Learning the Rule Base of a Fuzzy Controller by a Genetic Algorithm

Jörn Hopf and Frank Klawonn

Abstract

For the design of a fuzzy controller it is necessary to choose, besides other parameters, suitable membership functions for the linguistic terms and to determine a rule base.
This paper deals with the problem of finding a good rule base — the basis of a fuzzy controller. Consulting experts still is the usual but time–consuming and therefore rather expensive method. Besides, after having designed the controller, one cannot be sure that the rule base will lead to near optimal control. This paper shows how to reduce significantly the period of development (and the costs) of fuzzy controllers with the help of genetic algorithms and, above all, how to engender a rule base which is very close to an optimum solution.
The example of the inverted pendulum is used to demonstrate how a genetic algorithm can be designed for an automatic construction of a rule base.
So this paper does *not* deal with the tuning of an existing fuzzy controller but with the genetic (re–)production of rules, even without the need for experts. Thus, a program is engendered, consisting of simple "*IF* ... *THEN* ..." instructions.

1 Introduction

In opposition to classical control techniques, which are mainly based on a mathematical model of the process to be controlled, the idea of fuzzy control is to model the behaviour of a (hypothetical) operator who is able to control the process. When the basic principles of the process and the reaction caused by the control actions are well understood, it is possible to design a reasonable control strategy using "*IF* ... *THEN* ..." rules. But in most cases it is necessary to consult an operator who is able to control the process and who has an intuitive understanding of the behaviour of the process. However, the knowledge acquisition process is often very difficult, since the operator is not always aware of all the rules he uses and might not be able to formulate an appropriate description of his control strategy. Therefore, a rule obtained from an operator might not work as well as expected. In some cases it is even impossible

formulate a rule base, e.g. when a new process is to be controlled and no competent operator is available.
In this paper we propose the application of genetic algorithms to the problem of the design of a rule base for a fuzzy controller without the use of a priori information or the help of an operator.
The quality of the controller working with the rule base determined by the genetic algorithm depends on the choice of the evaluation function used in the genetic algorithm. In this way, we can prescribe the desired quality of the fuzzy controller to be generated.
The "IF ... THEN ..." rules appearing in the rule base of a fuzzy controller form an algorithm. Unlike a C–program, such an algorithm can be engendered by a genetic algorithm. The possibility of using an evolutionary process has its roots in the independence of these program modules. In genetic terms: changing the genotype does not affect the determination of the algorithm, what happens is only a phenotypical alteration – a changing of the algorithm's behaviour. The used language of a reduced instruction set code:

program	::	clause [clause ...]
clause	::	**IF** expression **THEN** expression
expression	::	variable **is** fuzzy–set [**and** expression]

Limited to:

clause	::	**IF** expression_1 **THEN** expression_2
expression_1	::	expression_2 **and** expression_2
expression_2	::	variable **is** fuzzy–set

2 The Control Problem

We consider the following control problem known as the inverted pendulum. A pole has to be balanced on a cart to be moved in the horizontal direction. A mass is fixed at the end of the pole. We neglect the frictional resistance here. The goal is to balance the pole by applying some force to the cart accelerating it. The force should be determined by the actual angle and angular velocity of the pole. The movement of the pendulum follows the differential equation

$$(m + M \cdot \sin^2\theta) \cdot l \cdot \ddot{\theta} + M \cdot l \cdot \sin\theta \cdot \cos\theta \cdot \dot{\theta}^2 - (M + m) \cdot g \cdot \sin\theta = -F \cdot \cos\theta$$

where g is the gravitational constant, l the pole length, M the mass at the head of the pole, m the mass at the foot of the pole and $-90° \leq \theta \leq 90°$ is required.
The two parameters to be optimized are the angle θ and the angular velocity $\dot{\theta}$.
We use a Sugeno fuzzy controller (see f.e. [Kruse et al., 1994]) to solve this control problem. The fuzzy sets for the fuzzy controller are denoted as usual with Negative–Big (NB), Negative–Medium (NM), Zero (ZE), Positive–Medium (PM) and Positive–Big (PB). The membership functions of all fuzzy sets are chosen as triangular functions and are uniformly distributed over the universe of discourse. For the output of each rule we chose a fixed value which we also associated with a linguistic expression of the above mentioned type.

3 How a Genetic Algorithm Works

Genetic algorithms represent a strategy to search efficiently for near optimal solutions in difficult search spaces. Each solution is represented by one individual of a whole population. New solutions can be engendered by combining selected old solutions.
To solve a problem, a genetic algorithm requires five components: [Davis, 1987]

1. A representation of the solution to a problem in the form of a chromosome (chromosomal representation).
2. An initial population of individuals (solutions).
3. An evaluation function which indicates the *fitness* of each individual. This fitness shows how well the individual is able to cope with the given environmental factors.
4. Genetic operators which determine which genes of which parent will be passed on to their offsprings in the process of reproduction.
5. Parameters, as there are: the size of the population and probabilities employed by the genetic operators.

A *population* corresponds in our application of genetic algorithms to fuzzy control to a family of rule bases. The *initial population* is chosen randomly.
The two most important genetic operators are *mutation* which means in our application of genetic algorithms to fuzzy control modifying a rule

Figure 1: Chromosomal representation of the solution

base by random, and *crossing over*, a recombination of two 'parent' rule bases. For a detailed discussion of these operators see f.e. [Beightler et al., 1979], [Davis, 1987], [Dewdney, 1986], [Goldberg, 1989], [Holland, 1992] and [Michalewicz, 1992].

3.1 Genetic Encoding of the Possible Solutions

To solve a problem with the help of a genetic algorithm it is first of all necessary to encode a general solution of the problem in a chromosomal representation. This representation has been chosen in correspondence with our 2–dimensional rule panel. Transferred to a 1–dimensional representation a rule base looks as shown in figure 1. Each box stands for one gene and is indexed with the premise of its corresponding rule in figure 1. The possible alleles (values) for each gene are the linguistic expressions for the output value in the rule base of the fuzzy controller. Note that we refrain from a binary representation here.
It is now the task of the genetic algorithm to fill out the rule base in figure 2 in a way that it contains the appropriate rules. The fuzzy controller on the basis of these rules must be able to hold the pole in the upright position both as quickly as possible and with the least possible deviation.

3.2 Generating the Initial Population

Many conventional optimization/search procedures use only one single starting point. Its position in the search space determines the next step which again leads us to another single point. This method, however, incorporates the following problem. When a local optimum has been found the global optimum might remain inaccessible, as the algorithm would have to give up the local optimum found before.
A genetic algorithm starts with a randomly generated initial population of possible solutions, i.e. a genetic algorithm relies on a set of starting points. In our case each chromosome represents a complete rule base for the fuzzy controller and determines the individual's position within the

search space. Some may sit in a low valley (poor solutions to the problem), others may be found on high mountains (good solutions). Thus, to search efficiently, the starting population should be spread evenly over the entire space.
First – and according to expectation – none of the individuals with its chromosomes will be able to solve the problem (holding the inverted pendulum upright) sufficiently. Now, before selecting and reproducing these individuals, the fitness of each individual must be determined by an appropriate evaluation function.

3.3 Evaluation

The evaluation function indicates the fitness of each individual (rule base) of the population and must be designed in such a way that the fittest individuals take part in the process of reproduction.
Along with an appropriate coding the evaluation function decides on the success of the genetic algorithm, see [Beightler et al., 1979], [Davis, 1987], [Dewdney, 1986], [Goldberg, 1989], [Holland, 1992] and [Michalewicz, 1992].
Each genetic algorithm requires a specific choice of the evaluation function taking the following aspects into account.

- For an evaluation function it is essential to assign high values to chromosomes that represent good solutions.
- On the other, an inhomogeneous gene pool with enough variety to find (near) optimal chromosomes (solutions) should be kept. Therefore, if the evaluation function favours good (but by far not optimal) chromosomes too strong against average chromosomes, the genetic algorithm will get stuck at some unsatisfactory solution.

Evaluation in this case is a tightrope walk between supporting those individuals with high fitness and keeping a rich variety in the gene pool.

3.3.1 Evaluation Function for the Inverted Pendulum

In order to compute the value of the evaluation function for a chromosome for the problem of the inverted pendulum we run several times a simulation of the inverted pendulum controlled by the fuzzy controller on the basis of the rule base associated with the chromosome. For each simulation different values for the initial angle and the velocity are chosen. During this simulation score points for the evaluation can be gained by the chromosome taking the following criteria into account.

(↓) angle \ (→) angular velocity					
force	NB	NM	ZE	PM	PB
NB	PM	NB	ZE	NM	PM
NM	NB	NM	NM	ZE	NM
ZE	NB	NM	ZE	PM	PB
PM	NM	ZE	PM	PM	PB
PB	PB	PM	PB	PB	PM

Figure 2: An engendered rule base

- A score can only be gained if the pendulum has been hold upright during the whole simulation.
- At the end of the simulation score points are granted for a small deviation from the upright position. The interpretation of small deviation is narrowed from generation to generation, since in the beginning usually no randomly generated chromosome will be able to hold the pendulum in the upright position.
- The time needed to reach a stable upright position of the pendulum is evaluated only indirectly by the chosen time of simulation.

The controller must be able to handle different initial situations. This is guaranteed by using a number of randomly chosen initial values for the simulation and evaluation. In an illustration, where good solutions are located on mountains and bad in valleys, this means an ever changing landscape. But although it is unlikely that the individuals are confronted with the same fitness landscape twice, an 'optimal' solution is engendered.

3.4 Genetic Operators

Genetic operators simulate changes of chromosomes in nature. Usually two operators are considered, namely cross over and mutation.

3.4.1 Cross Over

The cross over operator mixes the genes of two chromosomes in the phase of reproduction. In genetic algorithms cross over is realized in the following way. First of all pairs of chromosomes are selected from the population, usually randomly proportional to their fitness determined

by the evaluation function. The idea of the cross over operator is to combine the features, especially the positive ones, of the chromosomes by mixing the genes of each pair of chromosomes. In this way new chromosomes are generated that replace their parents. Mixing of genes is achieved choosing one gene randomly — the cross over point — and exchanging the genes of the pair of chromosomes behind this cross over point. In order to illustrate the cross over operator we consider the (binary) chromosomes

1 0 1 0 1 1 0 0 1 0 1 0 1	0 1 1 1 0 0 1 0 0 0 1 0

and

1 0 1 1 0 0 1 0 1 0 0 1 1	0 1 0 1 0 0 1 0 1 0 1 0

The cross over point is chosen behind the 13th gene. Thus the cross over operator yields the following two chromosomes.

1 0 1 0 1 1 0 0 1 0 1 0 1	0 1 0 1 0 0 1 0 1 0 1 0

and

1 0 1 1 0 0 1 0 1 0 0 1 1	0 1 1 1 0 0 1 0 0 0 1 0

Since chromosomes to participate in cross over are chosen randomly, but with a probability proportional to their fitness, better chromosomes have higher chances to produce offspring by cross over.
As a side remark, we should mention that the cross over operator described above is the most simple one but also the one yielding the worst results [Michalewicz, 1992]. Therefore, we preferred in opposition to this one point cross over the two point cross operator where two genes are selected randomly and the gene sequences between these genes are exchanged.

3.4.2 Mutation

Besides cross over the other genetic operator is mutation. Whereas cross over mixes genes of different chromosomes and can in this way combine good solutions, mutation changes genes in one chromosome randomly. The reason for the use of the mutation operator is the following.
Mutation avoids the convergence to a population with a homogeneous gene pool and thus guarantees for a certain variety of genes. It should be emphasized that without mutation chances for a thorough search through the space of possible solutions are quite small. If a certain allele (value) for one of the genes is not present in the population, this allele

cannot be generated by cross over. Therefore mutation is a necessary operator, even if it supports only a random search, not directly aimed to improve individual chromosomes.
Mühlenbein demonstrated [Mühlenbein and Schlierkamp-Voosen, 1993] that in most cases it is sufficient to work with a constant mutation rate (probability for changing one gene) of $R^M = \frac{1}{n}$, where n is the number of genes per chromosome.

3.4.3 The Building Block Hypothesis

The schema theorem (see f.e. [Michalewicz, 1992]) indicates that a genetic algorithm concentrates the search for an optimal solution on certain subspaces of the space of all possible solutions. These subspaces are characterized by schemata, chromosomes with undetermined genes. The subspace associated with a schema corresponds to the set of all genes that have the same alleles for those genes that are determined in the schema.
The building block hypothesis, derived from the schema theorem, states that a genetic algorithm mainly searches in those subspaces that are characterized by schemata with a short defining length, low order and high fitness. The defining length of a schema is the distance between the two outmost determined genes in a chromosome. The order of a schema is the number of determined genes. The fitness of a schema is the average fitness of all chromosomes in its corresponding search space. For our problem – finding a suitable rule base for a fuzzy controller – it is easy to see that the building block hypothesis is applicable. If a rule base should be able to handle a certain situation, only a small subset of neighbouring rules is needed. Since the coding of the chromosomes maintains at least partly this neighbourhood relation, (partially) good rule bases correspond to schemata with a short defining length, low order and high fitness. Therefore, it is possible to generate an overall good rule base from two partially good rule bases by using the cross over operator.

4 Simulation Results

The best rule base (chromosome) obtained after 33 generations with a population size of 200 is illustrated in figure 2. Starting the simulation with an upright standing pendulum but with a high initial angular velocity the fuzzy controller using this rule base is able to balance the pole finally with a deviation of constantly less than one degree. The protocol

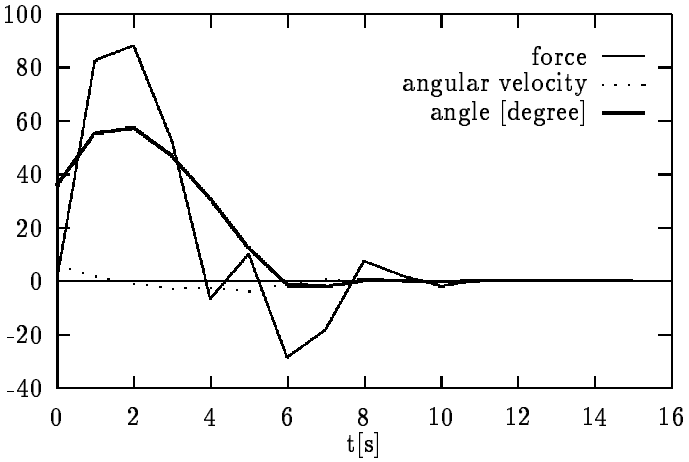

Figure 3: Change of angle, angular velocity and force due to a start impulse of 6.6 out of the neutral position.

of the simulation run is presented in figure 3, where the values for the angle, angular velocity, and the force are shown over the time. Note that this result is obtained by learning the rule base alone. The initial fuzzy sets remained unchanged.
The appendix shows a comparison between a fuzzy controller based on Lukasiewicz logic [Klawonn, 1992] designed by hand and the fuzzy controller obtained from the genetic algorithm.
Note that the genetic algorithm does only rely on the fitness of a rule base. Therefore, it is possible that the genetic algorithm finds a well working rule base that does not coincide with the rule base one would write down from an intuitive point of view. This is one of the reasons why the rule base in figure 2 is not symmetric. Some rules also nearly never apply and the entry in the table for these rule is not so important for the overall evaluation or fitness of the chromosome. This might also lead to deviations from the intuitively appealing rule base.

References

C. S. Beightler, D. T. Phillips and D. J. Wilde (1979). *Foundations of Optimization*. Prentice-Hall, Englewood Cliffs, NJ, 2. edition.

L. Davis, ed. (1987). *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann, Los Altos, Ca.

A. K. Dewdney (1986). *Computer-Kurzweil*. Spektrum der Wissenschaft, pages 4–11.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison–Wesley.

J. H. Holland (1992). *Genetische Algorithmen*. Spektrum der Wissenschaft, pages 44–51.

F. Klawonn (1992). *On a Łukasiewicz Logic Based Controller*. In *Proc. MEPP'92 International Seminar on Fuzzy Control through Neural Interpretations of Fuzzy Sets*, number 14 Ser. B. in *Reports on Computer Science & Mathematics*, pages 53–56, Turku, Finland. Åbo Academi.

R. Kruse, J. Gebhardt and F. Klawonn (1994). *Foundations of Fuzzy Systems*. Wiley, Chichester.

Z. Michalewicz (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin.

H. Mühlenbein and D. Schlierkamp-Voosen (1993). *Optimal Interaction of Mutation and Crossover in the Breeder Genetic Algorithm*. Technical Report, GMD, Bonn, Germany.

R. Sommer (1992). *Entwurf und Implementierung eines auf Łukasiewicz-Logik basierenden Fuzzy Controllers*. Studienarbeit, Institut für Betriebssysteme und Rechnerverbund, Technische Universität Braunschweig.

Appendix

The following diagrams compare the action of a fuzzy controller on the basis of Łukasiewicz logic *(left)* [Sommer, 1992] with that of the controller engendered by a genetic algorithm *(right)*.
Initial values: angle = 40.0°, angular velocity = −2.0