

Automatic Design of Hierarchical Fuzzy Controllers Using Genetic Algorithms

Frank Hoffmann, Gerd Pfister

University of Kiel

Institute of Applied Physics

24098 Kiel, Germany

Phone: +49 431 880 3935 , Fax: +49 431 880 4608

Email: hoefi@ang-physik.uni-kiel.de

Abstract

An automatic design method for hierarchical fuzzy controllers using genetic algorithms is proposed. A reorder operator for the genetic algorithm is introduced. We applied the method to the problem of controlling an autonomous vehicle with the task to reach a given location and avoiding obstacles on the way.

Keywords: Fuzzy Control, Genetic Algorithms, Autonomous Vehicle

1 Introduction

Design of linguistic variables and rules of a rule-based fuzzy controller requires expert knowledge. This knowledge is obtained by consulting or observing a human operator controlling a dynamic system.

Besides adaptive fuzzy systems and neuro fuzzy systems, genetic algorithms (GA) have been proposed for automatic design of fuzzy controllers. GAs are search algorithms, which use operations coming from natural evolution [1]. GAs were applied successfully in optimizing fuzzy controllers [2]. One approach in optimizing an existing controller deals with the modification of membership functions [3]. Another method optimizes the rule base of a fuzzy controller [4] using a GA.

In the following a design method for hierarchical fuzzy controllers using a GA is proposed. The architecture of the hierarchical fuzzy controller using hidden fuzzy variables is explained in section 2. Section 3 describes the GA used to optimize the fuzzy controller. A new reorder operator for the GA is introduced, preserving meaningful rule bases when combining strings of two parents by crossover. In section 4 an application of the method in designing a fuzzy controller for an autonomous vehicle is presented. The paper closes with concluding remarks in section 5.

2 Architecture of the hierarchical fuzzy controller

The number of rules in a fuzzy systems grows overproportional with the number of linguistic variables and connected linguistic terms. For a complete rule base with input variables $\{X_i | i = 1, \dots, n\}$ with terms $\{A_{ij} | j = 1, \dots, q_i\}$ and output variables $\{Y_i | i = 1, \dots, l\}$ with terms $\{B_{ij} | j = 1, \dots, p_i\}$ one needs N fuzzy rules with

$$N = \prod_{i=1}^n q_i \quad (1)$$

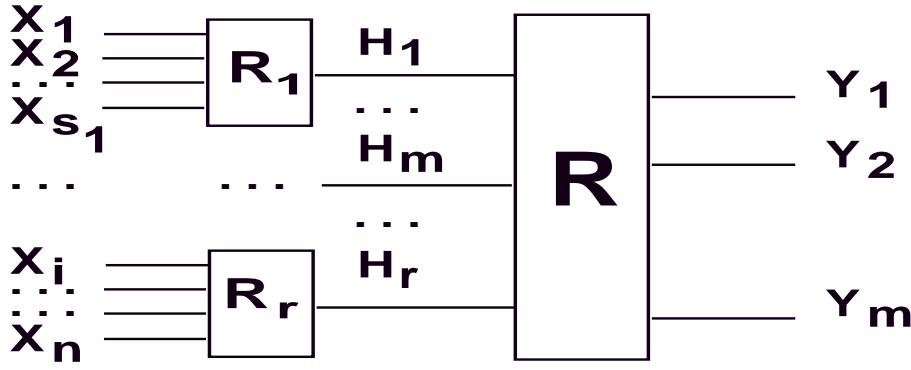


Figure 1: Architecture of the hierarchical fuzzy controller

The rules have the form:

if $X_1 = A_1$ and ...and $X_n = A_n$ then $Y_1 = B_1$ and ...and $Y_l = B_l$

with $A_i \in \{A_{ij}\}, B_i \in \{B_{ij}\}$. This large number of rules complicates the design of a rule base, because for each of the N different premises the expert must suggest a combination of terms for the output variables. It is possible to omit part of the rule base, if it could be guaranteed that certain combinations of input variables do not appear in the dynamic system during control. Such a proof requires an analytical model of the controlled system.

A hierarchical architecture of the rule base permits reduction of the number of rules. This could be done by introducing hidden fuzzy variables in analogy to hidden neurons in neural networks. The architecture of the hierarchical fuzzy controller is shown in Fig. 1. In a first step the n linguistic input variables X_i are divided into disjunct sets $\{Q_m | m = 1, \dots, r\}$ each with s_k input variables.

With each set Q_m a complete rule base R_m of rules such as

if $X_1 = A_1$ and ...and $X_{s_m} = A_{s_m}$ then $H_m = C_m$

is associated. There are no fuzzy sets associated to the hidden fuzzy variables H_m in the consequences. They are fully described by the membership values $\mu(C_{mj})$ of their fuzzy terms $C_m \in \{C_{mj} | j = 1, \dots, p_m\}$. The degrees of membership for fuzzy terms C_m are evaluated by inference of rule base R_m . Degree of membership $\mu(B_{ij})$ for output variables Y_i is obtained by inference of the overall rule base I with rules such as

if $H_1 = C_1$ and ...and $H_r = C_r$ then $Y_1 = B_1$ and ...and $Y_l = B_l$.

By defuzzification of fuzzy sets B_{ij} one obtains crisp values y_i for output variables. Grouping together different premises by associating them with the same hidden fuzzy term results in a fuzzy-OR of these premises. All premises of rule base R_m possessing the same hidden fuzzy term C_{mj} as consequence finally vote for the same consequence B_{ij} in rule base R . Let P_i^m be premises associated to the same term of hidden fuzzy variable H_m . The resulting overall rule base could be described by rules such as

**if (P_1^1 or P_2^1 or ...) and (P_1^2 or P_2^2 or ...) and ...
...and (P_1^r or P_2^r or ...) then $Y_1 = B_1$ and ...and $Y_l = B_l$.**

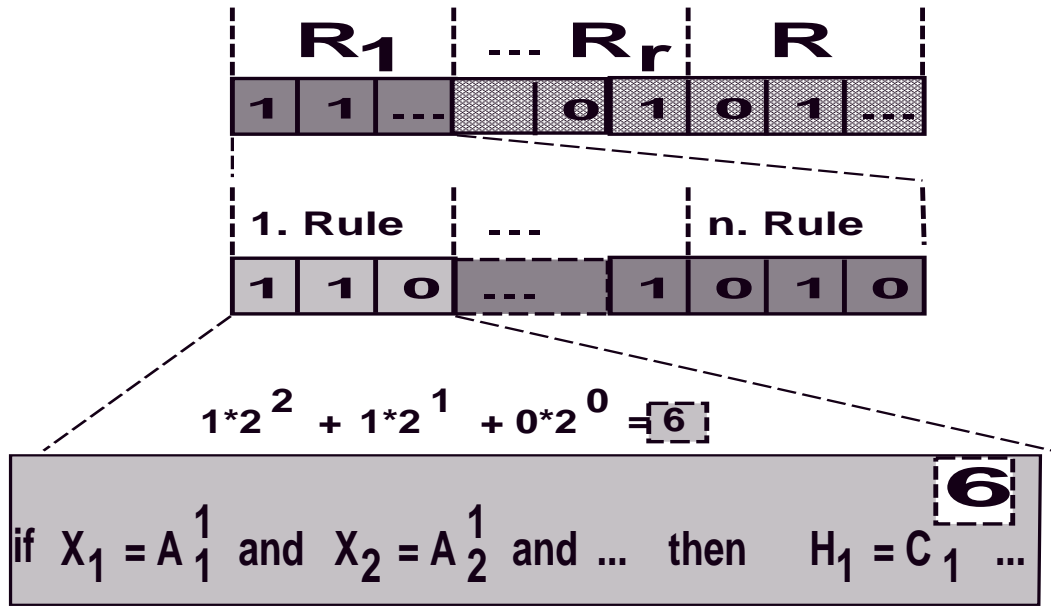


Figure 2: Coding of rule bases

By mapping different premises to the same hidden fuzzy term the overall number of rules N is drastically reduced to:

$$N = \sum_m N_{R_m} + N_R \quad (2)$$

where N_{R_m}, N_R are the number of rules for rule bases R_m, R , which could be obtained from equation (1), but with smaller number of input variables n compared to a non-hierarchical fuzzy controller.

3 Genetic Algorithm

GAs code parameters of the search space as binary strings S_i of fixed length. They employ a population $P = \{S_1, S_2, \dots, S_M\}$ of strings initialized at random, which evolve to the next generation by genetic operators such as selection, crossover and mutation. The fitness function evaluates the quality of solutions coded by strings. Selection allows strings with higher fitness to appear with higher probability in the next generation. Crossover provides a mechanism for combining two parents by exchanging parts of their strings, starting from a randomly chosen crossover point. This leads to new solutions inheriting desirable qualities from both parents. Mutation flips single bits in a string, which prevents the GA from premature convergence, by exploiting new regions in search space.

For optimizing the above described fuzzy controller using a GA, a coding scheme must be chosen. Consequences of rule bases R_m and R are coded by binary strings as shown in Fig. 2. Each conclusion $H_m = C_m$ obtains one of p_m different possible terms C_{mj} . A selected term $C_{mj}, j \in \{1, \dots, p_m\}$ is coded as an integer j by a substring TS_m of length $\text{ld}(p_m)$. The substring S_m for a rule base R_m is obtained by putting together substrings TS_m . In the same way consequences $Y_1 = B_1^k$ and $\dots Y_m = B_m^k$ are coded in a substring TS , where for each of the output variables Y_i the resulting output term B_{ij} is coded as an integer j . Linking of substrings TS results in substring S coding rule base R . The complete string for coding an entire fuzzy controller is obtained by adding all substrings $S_1 S_2 S_3 \dots S_r S$ in a row.

The hierarchical fuzzy controller is invariant under a permutation of indices j of terms C_{mj} of hidden fuzzy variables H_m if this permutation is applied to consequences in rule base R_m as

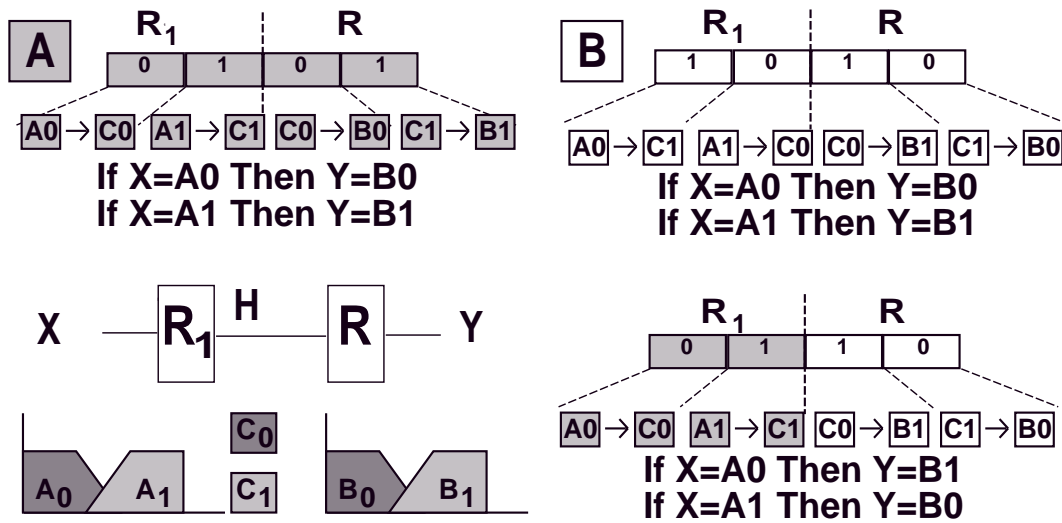


Figure 3: Disruption of hidden fuzzy terms by crossover

well as to premises of rules in rule base R . Crossover produces a new controller by combining two parent strings A and B . Crossover could cause disruption of parent strings as shown in Fig. 3. The fuzzy controller obtains input from variable X with only two fuzzy terms A_0 and A_1 . Only two rules for two different premises are needed in rule base R_1 which has hidden fuzzy variable H with two hidden fuzzy terms C_0 and C_1 as consequences. Inference to the output variable Y with two fuzzy terms B_0 and B_1 is done by rule base R which likewise contains two rules. Each rule is coded by a single bit, choosing between two possible consequences. Although this most simple hierarchical fuzzy controller could be easily replaced by a non-hierarchical one, it demonstrates well the difficulties occurring due to crossover. Both parent controllers show the same overall behaviour which could be described by **if $X = A_0$ then $Y = B_0$** and **if $X = A_1$ then $Y = B_1$** . Crossover between parent strings A and B behind the second bit results in a child with rule base R_1 inherited from parent A and rule base R inherited from parent B . Since the hidden fuzzy terms C_0 and C_1 are used in an opposite manner of parents A and B , the controller shows the opposite behaviour **if $X = A_0$ then $Y = B_1$** and **if $X = A_1$ then $Y = B_0$** as his parents. Premises in rule base R obtained by parent B , don't match with consequences in rule base R_1 obtained by parent A . To avoid this situation a matching between both parent strings must be performed in order to bring the hidden fuzzy terms C_0 and C_1 in a common order for both parents. For this purpose we introduce a reorder operator to assure coherent interpretation of fuzzy terms during crossover.

In a first step matching procedure between parents A and B is performed by comparing the premises P of hidden fuzzy terms C_i . In Fig. 4 hidden fuzzy term C_0 of parent A has one premise in common with hidden fuzzy term C_1 of parent B and two premises in common with term C_2 . Since terms C_0 of parent A and term C_2 of parent B have the largest correspondence these two terms are mapped to each other. In the same way term C_1 is mapped to C_0 and term C_2 to C_1 .

In the next step parent string A is recoded, so that the hidden fuzzy terms match those of parent B . Recoding must be done in rule base R_i as well as in R , but in different ways. In rule base R_i hidden fuzzy term C_j is part of the consequence, so the value of the gene has to be changed (Fig. 5). In rule base R hidden fuzzy term C_j is part of the premise, so the position of the gene in the string has to be changed (Fig. 5). This recoding operator has the property to leave the overall behaviour of the fuzzy controller unchanged, but assures unique interpretation of hidden fuzzy terms between both parents.

We also suggested another reorder operator [5], which leads to a common use of hidden fuzzy terms in the whole population. The recoding part is the same as described above, but matching is done between all strings in the population. This reorder operator has the disadvantage of premature convergence of the GA because of its global properties. The above described reorder

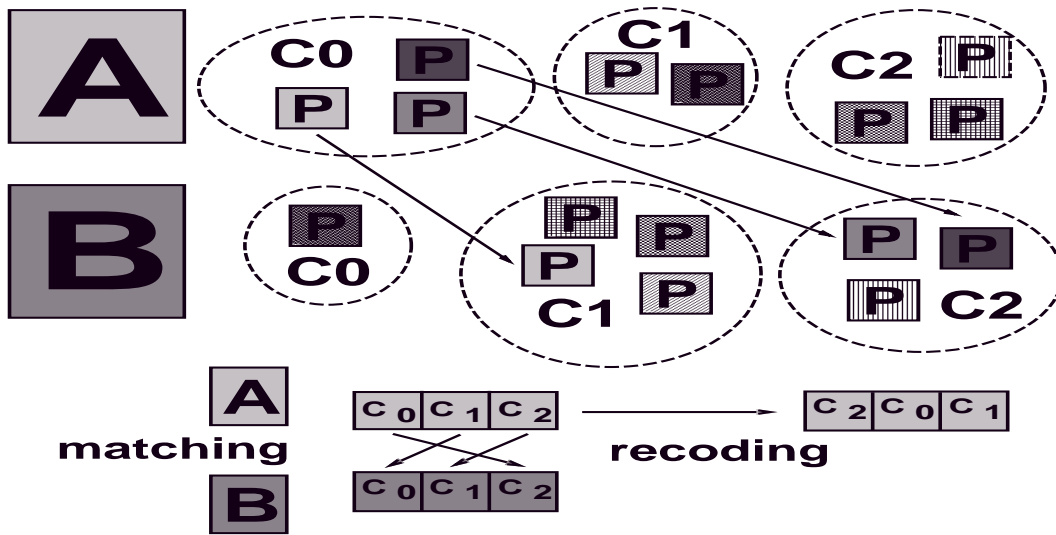


Figure 4: Matching of parent strings by comparing premises P of hidden fuzzy terms C_i

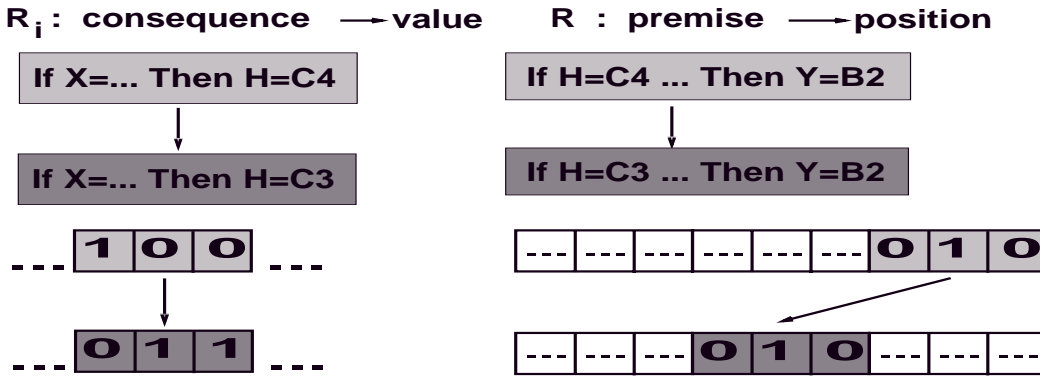


Figure 5: Recoding of hidden fuzzy terms in rule base R_i and R

operator matches only two strings and avoids premature convergence of the GA.

Finally a fitness function must be chosen which describes the performance of the fuzzy controller with reference to a desired behaviour of the dynamic system [4]. If there exists a reference data set of desired controlling actions for different states of the system, one could use the deviation of the actions proposed by the fuzzy controller from the reference data as fitness function [3].

4 Autonomous Vehicle

For an autonomous vehicle a fuzzy controller was designed with the above described method. The vehicle should reach given locations in indoor environments. Obstacles like walls or objects should be recognized and collisions should be avoided. Three ultrasonic sensors allow contactfree distance measurements of obstacles. The steering of the vehicle is done by two independent stepping motors. Position and orientation of the vehicle are obtained by counting steps of both driving wheels.

Input variables for the fuzzy controller are distances X_{d_left} , X_{d_right} and angles to obstacles X_{a_left} , X_{a_right} obtained from the ultrasonic sensors, containing three or two term sets.

Further input variables are distance $X_{distance}$ and relative orientation $X_{orientation}$ of vehicle to the goal point the vehicle should reach. $X_{distance}$ contains only two term sets *near*, *far*; $X_{orient.}$ contains five term sets *left behind*, ..., *ahead*, ..., *right behind*. Steering angle $Y_{steering}$ is used as output variable from which actual velocities v_1 and v_2 of stepping motors are calculated. $Y_{steering}$

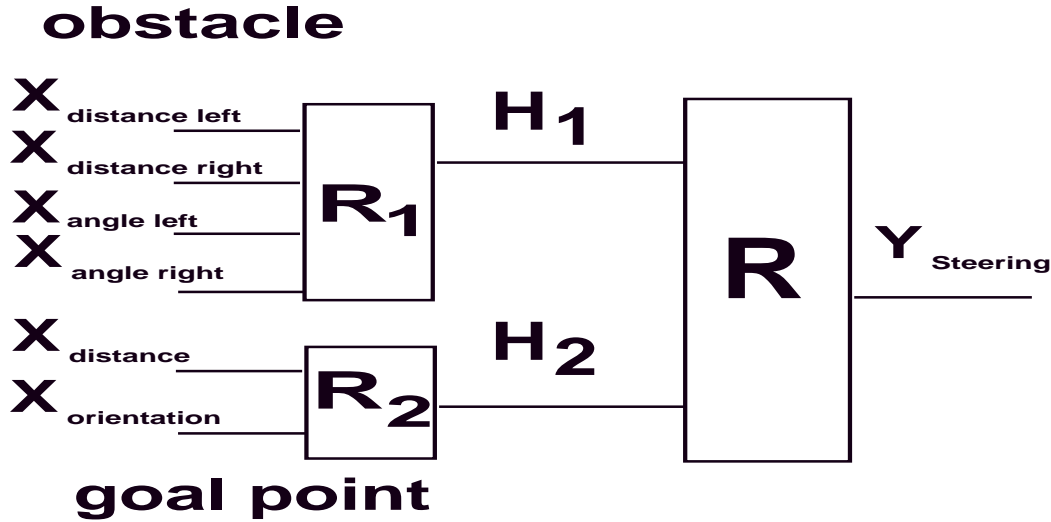


Figure 6: Architecture of fuzzy controller for an autonomous vehicle

contains seven term sets *very left*, ..., *straight ahead*, ..., *very right*. The controller uses Max-Min inference as inference method and Center-of-Gravity as defuzzification method to compute the crisp value of output variable $Y_{steering}$. For aggregation of premises the Minimum operator is employed.

Input fuzzy variables X_{d_left} , X_{d_right} , X_{a_left} , X_{a_right} are combined in rule base R_1 . Remaining input variables $X_{distance}$ and $X_{orientation}$ constitute rule base R_2 . Inference to output variable $Y_{steering}$ is done by rule base R which takes hidden fuzzy variables H_1 and H_2 as input. The architecture of the fuzzy controller is shown in Fig. 6. The task of rule base R_1 is to recognize the location of obstacles evaluating distance information coming from ultrasonic sensors. Rule base R_2 gathers information about the goal point to reach. This is similar to the approach of Ruspini [6], who uses context-dependent blending of behaviour between the two tasks of avoiding obstacles and reaching a goal point, leading to an overall strategy. Each of the hidden fuzzy variables contains five terms. The overall number of rules is 66 in comparison to a non-hierarchical controller needing 360 rules for a complete rule base.

Hidden fuzzy terms C_{1j} contain information whether an obstacle is left, straight ahead or right of the vehicle. Hidden fuzzy terms C_{2j} contain information whether the goal point is left, in front of or right from the vehicle.

Fuzzy controllers obtained from the GA were tested in a simulation on a computer with environments containing different kinds and numbers of obstacles and goal points. Fig. 7 shows two simulated runs, where the goal point is marked by a cross. Fuzzy controller achieved high fitness, if the vehicle reached the goal point without collision with one of the obstacles. The fitness function was designed to minimize the final distance from goal point, the length of the covered path and contains an additional punishment for collisions and if the vehicle falls short of a safety distance to obstacles. In addition to a standard GA, with a population size of hundred strings, we used the reorder operator. The GA tends to learn the geometry of the environments presented to the controller, instead of the general task. The obtained controllers worked well in the environments presented during the evolution process, but showed poor performance for new kinds and positions of obstacles. We modified the GA by keeping environments fixed for ten generations. After every tenth generation we used a new set of environments to test the controllers obtained from the GA. Instead of controllers achieving near optimal fitness in only certain environments, the evolution process is lead to controllers with sufficient performance in a large variety of environments. The fuzzy controller obtained from the GA is able to solve the control problem for environments with little complexity. The fuzzy controller fails in more complex environments like labyrinths because he possesses no memory and is only able to use local information from his sensors.

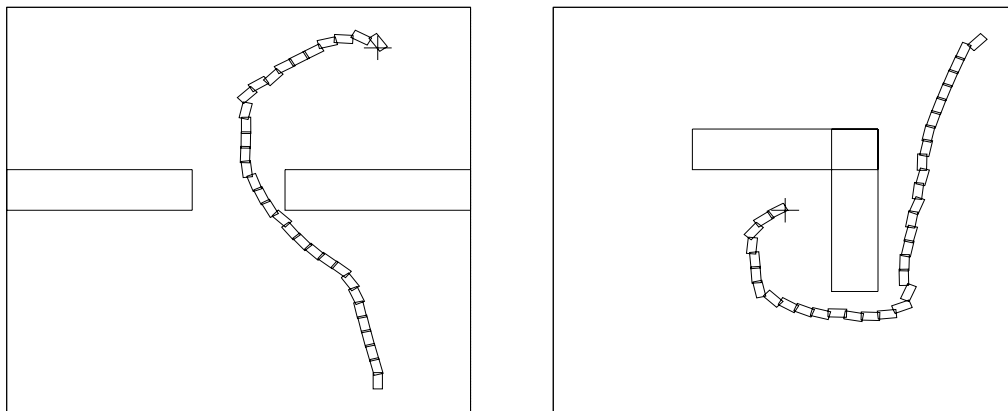


Figure 7: Examples of simulated runs of autonomous vehicle

5 Conclusions

In this paper a design method for hierarchical fuzzy controllers using genetic algorithms is proposed. By using a hierarchical architecture for the fuzzy controller the number of rules is reduced in comparison to a non-hierarchical controller. Hidden fuzzy variables are used to group several premises together which divide up the rule base in smaller parts. A two step reorder operator for the genetic algorithm which is applied before crossover is introduced. Matching and recoding of strings assure a unique interpretation of hidden fuzzy terms between both parents. Results obtained for the control of an autonomous vehicle have demonstrated the ability of our method to evolve a fuzzy controller solving the given task.

References

- [1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading Massachusetts: Addison-Wesley, 1989
- [2] C. L. Karr, L. M. Freeman, D. L. Meredith, "Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm", *SPIE Intelligent Control and Adaptive Systems*, vol. 1196, P. 274-288, 1989
- [3] Hartmut Surmann, Andreas Kanstein, Karl Gosser, "Self-Organizing and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems", *EUFIT 93 First European Congress on Fuzzy and Intelligent Technologies*, Aachen 1993, Vol. 2, P. 1097-1104
- [4] K. Kropp, U. G. Baitinger, "Optimization of Fuzzy Logic Controller Inference Rules Using a Genetic Algorithm", *EUFIT 93 First European Congress on Fuzzy and Intelligent Technologies*, Aachen 1993, Vol. 2, P. 1090-1096
- [5] F. Hoffmann, G. Pfister, "Optimierung Hierarchischer Fuzzy Regler mit Genetischen Algorithmen", *4. Dortmunder Fuzzy Tage*, 1994
- [6] Alessandro Saffiotto, Enrique H. Ruspini, Kurt Konolige, "A Fuzzy Controller for Flakey an Autonomous Mobile Robot", *3. Dortmunder Fuzzy-Tage*, Springer-Verlag, 1993, P. 3-12