Neural Networks Robot Controller Trained with Evolution Strategies

Antonio Berlanga
ScaLab, Univ. Carlos III de Madrid, SPAIN
30 Avda. Universidad
Leganés, Madrid 28911
aberlan@ia.uc3m.es

Pedro Isasi
ScaLab, Univ. Carlos III de Madrid, SPAIN
30 Avda. Universidad
Leganés, Madrid 28911
isasi@ia.uc3m.es

Araceli Sanchis
ScaLab, Univ. Carlos III de Madrid, SPAIN
30 Avda. Universidad
Leganés, Madrid 28911
masm@inf.uc3m.es

José M. Molina
ScaLab, Univ. Carlos III de Madrid, SPAIN
30 Avda. Universidad
Leganés, Madrid 28911
jmolina@ia.uc3m.es

Abstract – Neural Networks (NN) can be used as controllers in autonomous robots. The specific features of the navigation problem in robotics make the generation of good training sets for the NN very difficult. In this paper an evolution strategy (ES) is introduced to learn the weights of the NN instead of the learning method of the network. The ES is used to learn high-performance reactive behavior for navigation and collisions avoidance. No subjective information about "how to accomplish the task" has been included in the fitness function. The learned behaviors are able to solve the problem in different environments; so, the learning process has proven the ability to obtain a specialized behavior. All the behaviors obtained have been tested in a set of environment and the capability of generalization is showed for each learned behavior. A simulator based on mini-robot Khepera has been used to learn each behavior.

1 Introduction

Autonomous robots are sometimes viewed as reactive systems; that is, as systems whose actions are completely determined by current sensorial inputs. This is the base of the subsumption architecture (Brooks 1991), where finite state machines are used to implement robot behaviors. Other systems use fuzzy logic controllers instead (Ishikawa 1995). The rules of these behaviors could be designed by a human expert, designed "ad-hoc" for the problem or learned using different artificial intelligence techniques (Matellán 1995). In this work, the control architecture used to evolve the reaction (adaptation) is based on a neural network.

The neural network controller has several advantages: NN are resistant to noise, that exists in real environment, and are able to generalize their ability in new situations, a NN could easily exploit several ways of learning during its lifetime. The used of a feed forward network with the input units gathering sensor information and the output units directly connected to motors appears in previous works (Miglino 1995) as an efficient way to learn a simple behavior as "avoid obstacles". In this work the NN ought to learn more complex behavior: "navigation".

In the proposed model, the robot starts without information about the right associations between environmental signals and actions responding to those signals. And from this situation the robot is able to learn through experience to reach the highest adaptability grade to the sensors information. The number of inputs (robot sensors), the range of the sensors, the number of outputs (number of robot motors) and its description is the only previous information. These constrain makes the generation of good training sets for the NN very difficult. Instead of using a classical learning method to adjust the weights of the NN, an Evolutionary Strategy has been applied.

The fitness value of each individual in ES is computed using some objective quality measures related with the trajectory of the controller. The experiments have been carried out using a robot simulator with different environments. Each achieved solutions solve accurately the specific navigation problem in which has been trained. The controllers obtained have also the ability to adapt to environments in which they worse perform.

2 Evolution Strategies

Evolution strategies (ES) developed by Rechenberg (Rechenberg 1973) and Schwefel (Schewefel 1981), have been traditionally used for optimization problems with real-valued vector representations. As Genetic Algorithms (Goldberg 1989) (GA) the ES are heuristic search techniques based on the building block hypothesis. Unlike GA, however, the search is basically focused in the gene mutation. This is an adaptive mutation based on the likely the individual represents the problem solution. The recombination plays also an important role in the search, mainly in the adaptive mutation.

0-7803-5536-9/99/$10.00 ©1999 IEEE

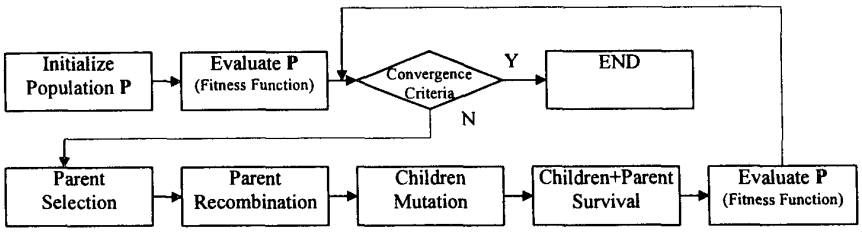

Figure. 1: Schema of an evolution strategy

Figure 1 shows a typical evolution strategy. First, it is necessary to codify each solution of the problem in a real-valued vector. Each vector represents a solution and also an individual. The method consists in evolving solution sets, called populations, in order to find better solutions. Selecting pairs of individuals (parents) that produce new individuals (children) via recombination, which are further perturbed via mutation performs the evolution of populations. The best individual (μ+1 selection) or the best individuals (μ+λ selection), in the set composed by parents and children, are selected to form the next population (Rechenberg 1989).

An individual is represented by $a = (x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n) \in \Re^n$, that are the n real values (x_i) and their corresponding deviations (σ_i) used in the mutation process for the (μ+λ) ES. The mutation is represented by equations (1) and (2).

$$\sigma_i' = \sigma_i \cdot \exp(N(0, \Delta\sigma)) \qquad (1)$$

$$x_i' = x_i + N(0, \sigma_i') \qquad (2)$$

Where x_i' and σ_i' are the mutated values, following a normal distribution ($N(\mu, \sigma)$).

However, when a (μ+1) ES is used the mutation process follows the 1/5 rule (Rechenberg 1989). In both cases, the recombination follows the canonical GA approach (Goldberg 1989).

3 Experimental Environment

The system has been developed using a simulator to prove different characteristics of the system. The task faced by the autonomous robot is to reach a goal in a complex environment avoiding obstacles found in the path. Different environments have been used to find the connections of the NN. Finally, a real robot has been used to test the proposed solution.

A simulator developed in a previous work (Sommaruga 1996) has been used as complete software for the simulation of mobile robot. Working with a simulation offers the possibility to evaluate several systems in different environments controlling the execution parameters. The robot simulator characteristics is based on a mini-robot Khepera (Mondada 1993) has been used, which is a commercial robot developed at LAMI (EPFL, Laussanne Switzerland). The robot characteristics are; 5.5 cm of diameter in circular shape, 3 cm of height and 70 gr. of weight. The robot has two wheels controlled by two motors that let any type of movement. The ES should specify the wheel velocity that could be read later by an odometer. Eight infrared sensors supply two kinds of incoming information: proximity to the obstacles and ambient light. Instead of using eight sensors individually, to reduce the amount of information six sensors are used and grouped (as Figure 2 shown) giving a unique value, the average, from two input values. Representing the goal by a light source, the ambient information lets the robot know the angle (the angle position in the robot of the ambient sensor receiving more light) and the distance (the amount of light in the sensor).

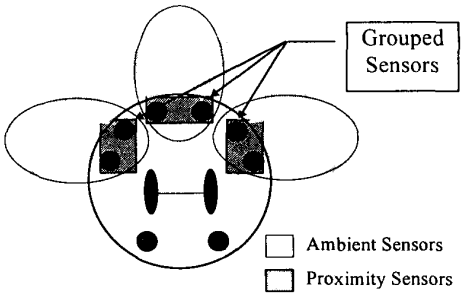

Figure. 2: Sensors considered in the real robot

The simulated world consists of a rectangular map of user defined dimensions, where particular objects are located. In this world it is possible to define a final position for the robot (the goal to reach), (Figure 3 (a)). In this case, the robot is represented with three proximity sensors and two special sensors to measure the distance and the angle to the goal.

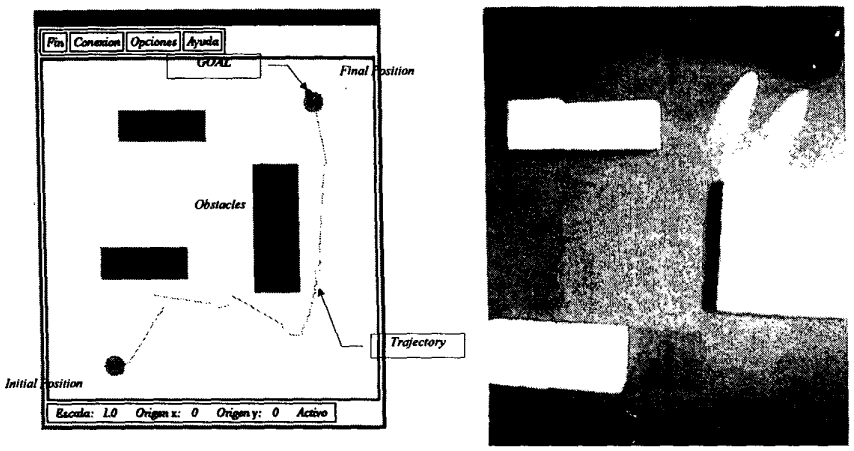

Figure. 3: (a) SimDAI Simulator (Example of one simulated environment). (b) Example of a real experimental environment.

Different simulated worlds that resemble real ones have been defined before being implemented in the real world. An example of these environments is shown in Figure 3 (a) and Figure 3 (b). The controlled developed is the same in both cases (simulated and real) except the differences in the treatment of the sensors.

4 Evolving NN connections by means of Evolution Strategies

It has been proved that by means of connections between sensors and actuators, a controller is able to solve any autonomous navigation robotic behavior (Braitenberg 1984). This theoretical approach is based on the possibility of finding the right connections of a feed-forward NN without hidden layers for each particular problem. The input sensors considered in this approach are the ambient and proximity sensors of Figure 2. The NN outputs are the wheel velocities. The velocity of each wheel is calculated by means of a linear combination of the sensor values using those weights (Figure 4):

$$v_j = f\left(\sum_{i=1}^{5} w_{ij} \times s_i\right) \qquad (3)$$

Where w_{ij} are searched weights, s_i are sensor input values and f is a function for constraining the maximum velocity values of the wheels.

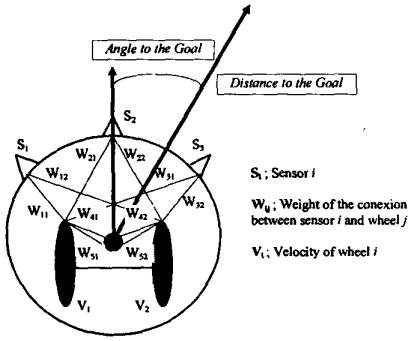

Figure. 4: Connections between sensors and actuators in the Braitenberg representation of a Khepera robot

Weight values depend on problem features. To find them automatically, an ES is proposed. In this approach each individual is composed by a 20 dimensional-real valued vector, representing each one of the above mentioned weighs and their corresponding variances. The individual represents one robot behavior consequence of applying the weights to the equation 3. The evaluation of behaviors is used as fitness function.

In order to make the problem more realistic no information about the location of the goal, neither direction nor distance, has been included in the evaluation function.

5 Experimental Results

Different experiments have been done all of them over the same set of environments. The environments have been generated by changing the goal position, number and location of obstacles. In a set of preliminary comparisons, it was found that results obtained with the software model did not differ significantly from the results obtained with the physical robot.

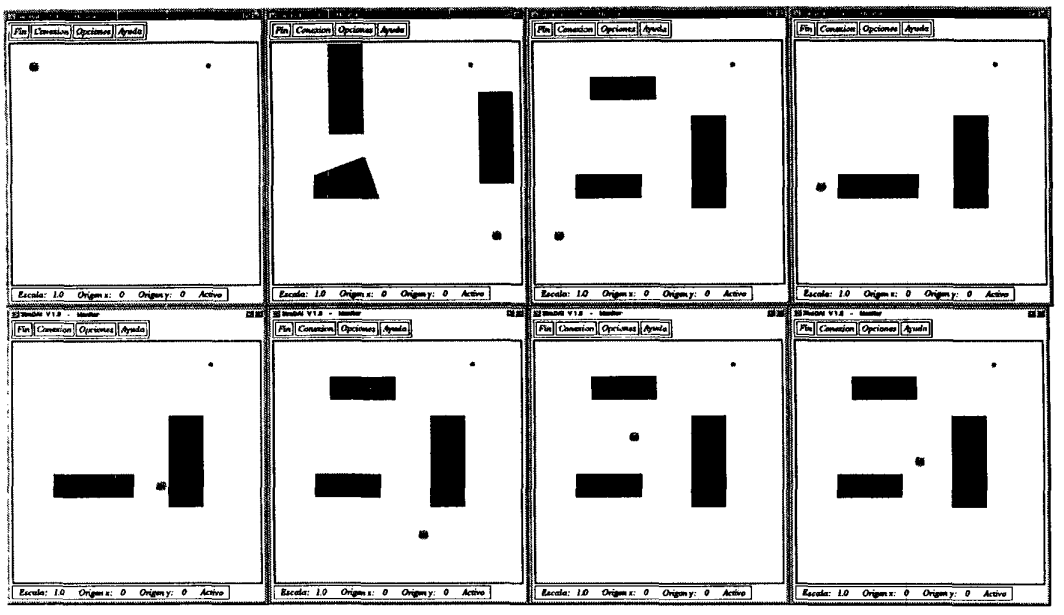

Figure 5. Some of the environments used to evolve the controller. Dark shapes are the obstacles, the big point is the starting location of the robot and the small point is the goal. The environments are closed.

An exploratory set of experiments was performed in simulation to adjust the quality measures used in the fitness function as well as the parameters of Evolution Strategy. A $(\mu+\lambda)$-ES, $\mu=6$, $\lambda=6$, were used.

The quality measures used to calculate the fitness value of a controller were the following:

- Number of collisions. (Collisions)
- Number of stops. Cycles of the simulation in which the robot stays in the same location. (Stand)
- Time needed to reach the goal. (Time)
- Length of the robot trajectory from the starting point to the final location. (Path Length)

The global evaluation depends linearly with these concepts: 10*Collisions + 10*Stand + 20*Time – 1,5*Path_Length. Each evaluated robot behavior ends over one environment when the goal has been reached or the time exceed some time out.

Five evolutionary runs of 70 generations each have been performed, for eight different environments, each one starting with a different seed for initializing the computer random functions.

The evolution of the quality measures used to calculate the fitness value shows a similar behavior over all environments. All the quality measures evolve in the way to get the optimal robot behavior. See Figures 6-10.

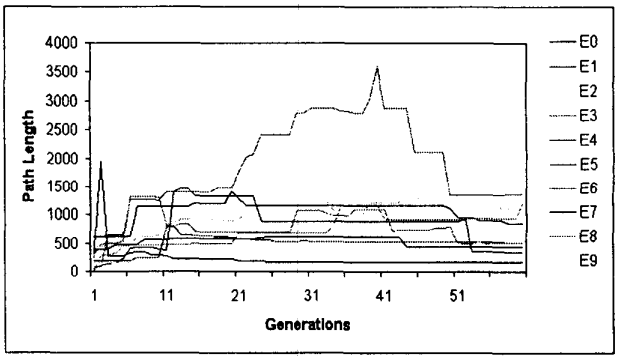

Figure 6. Evolution of the "Path Length" versus generations in each environment

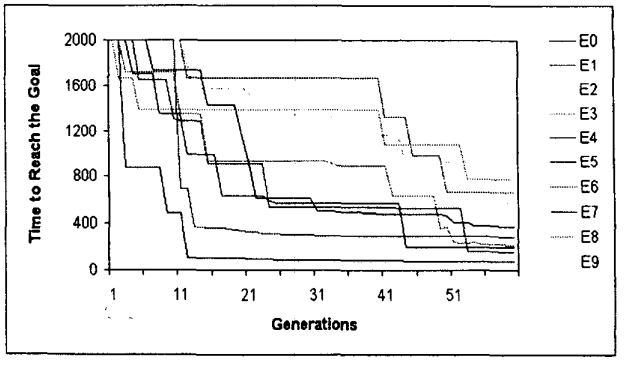

Figure 7. Evolution of "Time" needed to reach the goal versus generations in each environment

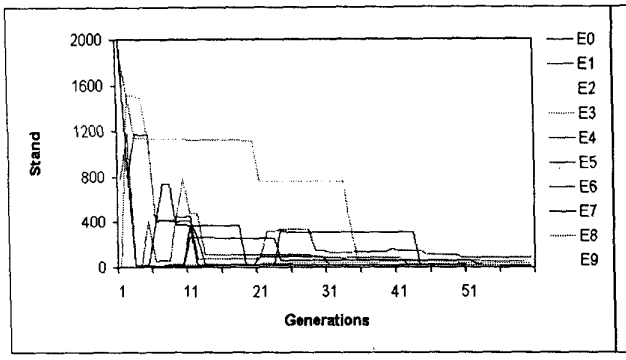

Figure 8. Evolution of the "Stand" versus generations in each environment

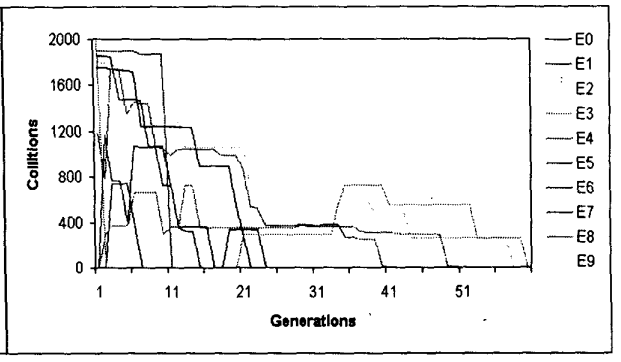

Figure9. Evolution of "Collisions" versus generations in each environment

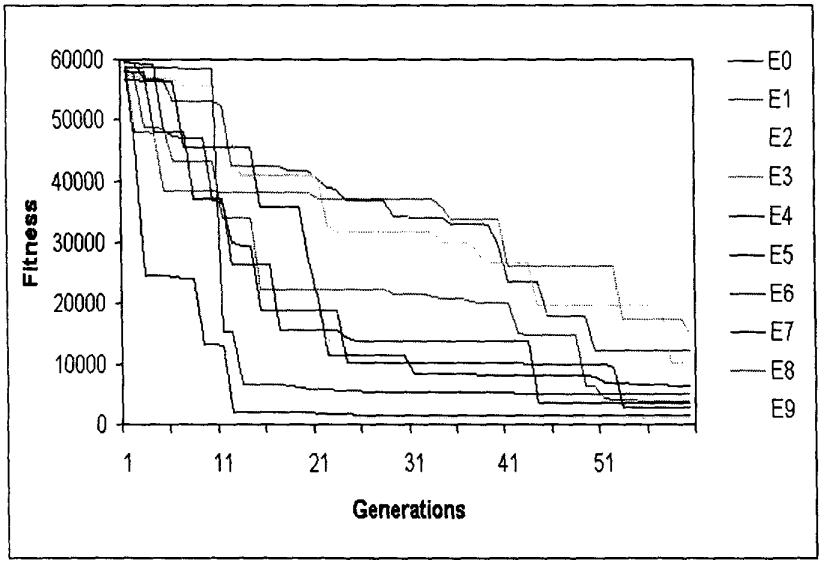

Figure 10. Evolution of the fitness value of the population's best controller versus generations in each environment

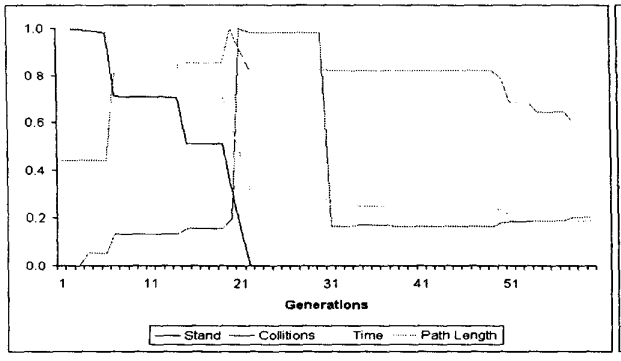

Figure 11. Evolution of the quality measures versus generations in environment 1

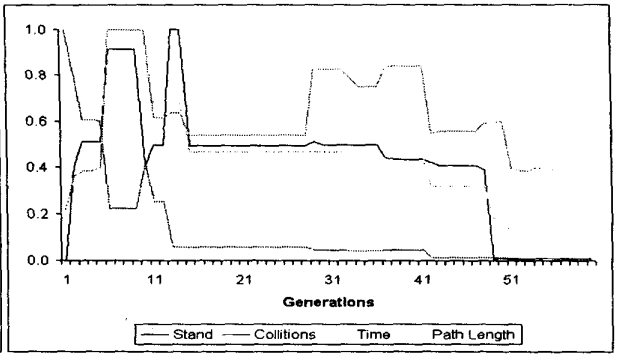

Figure 12. Evolution of the quality measures versus generations in environment 8

Figures 11 and 12 show the evolution of the quality measures of environments 1 and 8.

Very different behaviors are observed. For environment 1, which consists in an initial configuration without close objets, initially exploratory behaviors appear. The robot covers long distances but without avoiding obstacles. On the contrary, for environment 8, due to the proximity of obstacles to the starting point, the controller will not be able to explore the environment searching for the goal, until it does not acquire the ability to avoid obstacles. The environment guides the learning process.

The obtained controllers are valid for the environment in which they are trained. Figure 13 shows the behavior of a controller in the environment where it has been trained, as well as in other new environments.

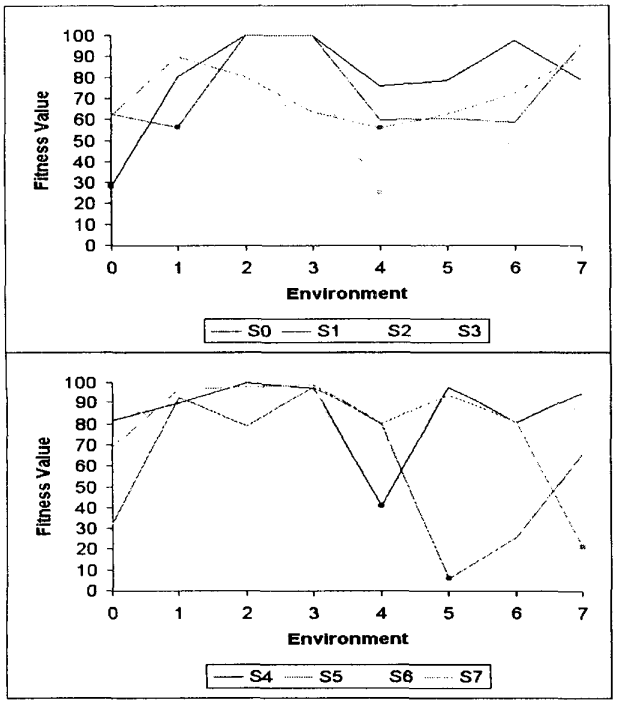

Figure 13. The fitness value of the solution (S_n) obtained in environment n is measured in all environments. The point shows the fitness value calculated in the training environment

Neural networks trained with an ES adjust precisely their weights to the training environment. This is an advantage when we want to obtain a good solution within a short processing time but a lack for getting generalized solutions. This behavior is displayed in Figure 14; where the solution trained in environment 3 is validated in environments 1,2 and 6.

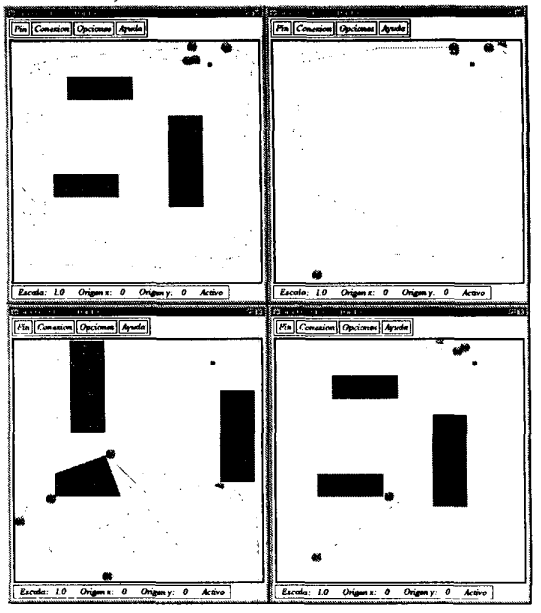

Figure 14. Solution trained in environment 3 plots its behavior in environments 3, 1, 2 and 6 respectively

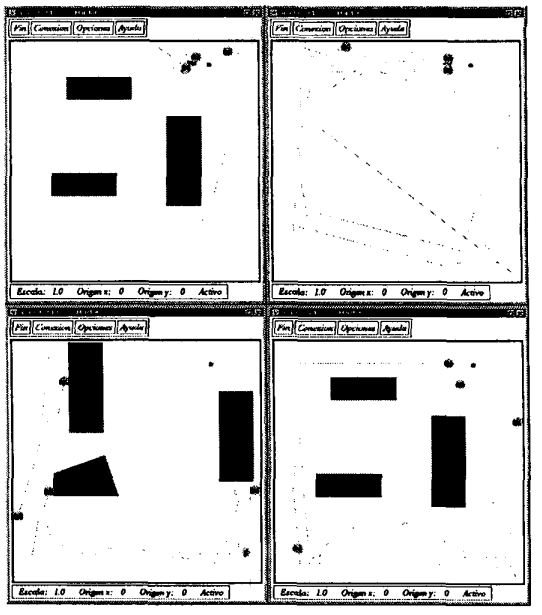

Figure 15. Solution trained in environment 6 plots its behavior in environments 6, 1, 2 and 3 respectively

In this case, it can be seen how when tested in environments 2 and 6 the robot is, in many cases, unable to reach the goal, having an erratic behavior. However in environment 1 the behavior is a good one, because of the resemblance between environments 1 and 3. Similar behaviors occur with all the controllers learned.

6 Conclusions and further work

The experiments prove the possibility of learning behaviors in an autonomous robot by means of an ES. The process has been applied on a simple NN where the directly associations between sensors and motors allows to solve a navigation problem. The learning process has specially proved their capability of learning for specific situations. The achieved controllers perform very well for the trained problems and make the process very useful in the search for "ad-hoc" solutions. However, they do not perform well when tested in environments different from the trained ones. Even in those cases, the efficacy of the learning process makes the adaptation of the solutions to any environment feasible. The method can be also extended to other more complex NN.

It is important to remark that the fitness functions doesn't include any subjective information "how to accomplish the task" but objective information about "how the task has been accomplished".

As a consequence, the learning process can be easily modified in order to consider new problems that could appear such as: surrounding an obstacle, or hiding from the light. The adaptation to new problems does not require too much effort because of no inclusion of local information about the problem in the fitness function.

Bibliography

Brooks (1991) Brooks R. A. "Intelligence without Representation". Artificial Intelligence, 47, 139-159, (1991).

Ishikawa (1995) Ishikawa S. "A Method of Autonomous Mobile Robot Navigation by using Fuzzy Control". Advanced Robotics, vol. 9, No. 1, 29-52, (1995)

Matellán (1995) Matellán V., Molina J.M., Sanz J., Fernández C. "Learning Fuzzy Reactive Behaviors in Autonomous Robots". Proceedings of the Fourth European Workshop on Learning Robots, Germany, (1995).

Miglino (1995) Miglino O., Hautop H., Nolfi S. "Evolving Mobile Robots in Simulated and Real Environment". Artificial Life 2: 417-434 (1995)..

Rechenberg (1973) Rechenberg, I. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog, Stuttgart (1973).

Schwefel (1981) Schwefel, H. P. Numerical Optimization of Computer Models. New York: John Wiley & Sons (1981).

Goldberg (1989) Goldberg D., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, (1989).

Rechenberg (1989) Rechenberg I., Evolution strategy: Nature's Way of Optimization. In H. W. Bergmann, editor, "Optimization: Methods and Applications, Possibilities and Limitations", Lecture Notes in Engineering, pag 106-26, Springer, Bonn (1989).

Mondada (1993) Mondada F. and Franzi P.I. "Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms". Proceedings of the Second International Conference on Fuzzy Systems. San Francisco, USA, (1993).

Sommaruga (1996) Sommaruga L., Merino I., Matellán V and Molina J. "A Distributed Simulator for Intelligent Autonomous Robots", Fourth International Symposium on Intelligent Robotic Systems-SIRS96, Lisboa (Portugal), (1996).

Braitenberg (1984) Braitenberg V. Vehicles: experiments on synthetic psychology. MIT Press Cambridge, Massachusets (1984).