Design of a fuzzy controller in mobile robotics using genetic algorithms

M. Mucientes *, D.L. Moreno, A. Bugarín, S. Barro

Department of Electronics and Computer Science, University of Santiago de Compostela, 15782 Santiago de Compostela, Spain

Abstract

The design of fuzzy controllers for the implementation of behaviors in mobile robotics is a complex and highly time-consuming task. The use of machine learning techniques, such as evolutionary algorithms or artificial neural networks for the learning of these controllers allows to automate the design process. In this paper, the automated design of a fuzzy controller using genetic algorithms for the implementation of the wall-following behavior in a mobile robot is described. The algorithm is based on the Iterative Rule Learning (IRL) approach, and a parameter (δ) is defined with the aim of selecting the relation between the number of rules and the quality and accuracy of the controller. The designer has to define the universe of discourse and the precision of each variable, and also the scoring function. No restrictions are placed neither in the number of linguistic labels nor in the values that define the membership functions.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Evolutionary algorithms; Fuzzy control; Mobile robotics; Wall-following behavior

1. Introduction

Fuzzy control has shown to be a very useful tool in the field of autonomous mobile robotics, characterized by a high uncertainty in the knowledge about the environment where the robot evolves.

The design of a fuzzy controller is generally made using expert knowledge about the task to be controlled. Expert knowledge is applied in order to decide the number of linguistic labels for each variable, to tune the membership functions, to select the most adequate linguistic values for the consequents, and to define the rules in the fuzzy knowledge base. This process is tedious and highly time-consuming. For this reason, automated learning techniques, such as evolutionary algorithms, have been employed for helping in some, or in all, of the tasks involved in the design process.

A knowledge base is formed of a database (number and definition of the linguistic values, and universe of discourse of each of the variables) and the rule base. Some authors have utilized evolutionary algorithms for learning or tuning fuzzy controllers in robotics [1–7]. In some of the approaches evolutionary algorithms are used just for tuning the membership functions [8,9]. In others, the complete rule base is learned, starting from a previously designed database [1,3–5]. Finally, some authors have used algorithms that learn both the database and the rule base [6].

In this paper, we describe the learning of a fuzzy controller for the wall-following behavior in a mobile robot. The designer has to define the universe of discourse and the precision of each variable, and also the scoring function. No restrictions are placed neither in the number of linguistic labels, nor in the values that define the membership functions. The methodology is based on the Iterative Rule Learning (IRL) approach [10]. The paper is organized as follows: in Section 2 some general comments about the evolutionary learning of knowledge bases are made, while in Section 3 the genetic learning methodology employed is explained. Section 4 describes the application of the proposed algorithm to the wall-following behavior, and Section 5 presents the results we obtained. Finally, conclusions and future work are pointed out in Section 6.

2. Evolutionary learning of knowledge bases

Learning of knowledge bases using evolutionary algorithms has three main approaches: Michigan, Pittsburgh, and IRL [11]. In the Michigan approach [12], each chromosome represents an individual rule, and the entire population is the rule base. Rules evolve along time due to their interaction with the environment.

* Corresponding author. Tel.: +34 981 563100x13570; fax: +34 981 528012.
E-mail addresses: manuel@dec.usc.es (M. Mucientes),
dave@dec.usc.es (D.L. Moreno), alberto@dec.usc.es (A. Bugarín),
senen@dec.usc.es (S. Barro).

The major problem of this approach is that of resolving the conflict between the performance of individual rules and that of the rule base. The objective is to obtain a good rule base, which means to obtain good individual rules, but also rules that cooperate between each other to get adequate outputs. This could be sometimes conflicting, for example, when an individual rule that receives a high payoff is not adequately cooperating with other rules. This problem is addressed in ref. [13].

This conflict is overcome by the Pittsburgh approach [14], where each chromosome represents a full knowledge base. Length of the chromosomes can be variable, which permits dealing with knowledge bases with a variable number of rules. This approach has a higher computational cost, because several knowledge bases have to be evaluated, while for the Michigan approach a single rule base is evaluated.

In the third approach (IRL [10]), each chromosome represents an individual rule, but contrary to the Michigan approach, a single rule is learned by the evolutionary algorithm and not the whole rule base. After each sequence of iterations, the best rule is selected and added to the final rule base. The selected rule must be penalized in order to induce niche formation in the search space. Niching is necessary for solving multimodal problems, as occurs with knowledge bases learning. In this case, each of the rules of the knowledge base is a solution (highly multimodal problem), and all the solutions must be taken into account to get the complete knowledge base. A common way to penalize the rules that have been obtained is to delete those training examples that have been covered by the set of rules that integrate the final rule base. The final step of the IRL approach is to check whether the obtained set of rules is a solution to the problem. In the case it is not, the process is repeated. A weak point of this approach is that the cooperation between rules is not taken into account when a rule is evaluated.

When learning knowledge bases, many approaches use a predefined number of linguistic labels for each variable, or the shape of the membership functions is constrained, facilitating the learning, but also limiting the solutions space and the strength of the method [1,3–5,8,9]. Learning of both the data and rule bases can be done simultaneously, or in different stages. The most common approach is to firstly learn the database and finally generate the rule base using the previously generated database [6,15].

3. Learning of fuzzy rule-based controllers

Our proposal consists on a learning method based on the IRL approach in which both the data and rule bases are simultaneously learned. The only predefined parameters are the universe of discourse and the precision of each variable. The number of linguistic labels, the shape of the membership functions and the rules' structure (a variable could not be considered in a rule) will be learned. The algorithm has the following steps:

(1) Obtain a rule for the system:
   (a) initialize population;

   (b) evaluate population;
   (c) eliminate bad rules and fill up population;
   (d) scale the fitness values;
   (e) while the maximum number of iterations is not exceeded
     (i) select the individuals of the population;
     (ii) crossover and mutate the individuals;
     (iii) evaluate population;
     (iv) eliminate bad rules and fill up population;
     (v) scale the fitness values.
(2) Add the best rule to the final rule set.
(3) Penalize the selected rule.
(4) If the knowledge base does not solve the problem, return to Step 1.

The rules that are going to be learned are conventional fuzzy rules like:

$$R^i = \text{If } X_1^i \text{ is } A_1^i \text{ and } \ldots \text{ and } X_{NA}^i \text{ is } A_{NA}^i \\ \text{Then } Y_1^i \text{ is } B_1^i \text{ and } \ldots \text{ and } Y_{NC}^i \text{ is } B_{NC}^i \quad (1)$$

where R^i, $i = 1, \ldots, \text{NR}$, is the ith rule, X_j^i, $j = 1, \ldots, \text{NA}$, and Y_k^i, $k = 1, \ldots, \text{NC}$, are linguistic variables of the antecedent and consequent parts, respectively. NR is the number of rules, NA the number of antecedents in a rule, NC the number of consequents, and A_j^i and B_j^i are linguistic values (labels) of these variables.

A set of examples has been chosen for learning the knowledge base. These examples cover the universe of discourse of all the variables in the antecedent part of the rule. The universes of discourse have been discretised, in order to minimize the search space, with a step or precision p_n, $n = 1, \ldots, \text{NV}$, where $\text{NV} = \text{NA} + \text{NC}$ is the number of variables. Prior to the application of the learning method, the best action for each one of the examples is determined. The function, SF, that scores the action of a rule over an example (not the fitness function) is application dependent.

An example, e^l, is covered by rule R^i if it complies with the following two conditions:

$$A_1^i(e_1^l) \wedge \cdots \wedge A_{NA}^i(e^l) > 0 \quad (2)$$

where $A_j^i(e_j^l)$ represents the membership degree of the value of variable j in the example e^l to A_j^i. A new parameter, δ, is defined with the aim of selecting the relation between the number of rules and the quality and accuracy of the controller. In that way, a second condition is imposed:

$$\frac{\text{SF}(R^i(e^l))}{\max(\text{SF}(e^l))} > \delta \quad (3)$$

where SF $(R^i(e^l))$ is the score assigned to the state reached by the system after applying rule R^i over example e^l, and $\max(S\text{-}F(e^l))$ is the maximum score that an action can obtain for example e^l. The value of parameter δ can be adjusted in a range between 0 and 1. A low value of δ produces a lower number of rules in the final knowledge base, but the quality and the accuracy of the controller decreases. On the contrary, a high
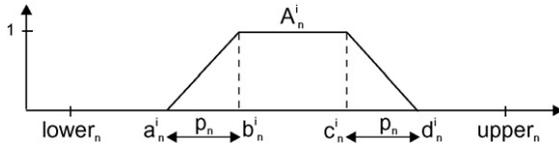
Fig. 1. Shape of the membership function for the linguistic value A_n^i.

value of δ increases the quality of the controller, but also the number of rules.

The use of δ can be clarified by means of the following example. Let us suppose a robot must reach a point by turning $30°$. Although this may be labeled as the best control action, also a rule proposing a turning of $20°$ should be considered as a good rule, even though the goal point is not fully reached. Parameter δ indicates the minimum quality a rule must have in order to be a valid rule for being added to the final knowledge base.

The shape of the membership functions that are going to be learned is shown in Fig. 1. Parameters b_n^i and c_n^i are learned, but points a_n^i and d_n^i are calculated as:

$$a_n^i = \max\{b_n^i - p_n, \quad \text{lower}_n\} \tag{4}$$

$$d_n^i = \min\{c_n^i + p_n, \quad \text{upper}_n\} \tag{5}$$

where lower_n and upper_n are the extreme points of the universe of discourse of variable n. In that way, it is not possible to exceed the range of the universe of discourse of the variables during the learning process. For example, for a variable n having the following values: $\text{lower}_n = 0$, $\text{upper}_n = 60$, $p_n = 10$, $b_n^i = 40$, and $c_n^i = 60$, then $a_n^i = \max\{40 - 10, 0\} = 30$ and $d_n^i = \min\{60 + 10, 60\} = 60$.

Chromosomes are real coded, and a rule as the one shown in (1) is encoded into a chromosome C^i as:

$$C^i = (b_1^i, c_1^i, \ldots, b_{NV}^i, c_{NV}^i) \tag{6}$$

The first step of the genetic algorithm consists in initializing the population. Rules in the initial population are created in the following way. An example currently not covered by any rule in the final knowledge base is randomly selected. The created rule is going to cover that single example, named e^l, at this initial generation. The membership functions for this rule are constructed as: $b_n^i = c_n^i = e_n^l$, while a_n^i and d_n^i are calculated using (4) and (5), respectively.

The evaluation of each individual of the population (each rule) is done with a two level fitness function. The first level (FF) consists on counting the number of examples that are covered by this rule, i.e., that fulfill (2) and (3), and that are not yet covered by a rule of the final knowledge base. If an example is covered by a rule of the final knowledge base, it will not contribute to the fitness value of any rule in the population. A second level for the fitness function is added in order to distinguish between rules with the same antecedent part but different consequents. It is the average value of the scoring

function, SF, for all the examples that verify (2):

$$\text{ASF}^i = \frac{\sum_{l=1}^{\text{NE}} \text{SF}(R^i(e^l))}{\text{NEC}^i} \tag{7}$$

where NE is the number of examples, and NEC^i is the number of examples that verify (2) for rule i. The second level of the fitness function is only used (in conjunction with the first level) when the final generation has been reached and the best rule of the population has to be added to the final knowledge base.

If for any example a rule verifies (2) but not (3), then this rule is deleted from the population. After the deletion of all the bad rules, the population must be filled up, until its size reaches NR. The rules that will be added are the best rules in the population. Finally the fitness values of the individuals of the population must be linearly scaled in order to prevent premature convergence of the population.

The selection procedure that has been employed is the stochastic remainder without replacement. An individual i will be selected $\text{int}(\text{FF}^i/\text{AFF})$ times, where FF^i is the value of the fitness function (first level) for individual i, and AFF is the average value of all FF^i. Taking into account $\text{frac}(\text{FF}^i/\text{AFF})$ the population is randomly filled up. After selection, the individuals are crossed (one-point crossover), mutated, and finally added to the new population. Elitism has been applied to avoid the loss of the best individuals due to crossover and mutation.

The mutation operator has three equally probable options to operate on a gene: increase or decrease its value in an amount of p_n, or leave it unchanged. This will provoke the extension or contraction of the membership function in a quantity equal to the precision of each variable, implementing a local search in that way. After crossover and mutation, each chromosome will be reordered for repairing bad definitions of the membership functions ($b_n^i > c_n^i$), and values of a_n^i and d_n^i will be calculated using (4) and (5), respectively.

Once the maximum number of iterations has been reached, the best rule of the population is added to the final knowledge base, and finally penalized. Penalization is done marking all the examples covered by this rule. In that way these examples will not contribute to the fitness value of the individuals in the next sequence of iterations. If all the examples are covered by the rules of the final knowledge base, then this is a solution to the problem and the algorithm ends.

4. Learning the wall-following behavior

The methodology presented in the previous section is applied here for the design of a fuzzy controller for the wall-following behavior in mobile robotics.

The wall-following behavior is usually implemented when the robot is exploring an unknown area, or when it is moving between two points in a map. A good wall-following controller is characterized by three features: to maintain a suitable distance from the wall that is being followed, to move at a high velocity whenever the layout of the environment is permitting, and finally to avoid sharp movements, making smooth and
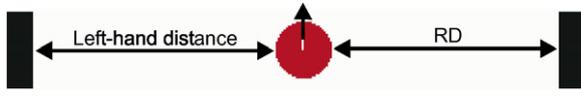
Fig. 2. Description of some of the distances used for the calculation of the input variables.

progressive turns and changes in velocity. The controller can be configured modifying the values of two parameters: the reference distance, which is the desired distance between the robot and the selected wall, and the maximum velocity attainable by the robot. In what follows we assume that the robot is going to follow a contour that is on its right side. Of course, the robot could also follow the left-hand wall, but this can be easily dealt with by simply interchanging the sensorial inputs.

The input variables of the control system are the right-hand distance (RD), the distances quotient (DQ), which is calculated as:

$$DQ = \frac{\text{left-hand distance}}{RD} \qquad (8)$$

As it can be seen (Fig. 2), DQ shows the relative position of the robot inside a corridor, which provides with information that is more relevant to the problem than simply using the left-hand distance. A high value for DQ means that the robot is closer to the right-hand wall, while a low value indicates that the closer wall is the left-hand one. The other input variables are the linear velocity of the robot (LV), and the orientation of the robot with respect to the wall it is following. A positive value of the orientation indicates that the robot is approaching to the wall, while a negative value means the robot is moving away from the wall. The output variables are the linear acceleration and the angular velocity.

All the information used to calculate distances and orientations comes from the ultrasound sensors of a Nomad 200 robot. The distances and the orientation are obtained in two ways: if any of the walls (left or right) can be modeled with a straight line using a least square mean of the raw sensor data, then the corresponding distance and orientation are measured from that line. Otherwise, distance is measured as the minimum distance of a set of sensors, and the orientation will be the orientation of that sensor with respect to the advance direction.

The function SF used at (3) is defined for this application as:

$$SF(R^i(e^l)) = \frac{1}{\alpha_1 + \alpha_2 + \alpha_3 + 1} \qquad (9)$$

where α_1, α_2, and α_3 are, respectively:

$$\alpha_1 = 100 \frac{|RD - \text{reference distance}|}{p_{RD}} \qquad (10)$$

$$\alpha_2 = 10 \frac{|\text{maximum velocity} - LV|}{p_{LV}} \qquad (11)$$

$$\alpha_3 = \frac{|\text{orientation}|}{p_{\text{orientation}}} \qquad (12)$$

Table 1
Universes of discourse and precisions of the variables

Variable	Universe of discourse: [lower$_n$, upper$_n$]	Precision (p_n)
RD	[0, 3.0]	0.2
DQ	[0, 2.7]	0.3
Orientation	[−450, 450]	50
LV	[0.0, 1.0]	0.125
Linear acceleration	[−1.0, 1.0]	0.125
Angular velocity	[−1.0, 1.0]	0.05

and p_{RD}, p_{LD}, and $p_{\text{orientation}}$ are the precisions of the respective input variables. Table 1 shows the values of the precisions and the universes of discourse of each variable. The precisions are used in these equations in order to evaluate the deviations of the values of the variables from the desired ones in a relative manner (the deviation of the value of variable n from the desired one is measured in units of p_n). This makes the comparison of the deviations of different variables possible and, as a consequence, the assignment of the weights for each one of the variables. These weights (100, 10, and 1 for (10–12), respectively) have been heuristically determined, and indicate how much important the deviation in the value of a variable is with respect to the deviation of other variables.

SF takes values in [0, 1]. The highest weight has been assigned to the distance, as small variations of RD with respect to the reference distance should be highly penalized. An intermediate weight is associated to velocity and, finally, the least important contribution to function SF is for the orientation of the robot.

DQ is implicitly included in SF, as its values are used in combination with RD in order to calculate the width of the passageway in which the robot is placed. If this width is too low (less than two times the reference distance), the value of the reference distance for that situation will be modified (half of the width will be selected), and the robot will follow the right wall closer and with a lower speed.

The defuzzification method that has been used for the learned fuzzy controller is the height defuzzifier [16]:

$$O_k = \frac{\sum_{i=1}^{NR} \bar{Y}_k^i B_k^i(\bar{Y}_k^i)}{\sum_{i=1}^{NR} B_k^i(\bar{Y}_k^i)} \qquad (13)$$

where $\bar{Y}_k^i$ is the center of gravity of the linguistic label B_k^i, and O_k is the defuzzified value for variable k. This method does not take into account the width of the membership functions from the consequent part of the rule. For this reason, and in order to simplify the learning process, the membership functions of the variables of the consequent part are crisp. This simplification does not affect the quality of the obtained controller.

5. Results

The system described in the previous sections has been implemented with a crossover probability of 0.2 and a mutation probability (per gene) of 0.4 (remember that mutation can increase, decrease or maintain unchanged a gene with equal
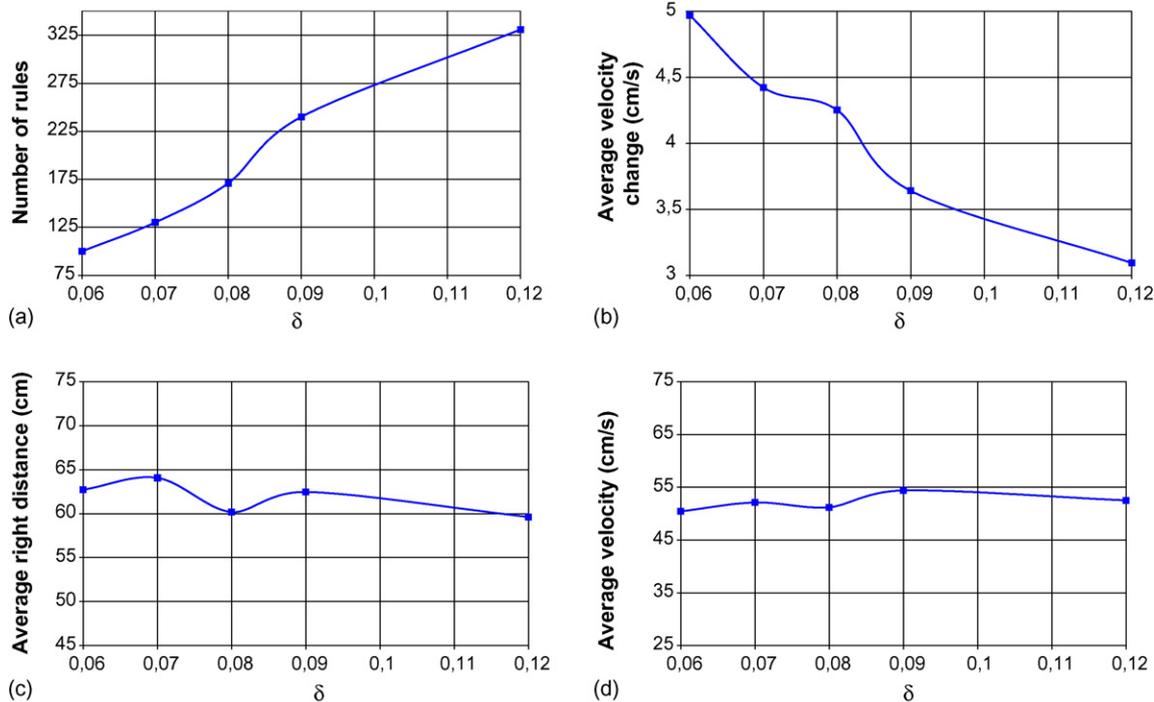
Fig. 3. Variation of some parameters with δ for the final knowledge base.

probability). These values have been selected in order to focus search in a few promising areas (low crossover probability), but exploiting those areas doing a local search due to the high mutation probability, in a similar way evolution strategies, for example $(1 + 1)$-ES, work. We noticed that a high crossover probability distracted the search due to the combination of rules from quite different areas of the search space, but a low crossover probability is still necessary to discover new promising areas.

The population size is 300 individuals, and the maximum number of generations is 50. Once that value is reached, the best rule of the population is added to the final knowledge base. The process is repeated until the final knowledge base covers all the examples. Different values for parameter $\delta(3)$ have been tried. All the parameters of the evolutionary algorithm have been heuristically obtained.

Fig. 3 shows the variation of the number of rules (a), the average velocity change (b), the average right distance (c), and the average linear velocity (d) with δ for the final knowledge base. These values have been measured for the environment shown in Fig. 4(b). Values of $\delta < 0.06$ have been discarded since they did not produce valid controllers. As δ increases the number of rules rises. The average velocity change evaluates the average change of the robot's velocity between two consecutive control iterations. Low values of the average velocity change indicate smooth and progressive changes in velocity, reflecting more accuracy and quality in the control actions. As can be seen in Fig. 3(b), as δ increases the average velocity change decreases, and consequently the accuracy and quality of the controller rises. The difference between the highest and lowest value of the average velocity change for the

different values of δ is of more than the 65%. As opposed to this, the difference in the extreme values of the right distance (c) and the velocity (d) is around 7% and 3%, respectively, which are not meaningful. Thus, a higher number of rules does not improve the average velocity or the average right distance too much, but gets smoother behaviors.

All the controllers that have been obtained were tested in three simulated environments using the Nomad 200 robot simulation software. We have to emphasize that the examples training set is different from the environments testing set.

Learning only depends on function SF, which has to be carefully selected, and also on the examples, that must be chosen covering the input space with an adequate precision (selection of the precision of the variables is also of high importance). None of the environments shown in this paper (Fig. 4) has been used during the learning process.

Fig. 4 shows as an example the robot path in three simulated environments for $\delta = 0.06$. The robot trajectory is represented by circular marks. A higher concentration of marks indicates lower velocity. The learned controller has 100 rules, the maximum velocity the robot can reach is 61 cm/s, and the reference distance at which the robot should follow the right wall is 51 cm. Ten tests have been done for each one of the environments. The average values measured for some parameters that reflect the controller performance are shown in Table 2.

Environment (b) is quite complex, with three concave corners and seven convex corners in a circuit of a length of 54 m. Convex corners are truly difficult situations, because the robot's sensors may cease to correctly detect the wall at some given moments, even though some of them may occasionally
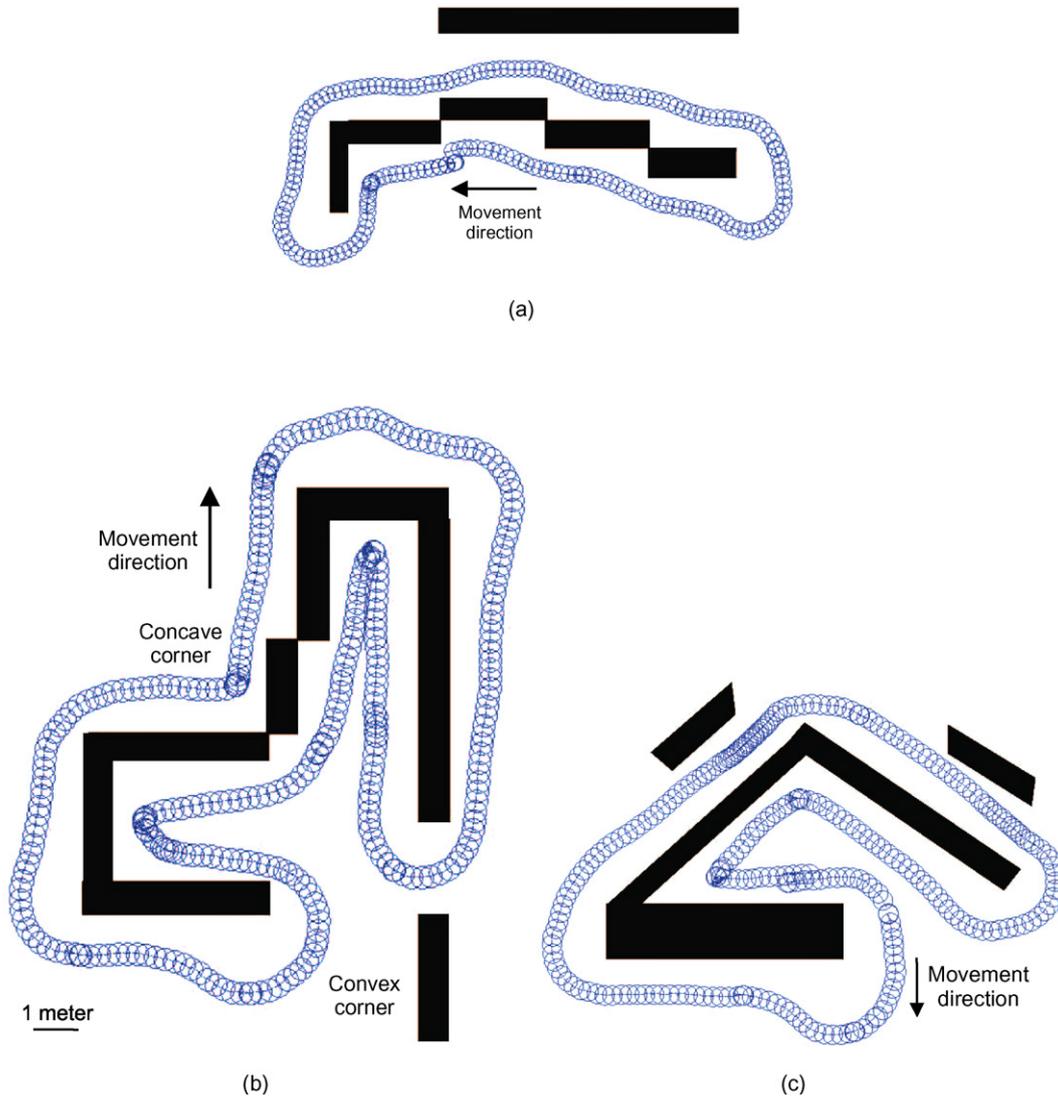
Fig. 4. Path of the robot in different simulated environments for $\delta = 0.06$.

detect it. The controller must also significantly reduce velocity at corners. In spite of these difficulties, the obtained average velocity has been quite high, and the distance at which the robot should follow the wall is near the desired reference distance. The difference between both distances is caused by the high number of corners, in which the orientation of the robot is very bad (at concave corners the robot is detecting two perpendicular walls, and sometimes at convex corners it detects no wall), and a fast turning is prioritized over a correct distance.

Nevertheless, some aspects could be improved. Thus, comparing this controller with refs. [17,18] (Fuzzy Temporal Rule-based controller, hand-designed involving 313 rules), the

Table 2
Average values of some parameters for the environments of Fig. 4

Environment	RD (cm)	Velocity (cm/s)	Velocity change (cm/s)	Time (s)
(a)	70	57	4.23	62
(b)	65	51	4.59	106
(c)	63	48	5.48	87

obtained behavior provokes sharp changes in velocity. For this reason, one of the aspects that characterizes a good wall-following controller (smooth and progressive turns and changes in velocity) is not completely fulfilled. Also, the controller proposed in refs. [17,18] gets a trajectory closer to the shape of the contour being followed. On the other hand, the time spent for the design of a hand-designed controller is much higher, and also a deep knowledge of the task to be controlled is required. Another advantage of the controller presented in this paper is the number of rules, much lower than in refs. [17,18].

6. Conclusions and future work

A genetic algorithm based on the IRL approach for the learning of fuzzy controllers has been described. Learning has no restrictions neither in the number of linguistic values for each variable, nor in the values that define the membership functions. The process only depends on function SF, and on the selected precisions of the variables, which are application dependent and must be carefully chosen. The algorithm has

been applied to the learning of the wall-following behavior. The learned control systems (for different values of δ) have been tested in a set of simulated environments with a high number of corners, showing a good performance both in the distance the wall was followed and in the average velocity.

Some aspects must be improved in the future, in order to get a better cooperation among rules of the final knowledge base, and enhance the interpretability of the rules. Another challenging work will be the learning of Fuzzy Temporal Rule-based controllers [17–19], which have a high degree of expressiveness and of analyzing the temporal evolution of variables, while taking past values into account.

Acknowledgement

References

[1] D.K. Pratihar, K. Deb, A. Ghosh, Optimal path and gait generations simultaneously of a six-legged robot using a ga-fuzzy approach, Robot. Auton. Syst. 41 (2002) 1–20.

[2] C. Zhou, Robot learning with ga-based fuzzy reinforcement learning agents, Inf. Sci. 145 (2002) 45–68.

[3] D. Gu, H. Hu, L. Spacek, Learning fuzzy logic controller for reactive robot behaviours, in: Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), 2003, pp. 46–51.

[4] D. Gu, H. Hu, J. Reynolds, E. Tsang, Ga-based learning in behaviour based robotics, in: Proceedings of the 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Kobe, Japan, (2003), pp. 1521–1526.

[5] H. Hagras, V. Callaghan, M. Collin, Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy-genetic system, Fuzzy Sets Syst. 141 (2004) 107–160.

[6] M. Mucientes, D.L. Moreno, C.V. Regueiro, A. Bugarín, S. Barro, Design of a fuzzy controller for the wall-following behavior in mobile robotics with evolutionary algorithms, in: Proceedings of the International Conference of Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'2004), Perugia, Italy, (2004), pp. 175–182.

[7] D. Leitch, Genetic algorithms for the evolution of behaviours in robotics, in: Genetic Algorithms and Soft Computing, Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, 1996, pp. 306–328.

[8] R. Braunstingl, J. Mujika, J.P. Uribe, A wall following robot with a fuzzy logic controller optimized by a genetic algorithm, in: Proceedings of the International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium, vol. 5, Yokohama, Japan, (1995), pp. 77–82.

[9] D.K. Pratihar, K. Deb, A. Ghosh, A genetic-fuzzy approach for mobile robot navigation among moving obstacles, International Journal of Approximate Reasoning 20 (2) (1999) 145–172.

[10] O. Cordón, F. Herrera, Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems, Fuzzy Sets Syst. 118 (2001) 235–255.

[11] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases, Advances in Fuzzy Systems—Applications and Theory, vol. 19, World Scientific, 2001.

[12] L. Magdalena, J.R. Velasco, Fuzzy Rule-based controllers that learn by evolving their knowledge base, in: Genetic Algorithms and Soft Computing, Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, 1996, pp. 172–201.

[13] A. Bonarini, Evolutionary learning of fuzzy rules: competition and cooperation, in: W. Pedrycz (Ed.), Fuzzy Modelling: Paradigms and Practice, Kluwer Academic Press, Norwell, USA, 1996, pp. 265–284.

[14] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, Fuzzy Sets Syst. 80 (1996) 273–293.

[15] B. Carse, T.C. Fogarty, A. Munro, Evolving temporal fuzzy rule-bases for distributed routing control in telecommunication networks, in: Genetic Algorithms and Soft Computing, Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, 1996, pp. 467–488.

[16] J.M. Mendel, Fuzzy logic systems for engineering: a tutorial, Proc. IEEE 83 (3) (1995) 345–377.

[17] M. Mucientes, R. Iglesias, C.V. Regueiro, A. Bugarín, S. Barro, A fuzzy temporal rule-based velocity controller for mobile robotics, Fuzzy Sets Syst. 134 (2003) 83–99.

[18] M. Mucientes, R. Iglesias, C.V. Regueiro, A. Bugarín, S. Barro, A fuzzy temporal rule-based approach for the design of behaviors in mobile robotics, in: Intelligent Systems: Technology and Applications, Fuzzy Systems, Neural Networks and Expert Systems of CRC Press International Volumes on Intelligent Systems Techniques and Applications, vol. 2, CRC Press, 2003, pp. 373–408.

[19] M. Mucientes, R. Iglesias, C.V. Regueiro, A. Bugarín, P. Cariñena, S. Barro, Fuzzy temporal rules for mobile robot guidance in dynamic environments, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 31 (3) (2001) 391–398.