ELSEVIER

Robotics and Autonomous Systems 25 (1998) 33–41

Robotics and Autonomous Systems

Genetic learning of fuzzy reactive controllers

Vicente Matellán *, Camino Fernández [1], José M. Molina [2]

Departamento de Informática. Universidad Carlos III de Madrid, Avd. Butarque 15, Leganés (Madrid), Spain

Received 17 April 1997; received in revised form 17 November 1997; accepted 2 March 1998

Abstract

This paper concerns the learning of basic behaviors in an autonomous robot. It presents a method to adapt basic reactive behaviors using a genetic algorithm. Behaviors are implemented as fuzzy controllers and the genetic algorithm is used to evolve their rules. These rules will be formulated in a fuzzy way using prefixed linguistic labels. In order to test the rules obtained in each generation of the genetic evolution process, a real robot has been used. Numerical results from the evolution rate of the different experiments, as well as an example of the fuzzy rules obtained, are presented and discussed.

Keywords: Autonomous robots; Fuzzy; Genetic algorithms; Learning

1. Introduction

Artificial Life has been defined as "a scientific discipline that studies how behavior of agents emerges and becomes intelligent and adaptative" [13]. Many experiments have been made in this sense using neural networks, classifier systems, etc., showing how these behaviors can emerge. This paper presents the evolution of the behavior of a robot using a different paradigm: fuzzy logic. In this way, the evolution of the control rules for an autonomous robot using a genetic algorithm is presented.

We have chosen fuzzy controllers as a symbolic representation for many reasons. First, because fuzzy rules and linguistic labels are close to the human way of expressing behavior rules. This means that we can easily evaluate the rules that have been obtained in a genetic way. Second, fuzzy controllers are very flexible, which makes them easily adaptative. Third, fuzzy sets theory is a well-suited paradigm that has shown its effectiveness in many autonomous systems. And finally, fuzzy theories are open to be shared among different agents [8], which is quite interesting in our further work.

Fuzzy controllers have two main parts. The first one is made up by the linguistic labels, and the second one by the fuzzy rules. Linguistic labels can be viewed as low level ideas or symbolic concepts. These concepts can be interpreted in different ways. For instance, the concept *near* will not have the same physical meaning for a one meter diameter robot as for a 5 cm one. Fuzzy rules express the relations among these concepts. In this paper we present a method for adapting these rules using genetic programming.

In our experiments, the robot starts up with no information about the right rules to move around. From this situation, the genetic method is able to reach a set of rules that represents the highest adaptability grade

* Corresponding author. E-mail: vmo@ia.uc3m.es

[1] E-mail: camino@inf.uc3m.es

[2] E-mail: molina@ia.uc3m.es

0921-8890/98/$ – see front matter
PII: S0921-8890(98)00035-9

to the information provided by sensors. The only given and fixed data are: the number of inputs (number of robot sensors), the partitions of the input domain (the range of the sensors), the number of outputs (number of robot motors) and the motors range, but it has no information about how do they relate to each other. This knowledge is obtained by the evolution process.

The rest of this paper is organized as follows. Section 2 is concerned with the fuzzy description of the problem and the particularities of evolving a fuzzy controller. Section 3 deals with the description of our experiments. Finally, in Section 4 the result of these experiments is discussed and our future work is presented.

2. Problem description

Fuzzy controllers have been widely used to control different kinds of autonomous robots in order to accomplish different tasks. For instance, in [11] the LIFIA architecture is presented. This is a hierarchy of layers, each one working asynchronously with its own level of data abstraction, where the navigation module consists of a reactive module based on fuzzy logic. Other example of an autonomous system controlled using fuzzy behaviors is given in [10]. This system consists of a car which navigates autonomously using a group of fuzzy rules sets. Another classical one is the architecture of the Flakey robot [12]. This is an approach for integrating planning and control based on "behavior schemas", which link the physical movements to abstract action descriptions by using the operations of multivalued logics, where goals and controllers can be combined to produce conjoint goals and complex controls.

In order to present the bases of these systems and the learning mechanism used to adapt the fuzzy control rules, the basic concepts of the fuzzy logic controllers are introduced in the next section. In Section 2.2 the strategy of design is presented and in Section 3 the adaptation process is described.

2.1. Fuzzy controller

The first step in the design of a fuzzy controller should be to select adequate descriptions of the relevant inputs for the control, such as the *distance* to obstacles, similar to those formulated by humans when they describe the perceived features. So, given the numerical distance d_i to an obstacle perceived by a sensor, D_i is defined as the range of all possible values of the computed distance d_i. To better cope with the intrinsic uncertainty that underlies the appearance of perceptual inputs (distorted after the acquisition process), the numerical values of the distances d_i can be mapped into qualitative symbolic labels through a fuzzification process [15], transforming the computed distances into linguistic variables.

A linguistic variable [15] is a variable whose values are sentences in a natural or artificial language, that is, a concatenation of atomic terms: labels (adjectives), hedges (modifiers such as very, much, slightly, etc.), negation and makers (parentheses). The meaning of a linguistic variable is defined as the fuzzy subset for which the value of the linguistic variable serves as a label. A fuzzy subset A of a universe of discourse U is characterized by a membership function $\mu : U \rightarrow [0, 1]$ which associates with each element y of U a number $\mu_A(y)$ which represents the degree of membership of y in A. The operation of fuzzification (application dependent) has the effect of transforming a nonfuzzy quantity into a fuzzy variable. The value of, for instance, a linguistic variable *distance* (a natural label such as *near*) represents a much less precise meaning than the numerical value of the inches to the obstacle.

Using these concepts, for each distance sensor d_i, a linguistic variable Ld_i is introduced together with a set of values $ld_{i1}, ld_{i2}, \ldots, ld_{im_i}$, whose cardinality is m_i. Each ld_{ij} in the set labels a fuzzy subset in the universe of discourse D_i (that is the range of possible values returned by the sensor) with membership function $\mu_{ldij}(di)$. Values of the membership function of a label are related to the difficulty of attributing this label to a numerical value d_i obtained from the sensors of the robot. The fuzzification operation adopted, affecting the numerical values d_i, will result in their transformation into a fuzzy singleton [15] or fuzzy subset whose support is a single point in D_i, with membership function equal to one.

After that, a fuzzy relational algorithm [15] (FRA) is used to store the knowledge required to control the autonomous robot through a fuzzy reasoning process, based on the linguistic labels of the inputs. The FRA is composed of a finite set of fuzzy conditional statements. Where the antecedent are conjunctions

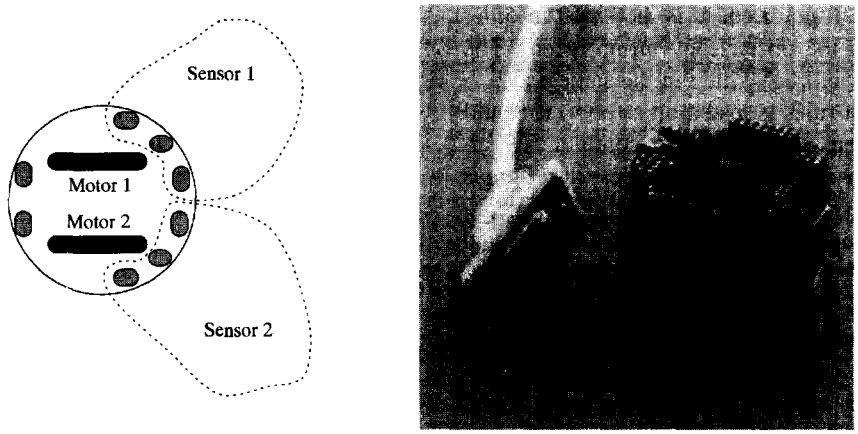


Fig. 1. Robot configuration.

about the linguistic variables Ld_i (linguistic variable defined over the distance measured by sensor d_i). Their consequents are also fuzzy conjunctions about Lm_k (linguistic variable defined over the speed that will be applied to motor m_k), whose possible values are $lm_{k1}, lm_{k2}, \ldots, lm_{kn}$. So, a rule of the fuzzy controller may be

IF Ld_1 **IS** ld_{11} **AND** Ld_2 **IS** ld_{23} **AND** ...
AND Ld_i **IS** ld_{ij} **THEN** Lm_1 **IS** lm_{14}
AND ... **AND** Lm_k **IS** lm_{kl}

An extension of the *modus ponens* rule is used as the inference mechanism to obtain the fuzzy subset induced in each Lm_k by the fuzzy conditional statements. As there can be several conditional statements forming the FRA, the meaning of each Lm_k is calculated as the intersection of the intermediate meanings resulting from each application of the CRI. The fuzzy subset of Lm_k is obtained after *max-min* product [16] of discretized versions of $\mu_{ld_{ir}}(d_i)$.

Finally, the adopted defuzzification process on Lm_k is a version of the center of gravity procedure. This method treats the rules separately. Each rule produces a level of activation, λ_{kl}, in the output labels lm_{kl}. Let $C_{lm_{kl}}$ be the numerical representative of each label, lm_{kl} (calculated using the method of the gravity center). Then, the output is taken as a type of weighted sum:

$$sum = \left(\sum \lambda_{kl} C_{lm_{kl}}\right) \Big/ \left(\sum \lambda_{kl}\right).$$

2.2. Designing an FRA

The design of the rules to control an autonomous system using an FRA is not a trivial issue. Let us consider a system, similar to the robot we have used, with eight input signals (sensors) and two outputs (motors), codifying each one of the sensors inputs with only two linguistic labels (*near, far*), and each one of the outputs with five labels (*very-forward, forward, stop, back, fast-back*); the number of possible fuzzy rules is 6400. But if we use a high level of granularity in our system, for instance, if every variable is codified with five linguistic labels, the number of rules will be 9 765 625. Using a typical eight linguistic labels for variable, the number of rules would be in the range of billions (1 073 741 824).

Of course, only a few rules are needed (typically less than a hundred) to get a sophisticated behavior. The problem is how to choose these rules. Until now, the most commonly used method has been asking human experts the rules. But humans usually think in an antropomorphical way, which can cause some problems.

Let us consider a simple autonomous robot such as the one in Fig. 1. This robot has only got two proximity sensors (*sensor*1 and *sensor*2), and two motors (*Motor*1 and *Motor*2). When a human is asked to write some rules to let the robot wander through the world without crashing, the obtained rules would be something like **IF** *sensor*1 **IS** *near* **THEN** *Motor*1 **IS** *fast*. This means that when an obstacle is perceived by the

left sensor, the left motor speed is increased in order to go away from the obstacle by turning right. The symmetrical rule will be written in the same way.

If this group of rules is tested on a robot, it would be proved that the robot begins to continuously increase its speed trying to avoid obstacles till it bumps into an obstacle. The problem can be defined as a continuous increase of entropy. It can be easily corrected by writing the rules in an opposite way – **IF** *sensor*1 **IS** *near* **THEN** *M*2 **IS** *slow* – and adding a new rule to increase the speed when there are no obstacles – **IF** (*sensor*1 **IS** *far* **AND** *sensor*2 **IS** *far*) **THEN** (*M*1 **IS** *forward* **AND** *M*1 **IS** *forward*). We want to test whether this problem would appear if the rules were obtained using genetic evolution.

Another problem with human rules is that they are designed in a theoretical world: the human mind. This means that the rules have to be tuned many times till they reach an acceptable performance. This is caused by the differences among sensors due to the differences in the manufacturing process. So, another factor to consider in the evaluation of our method will be its tolerance to hardware constrains. The use of a fuzzy controller to evaluate rules provides a softer decline in the performance of the system by reducing the dependence between a concept and its representation.

3. Adaptation of fuzzy behaviors

Different methods can be found in the literature about learning of fuzzy controllers. Most of them are based on the use of neural networks [3], but some use also genetic methods to learn plans for autonomous robots. For example, Ref. [1] presents a genetic method to formulate new sets of low-level decision rules for robot movements and pushing techniques. Each rule checks if certain conditions are true (obstacles detected, goal position, etc.), and it executes a number of corresponding operators (move forward, move backward, turn left or turn right). The genetic competition occurs among sets of behavior rules (plan) after testing the plans. The main problem in this approach is that it has only been tested on a simulator.

Some works have also been made in the genetic evolution of fuzzy rules. For instance, in [14] a genetic algorithm for the design of the fuzzy rules is presented to center a cart by applying a single force. Another example appears in [6] where the application of this method to three different physical systems is presented: a liquid-level system, a pH system and a satellite-rendezvous system. In each of these applications, the genetically designed fuzzy controllers outperforms the human designed ones. Fewer experiments have been made in the evolution of fuzzy linguistic labels. One of them is the genetic method presented in [7], which determines, in a TSK fuzzy model (where the output variables are computed as linear combinations of the input values), the membership functions, the number of fuzzy rules and the rule consequent parameters. This system is applied to the classical inverted pendulum control problem.

Finally, some works have also been proposed in the evolution of fuzzy controllers applied to the control of autonomous vehicles. For instance, in [2] a general method for the evolution of rule-based fuzzy controllers is presented. Another methodology based in a hierarchical prioritized structure using a messy genetic algorithm is applied in [5] to the control of an autonomous vehicle.

In the rest of this section we will briefly describe our approach, starting by the robot used in our experiments, and then we will discuss our work: genetic evolution of a fuzzy controller for the control of an autonomous robot.

3.1. The mini-robot Khepera

The robot that has been used is the mini-robot Khepera [9], which is a commercial mini-robot developed at LAMI (EPFL, Lausanne, Switzerland). This robot has a circular shape with a diameter of 5.5 cm, a height of 3 cm and a weight of 70 g. It moves using two wheels and two small Teflon balls. The wheels are controlled by two motors that let the former move in both directions. The robot has also eight infra-red proximity sensors.

The heart of the robot is the Motorola 68331 controller with 256 Kbytes of RAM and 512 Kbytes of ROM which manages all the input–output routines and can communicate via serial port with a host computer. It also has its own batteries which let it work autonomously. It is also possible for it to work attached to a workstation via the serial port. This lets the robot use workstation resources (such as the hard disk to store data).

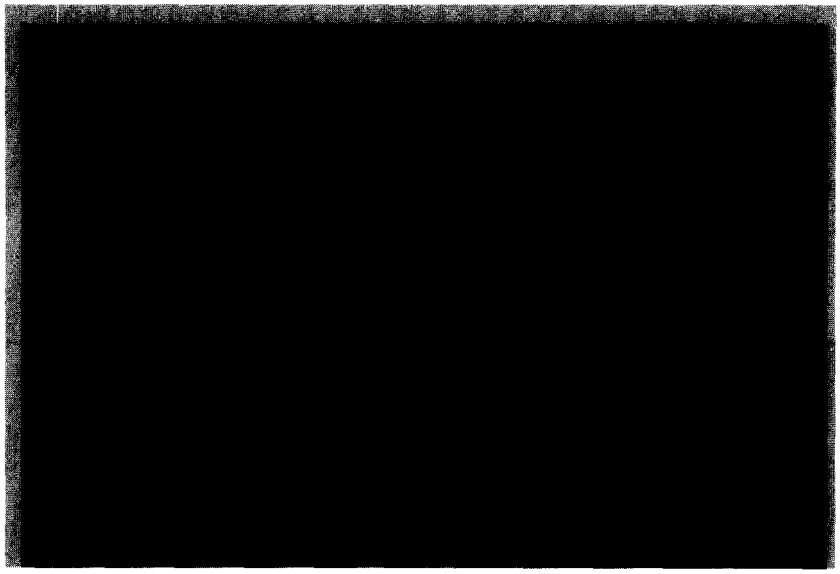


Fig. 2. Experimental environment.

We have preferred to use a real robot instead of a simulation for two reasons. First, because a perfect simulation of a simple robot as Khepera requires hard computations. For instance, the simulation of the sensors, taking care of the lab lightening, orientation of robot, etc; would require huge calculations. And second, even a very good simulation is unable to consider all the physical laws of a real robot such as inertia, friction, failures of the hardware, etc.

3.2. *Adaptation of fuzzy rules*

Our first goal was to test if the rules obtained by means of genetic evolution were able to control successfully an autonomous robot as it has been probed in other environments, [6,14]. So, the robot was given some fixed concepts such as near, far, etc. for the sensors and slow and fast for the motors. The genetic program should be able to successfully combine these concepts.

The robot was located in a simple environment. It consisted of a rectangular area with two obstacles situated as shown in Fig. 2. The walls were made of cardboard and the floor was the surface of a wooden table. This table was situated in an artificially illuminated laboratory, without windows, in order to avoid variations in the lightning conditions.

Each of the Khepera sensors returns an integer, whose range is an integer between 0 (no obstacle detected) and 1023 (an obstacle just in front of the sensor). The speed of both motors is also fixed by an integer. In this case, the range we have used goes from -10 (maximum speed backwards) to 10 (maximum speed forward). In order to design the controller we have grouped the six frontal sensors into two groups (using a simple average of the values returned by each sensor), as shown in Fig. 1. So, the fuzzy controller will have four linguistic variables (*sensor*1, *sensor*2, *motor*1 and *motor*2).

We have defined five fuzzy partitions for each one of the sensor and motor variables. These partitions are defined as is shown in Fig. 3. So, the behavior of an individual can be defined by two 5×5 tables, each one controlling one motor. Given a perceived situation, described in terms of the fuzzy partitions of the sensors, each table assigns a fuzzy partition of the motor variable to its corresponding motor.

These tables will be the chromosomes [4] that will be optimized by the genetic algorithm. The alleles will be integers {0, 1, 2, 3, 4, 5} which correspond to the five membership functions *Very-Backward, Backward, Stop, Forward*, and *Very-Forward* previously defined, plus *blank*, which indicates that there is no relation between the two membership functions. Both tables

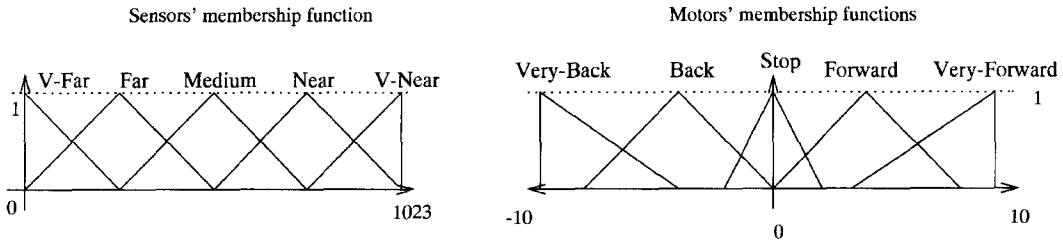


Fig. 3. Fuzzy partitions for sensors and motors.

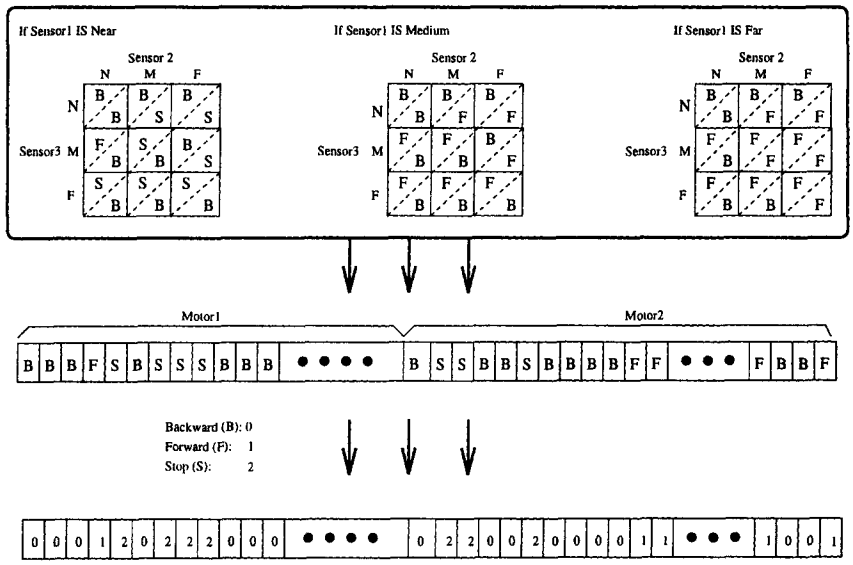


Fig. 4. Genetic codification of the fuzzy controller.

make up the *genotype* of each individual and its chromosome is consequently made up by 50 integer gens. This process is summarized in Fig. 4.

The *phenotype* is the behavior that the fuzzy controller produces. The behavior is obtained applying the usual fuzzy operations: fuzzification, max-composition and centroid defuzzification. These phenotypes are tested in the environment and the best one is promoted to the next generation. In order to measure the performance of the phenotypes we should define a *fitness* function.

The fitness criterion, Θ, was defined depending on three variables directly measured on the mini-robot Khepera, and a fourth one on the genotype, as

$$\Theta = \frac{V(1-\sqrt{D})(1-I)}{\text{rules}},$$

where the variable I represents the normalized value of the sensor which presents the highest level of activation:

$$I = \frac{\text{sensor}}{1023},$$

V is the rotation average speed of the two wheels:

$$V = \frac{\text{average}}{10}$$

and D is the normalized absolute value of the difference between the speed of the two wheels:

$$D = \frac{|\, v_1 - v_2 \,|}{20}.$$

This makes function Θ be maximized by obstacle avoidance, higher speed, straight direction, and fewer rules.

This function has been heuristically developed. The first attempt was built considering only the number of collisions. This value is given by the variable I (when the robot bumps into an obstacle one sensor gets its maximum activation value). The obtained behavior was that the robot tends to be quiet. Obviously, this is one of the best ways to avoid collisions.

In order to make the robot move, the variable V was introduced. This variable evaluates the speed of the robot, that is, forces robot movements. Unfortunately, the phenotype obtained was that the robot moves, but it moves in circles over its position. Again, this is the best way of moving without bumping into any obstacle.

The appropriate phenotype was achieved when the variable D was introduced. This variable penalize instant differences (turns) between the speeds applied to the wheels. This means that better fitness is obtained when the robot tries to move further. Finally, the number of rules was introduced in order to reward phenotypes using a reduced set of rules.

The evolutionary training was a standard genetic programming process. It consists on 100 generations of 100 individuals each. The mutation operator was defined to change a fuzzy code of the chromosome, either up or down a level, including the blank code. The crossover operator used was a standard two-point crossover. The life time of each individual has been set to 20 s. This time let the robot go over most of the environment of Fig. 2.

An *elite* strategy, meaning that the best individual is automatically promoted to the next generation, was used to generate new populations. The probability values for the genetic operators were heuristically chosen and the following figures show how these factors influence on the learning process.

3.3. Experimental discussion

An example of the behaviors obtained is shown in Fig. 5. They were obtained using the following probabilities configuration:

Obstacles	1
Mutation	0.2
Crossover	0.2

At first sight, the set of rules obtained looks OK. It deals properly with the entropy problem: the controller uses the label *Very-Forward* only when it does not detect any obstacle. Besides, the turns are made by reducing the speed of one the wheels instead of accelerating. However, some strange rules have been generated. For instance, there are some blanks. This means that no rule was generated for that situation. This is due to the fact that we are rewarding the reduction in the number of rules. In fact, this is not a very interesting parameter and in our further experiments the number of rules would be probably removed from the fitness function.

Another singular set of rules is, for instance, the one that makes the robot go backwards in some situations. There are two major causes. The first one is that back sensors are not being considered. The second one is that the fuzzy partitions of the sensors were not well defined. The main problem here is the sensibility of

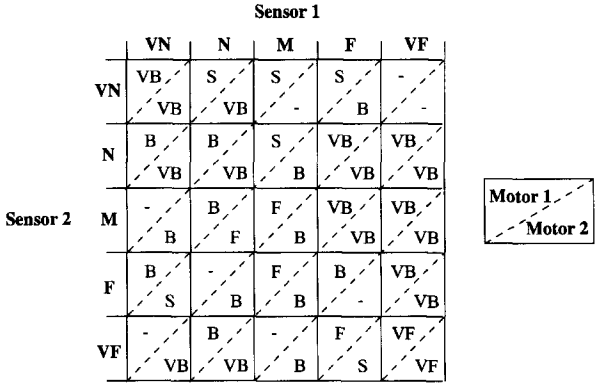


Fig. 5. Fuzzy controller obtained.

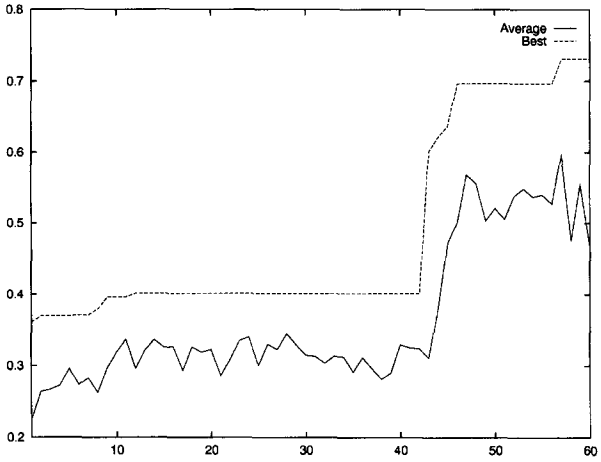


Fig. 6. Average vs. best *fitness*.

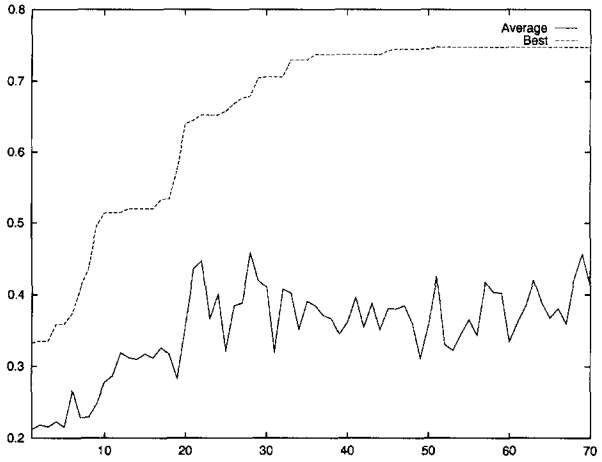


Fig. 7. Average vs. best *fitness*.

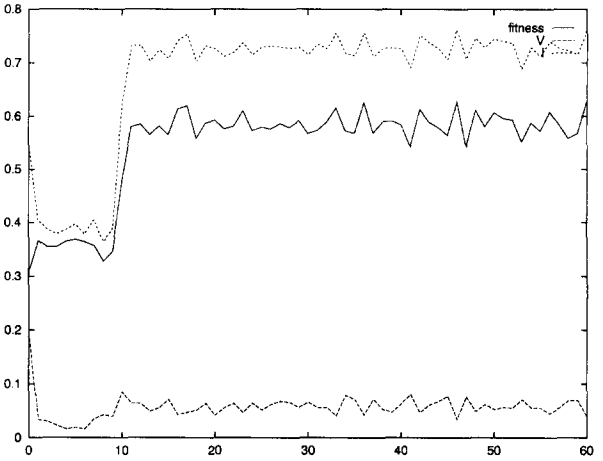


Fig. 8. *Fitness* vs. two main component factors.

the sensors. The definition made in Fig. 3 assumes that sensors sensibility is linear. This is not true, sensors have higher sensibility in some distances. Anyway, the behavior of the robot was the desired one. The robot moves in its environment as shown in the right part of Fig. 5. This means that the robot learns rules for the specific labels given. In future experiments, the evolution of the labels should also be considered.

Fig. 6 shows the evolution of the average fitness of each generation versus the fitness of the best individual of each generation for the same experiment. It can be appreciated that, although we allow 100 generations, around the 60th generation in most of the experiments, the population has learned to avoid obstacles.

Fig. 7 shows the same factors using different probabilities for the genetic operators:

Obstacles 1
Mutation 0.4
Crossover 0.4

This figure shows that if both the probabilities of mutation and crossover rise, then the speed of the learning process increases. We have also made experiments increasing only one of these factors and modifying the number of individuals in each generation and the number of generations. In this case, the increase of the speed is lower.

In Fig. 8 we show the contribution of the three different parts of the function Θ versus the total fitness. Both values (parts and total) shown in Fig. 8 correspond to the average of the individuals of each population. We have also tested the consequences of using a factor to improve one of the abilities, for instance, "to run faster" against "to keep safe". In these cases, the robot gets the behavior of avoiding obstacles but it gets an "specialized" behavior. This means that the robot, for instance in the case of keeping safe, develops rules that make it turn earlier. This makes this method of generating controllers really adaptable, so the only effort by the human designer is fixing the fitness function.

4. Conclusion

In this paper we have proposed a method to evolve high level rules using classical genetic algorithms. Using these evolutionary processes the mini-robot Khepera has been able to develop an autonomous behavior which allows it to move in an unknown environment without bumping into the obstacles. The role of researchers has been limited to provide the survival criterion and the structure of the fuzzy controller.

Artificial life community has not been very concerned about the emergence of symbolic concepts. On the other hand, the machine learning community has been mainly focussed on the improvement of the symbolic knowledge that a machine has about the world and how to operate in it. From this point of view, it has been forgotten how this symbolic knowledge can be obtained from raw data perceived in the real world.

The average set of rules of the individuals in the last generation shows that this method provides solutions that are similar to those designed by humans, or even better if the humans designers are not aware of problems like the increase of entropy.

Our current work is aimed at using the same approach in the emergence of the fuzzy membership functions and in the evolution of both parts of the fuzzy controllers at the same time. We are confident about the possibility of evolving both at the same time. We are also studying the possibility of mixing human designed controllers with genetically obtained ones.

References

[1] D.J. Cook, Using analytic and genetic methods to learn plans for mobile robots, in: Proceedings of the SPIE Conference on the Applications of Artificial Intelligence, 1993, pp. 327–336.

[2] M.G. Cooper, Evolving a rule-based fuzzy controller, Simulation 65 (1) (1995).

[3] J. Godjevac, N. Steele, Adaptive fuzzy controller for robot navigation, Conference FUZZ-IEEE96, New Orleans, USA, September 1996.

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

[5] F. Hoffmann, G. Pfister, Automatic design of hierarchical fuzzy controller using genetic algorithms, in: Proceedings of the EUFIT 94, Aachen, Germany, 1994.

[6] C. Karr, Applying genetics to fuzzy logic, AI Expert, March 1991, pp. 38–43.

[7] M.A. Lee and H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, in: Proceedings of the Second International Conference on Fuzzy Systems, 1993, pp. 612–617.

[8] V. Matellán, J.M. Molina, L. Sommaruga, Fuzzy cooperation of autonomous robots, in: Proceedings of the Fourth International System on Intelligent Robotics Systems, Lisboa, Portugal, July 1996.

[9] F. Mondada, F.P. Ienne, Mobile robot miniaturisation: A tool for investigation in control algorithms, in: Proceedings of the Third International Symposium on Experimental Robotics, Kyoto, Japan, 1993.

[10] F.G. Pin, Y. Watanabe, Driving a car using reflexive fuzzy behaviors, in: Proceedings of the Second International Conference on Fuzzy Systems, San Francisco, USA, 1993.

[11] P. Reignier, Fuzzy logic techniques for mobile robot obstacle avoidance, Robotics and Autonomous Systems 12 (1994) 143–153.

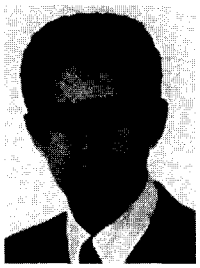
[12] A. Saffiotti, K. Konolige, E.H. Ruspini, A multivalued-logic approach to intergrating planning and control, Artificial Intelligence 76 (1–2) (1995) 481–526.

[13] L. Steels, The artificial life roots of artificial intelligence, Artifical Life 1 (1994) 75–110.

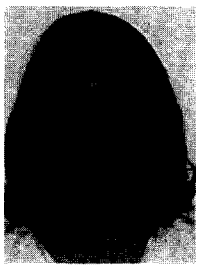
[14] P. Thrift, Fuzzy logic synthesis with genetic algorithms, in: Proceedings of the Third International Conference on Genetic Algorithms 1993, pp. 509–513.

[15] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, Transactions on Systems, Man and Cybernetics 1 (1) (1973).

[16] Z. Zhong, I.B. Turksen, An approximate analogical reasoning approach based on similarity measures, Transactions on Systems, Man and Cybernetics 18 (6) (1988).



Vicente Matellán received his Ph.D. in Computer Science from Universidad Politécnica de Madrid. He joined the Computer Science Department of the Universidad Carlos III de Madrid in 1993 and since then has been working in the Intelligent Agents Group. His research interests center around the application of multi-agent techniques, as well as other well-known paradigms of Artificial Intelligent, into the robotics field.



Camino Fernández received her degree in Computer Science from Universidad Politécnica de Madrid. M.S. degree in Artificial Intelligence from Facultad de Informátion (U.P.M.). She joined the Computer Science Department of the University Carlos III of Madrid in 1995. She is a member of the Artificial Life Group and her main research interests include the integration of symbolic and subsymbolic techniques for controlling autonomous agents.

José M. Molina received his Ph.D. in Telecommunication Engineering from Universidad Politécnica de Madrid. He was a member of the Systems, Signal and Radio Communications of the Universidad Politécnica de Madrid. He also joined the Computer Science Department in 1993 being enrolled in the Intelligent Agents Group. His current research focuses on the application of evolutionary computing techniques to engineering optimization problems, as the generation of adaptive behaviors for autonomous robots.