

# Evolutionary Learning of a Fuzzy Control Rule Base for an Autonomous Vehicle

Frank Hoffmann and Gerd Pfister

Institute of Applied Physics  
University of Kiel  
Olshausenstr. 40  
D-24098 Kiel  
hoefi@ang-physik.uni-kiel.de

## Abstract

This paper presents a hybrid learning method in which fuzzy logic controllers (FLC) are automatically designed by means of a genetic algorithm (GA). A messy coding scheme is proposed which allows a compact and flexible representation of the fuzzy rules in a genetic string. The complexity and size of the rule base are reduced which enables the GA to solve the design task even for FLCs with a large number of input variables. A dynamically weighted objective function is suggested for control problems with multiple tasks, which prevents the GA from premature convergence on FLCs that are specialized exclusively to the easier subtasks. In order to achieve a robust control behaviour for a broad spectrum of situations a second GA coevolves a set of training situations to evaluate the performance of the FLCs. The method is applied to learn a FLC, which implements a behaviour of a mobile robot. The robot is given the task to reach a goal point and to avoid collisions with obstacles on its way, which it perceives by means of ultrasonic sensors. The performance of the FLCs, which are learned in simulated environments, is tested afterwards in real world experiments with the mobile robot.

## 1 Introduction

Soft computing is concerned with the design of adaptive, intelligent and robust systems, which imitate the human mind in its ability to deal with imprecise, incomplete knowledge and partial truth. In order to achieve this objective, soft computing suggests the combination of fuzzy logic (FL), neural networks (NN) and genetic algorithms (GA) in a spirit of partner-

ship. The integration of different methods, adequate for their specific domain of problems, results in more powerful hybrid systems with higher machine intelligence, than using a single method exclusively. In the same way in which the complete diversity of colours emerges from mixing the basic colours red, blue and yellow, the full spectrum of possibilities to create new methodologies is exhausted by using a combination of the three complementary methods FL, NN and GA.

Thirty years ago LOTFI ZADEH founded the principles of fuzzy logic with his seminal paper on fuzzy sets [ZAD65]. Fuzzy systems are able to represent imprecise, uncertain knowledge. They use a mode of approximate reasoning, which allows them to make decisions based on vague and incomplete information in a way similar to human beings. A fuzzy system offers the advantage to describe its knowledge by means of linguistic concepts without requiring the complexity and precision of mathematical or logical models. Fuzzy control (FC) provides a flexible tool to model the relationship among input information and control output. FLCs have been employed for a variety of complex control problems [MAR94], including the control of mobile robots [BEO95]. They proved to be robust in respect to noise and variations of system parameters.

GAs are optimization methods guided by the principles of natural evolution, which they simulate in a computer. They imitate the underlying processes of evolution such as selection, recombination and mutation in order to solve difficult theoretical and practical problems. The GA processes a population of candidate solutions from one generation to the next. Each individual is represented by a genetic string of parameters called chromosome. In the course of time the artificial evolution produces more and more suitable solutions to the optimization problem. GA require only a scalar fitness function in order to adapt a system to its environment and to optimize it with respect to a given objective.

The combination of NN and FL in neuro fuzzy sys-

tems serves as a model for soft computing techniques. During the last few years GAs have been widely used for the design of FLCs [COR95].

## 2 Genetic algorithms for the design of fuzzy logic controllers

There are mainly two different approaches to design a FLC using a GA. The first method tunes the parameters of an already existent FLC. It is primarily useful in order to adapt the fuzzy system when the dynamics or the external conditions of the process vary in the course of time. The parameters of the knowledge base, such as scaling factors, the width and the center of membership functions, are coded in the chromosome and optimized by the GA in respect to a given reference data set. This method has the drawback that a human designer has to formulate a suitable rule base in advance.

The second approach adapts the rule base itself. For this purpose the genetic string encodes the label of the output fuzzy term, which occurs in the conclusion of a rule. The complete chromosome is built of the aligned labels of the output terms, that correspond to all possible combinations of antecedents. In contrast to the first method, which optimizes an existent fuzzy system, the automatic design of the rule base is a matter of learning the general control behaviour. The optimization task to find the right conclusions of fuzzy rules is much more difficult, because the GA has to design the FLC starting from scratch. There exist hybrid methods, in which the GA learns the fuzzy rules as well as their parametrized membership functions.

According to the building block hypothesis the way in which the coding scheme represents a candidate solution has a significant influence on the effectiveness of the GA. Therefore the coding of the controller's knowledge base in the genetic string plays an essential role in the design of the methodology. Previous approaches for the design of FLCs using GAs employ a position dependent, fixed length coding scheme which represents the complete rule base in a genetic string. The complexity of such a rule base grows rapidly with an increasing number of input variables. The efficiency and the robustness of the FLC is lost, with the result that the optimization task becomes less feasible for the GA.

### 2.1 Messy coding scheme

In imitation of messy GAs (mGAs)[GOL89] we suggest a new compact coding scheme for a fuzzy rule base. The coding does not depend on the position of genes and allows a flexible, efficient representation of

fuzzy rules of different structural complexity in the genetic string. In comparison with a conventional coding scheme it has the advantage of a redundancy free, compact coding of the fuzzy rule base. The GA is therefore able to form chromosomes in a very flexible manner in order to represent the relationship among control input and output. Because of the reduced complexity of the fuzzy rule base the fuzzy controller remains comprehensible and effective, helping the GA to solve the optimization task even for more complex control problems.

Chromosomes are composed of a variable number of genes. Their meaning is part of the coding and therefore independent of their position inside the string. The coding scheme is able to encode each conceivable combination of fuzzy variables and fuzzy terms into the chromosome. Therefore it is not subjected to any restrictions regarding the formulation of fuzzy rules. In order to maintain the clarity of the adapted knowledge base fuzzy rules are based on linguistic variables and terms previously defined by a human expert.

In order to encode the fuzzy rules in a genetic string the input and output variables and their corresponding fuzzy terms are numbered by labels. Fuzzy clauses constitute the basic element of the coding scheme. They are genetically encoded in a pair of labels, in which the first label represents the variable and the second defines the associated fuzzy term. Instead of using a binary representation all labels are coded by integer values.

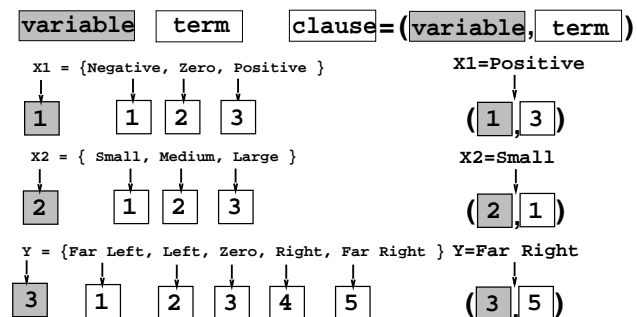


Figure 1: Messy coding of fuzzy clauses

Fig. 1 shows the messy coding of clauses for a simple FLC with two input variables namely  $X_1$  with terms *negative*, *zero*, *positive* and  $X_2$  with terms *small*, *medium*, *large* as well as a single output variable  $Y$  with five terms *far left*, *left*, *zero*, *right*, *far right*. The first integer specifies the meaning of a gene because it defines the variable of the fuzzy clause. The second number determines the value of a gene because it associates a fuzzy term with the variable. For example the gene  $(1, 3)$  represents the fuzzy clause  $X_1 = \text{positive}$ ,

because  $X_1$  is the first variable and *positive* is its third fuzzy term.

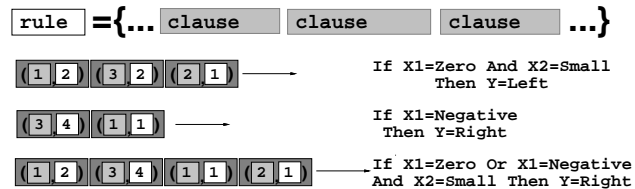


Figure 2: Messy coding of fuzzy rules

Complete fuzzy rules of variable length are built by the combination of multiple fuzzy clauses in any order whatever. The coding of a fuzzy rule is a sequence of genes each one representing a fuzzy clause. Fig. 2 shows some examples of possible gene combinations for three fuzzy rules. The sequence of genes **(1,2) (3,2) (2,1)** is the genetic representation of the fuzzy rule:

**If  $X_1$ =Zero **And**  $X_2$ =Small **Then**  $Y$ =Left**

Notice that the individual genes of a fuzzy rule can be arranged in any order whatever, because their meaning is independent of their position in the sequence. The sequence **(2,1) (1,2) (3,2)** for example is another valid coding of the same fuzzy rule above. Reordering of genes in the chromosome can be advantageous, because it enables a messy GA to form tight and efficient building blocks according to the schema theorem.

The drawbacks that result from a messy coding scheme are the twin problems of under- and overspecification of the genetic string. In our case underspecification is a desired feature of the coding scheme, because it enables the formation of compact fuzzy rules, in which the antecedent includes only a subset of all input variables. The complexity of the rule base is reduced because large regions of input space can be covered by a single rule, if it contains only a few fuzzy terms in its antecedent. Fuzzy clauses are simply omitted in the antecedent of the fuzzy rule, if the gene sequence contains no pair that corresponds to their variable. In the second example of Fig. 2 the chromosome **(3,4) (1,1)** includes no gene for variable  $X_2$ , which results in the quite general fuzzy rule:

**If  $X_1$ =Negative **Then**  $Y$ =Right**

Instead of this single rule a conventional fixed coding scheme, in which fuzzy rules always contain complete antecedents, requires three fuzzy rules with identical conclusions in order to genetically encode the same relationship among  $X_1$  and  $Y$ .

Overspecification occurs whenever a chromosome contains more than one different fuzzy clause for the same variable. It is resolved either by means of a dominance scheme, in which only the leftmost clause is expressed,

or by using a fuzzy **Or**-operator in order to associate multiple fuzzy terms with the same variable. In the third example of Fig. 2 the two fuzzy clauses corresponding to the conflicting pairs **(1,2)** and **(1,1)** for variable  $X_1$  appear both in the rule:

**If ( $X_1$ =Zero **Or**  $X_1$ =Negative) **And**  $X_2$ =Small **Then**  $Y$ =Right**

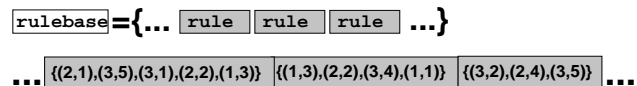


Figure 3: Messy coding of fuzzy rule base

In the same way in which a fuzzy rule is composed of multiple fuzzy clauses the complete rule base is constructed from an unordered set of fuzzy rules. As depicted in Fig. 3 the joined gene sequences of individual rules form the entire chromosome of a fuzzy rule base. The order of fuzzy rules inside the rule base does not matter, with the result that an additional numbering of gene sequences is not necessary. The coding scheme enables the GA to evolve fuzzy rule bases of different complexity and size, because the number of rules and their structural composition are variable and therefore part of the optimization process. See [HOF96] for more details on the coding scheme and the handling of under- and overspecification.

## 2.2 Genetic operators

Besides of a different coding scheme mGAs employ a modified crossover operator in order to recombine two genetic parent strings in a new offspring. The conventional crossover is replaced by the two operators cut and splice. The position of cuts on both parents is chosen independently and the resulting pieces are spliced in any order whatever. Cut and splice result in the inversion operator if both pieces are inherited from a single parent but are spliced in an inverse order.

Recombination of parent strings appears on both levels of the coding scheme. On the lower level the cut and splice operators create new fuzzy rules when they exchange fuzzy clauses of parent rules. Before recombination one of the parents is matched up with a set of candidates for mating in order to guarantee a minimal similarity between the two fuzzy rules to be merged together. Otherwise recombination will mainly generate purely random rules.

On the upper level recombination mixes two set of fuzzy rules together into a new rule base. Two parents that have already adapted suitable control rules for some region of input space benefit from each other if they exchange parts of their rule bases. This com-

plies with the basic idea of recombination to further exploit the chromosomes of those candidate solutions, that belong to promising regions of search space. A cut operation that is applied near the center of the chromosome reduces the number of the fuzzy rules nearly by half, with the result that the offspring loses an essential part of the parent's rule base. Therefore the position of a cut is not distributed with uniform probability along the chromosome. Instead it is chosen in a way that one parent contributes a larger number of fuzzy rules, while the other one transmits only a smaller part of its genes to the offspring.

### 3 Learning the behaviour of an autonomous agent

An autonomous agent is a behaviour based system, that is situated in a dynamic environment, which it perceives for example by means of sensors. Therefore its knowledge about the environment is incomplete and often unreliable. In response to these input informations it reacts with an appropriate control action, which enables it to change its state in the environment. This mapping from perceptive input to a control action realizes a specific behaviour of the agent. FLCs are suitable to implement such a behaviour, because their method of approximate reasoning predestine them to find a suitable control action even if their knowledge about the environment is imprecise and incomplete.

By means of a reinforcement signal provided by the environment the agent is able to detect which actions resulted in favourable states. The agent adapts its behaviour in order to maximize the reinforcement payoffs in the future. BONARINI [BON96] demonstrated that evolutionary algorithms are suitable to learn the behaviour of an autonomous agent implemented by fuzzy rules.

We applied our method to evolve a FLC for a mobile robot. The given task is to reach a specified goal point while avoiding collisions with obstacles on the way. BEOM et al. [BEO95] presented a method which builds a fuzzy rule base for the same task by means of reinforcement learning. In our case a GA learns the FLCs in simulated environments. After the learning period the FLCs are tested on the robot in real world situations.

In the simulations as well as in the experiments the controller's inputs are only the robot's position and five current distances to objects, measured by ultrasonic sensors on the real autonomous vehicle. Prior distance information is not available to the controller, so that it possesses no memory about previous states of its environment. The simulation employs a simplified

model of the real sensors, based on idealistic assumptions about the reflection characteristics of ultrasonic signals.

#### 3.1 Design of the objective function

To evaluate the performance of the FLCs the GA requires a scalar objective function in order to select the best individuals for reproduction of offspring. In the following two proposals for evaluation of FLCs are presented, which result in an improved, more robust control behaviour. First of all the usually constant set of training situations is replaced by a variable set of test cases, which adapts itself evolutionary to the performance of the current population of FLCs. This adaptation is achieved by a second simultaneous GA, so that the learning is based on the competition of two evolutionary processes. Therefore it is guaranteed, that those FLCs emerge, which demonstrate the desired behaviour for a broad spectrum of situations and environments. The FLCs are adapted to the *actual* given task, instead of merely learning their training situations by heart.

In case of the mobile robot the performance of each FLC is evaluated in thirty, simulated, two-dimensional environments, which include corridors, dead ends and detached obstacles. A second GA updates this set of training situations in one out of three generations of the FLC population. It varies the parameters of the environments, such as the width and the geometry of corridors as well as the position of obstacles, dead ends, the start and the goal point. A training situation achieves a high fitness value, if a high percentage of FLCs fails on goal point reaching or collision avoidance in this environment.

For control problems with multiple competing tasks the fitness function has to be designed carefully in order to take all objectives into account. The GA tends to converge prematurely on FLCs that are exclusively specialized to individual easy subtasks. If for example a static objective function is used in case of the mobile robot, the FLCs mainly succeed in collision avoidance but carry out inadequately the task of goal point reaching, which is demanded as well. In order to evolve FLCs capable of multiple subtasks, we suggest a dynamic objective function, in which a FLC achieves a high fitness value for a specific subtask, if it is not yet mastered by the population. Therefore a large incentive for FLC is created to learn *all* of the demanded subtasks.

Each FLC  $C_k$  of the current population is tested in simulations on the mobile robot in thirty different environments  $E_i$ . Its performance is afterwards evaluated in respect to the two tasks of goal point reaching and

collision avoidance. A single run of the robot stops either if a collision with an obstacle occurs, a maximum number  $N_{\max}$  of control steps is exceeded or the goal point is reached. For each of the three cases the FLC receives a different amount of reinforcement.

$$\begin{aligned} R_c(C_k, E_i) &= N_c/N_{\max} && \text{collision} \\ R_d(C_k, E_i) &= 1 - d_{\min}/d_0 && \text{no collision} \\ R_t(C_k, E_i) &= 1 && \text{goal point reached} \end{aligned} \quad (1)$$

If a collision happens the FLC receives only a small reward  $R_c$  proportional to the number of steps  $N_c$  made so far. If a collision is avoided but the goal point is missed the FLC is given an additional reward  $R_d$ , depending on how close the robot approached the goal point. In Eq. 1  $d_0$  is the distance between the start and the goal point while  $d_{\min}$  is the minimum distance between robot and the goal point, which occurred during  $N_{\max}$  steps. Notice that in this case the FLC receives the sum  $R_c + R_d$  of both rewards. The robot is considered to be successful in goal point reaching, if its distance to goal point falls below a minimal radius  $r_{\min}$ . In this case the FLC receives all of the three rewards  $R_c, R_d$  and  $R_t$ .

These raw reinforcement values are dynamically scaled according to the performance the whole population demonstrates on the individual subtasks. For this purpose the reinforcements  $R_c(C_k, E_i)$ ,  $R_d(C_k, E_i)$  and  $R_t(C_k, E_i)$  are divided by the sum of reinforcements achieved by the other FLCs on each individual contribution. The resulting fitness value  $F(C_k)$  of a FLC  $C_k$  is:

$$F(C_k) = \sum_i \sum_{x \in \{c,t,d\}} \left( R_x(C_k, E_i) / \sum_n R_x(C_n, E_i) \right) \quad (2)$$

The Fig. 4 compares the performance of two GAs in the course of thirty generations, one in which a static fitness function is employed and the other one using the dynamically weighted fitness evaluation. The percentage of goal points reached and of avoided collisions are shown, with each criterion averaged over all FLCs of the current population. All results are based on an average over ten runs for each of the two GAs. While both GAs achieve nearly the same performance on collision avoidance (upper graphs), the FLCs of the GA that uses a static fitness function demonstrate only minor improvements on goal point reaching.

In the first generations of the GA, which employs a dynamic fitness function, most of the FLCs are specialized on the objective of collision avoidance. Only a few FLCs manage to reach the goal point in at least one or two environments. These FLCs achieve a relatively high fitness value, because they have to share

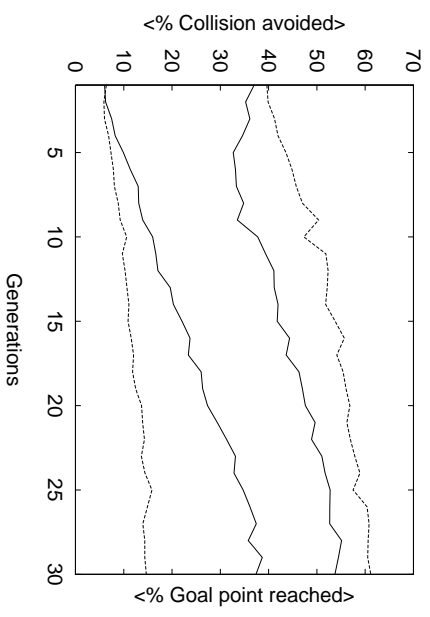


Figure 4: Average number of goal points reached and collisions avoided using a static (---) or a dynamic (—) fitness function

their reinforcements  $R_t$  with only a few competitors. Therefore they are given preference in the selection compared with other FLCs, even if they collide in the remaining environments more often with obstacles than an average FLC. By means of selection and recombination the number of FLCs increases during the course of evolution, that have learned to carry out both tasks simultaneously.

The fitness values  $F(E_i)$  of the coevolved environments  $E_i$  are calculated from the inverse sum of reinforcements, which the controllers achieved on them:

$$F(E_i) = 1 / (1 + \sum_n R_x(C_n, E_i)) \quad (3)$$

### 3.2 Experimental results with the mobile robot

The FLCs evolved by the GA proved their usefulness in respect to both tasks in simulated as well as experimental runs of the mobile robot. In conflicting situations, such as a goal point located inside an obstacle, collision avoidance is given a higher priority than goal point reaching as one expects meaningfully. The FLC for example is able to detect, whether a corridor provides enough space in order to turn the robot, which is remarkable in view of the lacking memory on previous sensor informations.

The Fig. 5 shows results of real world experiments in two corridors, in which the mobile robot starts with its heading in opposite direction to the goal point. In the left experiment with a narrow corridor the FLC tries several times to turn the robot. It interrupts the turning manoeuvre in time in order to prevent a collision. After leaving the corridor the FLC manages to turn the robot in the free area. In the Fig. 5 on

the right the corridor is wide enough, with the result that the FLC is able to perform a turning manoeuvre with the robot immediately. It first tries to turn round right, but stops the manoeuvre in the critical moment, when the robot comes too close to the lower wall. Now the robot has enough space at its disposal in order to complete the turning manoeuvre to the left. For reasons of clarity the path of the robot after its turn is not depicted, but in both situations the robot finally reaches the goal point.

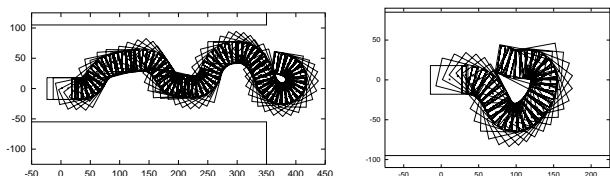


Figure 5: Paths of the mobile robot in a narrow and a broad corridor with goal point located at the left

Furthermore the adapted FLCs proved robust enough to demonstrate a meaningful control behaviour when using the real imprecise and unreliable sensor data, although they were trained in the evolution by means of a very simplified model of sonar distance measurements. In runs that were carried out in identical environments, only small differences between real world experiment and simulation were observed. The results show that FLCs which were adapted in a simulation can be transferred successfully to physical reality. The Fig. 6 shows two photos taken during an experiment, in which the robot gets out of the way of an obstacle.



Figure 6: Mobile robot during experiment

In order to investigate the FLCs capability for generalization, they were tested in environments that were not presented during the learning process in the GA. It turned out that they manage the given tasks in previously unseen environments, as long as their degree of difficulty is not significantly higher than those of the training situations. The FLCs are successful in environments with a geometry that is similar to those presented during evolution, while it fails in situations that differ substantially. If for example the training situations contained no corridors with dead ends the

robot collides with the wall, which blocks the exit of the corridor.

## 4 Conclusions

We suggested a messy coding scheme for the evolutionary design of FLCs, which allows a compact and flexible genetic representation of the fuzzy rule base. Because of the reduced complexity of the fuzzy rule base the fuzzy controller remains comprehensible and effective, which enables the GA to solve the optimization task even for more complex control problems. We applied our method successfully in order to design a FLC which implements the behaviour of a mobile robot.

## Acknowledgements

This research was partially supported by Stifterverband für die Deutsche Wissenschaft.

## References

- [BEO95] Hee Rak Beom, Hyung Suck Cho, "A Sensor-Based Navigation for A Mobile Robot Using Fuzzy Logic and Reinforcement Learning", *IEEE Trans. Syst. Man Cyber.*, vol. 25, no. 3, pp. 464-477, (1995)
- [BON96] Andrea Bonarini, "Learning behaviors implemented as Fuzzy Logic Controllers for Autonomous Agents", *Second Online Workshop on Evolutionary Computation*, (1996)
- [COR95] Oscar Cordon, Francisco Herrera, Manuel Lozano, "A Classified Review on the Combination Fuzzy Logic Genetic Algorithms Bibliography", *Technical Report #DECSAI-95129*, <http://decsai.ugr.es/herrera/fl-ga.html>, (1995)
- [GOL89] David E. Goldberg, Bradley Korb, Kalyanmoy Deb, "Messy Genetic Algorithms Motivation, Analysis, and First Results", *Complex Systems*, vol. 3, pp. 493-530, (1989)
- [HOF96] Frank Hoffmann, Gerd Pfister "Learning of a Fuzzy Control Rule Base Using Messy Genetic Algorithms", *Genetic Algorithms and Soft Computing*, Ed. F. Herrera, J. L. Verdegay, Physica-Verlag, Heidelberg, (1996)
- [MAR94] Robert J. Marks II (Ed. ), *Fuzzy Logic Technology and Applications*, IEEE Technology Update Series, (1994)
- [ZAD65] Lotfi Zadeh, "Fuzzy Sets", *Journal of Information and Control*, vol. 8, pp. 338-353, (1965)