomotion", Science, 524–527 (Aug. 1990).
3. L. Chen, Y. Wang, S. Ma and B. Li, "Analysis of Traveling Wave Locomotion of Snake Robot", Proc. IEEE Int. Conf. RISSP 365–369 (2003).
4. Y. Shan and Y. Koren, "Design and Motion Planning of a mechanical Snake", IEEE Trans. on Syst., Man, and Cyb.. 23(4):1091–1100 (1993).
5. M. Nilsson, "Snake Robot Free Climbing", IEEE Control Systems, 21–26, (1998).
6. Duro, R. J., Santos, J., Becerra, J. A. (2000), "Evolving ANN Controllers for Smart Mobile Robots", Future Directions for Intelligent Systems and Information Sciences, Springer-Verlag, pp. 34-64.
7. Marín, J. and Solé, R.V. (1999), "Macroevolutionary Algorithms: A New Optimization Method on Fitness Landscapes", IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp. 272-286.