25–29 July, 2004 • Budapest, Hungary

Obtaining a Fuzzy Controller with High Interpretability in Mobile Robots Navigation

Manuel Mucientes
Dept. Electronics and Computer Science
University of Santiago de Compostela
Spain, E-15782
E-mail: manuel@dec.usc.es

Jorge Casillas
Dept. Computer Science and Artificial Intelligence
University of Granada
Spain, E-18071
E-mail: casillas@decsai.ugr.es

Abstract— **The paper presents the design of a fuzzy controller for the wall-following behavior in mobile robotics using the COR (Cooperative Rules) methodology with Ant Colony Optimization. The system has been tested in several simulated environments using the Nomad 200 robot software, and compared with other controller based on genetic algorithms. The proposed approach obtains a highly interpretable knowledge base in a reduced time, and the designer only has to define the number of membership functions and the universe of discourse of each variable.**

I. Introduction

The field of mobile robotics is characterized by the high amount of uncertainty presented in real and unconstrained environments. Furthermore, information provided by robot sensors is noisy and unreliable. Fuzzy logic has shown to be a useful tool when dealing with this uncertainty, and has been widely used for the design of behaviors in robotics [1]. The design of Fuzzy Rule-Based Systems (FRBS) requires a deep knowledge on the task to be controlled and forces to spend long time tuning the controller [2]. Due to that, in the last few years the use of learning methods for the design of fuzzy controllers has been generalized. The main approaches are evolutionary algorithms [3] and neural networks [4].

In this paper we present the design of a fuzzy controller for the wall-following behavior in mobile robotics using the COR (Cooperative Rules) methodology [5], [6]. The main advantages of the proposed approach are the easiness in the design, as only the number of membership functions and the universe of discourse of each variable have to be defined. Other advantages are the speed in the obtaining of the knowledge base (due to the search space reduction) and its high degree of interpretability. Finally, the use of a function that scores the action of the controller over each of the examples of the training set (which cover the universe of discourse of all the variables) allows that the quality of the learned behavior does not depend on the environment, and also that the robot will be capable to face any situation.

The paper is organized as follows. Section II introduces the wall-following behavior. Section III presents the methodology that has been used. Section IV shows the obtained results and, finally, conclusions are discussed.

II. Learning the Wall-Following Behavior

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 possible, and finally to avoid sharp movements, making smooth and 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:

$$DQ = \frac{left - hand\,distance}{RD} \quad (1)$$

As it can be seen (Fig. 1), 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, whilst 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, whilst 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 is obtained from the ultrasound sensors of the 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

0-7803-8353-2/04/$20.00 © 2004 IEEE

of a set of sensors, and the orientation will be the orientation of that sensor with respect to the advance direction.

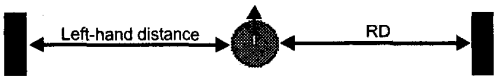

Fig. 1. Description of some of the distances used to calculate input variables

A set of examples (23085) 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 discretized, in order to minimize the search space, with a step or granularity g_n, where n is the variable. Function SF, that scores the action of the rule base over an example, is defined as:

$$SF\left(RB\left(e^l\right)\right) = \frac{1}{\alpha_1 + \alpha_2 + \alpha_3 + 1} \tag{2}$$

where α_1, α_2, and α_3 are respectively:

$$\alpha_1 = 100 \cdot \frac{|RD - reference\,distance|}{g_{RD}} \tag{3}$$

$$\alpha_2 = 10 \cdot \frac{|maximum\,velocity - LV|}{g_{LV}} \tag{4}$$

$$\alpha_3 = \frac{|orientation|}{g_{orientation}} \tag{5}$$

and g_{RD}, g_{LV}, and $g_{orientation}$ are the granularities of the respective input variables. The granularities 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 g_n). This makes possible the comparison of the deviations of different variables and, as a consequence, the assignment of the weights for each one of the variables. These weights (100, 10 and 1 for (3), (4), and (5) 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. 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.

The index that measures the global quality of the encoded rule set is:

$$f(RB) = \frac{1}{2 \cdot NE} \sum_{l=1}^{NE} \left\{ \left\{ 1 - \frac{SF\left(RB\left(e^l\right)\right)}{\max\left(SF\left(e^l\right)\right)} \right\} \cdot \omega \right\}^2 \tag{6}$$

where NE is the number of examples, ω is a scaling factor that has been set to 1000, and $\max\left(SF\left(e^l\right)\right)$ is the maximum score that an action can obtain for example e^l. These values are obtained before the beginning of the algorithm, trying and scoring all the possible actions for each example.

III. Learning Methodology Based on COR

The process followed to learn the fuzzy controller is based on the COR methodology (proposed in [5] and extended in [6]). We have selected this process due to its good properties to quickly obtain knowledge bases with a high interpretability. The three following subsections describe the learning methodology, an analysis of its main properties, and the proposed algorithm based on it.

A. COR Methodology

A family of efficient and simple methods to derive fuzzy rules guided by covering criteria of the data in the example set, called *ad hoc data-driven methods*, has been proposed in the literature in the last few years. Their simplicity, in addition to their quickness and easy understanding, make them very suitable for learning tasks. However, ad hoc data-driven methods usually look for the fuzzy rules with the best individual performance (e.g. [7]) and therefore the global interaction among the rules of the rule base is not considered, thus involving knowledge bases with a bad accuracy.

With the aim of addressing these drawbacks keeping the interesting advantages of ad hoc data-driven methods, the COR methodology is proposed [5]. Instead of selecting the consequent with the highest performance in each subspace like these methods usually do, the COR methodology considers the possibility of using another consequent, different from the best one, when it allows the FRBS to be more accurate thanks to have a knowledge base with better cooperation.

COR consists of two stages:

1) *Search space construction* — It obtains a set of candidate consequents for each rule.
2) *Selection of the most cooperative fuzzy rule set* — It performs a combinatorial search among these sets looking for the combination of consequents with the best global accuracy.

A wider description of the COR-based rule generation process is shown in Fig. 2.

B. Advantages of the COR Methodology to Learn Fuzzy Controllers in Mobile Robots Navigation

The above mentioned methodology has some interesting advantages that make it very useful to learn fuzzy controllers in mobile robots navigation. We can mainly highlight two characteristics:

1) *Search space reduction* — The COR methodology reduces the search space basing on heuristic information. This fact differences COR from other rule base learning methods [8] and allows it to be quicker and to make a better solution exploration. This is an important issue for the learning of fuzzy controllers, where a high number of examples is used. In the wall-following behavior presented in this paper, 23085 examples have been used, and the employed methodology spends only 20 minutes in order to obtain the controller. As opposed to this, a solution based on genetic algorithms with the same number of examples could spent several hours.

Inputs:

- *An input-output data set*—$E = \{e_1, \ldots, e_l, \ldots, e_N\}$, with $e_l = (x_1^l, \ldots, x_n^l, y_1^l, \ldots, y_m^l)$, $l \in \{1, \ldots, N\}$, N being the data set size, and n (m) being the number of input (output) variables—*representing the behavior of the problem being solved.*
- *A fuzzy partition of the variable spaces.* In our case, uniformly distributed fuzzy sets are regarded. Let $\mathcal{A}_i$ be the set of linguistic terms of the i-th input variable, with $i \in \{1, \ldots, n\}$, and $\mathcal{B}_j$ be the set of linguistic terms of the j-th output variable, with $j \in \{1, \ldots, m\}$, with $|\mathcal{A}_i|$ ($|\mathcal{B}_j|$) being the number of labels of the i-th (j-th) input (output) variable.

Algorithm:

1) *Search space construction*:

1.1. *Define the fuzzy input subspaces containing positive examples*: To do so, we should define the positive example set ($E^+(S_s)$) for each fuzzy input subspace $S_s = (A_1^s, \ldots, A_i^s, \ldots, A_n^s)$, with $A_i^s \in \mathcal{A}_i$ being a label, $s \in \{1, \ldots, N_S\}$, and $N_S = \prod_{i=1}^{n} |\mathcal{A}_i|$ being the number of fuzzy input subspaces. In this paper, we use the following:

$$E^+(S_s) = \{ \quad e_l \in E \mid \forall i \in \{1, \ldots, n\}, \; \forall A_i' \in \mathcal{A}_i, \mu_{A_i^s}(x_i^l) \geq \mu_{A_i'}(x_i^l) \; \} \tag{7}$$

with $\mu_{A_i^s}(\cdot)$ being the membership function associated with the label A_i^s.
Among all the N_S possible fuzzy input subspaces, consider only those containing at least one positive example. To do so, the set of subspaces with positive examples is defined as $S^+ = \{S_h \mid E^+(S_h) \neq \emptyset\}$.

1.2. *Generate the set of candidate rules in each subspace with positive examples*: Firstly, the candidate consequent set associated with each subspace containing at least an example, $S_h \in S^+$, is defined. In this paper, we use the following:

$$C(S_h) = \{ \quad (B_1^{k_h}, \ldots, B_m^{k_h}) \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_m \mid \exists e_l \in E^+(S_h) \; where \; \forall j \in \{1, \ldots, m\}, \; \forall B_j' \in \mathcal{B}_j, \; \mu_{B_j^{k_h}}(y_j^l) \geq \mu_{B_j'}(y_j^l) \; \}. \tag{8}$$

Then, the candidate rule set for each subspace is defined as $CR(S_h) = \{R_{k_h} = [\text{IF } X_1 \text{ is } A_1^h \text{ and } \ldots \text{ and } X_n \text{ is } A_n^h \text{ THEN } Y_1 \text{ is } B_1^{k_h} \text{ and } \ldots \text{ and } Y_m \text{ is } B_m^{k_h}] \text{ such that } (B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)\}$.
To allow COR to reduce the initial number for fuzzy rules, the special element $R_\emptyset$ (which means "do not care") is added to each candidate rule set, i.e., $CR(S_h) = CR(S_h) \cup R_\emptyset$. If it is selected, no rules are used in the corresponding fuzzy input subspace.

2) *Selection of the most cooperative fuzzy rule set* — This stage is performed by running a combinatorial search algorithm to look for the combination $RB = \{R_1 \in CR(S_1), \ldots, R_h \in CR(S_h), \ldots, R_{|S^+|} \in CR(S_{|S^+|})\}$ with the best accuracy. Since the tackled search space is usually large, approximate search techniques should be used.
An index $f(RB)$ measuring the global quality of the encoded rule set is considered to evaluate the quality of each solution. In order to obtain solutions with a high interpretability, the original function is modified to penalize excessive number of rules:

$$f'(RB) = f(RB) + \beta \cdot f(RB_0) \cdot \frac{\#RB}{|S^+|} \tag{9}$$

with $\beta \in [0, 1]$ being a parameter defined by the designer to regulate the importance of the number of rules, $\#RB$ being the number of rules used in the evaluated solution (i.e., $|S^+| - |\{R_h \in RB \; such \; that \; R_h = R_\emptyset\}|$), and RB_0 being the initial rule base considered by the search algorithm.

Fig. 2. COR algorithm

This search space reduction is performed by two constraints:

a) *Maximum number of fuzzy input subspaces*: The maximum number of fuzzy input subspaces, and therefore maximum number of fuzzy rules, is limited by the positive example sets. The constrains imposed to construct $E^+(S_s)$ (see eq. (7) in Fig. 2) divides the input space with a crisp grid bounded by the cross points between labels and, therefore, each example contributes to generate a single rule. It is a conservative subspace set selection that generates the least possible number of rules that guarantee a whole covering of the examples.
In our problem, since the example data are uniformly distributed in the whole input space (as described in Sect. II), no reduction of the number of fuzzy input subspaces is done. Nevertheless, the fact of assigning each example to only one subspace will involve to reduce the number of candidate consequents, since the positive example sets are reduced.

b) *Candidate rule set in each subspace*: Once the fuzzy input subspaces are defined, a second search space reduction is made by constraining the set of possible consequents for each antecedent combination, i.e., the candidate rules in each subspace. Again, we use a restrictive condition to construct $C(S_h)$ (see eq. (8) in Fig. 2) that generates a low number of candidate rules.

To illustrate the effect of this search space reduction, from the example data set proposed in Sect. II, and using the following number of linguistic terms for each input/output variable, $|\mathcal{A}_1| = 5$, $|\mathcal{A}_2| = 2$, $|\mathcal{A}_3| = 5$, $|\mathcal{A}_4| = 2$, $|\mathcal{B}_1| = 9$, $|\mathcal{B}_2| = 9$, our methodology generates a search space of $\prod_{S_h \in S^+} |C(S_h)| = 9.8\text{e+}89$ combinations, while the total of possible combinations (considering the $|S^+| = 100$ input subspaces analyzed) is $(|\mathcal{B}_1| \cdot |\mathcal{B}_2|)^{100} = 7.1\text{e+}190$.

2) *Interpretability issues* — The proposed methodology has also some interesting advantages from the interpretability of the obtained fuzzy knowledge point of view. This is an important issue in fuzzy control for mobile robot navigation, as the actions of the robot are easily understandable and can be communicated to other modules that supervise the behavior in the control architecture. Basically, we can remark the two following aspects:

- *Model structure and membership functions keep invariable for an excellent interpretability*: The COR methodology is an effort to exploit the accuracy ability of linguistic FRBSs by exclusively focusing on the rule base design. In this case, the membership functions and the model structure keep invariable, thus resulting in the highest interpretability. Indeed, instead of improving the accuracy by deriving the

shape of the membership functions or by extending the model structure (weighted rules, linguistic hedges, hierarchical knowledge bases, etc.), COR methodology improves the accuracy inducing cooperation among linguistic fuzzy rules.

- *Rule base reduction to improve interpretability and accuracy*: A problem when defining a rule base is that one can not be sure whether the rules are correctly defined, i.e., without redundant rules or rules that generate conflicts with others in certain situations. Moreover, a high number of rules is difficult to be interpreted, even when a linguistic fuzzy rule structure is considered.

  To face this problem, a *rule reduction post-processing* is usually developed. When no restriction to the interpretability is considered, the rules can be merged [3], thus generating a scatter structure where each fuzzy rule uses different fuzzy sets for each variable.

  On the other hand, if we want to obtain linguistic fuzzy rules with high interpretability, a selection process can be developed to obtain a subset of the original rule base. However, this approach does not seem to be appropriate to generate an accurate final rule set since it is not considered interdependency between the learning and reduction tasks. That is, it is sure that after reducing the rule set, the new set of rules that best cooperate will be different.

  The COR methodology achieves the reduction process at the same time as the learning one with the aim of improving the accuracy (the cooperation among rules and thus the system performance can be improved by removing rules) and interpretability (a model with less rules is more interpretable) of the learned model.

  This process is performed by adding the null rule ($R_\emptyset$) to the candidate rule set corresponding to each subspace, as shown in the step 1.2. of Fig. 2. In this way, if such a element is selected for a specific subspace, this will mean that no rules will take part for the corresponding antecedent combination. Notice that the addition of $R_\emptyset$ in each candidate rule set slightly increases the search space. Moreover, the objective function used to guide the search algorithm (see eq. (9) in Fig. 2) is modified to penalizing solutions with a high number of rules.

C. COR Methodology with Ant Colony Optimization

Since the search space tackled in step 2. is usually large, it is necessary to use approximate search techniques. In [5], accurate linguistic models have been obtained using simulated annealing. However, since one of our constrains is to deal with a computational expensive evaluation function, in this paper the use of ant colony optimization (ACO) [9] is proposed (the metaheuristic is briefly described in Appendix A). This section describes the components of the proposed algorithm.

1) Problem Representation for Learning Cooperative Fuzzy Rules: To apply ACO in the COR methodology, it is convenient to see it as a combinatorial optimization problem with the capability of being represented on a weighted graph. In this way, we can face the problem considering a fixed number of subspaces and interpreting the learning process as the way of assigning consequents vectors—i.e., labels of the output fuzzy partitions—to these subspaces with respect to an optimality criterion (i.e., following the COR methodology).

Therefore, according to Fig. 2, each node $S_h \in S^+$ is assigned to each candidate consequent $(B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$ and to the special symbol "don't care" ($R_\emptyset$) that stands for absence of rules in such a subspace.

2) Heuristic Information: The heuristic information on the potential preference of selecting a specific consequent vector, B^{k_h}, in each antecedent combination (subspace) is determined as described in Fig. 3.

For each subspace $S_h \in S^+$ do:

1) Build the sets $E^+(S_h)$ and $C(S_h)$ as shown in Fig. 2.
2) For each $B^{k_h} = (B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$, make use of an initialization function based on a covering criterion to give a heuristic preference degree to each choice. In this paper, we use the following:

$$\eta_{hk_h} = \max_{e_l \in E^+(S_h)} Min\left(\mu_{A^h}(x^l), \mu_{B_j^{k_h}}(y^l)\right). \quad (10)$$

3) For each $B^{k_h} \notin C(S_h)$, make $\eta_{hk_h} = 0$.
4) Finally, for the "don't care" symbol, make the following:

$$\eta_{h,|\mathcal{B}_1|\cdot\ldots\cdot|\mathcal{B}_m|+1} = \frac{1}{\max\limits_{k_h \in \{1,\ldots,|C(S_h)|\}} \eta_{hk_h}}. \quad (11)$$

Fig. 3. Heuristic assignment process

3) Pheromone Initialization: The initial pheromone value of each assignment is obtained as follows:

$$\tau_0 = \frac{1}{|S^+|} \sum_{S_h \in S^+} \max_{B^{k_h} \in C(S_h)} \eta_{hk_h}. \quad (12)$$

In this way, the initial pheromone will be the mean value of the path constructed taking the best consequent in each rule according to the heuristic information (a greedy assignment).

4) Fitness Function: The fitness function will be the said objective function, defined in eq. (9) in Fig. 2.

5) Ant Colony Optimization Scheme: Best-Worst Ant System Algorithm: Once the previous components have been defined, an ACO algorithm has to be given to solve the problem. In this contribution, the BWAS algorithm [10] is considered. Its global scheme is shown in Fig. 4.

IV. Results

The learned controller has been tested in four simulated environments using the Nomad 200 simulation software. These environments include very different situations that the robot usually faces during navigation: straight walls of different

1) Give an initial pheromone value, τ_0, to each edge.
2) While (*termination_condition* is not satisfied) do:
   a) Perform the track of each ant by the **solution construction process**.
   b) Apply the **pheromone evaporation mechanism**.
   c) Apply the **local search process** on the current-best solution.
   d) Update $S_{global\ best}$ and $S_{current\ worst}$.
   e) Apply the **Best-Worst pheromone trail update rule**.
   f) Apply the **pheromone trail mutation**.
   g) If (*stuck_condition* is satisfied) then apply **restart**.

Fig. 4. BWAS algorithm

lengths, followed and/or preceded of a number of concave and convex corners, ... thus covering a wide range of contours to follow and truly defining very complex test environments. It is important to remark that these environments have not been used during training. The training set is only composed of a list of examples that have been chosen covering the input space with an adequate precision. These conditions warrantee that the quality of the learned behavior does not depend on the environment, and also that the robot will be capable to face any situation.

Figure 5 shows the robot path along one of the environments (named *D*) used for testing. The robot trajectory is represented by circular marks. A higher concentration of marks indicates lower velocity. The learned controller has 77 linguistic rules and has been learned in less than 20 minutes. If for any situation no rule is fired, then a null linear acceleration and angular velocity are selected. 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 analyzed environments. The average values measured for some parameters that reflect the controller performance are shown in Table I. These parameters are the average distance to the right wall (the wall that is being followed), the average linear velocity, the time spent by the robot along the path, and the average velocity change. The latter parameter measures the change in the linear velocity between two consecutive cycles, reflecting the smoothness of the behavior.

In order to show the quality of the controller we are going to describe in detail the path of the robot in environment *D* (Fig. 5). This environment is quite complex, with four concave corners and six convex corners in a circuit of a length of 59 meters. 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 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

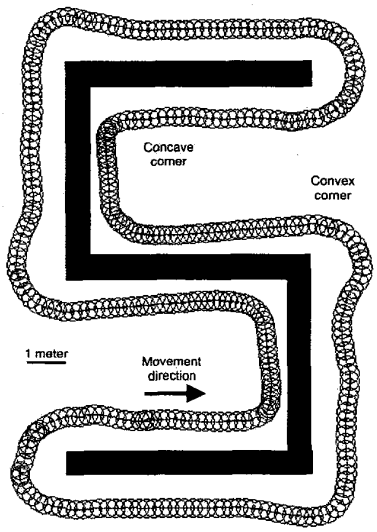

Fig. 5. Path of the robot along environment *D*.

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.

TABLE I
AVERAGE VALUES OF SOME PARAMETERS (77 LINGUISTIC RULES)

Env.	RD (cm)	Velocity (cm/s)	Vel. change (cm/s)	Time (s)
A	65	55	5.73	62
B	57	52	5.75	103
C	55	50	4.10	84
D	55	54	4.27	112

For evaluating the obtained controller, a comparison with another wall-following controller designed using genetic algorithms [3] has been done. Strict comparison of the results obtained with other control systems described in the literature is not possible, because it is not viable to propose a standard *test bank* which would allow us to compare controllers in similar environments, and to consider the different mechanical and dynamic features of the different robots. The design in [3] comprised two stages: learning of the data base and a general rule base (Pittsburgh approach), and reduction of the generated rule base merging adjacent membership functions. This reduction provokes a loss in the interpretability of the final knowledge base since a different fuzzy set is built for each fuzzy rule, thus losing the legibility provided by the use of linguistic variables with global semantics. That controller has 46 fuzzy rules, and the average values of some parameters for the test environments are presented in Table II. The time employed to learn the system is very high (several hours) compared with less than 20 minutes for the controller presented in this paper.

The controller described in this paper has a high number

TABLE II
AVERAGE VALUES OF SOME PARAMETERS FOR THE SYSTEM DESCRIBED IN [3] (46 FUZZY RULES)

Env.	RD (cm)	Velocity (cm/s)	Vel. change (cm/s)	Time (s)
A	59	46	8.69	72
B	54	43	9.70	120
C	54	36	7.75	114
D	54	46	7.18	127

of rules (77 vs. 46), but clearly improves the results of the controller presented in [3]. The average velocity is higher in all the environments, reducing the time spent by the robot in the circuit. The average right-hand distance is very similar in both controllers and, finally, the average velocity change is drastically reduced in the present controller, which reflects the quality of the obtained behavior as compared with [3]. On the other hand, the interpretability of the obtained rules is very high, as opposed to [3]. This makes easier to understand the actions taken by the robot. The controller selects a strong braking when, due to the orientation and velocity, the robot is going to be in the next iteration very close to a wall, or quite far from the wall it must follow. A medium or hard acceleration is selected when the following position of the robot is going to have a good right-hand distance. In resume, the objective is firstly to place the robot to an adequate distance from the right-hand wall, then to select a high velocity if possible, and finally to orient it parallel to the wall.

V. CONCLUSIONS

We have presented the design of a fuzzy controller for the wall-following behavior in mobile robotics using the COR methodology. The main advantages of the proposed approach are the easiness in the design, the speed in the obtaining of the knowledge base, the high degree of interpretability of that knowledge base and, finally, that the quality of the learned behavior does not depend on the training set.

The controller has been tested in a number of simulated environments showing good results in the average values of some parameters that reflect the quality of the behavior. The system has also been compared with a previous design based on genetic algorithms, increasing the quality and the interpretability of it.

APPENDIX A: ANT COLONY OPTIMIZATION

ACO algorithms constitute a new family of global search bio-inspired algorithms that has been recently proposed. Since the first proposal, the ant system algorithm [9]—applied to the traveling salesman problem—, numerous models have been developed to solve a wide set of optimization problems.

ACO algorithms draw inspiration from the social behavior of ants to provide food to the colony. In the food search process, ants deposit a substance called *pheromone*. Ants have the ability of smelling the pheromone. When an ant is located at a branch, it decides to take the path according to a probability defined by the amount of pheromone existing in each trail. In this way, the depositions of pheromone terminate in constructing a path between the nest and the food that can be followed by new ants. The shortest paths are finally the more frequently visited ones and, therefore, the pheromone concentration is higher on them.

The basic operation mode of ACO algorithms is as follows [9]: at each iteration, a population of a specific number of ants progressively construct different tracks on a graph representing the problem instance (i.e., solutions to the problem) according to a *probabilistic transition rule* that depends on the available information (heuristic information and pheromone trails). After that, the pheromone trails are updated by firstly decreasing them by some constant factor (corresponding to the evaporation of the pheromone) and then reinforcing the attributes of the constructed solutions considering their quality. This task is developed by the *global pheromone trail update rule*. Several extensions to this basic operation mode (different transition and update rules, new components, local search...) have been proposed.

ACKNOWLEDGMENT

This work was supported in part by the Spanish Ministry of Science and Technology under grants no. TIC2002-04036-C05-01, TIC2003-00877 and TIC2003-09400-C04-03. Part of this research has been developed during a research stay of the first author at the Department of Computer Science and Artificial Intelligence of the University of Granada that was supported by the *Dirección Xeral de I+D, Xunta de Galicia.*

REFERENCES

[1] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Computing*, vol. 1, no. 4, pp. 180–197, 1997.

[2] M. Mucientes, R. Iglesias, C. V. Regueiro, A. Bugarín, and S. Barro, "A fuzzy temporal rule-based velocity controller for mobile robotics," *Fuzzy Sets Sys.*, vol. 134, pp. 83–99, 2003.

[3] M. Mucientes, D. L. Moreno, C. V. Regueiro, A. Bugarín, and S. Barro, "Design of a fuzzy controller for the wall-following behavior in mobile robotics with evolutionary algorithms," in *Proc. Int. Conf. of Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'2004)*, 2004. Accepted.

[4] D. Floreano and F. Mondada, "Evolutionary neurocontrollers for autonomous mobile robots," *Neural Networks*, vol. 11, pp. 1461–1478, 1998.

[5] J. Casillas, O. Cordón, and F. Herrera, "COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules," *IEEE Trans. Sys., Man, and Cybern.—Part B: Cybern.*, vol. 32, no. 4, pp. 526–537, 2002.

[6] J. Casillas, O. Cordón, and F. Herrera, "COR methodology: a simple way to obtain linguistic fuzzy models with good interpretability and accuracy," in *Accuracy improvements in linguistic fuzzy modeling*, J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds. Springer, Heidelberg, Germany, 2003.

[7] L.-X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Sys., Man, and Cyber.*, vol. 22, no. 6, pp. 1414–1427, 1992.

[8] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. on Genetic Algorithms*, R.K. Belew and L.B. Booker, Eds., San Mateo, CA, USA, 1991, pp. 509–513, Morgan Kaufmann Publishers.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Trans. Sys., Man, and Cybern.—Part B: Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[10] O. Cordón, F. Herrera, I. Fernández de Viana, and L. Moreno, "A new ACO model integrating evolutionary computation concepts: the best-worst ant system," in *Proc. 2nd International Workshop on Ant Algorithms*, Brussels, Belgium, 2000, pp. 22–29.