Iterative Rule Learning of Quantified Fuzzy Rules for Control in Mobile Robotics

Ismael Rodríguez-Fdez, Manuel Mucientes, Alberto Bugarín
Dep. Electronics and Computer Science
University of Santiago de Compostela
{ismael.rodriguez, manuel.mucientes, alberto.bugarin.diz} @usc.es

***Abstract*—Learning controllers in mobile robotics usually requires expert knowledge to define the input variables. However, these definitions could be obtained within the algorithm that generates the controller. This cannot be done using conventional fuzzy propositions, as the expressiveness that is necessary to summarize tens or hundreds of input variables in a proposition is high. In this paper the Quantified Fuzzy Rules (QFRs) model has been used to transform low-level input variables into high-level input variables, which are more appropriate inputs to learn a controller. The algorithm that learns QFRs is based on the Iterative Rule Learning approach. The algorithm has been tested learning a controller in mobile robotics and using several complex simulated environments. Results show a good performance of our proposal, which has been compared with another three approaches.**

***Index Terms*—mobile robotics, Quantified Fuzzy Rules, Iterative Rule Learning.**

I. Introduction

The selection of the set of actions that a mobile robot must take in order to complete a prefixed task, depends not only on its state, but also on its situation in the environment. Thus, an augmented state, which comprises both the state of the robot and the description of the environment, must be considered in order to select the most appropriate action. Data describing the environment is acquired with the sensors of the robot. These data must be transformed in order to be used by the control system that implements the task. This mapping between the augmented state of the robot (including the sensors data) and the actions depends on the task.

This problem is usually solved with the translation of the sensors data (or low-level input variables) into high-level data, which includes all the information that is relevant to solve the task. This translation of data is done using expert knowledge. Our proposal in this paper is to present a method for learning a controller starting from the low-level input variables (sensors raw data), and without using expert knowledge in the transformation of the variables. This means that the mapping between low and high-level input variables is done automatically during the learning phase of the controller.

This kind of learning requires a fuzzy rules model that is able to represent information from sets of low-level input variables. This information is not just an average value, a maximum or a minimum value. More expressive and meaningful information is required, i.e., the propositions must summarize data using expressions like *"most of the variables in the set take a high value"*. These propositions are referred to as Quantified Fuzzy Propositions (QFPs) [1].

The learning of knowledge bases of Quantified Fuzzy Rules (QFRs) requires the use of an evolutionary algorithm with the ability to cope with individuals with different structures defined through a context-free grammar. The reason is that a fuzzy rule can contain both conventional and quantified propositions (with different structures). Moreover, grouping low-level variables into high-level input variables makes the number of propositions in the antecedent of the rules very variable. Therefore, genetic programming, where each chromosome of the population is represented as a tree of variable length, is the most appropriate choice.

In this paper we propose a genetic based approach to learn Quantified Fuzzy Rules (QFRs), defined as fuzzy rules of variable structure involving QFPs. The algorithm is based on the genetic programming paradigm and has been designed to solve regression problems in mobile robotics having as input variables the state of the robot and the sensors data, but without using expert knowledge for transforming these data. The algorithm is based on the Iterative Rule Learning (IRL) approach.

The main contributions of the paper are: (i) the proposed algorithm is able to learn with the state of the robot and the sensors data, without any preprocessing; (ii) the algorithm uses QFPs, a model able to represent information in a more expressive way. Furthermore, the proposal presented in this paper is able to learn rules involving linguistic labels with multiple granularity. On the contrary, previous approaches [2] were based on the Genetic Cooperative-Competitive Learning (GCCL) approach, and the rules involved approximative labels.

The paper is structured as follows: Sec. II presents the QFRs model and its advantages in mobile robotics, while Sec. III describes the algorithm that has been used to learn the QFRs. Sec. IV presents some results and, finally, Sec. V points out the conclusions.

II. Quantified Fuzzy Rules (QFRs)

A. Motivation for Mobile Robotics

Learning of a controller in mobile robotics requires the definition of the input variables. These variables usually need to be obtained through a preprocessing of the sensors data. For example, in [3], [4] two successful approaches to learn fuzzy rules for the control of a robot in two different tasks were described. In both cases, the learned rules were conventional fuzzy rules and all the input variables were defined by a human expert. For the wall following task these variables were *right distance*, *left distance*, *orientation*, and *velocity*. For the moving object tracking task the variables were *distance to the object*, *deviation*, *difference in velocity with the object* and *difference in angle*.

In general, two categories can be established for the input variables:

- High-level input variables: variables that provide, by themselves, information that is relevant and meaningful to the expert for modeling the system (e.g. the linear velocity of the robot or the right-hand distance from the robot to a wall).
- Low-level input variables: variables for which their individual values do not contribute to model the system. Their importance stems from the analysis of sets of these variables (e.g. the distance measured by a single beam of a laser range finder).

However, the values of a group of low level variables can provide valuable and meaningful information. For example, the "frontal sector distance" of a laser range finder is a high-level variable made up of a set of distances of single beams (low-level variables). Within this context Quantified Fuzzy Propositions (QFPs) such as *"some of the distances of the frontal sector are low"* are useful for representing relevant knowledge for the experts and therefore for performing intelligent control. Modeling using QFPs as in this example demands the definition of several elements:

- *Some*: how many distances of the frontal sector must be low?
- *frontal sector*: which beams belong to the frontal sector?
- *low*: what is the actual semantics of low?

This example clearly sets out the need to use propositions different from the conventional ones. In this paper QFPs (as "X is A in Q of S") are used for representing knowledge about high-level variables that are defined as the grouping of low-level variables. Conventional fuzzy propositions ("X is A") are used to represent other high-level variables, non related to low-level ones.

B. QFRs Model

An example of QFR is shown in Fig. 1, and involves both QFPs (Eq. 1) and conventional ones (Eq. 2):

$$\text{IF } d(h) \ \textit{is HIGH in most of } F^1_{\textit{sector}} \ \text{ and} \tag{1}$$

$$\ldots$$

$$\textit{velocity is } F_{\textit{vel}} \tag{2}$$

$$\text{THEN } \textit{linear velocity is } F_{lv} \ \text{ and } \ \textit{angular velocity is } F_{av}$$

Fig. 1. A typical QFR to model the behavior of a mobile robot.

The general expression for QFPs in our case is:

$$d(h) \ \textit{is } F^i_d \textit{ in } Q^i \textit{ of } F^i_{\textit{sector}} \tag{3}$$

where, for each i=1,...,g^{max}_b (g^{max}_b being the maximum possible number of sectors of distances):

- $d(h)$ is the signal. In this particular case, it represents the distance measured by beam h.
- F^i_d is a linguistic value for variable $d(h)$.
- Q^i is a (spatial, defined in the laser beam domain) fuzzy quantifier.
- $F^i_{\textit{sector}}$ is a fuzzy set in the laser beam domain (e.g., the "frontal sector").

Evaluation of the Degree of Fulfillment (DOF) for Eq. 3 is carried out using Zadeh's quantification model for proportional quantifiers (such as "most of", "part of", ...) [5], that allows to consider non-persistence, partial persistence and total persistence situations for event "$d(h)$ is F^i_d" in the range of laser beams (spatial interval $F^i_{\textit{sector}}$). This is a relevant feature of this model, since it allows to consider partial, single or total fulfillment of an event within the laser beams set.

Automatic learning of QFRs for this application (Fig. 1) demands an algorithm with the ability to represent rules with different structures, as the number of analyzed sectors of distances, and therefore the number of QFPs per rule (Fig. 1) can change among rules. Moreover, the number of elements involved in the definition of a proposition is higher in QFP than in conventional propositions (3 vs. 1).

III. QFRs Learning Algorithm for Robotics

A. Evolutionary Learning of Knowledge Bases

According to [6], [7], evolutionary learning of knowledge bases has different approaches to represent the solution to the problem: Pittsburgh, Michigan, IRL [8], and GCCL [9]. We focus the discussion on those approaches for which an individual represents a rule, discarding the Michigan approach as it includes reinforcement learning techniques in order to obtain the fitness. Therefore, we analyze the IRL and GCCL approaches.

In the IRL approach, a single rule (and not the whole rule base) is learned by the evolutionary algorithm. 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. A common way to penalize the obtained rules is to delete the training examples that have been covered by the set of rules in 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 among rules is not taken into account when a rule is evaluated. For example, a new rule could be added to the final rule base, deteriorating the behavior of the rule base over a set of examples that were already covered. The cooperation among rules can be improved with a posterior selection algorithm.

In the GCCL approach rules evolve together but competing among them to obtain the higher fitness. For this type of algorithm it is fundamental to include a mechanism to maintain the diversity of the population (niche induction). The mechanism must warrant that individuals of the same niche compete among themselves, but also has to avoid deleting those weak individuals that occupy a niche not covered by other individuals of the population. This can be usually done using token competition. Although this approach works well for classification problems [1], the same does not occur for regression problems [2]. The main drawback is the balance between cooperation and competition, which is difficult to adjust in regression problems. It is frequent that an individual tries to capture examples seized by other individual, improving the performance on many of them, but decreasing the accuracy over a few ones. In the subsequent iterations, new and more specific individuals replace the rule that was weakened. As a result, the individuals improve their fitness, but the performance of the knowledge base does not increase. In particular, for mobile robotics, the obtained knowledge bases over-fit the training data due to a *polarization* effect of the rule base: a few very general rules and many very specific rules. Moreover, many times, the errors of the individual rules compensate each other generating a good output of the rule base but, of course, only in the training phase.

Our proposal, called IQFRL (Iterative Quantified Fuzzy Rule Learning, fig. 2), is based on IRL. The learning process is divided in epochs (set of iterations), and at the end of each epoch a new rule is obtained. The following sections describe each of the stages of the algorithm.

```
1: $KB_{cur} := \varnothing$
2: repeat
3:     $it := 0$
4:     $equal_{ind} := 0$
5:     Initialization
6:     Evaluation
7:     repeat
8:         Selection
9:         Crossover and Mutation
10:        Evaluation
11:        Replacement
12:        if $best^{i-1}_{individual} = best^{i}_{individual}$ then
13:            $equal_{ind} := equal_{ind} + 1$
14:        else
15:            $equal_{ind} := 0$
16:        end if
17:        $it := it + 1$
18:    until $(it < it_{min} \vee equal_{ind} < it_{check}) \wedge (it < it_{max})$
19:    $KB_{cur} := KB_{cur} \cup best_{ind}$
20:    $uncov_{ex} := uncov_{ex} - cov_{ex}$
21: until $uncov_{ex} = \varnothing$
```

Fig. 2. IQFRL algorithm.

B. Examples and Grammar

The learning process is based on a set of training examples. In mobile robotics, each example can be composed of some variables that define the state of the robot (position, orientation, linear and angular velocity, etc.), and the data measured by the sensors. If the robot is equipped with laser range finders, the sensors data is a vector of distances. A laser range finder provides the distances to the closest obstacle in each direction with a given angular resolution (number of degrees between two consecutive beams). In this paper, each example e^l is represented by a tuple:

$$e^l = (d(1), \ldots, d(N_b), \textit{velocity}, \textit{vlin}, \textit{vang}) \quad (4)$$

where $d(h)$ is the distance measured by beam h, N_b is the number of beams, *velocity* is the measured linear velocity of the robot, and *vlin* and *vang* are the output variables (velocity control commands for the linear and angular velocities).

The structure of the individuals of the population can be very different among them: some propositions are conventional and others are QFPs, and the number of inputs is variable. Therefore, genetic programming is the most appropriate approach, as each individual is a tree of variable size. In order to generate valid individuals of the population, and to produce right structures for the individuals after crossover and mutation, some constraints have to be added. With a context-free grammar all the valid structures of a tree (genotype) in the population can be defined in a compact form. A context-free grammar is a quadruple (V, Σ, P, S), where V is a finite set of variables, Σ is a finite set of terminal symbols, P is a finite set of rules or productions, and S is an element of V called the start variable.

The grammar is described in Fig. 3. The first item enumerates the variables, then the terminal symbols, in third place the start variable is defined, and finally the rules for each variable are enumerated. When a variable has more than one rule, rules are separated by symbol |. Fig. 4 represents a typical chromosome generated with this context-free grammar. Terminal symbols (leafs of the tree) are represented by ellipses, and variables are shown as rectangles. There are two different types of antecedents:

- The sector antecedent. Consecutive beams are grouped in sectors in order to generate more general (high-level) variables (frontal distance, right distance, etc.). This type of antecedent is defined by the terminal symbols F_d, F_b, Q, that are the linguistic labels for the measured distances ($HIGH$ in Fig. 1, prop. 1), the definition of the sector, i.e., which beams belong to the sector (F^1_{sector} in Fig. 1, prop. 1) and the quantifier ($most$ in Fig. 1, prop. 1) respectively.
- The measured linear velocity of the antecedent is defined by the F_v linguistic label.

Finally, F_{lv} and F_{av} are the linguistic labels of the linear and angular velocity control commands respectively, which are the consequents of the rule.

The linguistic labels of the antecedent (F_v, F_d, F_b) are defined using a multiple granularity approach. The universe of discourse of a variable is divided into a different number of labels for each granularity. Specifically, a granularity g^i_{var}

- V = { rule, antecedent, consequent, sector }
- Σ = { F_{lv}, F_{av}, F_v, F_d, F_b, Q }
- S = rule
- P:
  - rule $\longrightarrow$ antecedent consequent
  - antecedent $\longrightarrow$ sector F_v | sector
  - consequent $\longrightarrow$ F_{lv} F_{av}
  - sector $\longrightarrow$ F_d Q F_b sector | F_d Q F_b

Fig. 3. Context-free grammar.

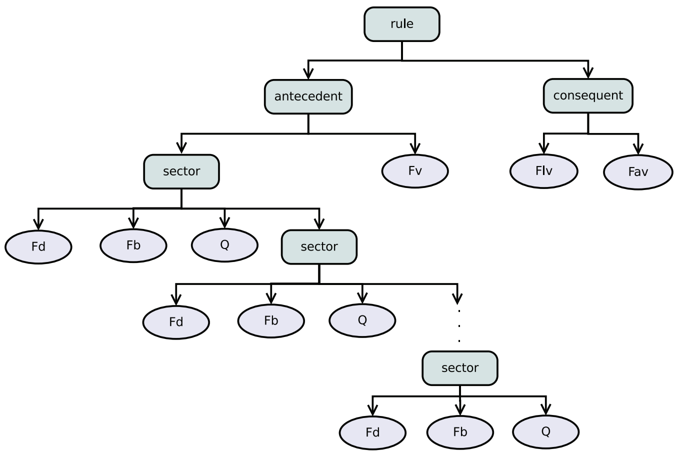


Fig. 4. A genotype representing a QFR to model the behavior of a robot.

divides the variable var in i uniformly spaced labels, i.e., $A^i_{var} = \{A^{i,1}_{var}, ..., A^{i,i}_{var}\}$. Fig. 5 shows a partitioning up to granularity five. g^{min}_{var} is the minimum granularity defined for the variable var and g^{max}_{var} is the maximum one. On the other hand, the linguistic labels of the consequents (F_{lv}, F_{av}) are defined by a single granularity approach.

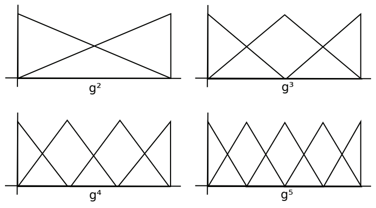


Fig. 5. Multiple granularity approach.

C. Initialization

A genotype (Fig. 4) is generated for each example in the training set. All the fuzzy sets of conventional fuzzy propositions (F_v) and those of the consequent part (F_{lv} and F_{av}) are initialized as $F_{var} = A^{g^{max}_{var},\beta}_{var}$ where $\beta = argmax_j \; \mu^{g^{max}_{var},j}_{var}\left(e^l\right)$, i.e., the label with the largest membership value for the example.

The initialization of the quantified propositions (sectors) is quite more complex. In the first place, the number of sectors that are going to be included in an individual is the number of labels for sector definition in the partition with maximum granularity (g^{max}_b). Afterwards, for each sector (F_b), the other components (F_d and Q), have to be calculated with the following steps:

1) The sector is divided into groups of consecutive laser beams whose deviation does not exceed a certain threshold (σ_{bd}).
2) The group with the largest number of beams is chosen and its average distance ($\bar{d}$) is calculated.
3) The distance is initialized as $F_d = A^{g^{max}_d,\beta}_{var}$ where $\beta = argmax_j \left(\mu^{g^{max}_{var},j}_{var}\left(\bar{d}\right)\right)$, i.e., the label that is closest to the average distance of the sector.
4) Finally, Q is calculated as the percentage of beams of the sector ($h \in F_b$) that fulfill F_d:

$$Q = \frac{\sum_{h \in F_b} min\left(\mu_{F_d}(d(h)),\ \mu_{F_b}(h)\right)}{\sum_{h \in F_b} \mu_{F_b}(h)} \tag{5}$$

D. Evaluation

The fitness of an individual of the population is calculated as follows. Firstly, it is necessary to estimate the probability that an example e^l matches the output associated to rule (individual) j (C_j):

$$P\left(C_j|e^l\right) = \exp\left(-\frac{\mathit{error}^l_j}{\mathit{ME}}\right) \tag{6}$$

where error^l_j is the difference between output C_j and the output codified in the example, and *ME* is a parameter that defines the meaningful error for the application. error^l_j can be defined as:

$$error^l_j = \sum_k \left(\frac{y^l_k - c_{j,k}}{max_k - min_k}\right)^2 \tag{7}$$

where y^l_k is the value of the k-th output variable of example e^l, $c_{j,k}$ is the output of the k-th output variable associated to individual j, and max_k and min_k are the maximum and minimum values of output variable k. In regression problems, an example can have several outputs that are different from the one codified in the example, but that produce small errors, i.e. that are very similar to the desired output. Thus, $P\left(C_j|e^l\right)$ can be interpreted as a normal distribution with covariance ME and $error^l_j$ is the square of the difference between the mean (output codified in the example) and the output value proposed in the rule codified by the individual.

In an IRL approach, $C_j = C_{R_j}$, i.e., the output associated to rule j is the output codified in individual j. The fitness of an individual in the population is calculated as the combination of two variables. On one hand, the accuracy with which the individual covers the examples, called confidence. On the other hand, the ability of generalization of the rule, called support. The confidence can be defined as:

$$confidence = \frac{quality_u}{\sum_l DOF_{R_j}(e^l_u)} \tag{8}$$

where $DOF_{R_j}(e^l_u)$ is the degree of fulfillment of e^l_u for rule j, e^l_u is an example that fulfills $e^l \in uncov_{ex}$ and $uncov_{ex}$ is defined as:

$$uncov_{ex} = \{e^l \ : \ DOF_{KB_{cur}}(e^l) < DOF_{min}\} \tag{9}$$

i.e., the set of examples that are covered with a degree of fulfillment below DOF_{min} by the current final knowledge base (line 20, fig. 2), and $quality_u$ can be defined as:

$$quality_u = \sum_l DOF_{R_j}(e_u^l) \ : \ P\left(C_{R_j}|e_u^i\right) > P_{min} \\ and\ DOF_{R_j}(e_u^l) > DOF_{min} \tag{10}$$

where P_{min} is the minimum admissible quality. Therefore, the higher the accuracy over the covered examples, the higher the confidence. Support is calculated as:

$$support = \frac{quality_u}{\#uncov_{ex}} \tag{11}$$

Thus, the support measures the percentage of examples that are covered with quality related to the total number of uncovered examples. Finally, we define $fitness$ as a linear combination of both values:

$$fitness = \alpha_f \cdot confidence + (1 - \alpha_f) \cdot support \tag{12}$$

which represents the strength of an individual over the uncovered examples. $\alpha_f \in [0,\ 1]$ is a parameter that codifies the trade-off between accuracy and generalization of the rule.

E. Crossover

The pairs of individuals that are going to be crossed are selected following a probability distribution defined as:

$$P_{close}(i,j) = 1 - \frac{\sum_k^{N_c}(\frac{c_{i,k} - c_{j,k}}{max_k - min_k})^2}{N_c} \tag{13}$$

where $c_{i,k}$ ($c_{j,k}$) is the value of the k-th output variable of individual i (j), and N_c is the number of consequents. With this probability distribution, the algorithm selects with higher probability mates that have similar consequents. The objective is to extract information on which propositions of the antecedent parts of the rules are important, and which are not.

The crossover operator only modifies one proposition of the antecedent part of the rule. As individuals have a variable number of antecedents, the total number of propositions could be different for two individuals. Moreover, the propositions can be defined using different granularities. Therefore, the first step is to select the propositions to be crossed in both individuals as follows:

1) Randomly select an antecedent $m \in [1, N_a = g_b^{max} + 1]$ where N_a is the maximum number of antecedents plus one, due to the velocity proposition.
2) Check the existence of this antecedent in both individuals, according to the following criteria:
    a) If the antecedent m is a sector, then calculate for each proposition of each individual the overlap between $A_b^{g_b^{max},m}$ (linguistic label that defines sector m) and F_b (the definition of the sector for the proposition). Then, select for each individual the proposition with higher overlap.
    b) If the antecedent m is the velocity, then the corresponding proposition is F_v (in case it exists).

TABLE I
CROSSOVER OPERATIONS

Individual 1	Individual 2	Action
no	no	try another antecedent
no	yes	insert antecedent 2 in individual 1
yes	no	delete antecedent
yes	yes	combine antecedents

Once the propositions used to generate a new individual (the process will be repeated to generate the second new individual) are selected, an operation must be picked depending on the existence of that antecedent in both individuals (table I):

- If the antecedent does not exist in both individuals, this means that the proposition is not relevant and the process to select a new antecedent is repeated.
- If the antecedent does not exist in the first individual but it exists in the second one, then the proposition of the second individual is copied to the first one, as this proposition could be meaningful.
- If the situation is the opposite to the previous one, then the proposition of the first individual is deleted, as it could not be important.
- If the antecedent exists in both individuals, then both propositions are combined in order to obtain a proposition that generalizes both antecedents.

If the antecedents need to be combined, then the degree of overlap between them indicates which action is executed. If the antecedent is of type sector, the overlap takes into account both F_b and F_d labels. Only when both overlaps are partial, the antecedents are merged:

- If the overlap is null, then the antecedents correspond to different situations. For example, the definition of the sectors is the same, but the distances label is *high* for one and *low* for the other. This means that the proposition of the first individual does not contain meaningful information and it can be deleted to generalize the rule.
- If the overlap is total, then, in order to obtain a new individual with different antecedents, the proposition is eliminated.
- Finally, if the overlap is partial, then the propositions are merged in order to obtain a new one that combines the information provided by the two original propositions. Thus, the individual is generalized. The merge action is defined as the process of finding the label with the highest possible granularity that overlaps with the labels of both original propositions. This is done for both F_b and F_d labels. Q will be calculated as the minimum Q of the two individuals.

F. Mutation

When crossover is not performed, both individuals are mutated. Mutation implements two different strategies: generalize or specialize a rule. For generalization, the following steps are performed:

1) Select an example $e^{sel} \in uncov^j_{ex}$, where $uncov^j_{ex}$ is the set of examples that belong to $uncov_{ex}$ and are not covered by individual j. The example is selected with a probability distribution given by $P\left(C_{R_j}|e^l_u\right)$. The higher the similarity between the output of the example and the consequent of rule j, the higher the probability to be selected.
2) The individual is modified in order to cover e^{sel}. Therefore, all the propositions with $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$ are selected for mutation. These are the propositions that are not covering the example.
   a) For sector propositions, there are three different ways in which the proposition can be modified: F_d, F_b, and Q. F_d and F_b are generalized, choosing the most similar label in the adjacent partition with lower granularity. The process is repeated until $\mu_{prop}\left(e^{sel}\right) > DOF_{min}$. On the other hand, Q is decreased until $\mu_{prop}\left(e^{sel}\right) > DOF_{min}$. The mutation is selected among the three possibilities, with a probability proportional to $\mu_{prop}\left(e^{sel}\right)$.
   b) For velocity propositions, generalization is done choosing the most similar label in the adjacent partition with lower granularity until $\mu_{prop}\left(e^{sel}\right) > DOF_{min}$.

For specialization, the process is equivalent:

1) Select an example $e^{sel} \in cov^j_{ex}$, where cov^j_{ex} is the set of examples that belong to $uncov_{ex}$ and are covered by individual j. The example is selected with a probability distribution that is inversely proportional to $P\left(C_{R_j}|e^l_u\right)$. The higher the similarity between the output of the example and the consequent of rule j, the lower the probability of being selected.
2) Only one proposition needs to be modified to specialize, and it is selected randomly.
   a) For sector propositions, there are again three different ways in which the proposition can be modified: F_d, F_b, and Q. F_d and F_b are specialized, choosing the most similar label in the adjacent partition with higher granularity. The process is repeated until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$. On the other hand, Q is increased until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$. The mutation is selected among the three possibilities, with a probability that is inversely proportional to $\mu_{prop}\left(e^{sel}\right)$.
   b) For velocity propositions, specialization is done by choosing the most similar label in the adjacent partition with higher granularity until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$.

Finally, after the mutation operation (generalization or specialization) of the antecedent is performed, the consequent is mutated. Again, this mutation requires the selection of an example. If generalization was selected for the mutation of the antecedent, then the example will be e^{sel}. On the other hand, for specialization an example is randomly selected from those currently in cov^j_{ex}. For each variable in the consequent, the label of the individual is modified following a probability distribution (Fig. 6). Thus, the labels closer to the label of the individual have a higher probability, while the labels closer to the example label have a lower one.

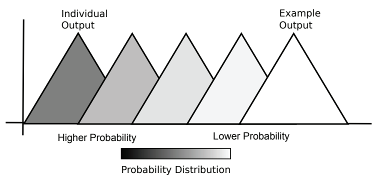


Fig. 6. Probability distribution for output mutation.

G. Selection and Replacement

Selection has been implemented with a binary tournament selection. On the other hand, replacement follows an steady-state approach. The new individuals and those of the previous population are joined, and the best $pop_{maxSize}$ individuals are selected for the final population.

H. Epoch Loop

An epoch is a set of iterations at the end of which a new rule is added to *KB*$_{cur}$. The stopping criterion of each epoch (inner loop in fig. 2) is the number of iterations, but this limit varies according to the following criteria: once the number of iterations reaches it_{min}, the algorithm stops if there are it_{check} consecutive iterations with no change in $best_{ind}$. If the number of iterations reaches it_{max} iterations, then the algorithm stops regardless of the previous condition.

When the epoch ends, the rule defined in $best_{ind}$ is added to *KB*$_{cur}$. Moreover, the examples that are covered with quality (according with Eq. 10) are marked as covered by the algorithm (line 20, fig. 2). Finally, the algorithm stops when either no uncovered examples remain or no example is covered by the individual learned in the last epoch.

I. Rule Subset Selection

After the end of the iterative part of the algorithm, the performance of the obtained rule base can be improved selecting a subset of rules with better cooperation among them. The rules selection algorithm described in [1] has been used. It is based on the Iterated Local Search (ILS) algorithm [10].

IV. Results

The proposed algorithm has been validated with a well-known behavior in mobile robotics: wall-following. The main objectives of a controller for this behavior are: to keep a suitable distance to the wall, to move at the highest possible velocity, and to implement smooth control actions.

The examples that have been used for learning were generated for three different situations that have been identified by an expert:

1) Convex corner: is a situation in which the robot finds a wall in front of it.
2) Concave corner: is characterized by the existence of a gap in the wall (like an open door).
3) Straight wall: any other situation.

For each of the above situations, the robot was placed in different positions and the associated control order was decided by an expert. Therefore, each example consists of 722 distances (one for each laser beam), the current linear velocity of the robot, and the control commands (linear and angular velocity). The expert always tried to follow the wall at, approximately, 50 cm and the maximum values for the linear and angular velocities were 60 cm/s and $45^o s^{-1}$. 468 training examples were selected for the straight wall situation, 1092 for the convex corner and 252 for the concave corner.

The values that have been used for the parameters of the evolutionary algorithm are: $ME = 0.02$, $DOF_{min} = 0.001$, $\alpha_f = 0.9$, $P_{cross} = 0.8$, $pop_{maxSize} = 70$, $it_{min} = 60$, $it_{check} = 10$, $it_{max} = 100$, $\sigma_{bd} = 0.01$ and $P_{min} = 0.17$. The granularities and the universe of discourse of each component of a rule are shown in table II. P_{min} is a parameter that has a high influence in the performance of the system. Only one value of P_{min} has been tried in our tests, and it has been obtained with Eq. 6 when the error for each consequent is of one label (Eq. 7).

TABLE II
GRANULARITIES AND UNIVERSE OF DISCOURSE

Variable	Min	Max	Granularities
Distance	0	1.5	$\{3, 5, 7, 9, 11, 13\}$
Beam	0	721	$\{2, 4, 6, 8\}$
Quantifier	10	100	–
Velocity	0	0.6	$\{2, 3, 4, 5\}$
Lineal velocity	0	0.6	$\{9\}$
Angular velocity	$-\pi/4$	$\pi/4$	$\{19\}$

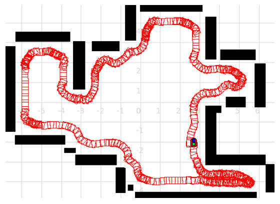
TABLE III
CHARACTERISTICS OF THE ENVIRONMENTS FOR THE WALL-FOLLOWING BEHAVIOR.

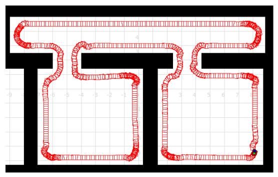
Environment	Dim. $(m \times m)$	Length (m)	#CC	#CX	#doors
1	15×15	70	4	7	1
2	14×10	43	10	6	0
3	19×12	86	12	6	4
4	26×15	146	23	10	8

In order to analyze the performance of the proposed learning algorithm, several tests have been done in four environments. Three of them are shown in Fig. 7. The trace of the robot is represented by marks, and the higher the concentration of marks, the lower the velocity. Table III shows some of the characteristics of the four environments: the dimensions of the environment, the length of the path, the number of concave (#CC) and convex (#CX) corners, and the number of times that the robot has to cross a door (#doors). The action of crossing a door represents a high difficulty as the robot has to negotiate a convex corner with a very close wall in front of it.

Our proposal (IQFRL) has been compared with three different algorithms:

- Cooperative-Iterative Rule Learning (CIRL): it is based on the described IQFRL algorithm and was designed only


(a) *Environment 2*


(b) *Environment 3*

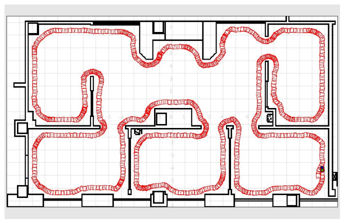

(c) *Environment 4*

Fig. 7. Path of the robot along three of the environments.

for comparison purposes. The only difference between them is that for CIRL $C_j = C_{KB_j}$, with $KB_j = KB_{cur} \cup R_j$. Thus, what is going to be evaluated is the output of a knowledge base composed of the current final knowledge base and the j-th rule of the population, rather than the output of the individual. Moreover, fitness is calculated as the fitness defined for IQFRL (Eq. 12) weighted by the accuracy over the covered examples (as defined in Eq. 8, but over the set of covered examples). In this way, CIRL also takes into account the ability of the rule to cooperate with the current final knowledge base. The values of the parameters used for CIRL are the same as for IQFRL.
- Genetic Cooperative-Competitive Learning (GCCL) [2]: uses the same crossover and mutation operators defined for IQFRL, and the same values for the parameters, except for $pop_{maxSize} = 300$ and the number of iterations was set to 300.
- Different neural networks (NN)[1] were also trained with Backpropagation and Resilient Backpropagation without weights [12] with the following parameters: $threshold = 0.01$, $stepmax = 1e+5$, $rep = 1$, $lRateL = NULL$, $lRateF = (-0.5, +1.2)$ and the error function was the root mean error. As it is not possible to learn with 723 input variables, the following preprocessing was implemented to obtain the inputs to the network:
    1) The distances of the beams are divided into r sectors. Five different number of sectors were used: $r = \{2, 4, 8, 16, 32\}$

[1]The package Neuralnet of the statistical software R [11] was used.

2) The number of hidden neurons was set to r.
3) For each sector, the input distance was calculated as the initialization stage of IQFRL.
4) For each of the three situations (convex and concave corner, and straight wall), two neural networks were trained, one for the linear velocity and another for angular velocity.

TABLE IV
NUMBER OF RULES LEARNED

Alg.	$\#R_{straight}$	$\#R_{convex}$	$\#R_{concave}$	$\#R_{total}$
IQFRL	54	124	31	209
IQFRL+S	42	100	29	171
CIRL	74	121	33	228
CIRL+S	71	119	28	218
GCCL	84	166	46	296
GCCL+S	61	137	43	241

TABLE V
AVERAGE RESULTS ($x \pm \sigma$) FOR DIFFERENT ENVIRONMENTS

Alg.	Env.	Dist.(cm)	Vel.(cm/s)	Vel.ch.(cm/s)	Time(s)
IQFRL	1	72.64 ± 2.70	32.71 ± 0.38	4.37 ± 0.76	206.27 ± 3.35
IQFRL	2	71.38 ± 5.18	24.81 ± 1.72	6.71 ± 0.55	161.13 ± 23.19
IQFRL	3	63.86 ± 4.26	32.44 ± 1.41	4.10 ± 0.77	261.73 ± 8.63
IQFRL	4	66.07 ± 2.62	32.49 ± 0.44	5.60 ± 0.49	433.23 ± 6.88
IQFRL+S	1	48.60 ± 0.82	30.70 ± 0.32	6.65 ± 0.86	208.33 ± 2.57
IQFRL+S	2	54.49 ± 0.31	22.41 ± 0.89	7.38 ± 0.12	178.17 ± 2.14
IQFRL+S	3	51.86 ± 0.29	30.37 ± 0.51	7.26 ± 0.63	270.40 ± 1.80
IQFRL+S	4	53.82 ± 0.52	27.33 ± 0.11	7.11 ± 0.85	511.93 ± 1.66
CIRL	1	57.32 ± 1.39	31.14 ± 0.60	5.04 ± 0.49	213.07 ± 1.63
CIRL	2	60.07 ± 0.56	19.69 ± 0.51	5.98 ± 0.31	215.27 ± 4.24
CIRL	3	56.91 ± 0.39	29.50 ± 0.24	3.89 ± 0.37	285.33 ± 2.10
CIRL	4	59.62 ± 0.26	25.83 ± 0.14	5.33 ± 0.44	563.17 ± 3.64
CIRL+S	1	66.00 ± 0.72	31.74 ± 0.72	5.08 ± 0.81	209.53 ± 1.32
CIRL+S	2	54.49 ± 0.56	22.41 ± 0.31	7.38 ± 0.37	178.16 ± 2.14
CIRL+S	3	65.40 ± 1.40	30.95 ± 0.28	4.91 ± 0.62	267.67 ± 1.43
CIRL+S	4	70.97 ± 1.66	27.55 ± 0.13	6.06 ± 0.58	519.93 ± 1.90
GCCL	1	53.03 ± 0.83	17.75 ± 0.37	2.65 ± 0.18	362.43 ± 5.69
GCCL	2	60.76 ± 0.69	13.32 ± 0.03	3.57 ± 0.62	308.73 ± 9.32
GCCL	3	57.23 ± 0.93	19.14 ± 0.33	2.1 ± 0.12	437.63 ± 6.88
GCCL	4	54.4 ± 0.27	15.12 ± 0.16	3.02 ± 0.23	935.27 ± 10.28
GCCL+S	1	53.48 ± 1.29	18.17 ± 0.31	3.11 ± 0.41	351.50 ± 4.45
GCCL+S	2	61.75 ± 0.71	12.25 ± 0.34	3.65 ± 0.38	324.03 ± 4.90
GCCL+S	3	58.47 ± 0.61	18.37 ± 0.47	2.75 ± 0.27	442.43 ± 6.30
GCCL+S	4	—	—	—	—
NN	—	—	—	—	—

Table IV shows the number of rules learned for the three different situations by each of the methods based on QFRs, and also for these methods with a posterior rule subset selection (+S). The values of the parameters for rules selection [1] were $radius_{nbhood} = 1$, $maxRestarts = 1$. Table V contains the results of each of the four controllers for the different environments. In order to evaluate the quality of the controllers we have measured four different indicators: the right distance (Dist.), the linear velocity (Vel.), the change in the linear velocity between two consecutive cycles (Vel.ch.) —which reflects the smoothness in the control—, and the time. The average values of the indicators are calculated for each lap that the robot performs in the environment. Results presented in the table are the average and standard deviation values over three laps of the average values of the indicators over one lap.

As can be seen, the best controller without rule selection is CIRL, as it obtains the best average distance in two of the environments, and is closer to GCCL in the other two. IQFRL performance is not adequate due to the low cooperation among the rules. On the other hand, none of the neural networks was able to complete a lap in any of the environments, thus showing the advantage of using QFRs. After selection, the performance of IQFRL highly increases due to a better cooperation. The distance to the wall is very close to the reference distance (50 cm) in all the environments, and the average velocity is close to the best one (IQFRL) in all the tests. On the contrary, the performance of CIRL+S and GCCL+S decreases, as the cooperation among rules was taken into account during the learning phase. In summary, the performance of the proposed algorithm with selection (IQFRL+S) is exceptional in all the situations, both for the distance and velocity parameters. Moreover, the number of rules (table IV) is also the lower one.

V. CONCLUSIONS

An algorithm based on IRL to learn controllers in mobile robotics using the state of the robot and the sensors data has been presented. Our proposal does not use explicit preprocessing of the data, as the transformation of the low-level variables into high-level variables is done through the use of QFPs. The algorithm has been tested with the wall-following behavior in four different complex environments. Also, it has been compared with other three proposals, showing a very good performance.

ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Science and Innovation under grant TIN2008-00040. M. Mucientes is supported by the *Ramón y Cajal* program of the Spanish Ministry of Science and Innovation.

REFERENCES

[1] M. Mucientes and A. Bugarín, “People detection through quantified fuzzy temporal rules,” *Pattern Recognition*, vol. 43, pp. 1441–1453, 2010.

[2] M. Mucientes, I. Rodríguez-Fdez, and A. Bugarín, “Evolutionary learning of quantified fuzzy rules for hierarchical grouping of laser sensor data in intelligent control,” in *Proceedings of the IFSA-EUSFLAT 2009 conference*, Lisbon (Portugal), 2009, pp. 1559–1564.

[3] M. Mucientes, J. Alcalá-Fdez, R. Alcalá, and J. Casillas, “A case study for learning behaviors in mobile robotics by evolutionary fuzzy systems,” *Expert Systems With Applications*, vol. 37, pp. 1471–1493, 2010.

[4] M. Mucientes and J. Casillas, “Quick design of fuzzy controllers with good interpretability in mobile robotics,” *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 636–651, 2007.

[5] L. Zadeh, “A computational approach to fuzzy quantifiers in natural languages,” *Computers and Mathematics with Applications*, vol. 9, pp. 149–184, 1983.

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

[7] F. Herrera, “Genetic fuzzy systems: Taxonomy, current research trends and prospects,” *Evolutionary Intelligence*, vol. 1, pp. 27–46, 2008.

[8] O. Cordón and F. Herrera, “Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems,” *Fuzzy sets and systems*, vol. 118, pp. 235–255, 2001.

[9] D. Greene and S. Smith, “Competition-based induction of decision models from examples,” *Machine Learning*, vol. 3, pp. 229–257, 1993.

[10] H. Lourenço, O. Martin, and T. Stützle, *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003, ch. Iterated Local Search, pp. 321–353.

[11] Stefan Fritsch and Frauke Guenther, “neuralnet: Training of neural networks,” http://cran.r-project.org/web/packages/neuralnet/index.html.

[12] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: The RPROP algorithm,” in *Neural Networks, 1993., IEEE International Conference on*. IEEE, 2002, pp. 586–591.