A General Learning Co-Evolution Method to Generalize Autonomous Robot Navigation Behavior

A. Berlanga
SCA-LAB,
Universidad Carlos III,
Avda Universidad, 30,
28911, Madrid, SPAIN
aberlan@ia.uc3m.es

A. Sanchis
SCA-LAB,
Universidad Carlos III,
Avda Universidad, 30,
28911, Madrid, SPAIN
masm@inf.uc3m.es

P. Isasi
SCA-LAB,
Universidad Carlos III,
Avda Universidad, 30,
28911, Madrid, SPAIN
isasi@ia.uc3m.es

J.M. Molina
SCA-LAB,
Universidad Carlos III,
Avda Universidad, 30,
28911, Madrid, SPAIN
molina@ia.uc3m.es

Abstract- In this paper a new coevolutive method, called Uniform Coevolution, is introduced, to learn weights of a neural network controller in autonomous robots. An evolutionary strategy is used to learn high-performance reactive behavior for navigation and collisions avoidance. The coevolutive method allows evolving the environment, to learn a general behavior able to solve the problem in different environments. Using a traditional evolutionary strategy method, without coevolution, the learning process obtains a specialized behavior. All the behaviors obtained, with/without coevolution have been tested in a set of environments and the capability of generalization is shown for each learned behavior. A simulator based on mini-robot Khepera has been used to learn each behavior. The results show that Uniform Coevolution obtains better generalized solutions to examples-based problems.

1 Introduction

A fundamental requirement for autonomous mobile robots is navigation. This task gets the robot from place to place with safety and no damage. Approaches based on the classical paradigms (abstraction, planning, heuristic search, etc.) were not completely suitable for unpredictable and dynamic environments. Other approaches consider reaction as the new paradigm to built intelligent systems. One classical instance of this kind of architecture is the subsumption architecture which was proposed by Brooks (Brooks 1991) and has been successfully implemented on several robots of MIT and other institutes. The base of the subsumption architecture is "behavior". Each behavior reacts in a situation and the global control is a composition of behaviors. Different systems, from finite state machines to fuzzy controllers (Ishikawa 1995), have been used for the implementation of these behaviors. 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.

Machine learning has been applied to shape the behavior of autonomous agents. Some of these techniques become inapplicable to the learning reactive behavior problem because they require more information than the problem constraints allow. Thus, it would seem reasonable to use an automatic system that gradually builds up a control system of an autonomous agent by exploiting the changing interactions between the environment and the agent itself. Some approaches use Genetic Algorithms to evolve fuzzy controllers (Matellán *et all* 1998), Classifier Systems to learn controllers (Molina 1998 and Sanchis 1999) or Neural Networks to learn behaviors (Mondada 1993).

The control architecture used to evolve the reaction (adaptation) is based on a neural network. The neural network controller has several advantages (Miglino 1995):

- Neural Networks, NN, are resistant to noise, those exist in real environment, and are able to generalize their ability in new situations.
- The primitives manipulated by the Evolutionary Strategy, ES, are at the lowest level in order to avoid undesirable choices made by the human designer.
- A NN could easily exploit several ways of learning during its lifetime.

The used of a feed forward network with eight input units and two output units directly connected to motors appears in previous works (Miglino 1995) as an efficient way to learn a behavior: "avoid obstacles" using Genetic Algorithms. 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. 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. From the initial situation the robot is able to learn through experience the optimal associations between inputs and outputs.

When a system is learning from examples, for testing a solution is necessary to face it with different situations (examples set). More over, in many cases, the solutions should not fit a particular examples set, they have to be generalized solutions, useful over any possible example. This problem becomes harder when using evolutionary methods. An excessive adaptation to the examples set could abort the generalization capability of a solution. The evolution of the examples tries only to generate harder examples for the solutions.

Coevolution refers to the simultaneous evaluation of several species, where the survival of each specie depends on one each other. When talking about coevolution in computational terms, this is referring to the ability of a system to improve its performance by means of mutual adaptation of its different constituents. The final performance of the system is improved as a consequence of the incremental adaptation among constituents. These ideas of coevolution were first explored in evolutionary computation in some works in the Iterated Prisioner's Dilemma (Axelrod 1984, Axelrod 1989 and Lindergren 1994). One of the first authors in applying the coevolution in an optimisation problem was Hillis with his work over the coevolution of parasites for improving solutions in a sorting network problem (Hillis 1991). More recently, some works for establishing the theoretical basis in coevolution have been done (Paredis 1996 and Rosin 1997). All these previous works have proven the usefulness of coevolution to improve the evolutionary computation techniques from different perspectives.

In this paper, a new method, based on Hillis ideas to use coevolution ideas to learn generalized autonomous robot navigation behaviours. Section 2 is related to the experimental environment and the goals of the work. In section 3, we outline the general theory of coevolutive method. The experimental results are shown in Section 4. The last section contains some concluding remarks.

2 Robot Controller and Experimental Environment

The task faced by the autonomous robot is to reach a goal in a complex environment while avoiding obstacles found in its path. Different environments have been used to find the connections of the NN.

2.1 Evolving Controllers by means of Evolutionary Strategies

It has been proven 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, see Figure 1. The input sensors considered in this approach are the ambient and proximity sensors, s_i, of Figure 2. The NN outputs are the wheel velocities.

The NN architecture is shown in Figure 1.

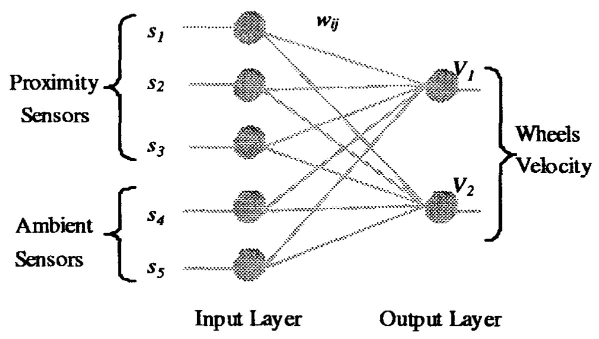
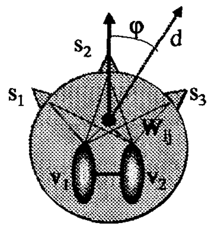


Figure 1: Neural Network Architecture.



s_i: Input of i-sensor
v_j: Velocity of j-wheel
d: Goal distance
φ: Goal angle
W_{ij}: Weight between i-sensor and j-wheel

Figure 2: Connections between sensors and actuators in the Braitenberg representation of a Khepera robot.

The velocity of each wheel is calculated by means of a linear combination of the sensor values, using those weights (Figures 1 and 2):

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

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.

Weight values depend on problem features. To find them automatically, an ES is proposed (Berlanga 1999b). 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 1. 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.

Evolution strategies (ES) developed by Rechenberg (Rechenberg 1973, 1989) and Schwefel (Schwefel 1981), have been traditionally used for optimization problems with real-valued vector representations. As Genetic Algorithms, GA, (Goldberg 1989) 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.

2.2 Environment

In this work, a simulator based on an autonomous robot named Khepera (Mondada 1993) is used. The mini-robot Khepera 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 sensory inputs come in from eight infra-red proximity sensors. These sensors are composed of two devices: an IR emitter and a receiver. The emitter and the receiver are independent, then it is possible to use the receiver to measure the reflected light (with the emitter active) or to measure the environmental light (without emission). The reflected light measurement can give some information about the obstacles. In fact, this measure is not only a function of the distance to an object in front of the emitter but also the environmental light and the object nature (color and texture). So the value of distance is modified by the measure of the ambient light and the object nature, the light used is constant and all the obstacles used have the same color and texture. The robot has two wheels controlled by two independent DC motors with incremental encoder that allow any type of movement. Each wheel velocity could be read by a speedometer.

Using the ambient sensors it is possible to measure the distance and the angle to a light source. The distribution of the amount of light coming into the eight sensors is used to evaluate the distance and the angle to the source, see figure 3. The amount of light received in the sensor depends on the distance of the light source. The response curve of each real sensor is described by a sigmoidal function (Mondada1993).

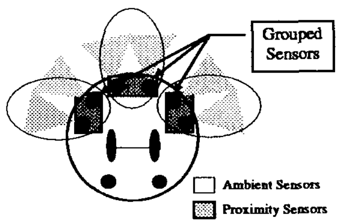


Figure 3: Sensors considered in the real robot.

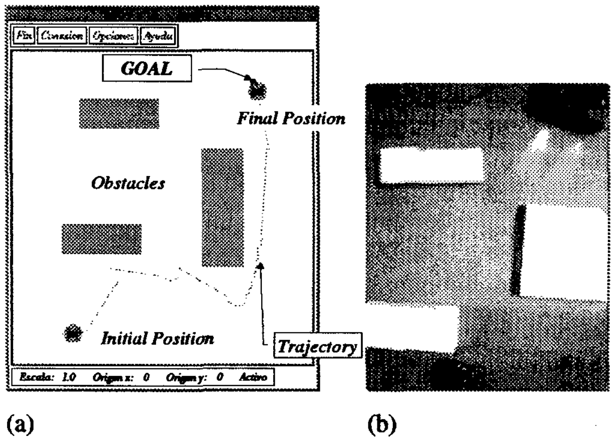


Figure 4: (a) SimDAI Simulator (Example of one simulated environment). (b) Example of a real experimental environment.

Experiments take a long time of continuous functioning of the hardware. In order to prove the different configurations of the controllers, a simulator developed in a previous work (Sommaruga 1996) has been used, the SimDAI one. In the simulator, the characteristics of the turtle robot model (McKerrow 1991) and the physical restrictions of the Khepera robot have been considered. SimDAI is a working prototype of a mobile robot simulation environment for experimenting with robot navigation and control algorithms. Each mobile robot is completely independent, can navigate and interacts with other robots in a 2-D simulated world of obstacles, which is separately monitored. This simulator has been used in many other works (Isasi 1997, Matellán 1998, Molina 1998, Sanchis 1999). The simulation 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. 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 4 (a) and (b)).

3 Uniform Coevolution

The uniform coevolution method, CoevU, is based on that proposed and developed by Hillis, (Hillis 1991). In coevolutive dynamics not only the population of solutions but also the training environment evolve. In this case, they compete in a called "Arms Race" dynamic. The environment is formed by codified evaluation examples, on which genetic operators will also be applied.

The CoevU faces two systems, the solutions system and the seeds blocks system. The solutions system is constituted by the population of neural nets. Each solution is associated to a set of seeds, called "block seeds" (BS), of size *m*. Applying the incremental mutation operator (Berlanga 1999a), each seed gives rise to a set of environment examples of size *n*. In this way a controller is evaluated in *m x n* environment examples. The definition of the autonomous navigation problem is the robot starting position which is defined by *x,y* coordinates and direction of the robot movement.

An evolutionary strategy will be applied on the solutions set, see figure 5. The evolution of the seeds block system is performed by the uniform crossover and mutation as it is used in canonic genetic algorithms (Goldberd 1989).

The sequence to calculate the global fitness value of a member of the solution system, f_{sol}, is the following:

- Calculate the specific fitness value, f_j^i, of a controller on its associated environments set.
- Apply equations 2, 3, 4, 5, and 6 to obtain the fitness value of the seeds block, Bf^i.
- Finally, equation 7 calculates the global fitness value f_{sol} used by the GA reproduction operator.

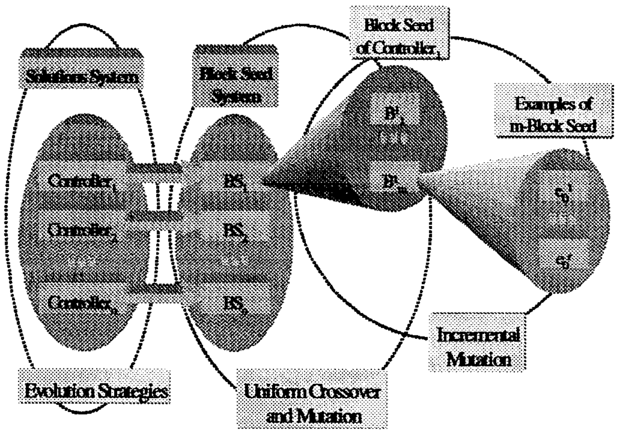


Figure 5: Evolutive process of Uniform Coevolution.

$$a^i = 1 - \frac{\sum_{j=1}^{N_{examples}} f_j^i}{N_{examples} F_{MAX}} \qquad (2)$$

Where F_{max} is the worst fitness value that can be reach in the evalutation of a solution over an environment.

$$\begin{cases} f_{max} = f_{min} \;, & \beta_j^i = 0 \\ f_{max} \neq f_{min} \;, & \beta_j^i = \dfrac{f_j^i - f_{min}}{f_{max} - f_{min}} \end{cases} \qquad (3)$$

Where f_{max} is the worst fitness value reached in an example of a seed block for a generation, and f_{min} the best.

$$w_j^i = \frac{\left(e^{a^i} - 1\right)\left(e^{\beta_j^i} - 1\right) + \left(e^{1-a^i} - 1\right)\left(e^{1-\beta_j^i} - 1\right)}{(e-1)^2} \qquad (4)$$

Figure 6 shows the landscape of the function corresponding to parameter *w*.

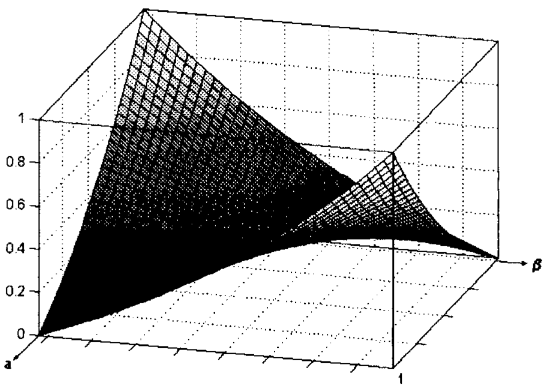


Figure 6: Graph of the shape of selection pressure control, w_j^i.

$$\alpha_j^i = \frac{w_j^i}{\sum_{k=1}^{N_{examples}} w_k^i} \qquad (5)$$

$$Bf^i = \sum_{j=1}^{N_{examples}} \alpha_j^i f_j^i \qquad (6)$$

$$f_{SOL} = \sum_{i=1}^{N_{blocks}} \frac{Bf^i}{N_{blocks}} - K\sigma_{fB} \qquad (7)$$

were K is a constant experimentally obtained and σ_{fB} is the standard deviation of the *Bf* values.

Factors ω and α in equations 4 and 5 smooth the fitness landscape. These factors are called "selection pressure control" (Berlanga 1999a). At the beginning of the evolution, see equation 8 and Figure 7, good behaviors within a block contribute with a higher weight in the fitness value of the seed block.

$$f_j^i \approx F_{MAX}$$
$$a \approx 0 \qquad (8)$$
$$w \approx \frac{e^{1-\beta}-1}{e-1}, \quad 0 \le \beta \le 1$$

As the process carries on, bad behavior weight increases, until equilibrium is reached. At the end of the evolution, to calculate the fitness value of seeds block, bad behaviors have the highest weight, as it is shown in equation (9) and Figure 7.

$$f_j^i \approx 0$$
$$a \approx 1 \qquad (9)$$
$$w \approx \frac{e^{\beta}-1}{e-1}, \quad 0 \le \beta \le 1$$

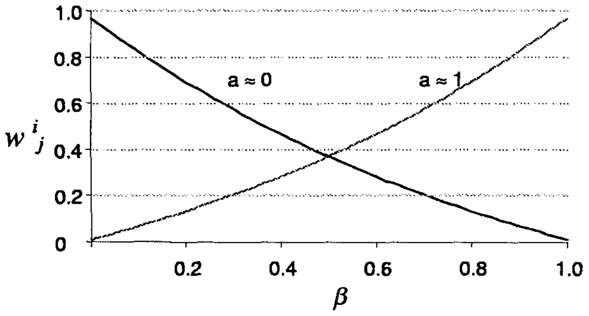


Figure 7: Values of the w_j^i in different evolutive instants.

In order to generate training examples from a seed, a new operator, called incremental mutation operator is introduced.

Thus, the first example is generated from the seed, the second example is generated from the first one and so on. The incremental mutation operator establishes a distance among different training examples, see equation 8. Learning process starts using similar examples, in order to avoid the disorientation-searching problem. As it proceeds, the difficulty of the example increases, that is, the examples are more and more different.

$$\mathrm{MI} = \frac{b^2 (e^{\frac{2Bf}{F_{max}} \ln\left(\frac{c-b}{b}\right)} - 1)}{c - 2b} \qquad (10)$$

Figure 8 shows the incremental mutation function of equation 10. Parameter c determines the maximum distance between two examples when the fitness value is the optimum.

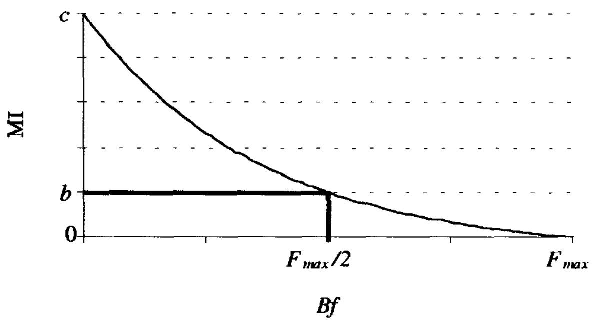


Figure 8: Graph of the mutation distance.

4 Experimental Results

Two different kinds of experiments have been performed. In both cases, an Evolutionary Strategy is used, (μ+λ)-ES, μ=6, λ=4, in order to find the network connections weights. Experiments differ in the way they are evaluated on the learning environments. One of the experiments, which will be referred as *fixed*, is trained in the same environment during all the evolutive process; that is, starting and goal positions, as well as the obstacle configuration are constant. On the other hand, those experiments that use the uniform coevolution algorithm, *coevU*, evolve the robot starting position and orientation, while they keep the goal position and obstacles configuration fixed.

4.1 Measure of the controllers fitness

To obtain a controller fitness value, the simulation has been run for a period of 2000 cycles. Simultaneously, a log of its behavior is recorded. The measures that will be taken into account to calculate the fitness value are the following:

- Number of cycles necessary to reach the goal, T. If the goal is not reached the value is 2000.
- Length of the robot's trajectory, L.
- Number of collisions, C.
- Number of cycles in which the robot stayed in the same position, S.
- Euclidean distance between the robot's starting and final position, D_o.
- Euclidean distance between the robot's starting position and the goal position, D_m.

Equation 11 shows the lineal combination and weights used to obtain the fitness value of a controller, obtained from the measurements of its behavior.

$$f_j^i = 20T - 1.5L + 10C + 10S + 10D_m - 1.5D_o \qquad (11)$$

The direction of the learning process is to minimize the fitness value. For the *fixed* experiment the fitness function is the base measurement used to apply the selection operator. For the *coevU* experiment, this is the value applied in equations 4, 5, and 8 to calculate the block fitness value. In these experiments, constant K in equation 7 has an experimental value of 0.25.

4.2 Generalization capabilities of fixed experiments

Figure 9 shows the training environments. Five runs with 200 generations in each one were performed over each environment.

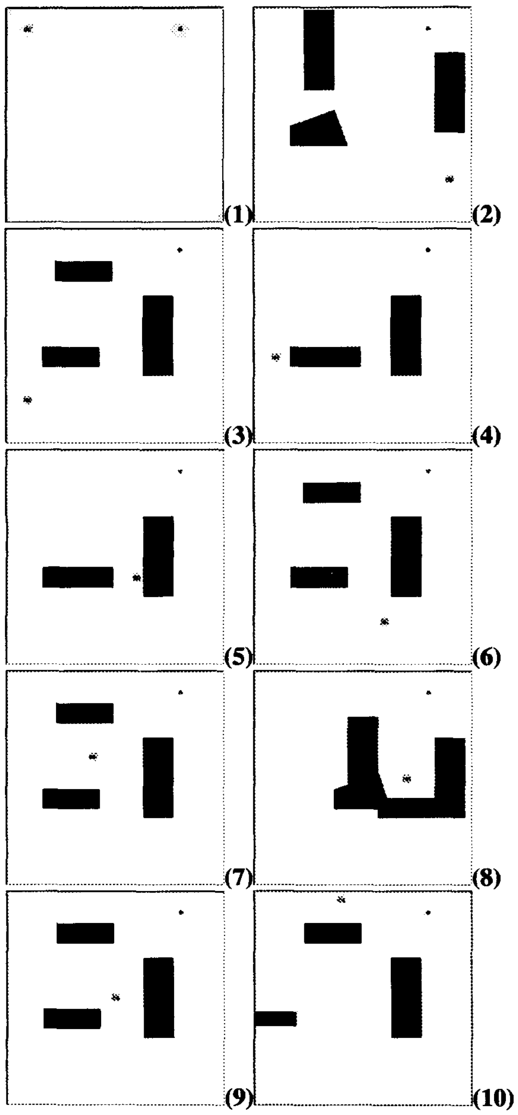


Figure 9: Evaluation environments.

The obtained validation values for the fixed experiments are shown in Figure 10. Each graph represents the fitness value obtained by the best controller that has been trained in an environment and evaluated in the rest. The first graph corresponds to the fitness value obtained in environment 1, the second graph to that obtained in environment 2, etc. The shadow column represents the fitness of the best controller in the environment in which the controller has been evolved.

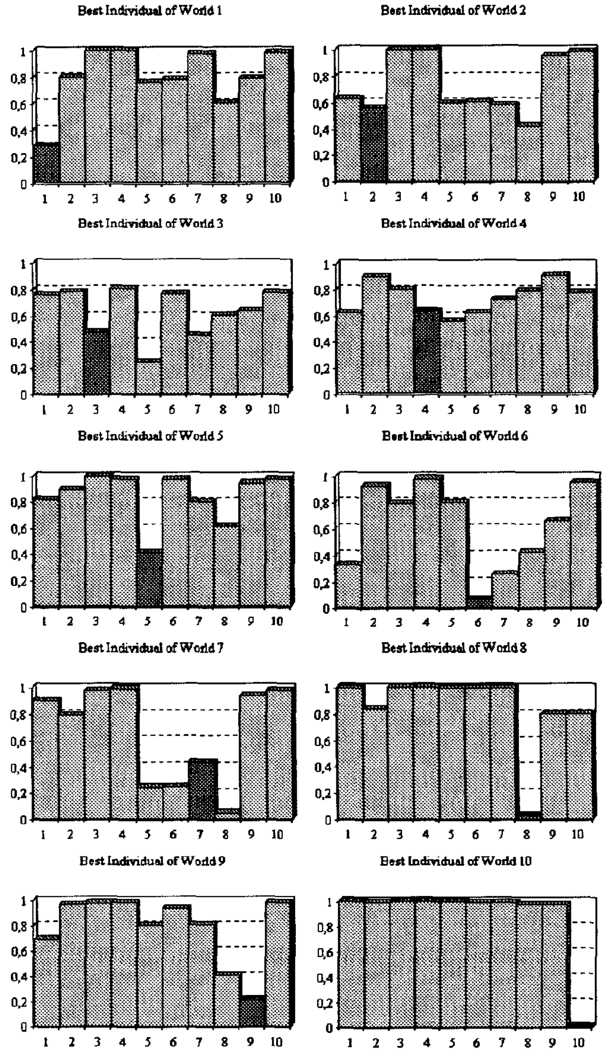


Figure 10: Fitness of the best solution obtained in a world and tested in all the other worlds.

These graphs (Figure 10) show that the obtained controllers are not general, since they are very adapted to the training environment. Darker bars correspond to the value obtained in the training environment and in most cases that is the lowest value. It is also shown, that depending on the training environment, solutions are more or less general. Thus, the controller trained in environment 3 seems to be the most general, while that obtained in environment 10 is the lowest.

The same information is shown in a different way in Figure 11. The graphs represent the behavior in a specific environment of the best solution obtained for each of the evaluated environments. In environment 1 for the first graph, in environment 2 for the second graph and so on.

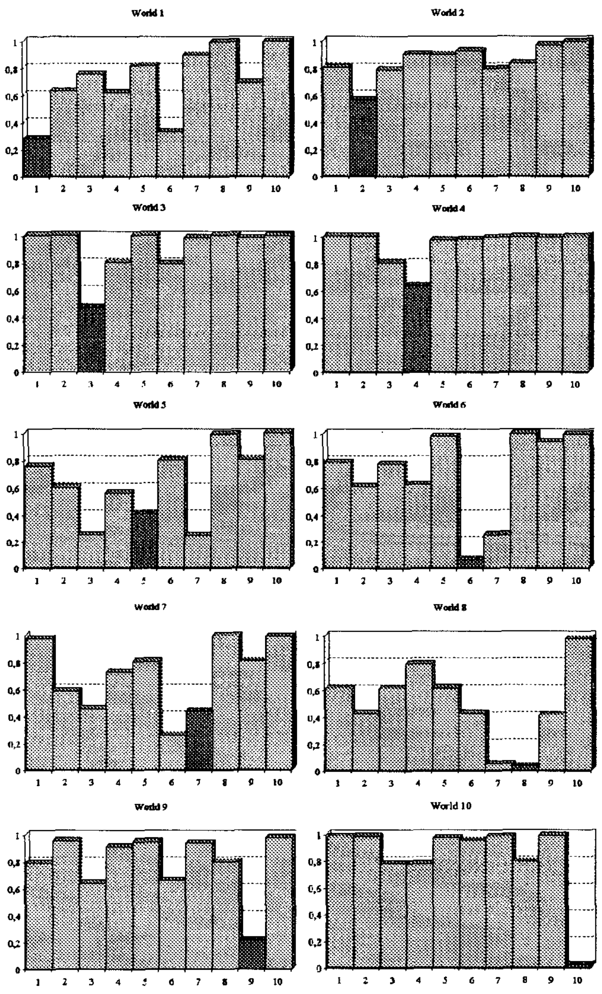


Figure 11: Fitness of the set of best solutions (one of each world) in each world.

Graphs in Figure 11 show the capability of environments to obtain general solutions. For example, environment 8 gives more general solutions in contrast to environment 10.

4.3 Uniform Coevolution results

The fixed type experiments have two main problems: the overadaptation problem and the quality of the solutions that depends on the training examples set. Thus, the necessity of an evolutive algorithm to improve the existing one is justified.

Coevolutive experiments have not been performed in all the environments since some of these only differ in the starting position, thus for example 3, 6, 7 and 9 in figure 8 are the same environment.

The fitness values of $fixed_1$, $fixed_3$ and $CoevU_1$, $CoevU_3$ are related with the evolution in worlds 1 and 3. These two worlds are the most general ones from Figure 9. In table 1, the validation of the obtained controllers is shown. These controllers have been learned in worlds 1 and 3 (of Figure 9) and tested in worlds 1, 3, 5 and 10. For comparing, the obtained values corresponding to fixed experiments (section 4.2) in worlds 1 and 3 have also been included.

Table 1: Resume of the fitness value obtained in fixed experiments and CoevU experiments.

	W_1	W_3	W_5	W_{10}
$Fixed_1$	67 ± 4	90 ± 6	87 ± 6	79 ± 6
$Fixed_3$	73 ± 5	96 ± 3	95 ± 3	91 ± 4
$CoevU_1$	2 ± 1	61 ± 10	67 ± 9	32 ± 11
$CoevU_3$	6 ± 2	22 ± 4	22 ± 7	6 ± 3

The validation process has been carried out making 1000 executions over worlds 1, 3, 5 and 10. Each execution has different initial position and orientation of the robot, randomly generated.

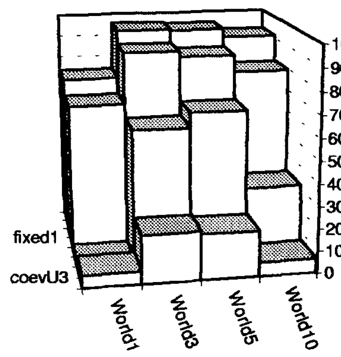


Figure 12: Representation of the fitness values compiled in table 1.

Both Table 1 and Figure 12 compiles the mean of the fitness value with a 95% confidence interval, over the 1000 running. The controller denoted through *CoevU1* shows the best generalization results, better that any one of the fixed controllers. Moreover, it also can be seen that *CoevU1* has a very specialized behavior in the world 1, comparing with the results of *CoevU3*. This last controller shows better results in the four worlds considered. *CoevU3* improves the fixed controllers in about an 80% and about a 20% over *CoevU1*.

5 Conclusions and further work

The experiments prove the possibility of learning general behaviors in an autonomous robot by means of an ES. The uniform coevolution method has shown to be able to improve the generalization of solution in about a 70%. The process has been applied on a simple NN where the direct associations between sensors and motors allows to solve a navigation problem.

It can be also extended to other more complex NN and to use another evolutionary algorithm.

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.

Uniform Coevolution has proven its capability to obtain better generalized solution in problems examples-based.

6 Bibliography

Axelrod, R. (1984) "The Evolution of Cooperation", Basic Books, New York.

Axelrod,R. (1989) "Evolution of Strategies in the Iterated Prisioner's Dilemma". Davies ed. of Genetics Algorithms and Simulated Annealing, 32-41, Morgan-Kaufman.

Berlanga A., Isasi P., Sanchis A., Molina J. M. (1999a) "Distance Modulation Competitive Coevolution method to find initial configuration independent cellular automata rules" in IEEE International Conference on Systems, Man and Cybernetics. 607-612, Japan.

Berlanga, A., Sanchis, A., Isasi, P., Molina, J.M. (1999b) "Neural Networks Robot Controller Trained with Evolution Strategies", Proc. of 1999 Congress on Evolutionary Computation, CEC99.

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

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

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

Hillis, W.D. (1991) "Co-evolving parasites improve simulated evolution as an optimization procedure". In C.G. Langton, editor, Artificial Life II, 313-324, Reading, MA, Santa Fe Institute, Addison Wesley.

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

Isasi P., Berlanga A., Molina J. M., Sanchis A., (1997) "Robot Controller against Environment, a Competitive Evolution", Special Session on Evolution Computation, 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics, (Germany).

Lindergren, K., Nordahl, M.G. (1994) "Artificial Food Webs", Artificial Life III, 73-103, Addison-Wesley.

Matellán V., Fernández C., Molina J.M. (1998) "Genetic Learning of Fuzzy Reactive Controllers", Robotics and Autonomous Systems, 25, (1-2), 33-41.

McKerrow P.J. (1991) "Introduction to robotics", Addison-Wesley Publishing Company Inc.

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

Molina, J.M., Sanchis, A., Berlanga, A. And Isasi, P. (1998) "An Enhanced Classifier System for Autonomous Robot Navigation in Dynamic Environments". Intelligent Automation and Soft Computing. Autosoft Press, in Press.

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

Paredis, J. (1996) "Coevolutionary Computation", Artificial Life, 2, 355-375, Addison-Wesley.

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

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

Rosin, C.D., Belew, R.K. (1997), "New Methods for Competitive Coevolution", Evolutionary Computation, 5,1-29.

Sanchis, A., Molina, J.M., Isasi, P. And Segovia, J. (1999) "RTCS: a Reactive with Tags Classifier System", Journal of Intelligent and Robotic Systems, in press.

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

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