A Hybrid Evolutionary Search Concept for Data–based Generation of Relevant Fuzzy Rules in High Dimensional Spaces

Extended Version[1]

T. Slawinski, A. Krone, M. Lindenblatt, P. Krause

Faculty of Electrical Engineering, University of Dortmund, D-44221 Dortmund
Tel.: +49 231 755 4524, Fax: +49 231 755 2752
e–mail: slawinski@esr.e-technik.uni-dortmund.de

U. Hammel, D. Wiesmann

Faculty of Computer Science, University of Dortmund, D-44221 Dortmund
Tel.: +49 231 755 900 977, Fax: +49 231 9700 959
e–mail: hammel@LS11.informatik.uni-dortmund.de

Abstract

In this paper we propose a hybrid fuzzy–evolutionary system for fuzzy modelling in high dimensional spaces. The system architecture is based on a Michigan–style approach (one individual represents one fuzzy rule). The design of the evolutionary algorithm makes use of a distance measure in the search space that in turn reflects some heuristic assumptions about the fitness landscape. Additionally, strategy parameters are dynamically adapted by means of a fuzzy controller. The approach is successfully applied to a complex benchmark problem as well as to several real–world modelling tasks such as the cancellation behaviour of insurance clients and the classification of automatic gearboxes.

1 Introduction

The applicability and acceptance of data–based fuzzy modelling methods are limited mainly by two factors. First, their acceptance depends very much on the method's ability to generate small, transparent rule bases. Second, the computing time must be acceptable even in the presence of many linguistic variables and values.

The Fuzzy–ROSA[2] method [1, 2, 3] was developed to meet these goals. Several successful applications in the domains of fuzzy modelling, robotic control, classification and prediction [4, 5, 6, 7, 8] show the usefulness of this concept, which is briefly described in Section 1.1.

In order to achieve reasonable computation times in high dimensional spaces, rule searches in the Fuzzy–ROSA method are based on an evolutionary algorithm (EA). In Section 1.2, this EA is compared to other approaches on evolutionary data–based rule generation from the literature. A novel implementation of an evolutionary algorithm based on a distance measure in the set of fuzzy rules is described in Section 2. This concept is elaborated in Section 3 yielding a hybrid adaptive EA. Finally, in Section 4 two practical applications are presented.

1.1 Fuzzy–ROSA Method

The basic idea of the Fuzzy–ROSA method is to apply a statistical relevance test to fuzzy rules. The test assess the potential of single fuzzy rules to describe a relevant aspect of the system under consideration

[1] Additional passages are printed in cursive and additional sections are marked by $^+$

[2] **R**ule**O**rientated **S**tatistical **A**nalysis

[9, 10, 11]. This reduces the problem of finding a good rule base to the problem of finding single relevant rules. On the other hand, since each rule with high relevance is supposed to express an important aspect of the system, such rules are meaningful by themselves, which leads to more transparent and comprehensible rule bases.

The Fuzzy–ROSA method can also deal with generalizing (incomplete) rules, where the length of premises may differ, i.e., consist of a variing number of linguistic expressions (Section 2.2.1). If a single rule is composed of fewer expressions than the total number of input variables, the rule covers several linguistic input situations. In particular, for large search spaces with a large number of input variables, such rule bases turn out to be much smaller than rule bases consisting of complete rules only.

The Fuzzy–ROSA approach does not aim at locating the global optimum (which is in the general case not achievable in polynomial time and therefore impractical for the majority of applications), but at finding satisfactory solutions in an acceptable time. Therefore, the rule generation process is divided into four main steps. There are alternative strategies available for each step, so that the method can be adapted to different application requirements (e.g., for modelling, classification, approximation or prediction) and problem sizes (e.g., numbers of variables, linguistic values and data sets).

Project Definition: Prior to rule generation, the membership functions of the input and output variables of the system under consideration are extracted by cluster analysis (e.g., [12]), heuristics or domain knowledge. A maximal combination depth of linguistic statements in the premise must be chosen to reduce the computational effort.

Rule Generation: Depending on the size of the search space, a complete search, an evolutionary search [13], or a combination of both is selected. The rule base is generated by iteratively collecting relevant rules.

Off–line Rule Reduction: The number of generated rules is reduced by off–line rule reduction methods [14] that meet different requirements, e.g., completely covering all input situations, restricting the modelling error or following a strictly bounded number of rules.

Analysis and Optimization: The analysis of the rule base allows an assessment of the modelling process as well as of model quality and provides feedback for problem formulation. Finally, the input–output behaviour of the fuzzy system obtained is optimized by adjusting the remaining free parameters.

1.2 Evolutionary Search Concepts for Data–based Fuzzy Modelling

In the literature, we find three main application areas of evolutionary algorithms in the field of fuzzy modelling: optimization of membership functions (e.g., [15]), optimization (generation) of rules (e.g., [16]) and simultaneous optimization of both (e.g., [17]).

In the case of rule base optimization (generation), most evolutionary algorithms use a fixed rule base structure of complete rules (e.g., [17]), where one individual in the evolving population represents a whole rule base (Pittsburgh style [18, 19]). Consider a fairly small example with six input variables and one output variable, with five linguistic values each. This leads to 15,625 different linguistic input situations with one rule for each input case. Obviously, it is almost impossible to interpret such a rule base. On the other hand, the size of the associated search space grows exponentially with the number of input situations (in the above example more than 10^{10000} different rule bases with complete rules). For these reasons, it is always time-consuming and often impossible to efficiently identify high quality rule base using this approach. In the case of simultaneous optimization of membership functions and rules the situation becomes even worse. Therefore, the Pittsburgh style is often not practicable for more than three or four variables and more complex applications are rarely published.

Contrarily, in our approach, which is depicted in Figure 1, each individual represents a single fuzzy rule (Michigan style [18, 19]). While the evolutionary algorithm is searching for the 'best' rule, many good

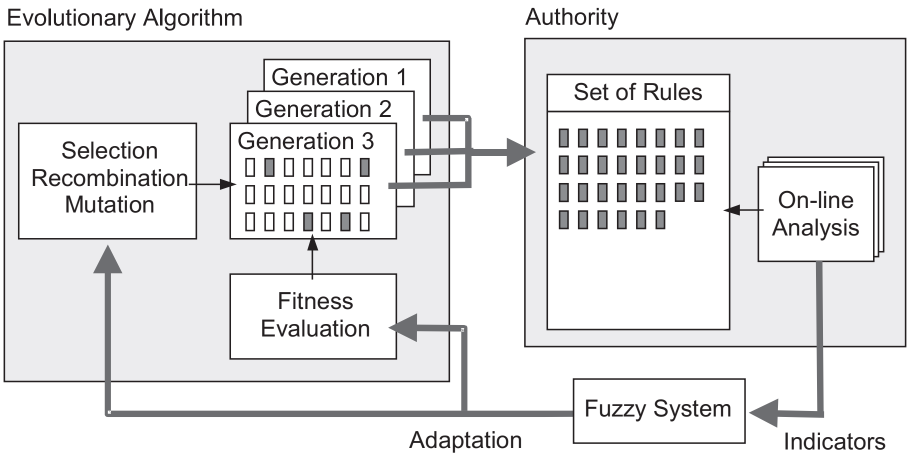


Figure 1: Hybrid Evolutionary Search Concept

rules are generated. The basic idea is to collect these good (relevant) rules during the course of evolution (Figure 1).

In comparison to other approaches that also take single rules as individuals (e.g., [17, 16]), our concept is based on one population and a single evolution process.

Since the Fuzzy–ROSA approach makes use of generalizing (incomplete) rules, a sophisticated genetic representation and extended genetic operators had to be developed (Section 2.3).

As described above, the fitness evaluation of individuals is based solely on the relevance of single rules. Thus, the overall quality of the rule base collected at the current generation does not influence the evolutionary search process directly. In some situations this might lead to a rule base consisting of a few rules of high relevance but poor overall performance. Consequently, additional measures must be introduced in order to avoid premature stagnation and to keep the algorithm exploring the search space. This can be achieved by dynamically adapting some of the strategy variables depending on the performance of the current rule base collected during the course of evolution. The implementation is based on a hybrid fuzzy–evolutionary approach (Section 3).

2 Evolutionary Fuzzy Rule Search

2.1 Design of the Evolutionary Algorithm

A common approach for the design of evolutionary algorithms is to choose an instance from the set of "canonical" evolutionary algorithms, such as a genetic algorithm (GA) or an evolution strategy (ES)[20]. This requires a mapping, which translates the real–world problem formulation into the standard genetic representation of the choosen algorithm. A disadvantage of this mapping between the genotype and phenotype spaces is that the effects of genetic operators (mutation and recombination) on fitness variations are difficult to determine. An ill-defined mapping may introduce a bias (independent of selection) into the search process as well as a weakening of the causality between variations of the genotype and the corresponding fitness values [21, 22].

In order to avoid these difficulties, the design of the evolutionary search is based on the concept of the Metric Based Evolutionary Algorithm (MBEA)[23]. The basic idea of the MBEA is to express domain knowledge as a metric on the phenotype space, such that similar individuals (according to the metric) have similar fitness values. Depending on the mapping, this metric must be translated to the genotype space. Based on this metric, a set of formal requirements for the genetic mutation and recombination operators can be defined, e.g., bias free operation, locality (small mutations occur more frequently than large ones), reachability (any point in the search space can be reached from any other point in a finite sequence of mutation steps) and feature preservation (the distance from parents to offspring is limited).

For our application, it is difficult if not impossible to meet all the requirements of the MBEA, due to the complexity of the search space, i.e., "What is the distance between two fuzzy rules?". On the other hand, there are some reasonable heuristic assumptions about the effect of modifications of a fuzzy rule, e.g., a change of a linguistic value is assumed to have a smaller effect than a change of a linguistic variable. These heuristic assumptions are implemented as probabilities of certain mutation events.

2.2 Heuristic Distance Measure for Fuzzy Rules$^+$

The heuristic distance measure (HDM) for fuzzy rules, presented in this section, was developed for two reasons. First, for the design of a MBEA it is indispensable to quantify the effects of modifications of the individuals (fuzzy rules). Second, the distance between individuals can be used as a measure for the diversity of the population (Section 3.2.1). The introduction of the HDM is structured as follows. In Section 2.2.1 the structure of fuzzy rules with varying combination depth is briefly described. Then an extended distance measure for linguistic expressions is proposed (Section 2.2.2). Finally in section 2.2.3, a kind of average distance between the linguistic expressions of two rules is introduced as HDM.

2.2.1 Structure of Rules

In the Fuzzy–ROSA system IF–THEN–rules have the following form

$$\text{IF} \quad P \quad \text{THEN} \quad C \qquad \text{with} \qquad P = e_1 \wedge \cdots \wedge e_d \quad , \tag{1}$$

where the premise part P consists of a conjunction of d linguistic expressions e_i and C denotes the conclusion. The number of linguistic expressions d is called the *combination depth*. Given the input variable v_i with a corresponding linguistic value w_i and the output variable y with a corresponding linguistic value w_c, the linguistic expressions e_i and the conclusion C are defined by

$$e_i = (v_i, t_i, w_i, n_i) \qquad \text{and} \qquad C = (w_c) \quad , \tag{2}$$

where e_i is interpreted as

$$v_i(T - t_i) = \begin{cases} w_i & \text{if} \quad n_i = 0 \\ \neg w_i & \text{if} \quad n_i = 1 \end{cases} \quad . \tag{3}$$

The Fuzzy–ROSA method allows to include the history of the variable v_i which is reflected by the time delay $t_i \in \mathbb{N}$, and $T \in \mathbb{N}$ denotes the current time. Depending on the value of flag $n_i \in \{0, 1\}$ expression e_i is interpreted as *"if variable v_i at time $(T - t_i)$ is equal to w_i"* in the case of $n_i = 0$ and *"if variable v_i at time $(T - t_i)$ is not equal to w_i"* in the case of $n_i = 1$.

Likewise the conclusion C is interpreted as

$$y(T) = w_c \tag{4}$$

The Fuzzy–ROSA method aims at identifying non contradictory premises. Therefore, the linguistic expressions of a single premise have to differ in either the linguistic variable or time depth.

Definition 1 (Combination Restriction) *For a given premise P the following condition must hold:*

$$\forall e_i, e_j \in P \mid i \neq j \Rightarrow v_i \neq v_j \vee t_i \neq t_j \tag{5}$$

2.2.2 Distance between Linguistic Expressions

In the literature several measures for the distance between linguistic expressions (fuzzy sets) can be found. In order to meet all requirements of the Fuzzy–ROSA method, i.e. negated linguistic expressions, the measure proposed in [24] has to be extended:

Definition 2 (Distance between linguistic expressions) *The distance $\Delta(e,e')$ between two linguistic expressions e and e' for the same linguistic variable v and time t is defined as:*

$$\Delta(e,e') = \begin{cases} \dfrac{\mid w - w' \mid}{s_v - 1} & , \textit{if } \ n = n' \\ 1 - \dfrac{1}{s_v - 1} & , \textit{if } \ n \neq n' \end{cases} \tag{6}$$

with

$$\begin{aligned} e &= (v, t, w, n) \\ e' &= (v, t, w', n') \\ s_v \in \mathbb{N} &: \textit{number of linguistic values of variable } v \end{aligned}$$

In the first part of Equation 6 the distance $\mid w - w' \mid$ between the linguistic values is normalized by the maximum distance $(s_v - 1)$ (Figure 2 A).

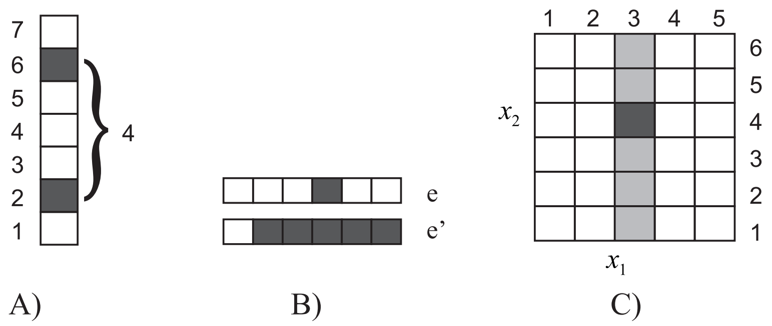


Figure 2: Heuristic Distance Measures

The case $n \neq n'$ is somewhat more complicated, since a negated linguistic expression covers more linguistic input situations compared to a positive linguistic expression (e.g. $n = 0$ and $n' = 1$ in Figure 2 B). It seems reasonable to assume, that larger differences in the number of covered input situations should lead to a larger distances. Therefore in the second part of Equation 6 the ratio of covered input situations $1/(1 - s_v)$ is subtracted from one. The following cases must be considered separately:

$s_v = 1$	:	does not appear, because negation is not possible
$s_v = 2$	:	does not appear, because instead of negation the linguistic value can be changed
$s_v \gg 1$	:	the distance is maximal $\Delta(e,e') = (1 - \frac{1}{s_v - 1}) \approx 1$

2.2.3 Distance between Fuzzy Rules

In the evolutionary search of the Fuzzy-ROSA method the fitness for a premise P is computed as the maximum fitness of all valid fuzzy rules with premise P [13]. Therefore, we may neglect the conclusion C when computing the distance of two fuzzy rules.

Since the Fuzzy–ROSA method can deal with generalizing rules an additional measure must be defined for the distance between a single linguistic expression and a generalizing premises. Take for instance the following two rules (see also Figure 2 C):

rule 1:	*IF* $(x_1 = 3) \wedge (x_2 = 4)$	*THEN* ...
rule 2:	*IF* $(x_1 = 3)$	*THEN* ...

Rule 2 is more general than *rule 1*, because the premise of *rule 2* holds for all linguistic values of the variable x_2. Analogous to the approach for the negated linguistic expressions in Equation 6, the definition of the distance is based on the ratio of the covered input situations.

Definition 3 (Distance between a linguistic expression and a generalizing premise) *The distance Δ_G of a linguistic expression $e = (v, t, w, n)$ to a generalizing premise, which covers all input situations specified by the linguistic variable v and the time t is calculated as follows:*

$$\Delta_G(e) = \begin{cases} 1 - \dfrac{1}{s_v} & \text{, if } n = 0 \text{ (e is not negated)} \\ \dfrac{1}{s_v} & \text{, if } n = 1 \text{ (e is negated)} \end{cases} \tag{7}$$

with $s_v \in \mathbb{N}$: number of linguistic values of variable v.

Consequently, a larger difference in the number of covered input situations leads to a larger distance Δ_G. The following cases must be considered separately:

$$\begin{array}{rcll} s_v = 1 & : & \text{no generalization possible, distance } \Delta_G = 0 & \\ s_v = 2 & : & \text{distance } \Delta_G = 0.5 & \\ s_v \gg 1 & : & \text{the distance is maximal } \Delta_G(e) = (1 - \frac{1}{s_v}) \approx 1 & \text{, if e is not negated} \\ & & \text{the distance is minimal } \Delta_G(e) = (\frac{1}{s_v}) \approx 0 & \text{, if e is negated} \end{array}$$

Based on Definition 2 and 3 the HDM is calculated as an average over the distance between the linguistic expressions of two premises.

Definition 4 (Distance between premises) *The distance D between two premises P_1 and P_2 is defined as follows:*

$$D(P_1, P_2) = \frac{\sum\limits_{e,e' \in P_{eq}} \Delta(e, e') + \sum\limits_{e'' \in P_{df}} \Delta_G(e'')}{\mid P_{eq} \mid + \mid P_{df} \mid}$$

where

$$\begin{array}{rl} P_{eq} := \{\, e_1 = (v_1, t_1, w_1, n_1) \mid & e_1 \in P_1 \\ & \wedge\ (\exists (v_2, t_2, w_2, n_2) \in P_2 : v_2 = v_1 \wedge t_2 = t_1)\} \\ P_{df} := \{\, e_1 = (v_1, t_1, w_1, n_1) \mid & (e_1 \in P_1 \vee e_1 \in P_2) \wedge \\ & (\forall (v_0, t_0, w_0, n_0) \in P_{eq} : v_0 \neq v_1 \vee t_0 \neq t_1)\} \end{array}$$

To summarize, D is the sum of the distance between linguistic expressions, which are equal in the linguistic variable v and time t, and the distance of the remaining linguistic expressions after Definition 3 divided by the number of addends.

2.3 Genetic Representation and Genetic Operators^{+}

As described in the previous section the search space consists of all feasible premises P of the form $P = e_1 \wedge \cdots \wedge e_d$. Feasibility in the sense of Definition 1 has to be preserved by the genetic operators.

Genetic operators like mutation and recombination modify the genetic information through primitive operations such as bit flips or concatenation in the case of binary strings. By the term *genetic representation* (of the search space) we regard an abstract datatype which provides these primitive operators. Thus, the design of the genetic operators depends on the algebraic properties of the genetic representation, such as arithmetic operators and order relations. Therefore, the efficiency of the search process strongly depends on the choice of the genetic representation as well[3].

The approach suggested here is based on the recent work of Krone and Kiendl [13], where the authors proposed an Integer–GA–like search strategy. But a careful review showed some deficiencies of this algorithm due to the properties of the underlying representation. In the following we describe some aspects of the original algorithm since this illustrates some pitfalls of EA-design frequently found in the literature.

Each individual of the original algorithm was defined as a list of linguistic expressions $\overline{e}_i = (\overline{g}_i, \overline{d}_i, \overline{n}_i, \overline{t}_i)$, where $\overline{g}_i$ is called an *event* at time $\overline{t}_i$, and $\overline{d}_i$ and $\overline{n}_i$ are two flags for activation and negation of the linguistic expression $\overline{e}_i$. A sample individual is depicted in figure 3.

linguistic expression 1					linguistic expression d			
$\overline{g}_1$	$\overline{d}_1$	$\overline{n}_1$	$\overline{t}_1$	$\cdots$	$\overline{g}_d$	$\overline{d}_d$	$\overline{n}_d$	$\overline{t}_c$

Figure 3: Representation of a premise (individual)

The events $\overline{g}_i$ carry two kinds of information: The variable name and the linguistic value. Since the set of variables as well as the set of linguistic values are finite this is achieved by just ordering the set of all possible events in a somewhat arbitrary way, like in the following example:

variable		value	index
temp	=	*cold*	1
temp	=	*hot*	2
pres	=	*low*	3
pres	=	*medium*	4
pres	=	*high*	5
cur	=	*low*	6
cur	=	*high*	7

This means that the genotype-phenotype mapping does not preserve similarities since similar values of $\overline{g}_i$, e.g., $\overline{g}_i = 2$ and $\overline{g}_i = 3$, have completely different interpretations in the phenotype-space.

Mutations on the $\overline{g}_i$ are performed by addition of a random value, where small changes occur more frequent than large ones, i.e., the mutation operator makes use of the arbitrary order on the set of events. Thus, the disruptiveness of even the smallest mutation is completely different depending on the current value of $\overline{g}_i$ and the order on the set of events.

[3]It is also true, that the efficiency of the primitive operators depending on the chosen datastructure has a great impact on the overall performance. I.e., if the runtime of primitive operations scale exponentially with, say, the cardinality of individuals this would limit the applicability of the design just as a poor design of the genetic operators.

Recombination on the $\overline{g}_i$ is performed by averaging the values at corresponding positions of the parents. Again, recombination does not preserve the properties of the parents, since the interpretation of the offspring might be completely different compared to the parents. Additionally, recombination introduces a strong bias to values of $s/2$ if s is the number of states.

Finally, the genetic operators do not guarantee the feasibility of premises according to definition 1 of the previous chapter. Since there is a large set of infeasible premises the algorithm spends a lot of time in evaluating and discarding infeasible solutions.

Motivated by this observations and in order to obtain transparent strategy parameters for the fuzzy-adaptation, we decided for a redesign of the genetic representation and operators. The redesign was based on the design rules for the MBEA proposed by Droste und Wiesmann [23].

2.3.1 New Representation based on Relational Algebra

The elements of the search space, i.e. the set of feasible premises, have the following properties:

Not ordered: A premise is a conjunction of linguistic expressions. Since the conjunction is commutative there is no inherent order on the set of linguistic expressions of a fuzzy rule.

Uniqueness: The linguistic expressions are unique in the sense that they can be unambiguously distinguished by the values of their components.

Varying length: The length of the premise, i.e. the number of linguistic expressions, is not constant.

Combination Restrictions: Contradictions of the Form IF $(v(t) = w)$ AND $(v(t) = w')$ THEN ..., where $w \neq w'$ are infeasible.

The first step in the design process was to define an abstract datatype, revealing these and only these properties. (Remember, that the reason for the weakness of the original EA resulted from introducing an artificial ordering on the set of events, which in turn influenced the design of the genetic operators.)

Therefore, we define a premise as an unordered set of linguistic expressions. The feasibility of the premises are guaranteed by introducing special key conditions. The following specification is based on relational algebra, gleaned from database design[4].

Definition 5 (Universe μ) *The universe $\mu(E)$ denotes the set of all feasible objects of type E.*

Definition 6 (State σ) *The state $\sigma(E)$ denotes a set of objects of type E, where $\sigma(E) \subseteq \mu(E)$ and $\sigma(E)$ is finite.*

Definition 7 (State σ_i) *The state $\sigma_i(E)$ denotes the i-th set of objects of type E, where $\sigma_i(E) \subseteq \mu(E)$ and $\sigma_i(E)$ finite.*

Definition 8 (Type R : linguistic expression) *The object type R describes the structure of the object "linguistic expression". The structure consists of several object attributes. The relation R has the form:*

$$R(\underline{v : D_v, t : D_t}, w : D_w, n : D_n) \tag{8}$$

[4]Please do not confuse the meaning of the letters μ (universe) and σ (state) with the meaning of μ (population size) and σ (step size) in the traditional notation for evolutionary algorithms.

Here v, t, w, n denote the linguistic variable, the time depth, the linguistic value and the negation flag, respectively, and $D_v, \ldots, D_n$ denote the corresponding data structures. The underlined attributes in (8) are treated as key conditions.

A tuple $e = (v, t, w, n) \in \mu(R)$ denotes an object of type R, i.e. a linguistic expression.

Definition 9 (Key condition) *Given $R(v, t, w, n)$. $\{v, t\}$ is a key condition of R if*

$$\forall\sigma\forall e, e' \in \sigma(R) : e \neq e' \Rightarrow v \neq v' \vee t \neq t', \quad \textit{where } e = (v, t, w, n) \textit{ and } e' = (v', t', w', n')$$

The key condition prohibits the existence of two linguistic expressions with identical v and t in a single premise $\sigma(R)$.

Definition 10 (Premise in the evolutionary search) *The i-th set of linguistic expressions (of type R) is a premise $\sigma_i(R)$ in the evolutionary search, where $\sigma_i(R) \subseteq \mu(R)$ and $\sigma_i(R)$ finite.*

In figure 4 the relationship of universe $\mu(R)$, premise $\sigma_i(R)$ and a single linguistic expression of type R is illustrated.

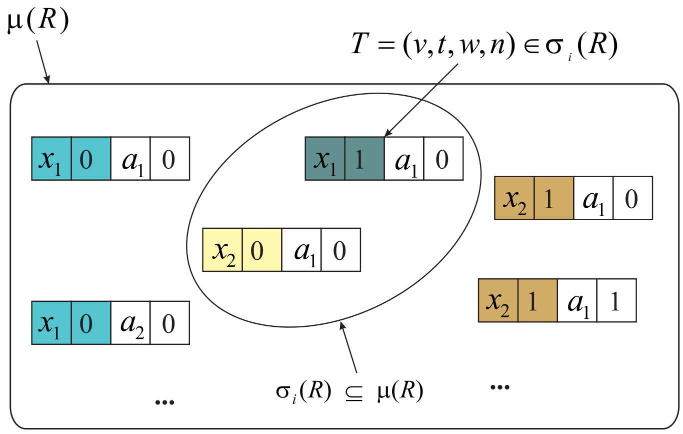


Figure 4: Relationship of universe, premises and linguistic expressions.

While in the universe $\mu(R)$ elements with identical key values exist, all key values in a single premise are unique.

Definition 11 (Individual and rule) *An individual I which represents a rule, consits of a premise $\sigma_i(R)$ for which the key condition holds as well as a conclusion $a \in \mu_c(R)$. The set $\mu_c(R)$ contains all linguistic expressions of the conclusion.*

To summarize, the premises P are defined as sets where additional key conditions have to be fulfilled. Thus, the definition of genetic operators have to be based on set operations:

Definition 12 (Basic operations for modification of a premise $\sigma_i(R)$) *Let $e, e_1, e_2 \in \mu(R)$.*

$$\begin{aligned} \textit{insert}(R, i, e) &: \sigma_{i,new}(R) = \sigma_{i,old}(R) \cup \{e\} & (9)\\ \textit{delete}(R, i, e) &: \sigma_{i,new}(R) = \sigma_{i,old}(R) - \{e\} & (10)\\ \textit{change}(R, i, e_1, e_2) &: \sigma_{i,new}(R) = [\textit{delete}(R, i, e_1); \textit{insert}(R, i, e_2)] & (11) \end{aligned}$$

The genetic operators defined in the following sections guarantee that the key conditions hold. Thus the key conditions are not checked by the basic operators.

2.3.2 Mutation

In accordance to the design rules given in [23] the mutation operator should have no bias, and every element of the search space should be reachable by a finite sequence of mutation steps. Furthermore, small mutations have to occure more frequently than large ones. To measure the distance between two rules, we use the heuristic distance measure for fuzzy rules presented in section 2.2.

To obtain mutation steps of different length, we defined five basic mutation operators:

Mutation of value (changeValue): changes the value of a linguistic expression

Mutation of variable (changeVariable): changes the linguistic variable of a expression

Mutation of structure (addTerm, deleteTerm): adds or deletes a linguistic expression

Mutation of negation (negate): changes the negation flag of a linguistic expression

Mutation of time depth (changeTime): changes the time depth of a linguistic expression

During a single mutation step these operators are activated with different probabilities reflecting the disruptiveness of different mutation events. A similar approach is proposed in [25].

Definition 13 (mutation) *The mutation can be described by the 9-tupel:*

$$\omega_M = (f_{Num}, f_{Scale}, n_{mut}, p_{cVal}, p_{neg}, p_{addT}, p_{delT}, p_{cVar}, p_{cT})$$

with

$$\begin{array}{lcll} f_{Num} & : & \mathbb{N}_0 \to \mathbb{N}_0 & \textit{determine the number of basic mutations} \\ f_{Scale} & : & [0,1] \to [0,1] & \textit{determine the mutation strength} \end{array}$$

and $p_{cVal}, p_{neg}, p_{addT}, p_{delT}, p_{cVar}, p_{cT}$ denote the probabilities for the activation of the basic mutation operators. The parameter n_{mut} controls the number of basic mutations f_{Num} as well as the probability distribution of f_{Scale}. The function f_{Scale} is used by the value-mutation only and computes the variance of the distribution by which the linguistic value is determined.

One complete mutation cycle works as follows

```
for each Individual do
  determine num;            // Number of basic mutation steps
  for i=0 to num do
    determine f_mutate;     // selection of one of the basic mutation operators:
                            // changeValue, addTerm, deleteTerm,
                            // changeTime, changeVariable, negate
    f_mutate Individual;    // mutation
  done
done
```

Figure 5: The mutation cycle

After extensive experimentation the following mutation probabilities have been choosen:

p_{cVal}	p_{neg}	p_{addT}	p_{delT}	p_{cVar}	p_{cT}
0.3	0.1	0.2	0.1	0.2	0.1

Definition 14 (mutation of value: changeValue) *The mutation of a linguistic value w is performed by randomly choosing one of the linguistic expressions $e = (v, t, w, n)$ from the premise $\sigma_i(R)$ and replacing the linguistic value w by w', which results in the new premise $\sigma_{i,new}(R)$.*

$$\sigma_{i,new}(R) = \begin{cases} change(R, i, e, e') & , \text{if } \mu_w \neq \emptyset \wedge e' \in \mu_w \\ \sigma_i(R) & , \text{otherwise (mutation infeasible)} \end{cases}$$

and

$$\mu_w := \{ \ e' = (v', t', w', n') \mid \quad e' \in \mu(R) \wedge v' = v \wedge t' = t \wedge n' = n \wedge w' \neq w \ \}$$

The expression e' is randomly choosen from μ_w according to a discretized Gaussian distribution with mean w, and the variance is computed by the function f_{Scale}.

Since the operation has no impact on v and t the key condition still holds.

Definition 15 (mutation of negation flag: negate) *The mutation of the negation flag is performed by randomly choosing one of the linguistic expressions $e = (v, t, w, n)$ from the premise $\sigma_i(R)$ and flipping the negation flag, which results in the new premise $\sigma_{i,new}(R)$.*

$$\sigma_{i,new}(R) = \begin{cases} change(R, i, e, (v, t, w, 1)) & , \text{if } n = 0 \\ change(R, i, e, (v, t, w, 0)) & , \text{if } n = 1 \end{cases}$$

Since the operation has no impact on v and t the key condition still holds.

Definition 16 (mutation of structure: addTerm) *The expansion of a premise is performed by inserting a randomly choosen feasible linguistic expression e into the premise $\sigma_i(R)$, such that the key condition holds.*

$$\sigma_{i,new}(R) = \begin{cases} insert(R, i, e) & , \text{if } |\sigma_i(R)| < c_{max} \\ \sigma_i(R) & , \text{otherwise} \end{cases}$$

where e is randomly choosen from μ_{vt} with uniform probability, and

$$\mu_{vt} := \{ \ e' = (v', t', w', n') \mid \quad \begin{array}{l} e' \in \mu(R) \ \wedge \\ \forall \ (v^0, t^0, w^0, n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t' \ \} \end{array}$$

The operation is permitted only if the lenght of the premise doesn't exceed c_{max}. The key condition holds by definition of μ_{vt}.

Definition 17 (mutation of structure: deleteTerm) *Delete the linguistic expression e from the premise $\sigma_i(R)$:*

$$\sigma_{i,new}(R) = \begin{cases} delete(R, i, e) & , \text{if } |\sigma_i(R)| > 1 \\ \sigma_i(R) & , \text{otherwise} \end{cases}$$

where e is randomly choosen from $\sigma_i(R)$ with uniform probability.

The operation is permitted only if the premise will not become empty.

Definition 18 (mutation of variable : changeVariable) *The mutation of a linguistic variable v is performed by randomly choosing a linguistic expression $e = (v, t, w, n)$ from the premise $\sigma_i(R)$ and exchanging the linguistic variable v to v', which results in the new premise $\sigma_{i,new}(R)$.*

$$\sigma_{i,new}(R) = \begin{cases} change(R, i, e, e') & , \textit{if } \mu_v \neq \emptyset \wedge e' \in \mu_v \\ change(R, i, e, e') & , \textit{if } \mu_{vt} \neq \emptyset \wedge e' \notin \mu_v \wedge e' \in \mu_{vt} \\ \sigma_i(R) & , \textit{otherwise (mutation infeasible)} \end{cases}$$

and

$$\begin{aligned} \mu_v := \{\ e' = (v', t', w', n') \mid \quad & e' \in \mu(R) \wedge t' = t \wedge n' = n\ \wedge \\ & \forall\ (v^0, t^0, w^0, n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t'\ \} \\ \mu_{vt} := \{\ e' = (v', t', w', n') \mid \quad & e' \in \mu(R)\ \wedge \\ & \forall\ (v^0, t^0, w^0, n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t'\ \} \end{aligned}$$

By definition of μ_v and μ_{vt} the key condition holds.

Definition 19 (mutation of time depth: changeTime) *The mutation of the time depth t is performed by randomly choosing a linguistic expression $e = (v, t, w, n)$ from the premise $\sigma_i(R)$ and modification of the time depth t, which results in the new premise $\sigma_{i,new}(R)$.*

$$\sigma_{i,new}(R) = \begin{cases} change(R, i, e, e') & , \textit{if } \mu_t \neq \emptyset \wedge e' \in \mu_t \\ change(R, i, e, e') & , \textit{if } \mu_{vt} \neq \emptyset \wedge e' \notin \mu_t \wedge e' \in \mu_{vt} \\ \sigma_i(R) & , \textit{otherwise (mutation infeasible)} \end{cases}$$

and

$$\begin{aligned} \mu_t := \{\ e' = (v', t', w', n') \mid \quad & e' \in \mu(R) \wedge v' = v \wedge n' = n \wedge w' = w\ \wedge \\ & \forall\ (v^0, t^0, w^0, n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t'\ \} \\ \mu_{vt} := \{\ e' = (v', t', w', n') \mid \quad & e' \in \mu(R)\ \wedge \\ & \forall\ (v^0, t^0, w^0, n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t'\ \} \end{aligned}$$

Again, by definition of μ_v and μ_{vt} the key condition holds.

2.3.3 Recombination

Recombination is performed by selection of two parents and combination of their genotypes. After selection, which is done either fitness-proportional or with equal probability, the premises of the parents are copied unchanged into the offsprings genotype with two exceptions:

1. If the maximum length c_{max} is exceeded c_{max} elements are randomly chosen from both parents with equal probability.
2. If two linguistic expressions have identical keys the linguistic value is either taken from one of the parents (discrete recombination) or defined as the average of the parental information (intermediate recombination). The negation flag is always copied from the first parent.

Definition 20 (recombination: recombine) *The offspring $\sigma_{offspring}(R)$ is built from of the two parental premises $\sigma_1(R)$ and $\sigma_2(R)$ as:*

$$\sigma_{offspring}(R) = \sigma_{Equal}(R) \cup \sigma_{TM}(R)$$

where

$$\begin{aligned}
\sigma_{Equal}(R) := \{\ e = (v,t,w,n) \mid\ & (\exists (v^1,t^1,w^1,n^1) \in \sigma_1(R) : v^1 = v \wedge t^1 = t) \\
& \wedge\ (\exists (v^2,t^2,w^2,n^2) \in \sigma_2(R) : v^2 = v \wedge t^2 = t) \\
& \wedge\ (w = f_{Cross}(w^1,w^2) \wedge n = n^1)\} \\
\sigma_{Diff}(R) := \{\ e = (v,t,w,n) \mid\ & (e \in \sigma_1(R) \vee e \in \sigma_2(R)) \\
& \wedge\ (\forall (v^0,t^0,w^0,n^0) \in \sigma_{Equal}(R) : v^0 \neq v \vee t^0 \neq t)\} \\
\sigma_{TM}(R) \subseteq \sigma_{Diff}(R)\ & \text{, random choice with equal probability and:} \\
& \mid \sigma_{TM}(R) \mid\ \leq\ c_{max} - \mid \sigma_{Equal}(R) \mid
\end{aligned}$$

The recombination process is illustrated in figure 6.

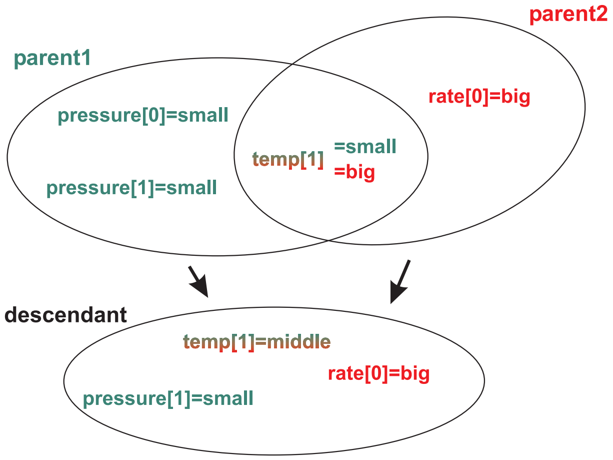


Figure 6: The recombination process.

2.4 Initialization

The initialization operator generates the λ individuals of the first generation.

Definition 21 *initialization*

For each individual a length $k \in \{1, \ldots, c_{max}\}$ is randomly chosen. The individual is constructed by applying the following operation k-times:

$$\begin{aligned}
\sigma_{i,new}(R) = insert(R, i, (v,t,w,n)), \quad & where \\
e = (v,t,w,n) \in \mu_{vt}, \quad & and \\
\mu_{vt} := \{\ e' = (v',t',w',n') \mid\quad & e' \in \mu(R)\ \wedge \\
& \forall\ (v^0,t^0,w^0,n^0) \in \sigma_i(R) : v^0 \neq v' \vee t^0 \neq t'
\end{aligned}$$

By definition of μ_{vt} the feasibility is guaranteed for the resulting individual.

3 Fuzzy Adaptation of Strategy Parameters

Due to the complex nature of the search space the evolutionary search of the Fuzzy–ROSA method is built on several basic operators with many free parameters. Since, these parameters have a great influence on the search process and their tuning depends very much on the problem at hand, it is desirable to dynamically adapt these parameters according to the state and history of the search. On the other hand, as described in section 1.2, selection in the evolutionary search of the Fuzzy–ROSA method is based on the relevance of single rules, rather than the overall quality of the rule set. This information can be fed back into the search process by dynamic adaptation of strategy parameters as well.

Adaptation is an important issue in evolutionary computation, and different approaches can be found in the literature [26]. Since, in the case of the Fuzzy–ROSA method there exist some reasonable assumptions about "good search behavior" (i.e., domain knowledge) it seemed appropriate to apply a fuzzy controller to the problem of parameter adaptation. A general introduction to the use of fuzzy systems to control the exploitation/exploration behavior in genetic algorithms is given in [27].

3.1 Indicators of Search Progress

The dynamic adaptation of strategy parameters (see next Section) again makes use of heuristic assumptions about the relationship between the state of the evolutionary search process and the performance of the resulting rule base. In order to assess the process state, the following set of global measures are chosen, based on extensive experimentation.

Diversity: Using the heuristic distance measure, the diversity of the population can be measured directly by the mean value of the distances between individuals. Alternative diversity measures are the entropy and the variation of the linguistic values.

Qualitative/Quantitative Progress: The best and the mean fitness of individuals are used to measure the quality of new rules in one generation. Rules are considered to be new, if they are not already elements of the current rule set. The total number of new rules in the current generation quantifies the search progress.

Rule Structure: In our experiments, the combination depth of the rules is an important criterion for the rule set structure. The lack of short as well as of complex rules can have a negative effect on the search process.

3.2 Dynamic Adaptation of Strategy Parameters

In the hybrid approach most of the indicators described above are used as inputs to a fuzzy system that dynamically adapts the strategy parameters of the evolutionary algorithm (Figure 7).

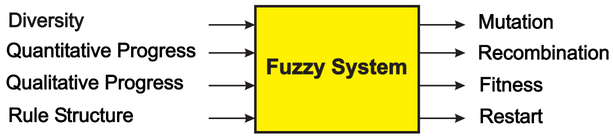


Figure 7: Structure of a fuzzy system for adapting strategy parameters

The control strategy for adaptation of the mutation operator is to preserve an average diversity in the population (*Section 3.2.1*). A more complex strategy is required to avoid a tendency towards short or long rules. This is achieved through the introduction of a dynamic penalty function *(Section 3.2.2)*. Finally, stagnation of the search process can be assumed if certain indicators exceed some critical values.

In this case, an automatic restart is performed (*Section 3.2.3)*.

It should be noted that the properties of the search space depends very much on the application and data at hand. In our approach, therefore, the priority was to achieve a robust and transparent adaptation mechanism.

3.2.1 Adaptation of the Mutation Operator^{+}

The effect of the mutation operator depends strongly on the parameter n_{mut} (number of mutations). As described in Section 2.3.2 the mutation rate and the number of mutations are determined by this parameter. Therefore n_{mut} seems to be a suitable strategy parameter and is chosen for the fuzzy–adaptation.

Our experiments have shown, that the maximum number of mutations can be restricted to 300 and consequently n_{mut} is normalised to this maximum value.

The diversity of the population can be measured by the (estimated) average distance $\bar{d}$ between the individuals (rules). In our algorithm the distance d between two fuzzy–rules is defined by the heuristic distance measure, described in Section 2.2.3. In order to save computing time only those pairs of rules, chosen for recombination, are taken to calculate the (estimated) mean value $\bar{d}_R$.

Based on numerous experiments the membership functions for n_{mut} and $\bar{d}_R$ are designed as shown in figure 8.

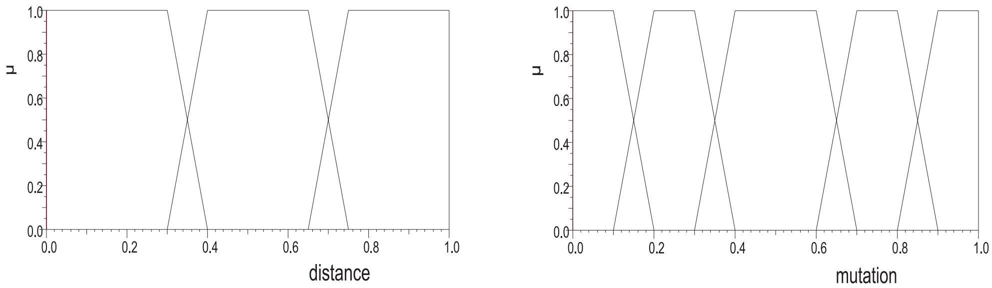


Figure 8: Membershipfunctions for the average **distance** $\bar{d}_R$ and the number of **mutations** n_{mut}

.

The number of mutations is divided into the five linguistic values: very small (xs), small (s), middle (m), large (l) and very large (xl). The three linguistic values for the average distance are *small*, *middle* and *big*.

In our experiments the results are usually better, if the average distance $\bar{d}_R$ is not too *small* and not too *big*. The most important effects concerning the diversity in the population derived from the mutation operator are:

- A premature stagnation of the search process can mainly be observed for a *(very)small* number of mutations n_{mut}. This stagnation is indicated by a very *small* average distance $\bar{d}_R$, i. e. the rules in the population are very similar.
- A *(very) large* number of mutations n_{mut} can lead to a disruptive search behaviour. Almost all individuals are mutated one or more times in each generation. Consequently the average distance $\bar{d}_R$ in the population increases which might lead to divergence of the search process, i. e. fewer rules with high relevance are generated.

Therefore the control strategy for the adaptation of the number of mutations n_{mut} is to preserve a *mean* average distance $\bar{d}_R$. If $\bar{d}_R$ is too *small* n_{mut} is increased and if $\bar{d}_R$ is too big n_{mut} is decreased. In the case, that a *mean* average distance $\bar{d}_R$ has been achieved, the strategy is to avoid extreme linguistic values, i. e. *xs* and *xl*, for the number of mutations n_{mut}.

# mutation n_{mut}	distance $\bar{d}_R$		
	small	middle	big
xs	s	s	xs
s	m	m	xs
m	l	m	s
l	xl	m	m
xl	xl	l	l

Table 1: Rule base for the adaptation of the number of mutations n_{mut}

Based on this control strategy, the rule base, shown in Table 1, was developed for the adaptation of n_{mut}. In the resulting fuzzy–system this rule base is used with the minimum operator for logical–AND and Centre of Gravity (COG) for defuzzification.

3.2.2 Adapting the Fitness Function

There are efficient methods available for the generation of short (generalising) rules, i. e. complete search or Monte Carlo search. Therefore, the main purpose of the evolutionary search is to find more complex rules, especially in high dimensional search spaces. In this section a fuzzy–system is presented, that enforces the search process towards longer (or if necessary towards shorter) rules.

In aim of obtaining a transparent strategy parameter for the adaptation, the evaluation of the individuals (rules) has to be extended by a penalty function. The original standard fitness value $f_{std}(\tau_{age}, \delta_{red})$ is calculated as described in [13]. The individuals are devaluated according to the parameter τ_{age} after exceeding a maximum age. Additionally, if individuals (rules) in the population are covered by better and more general rules in the rule base, collected from the authority, then their fitness values are multiplied with a penalty factor for covered rules $\delta_{red} < 1$.

The calculation of the extended fitness value f_{ext} is given in equation 12 with c_{max} as the maximum combination depth of linguistic statements in the premise:

$$f_{ext}(\tau_{age}, \delta_{red}, \alpha_{len}) = f_{std}(\tau_{age}, \delta_{red}) \cdot (\alpha_{len} + \frac{1-\alpha_{len}}{c_{max}} \cdot c_{rule}) \qquad (12)$$

The combination depth of the considered individual (rule) is c_{rule}. As shown in figure 9 the degree of devaluation for shorter rules is defined by parameter α_{len}.

The fuzzy system, designed for the adaptation, recommends a simultaneous relative change ζ of δ_{red} and α_{len}. To enforce the search towards more complex rules, δ_{red} is increased and α_{len} is decreased and vice versa if shorter rules are desired. The change of each parameter is proportionl (i. e. 10 %) to its actual permissible variation range. This conservative strategy is chosen, to avoid too large adaptation steps and infeasible values of the adapted parameters. Based on our experiments the following seven linguistic values are defined for the relative change ζ (Figure 10): negative big (*nb*), negative (*n*), negative small (*ns*), zero (*z*), positive small (*ps*), positive (*p*) and positive big (*pb*).

The rule structure in the population is characterised by the following indicators:

Normalised Average Rule Length ($\bar{c}_{NRL} = \bar{c}_{rule}/c_{max}$): By this indicator extreme search progresses can be characterised, i. e. a population, which almost consists of short or long rules. For a medium value of $\bar{c}_{NRL}$ it is almost impossible to draw any conclusion about the distribution of c_{rule} in the population.

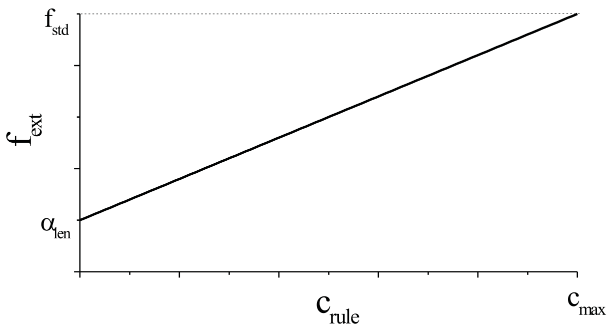


Figure 9: Penalty function for the devaluation of shorter rules.

Proportion of short/long rules (PSR/PLR): The proportion of short/long rules in the population is defined by

$$PSR = c_{max} \cdot \frac{\#\text{ rules with } c_{rule} = 1}{\#\text{ rules in the population}} \quad \text{and} \quad PLR = c_{max} \cdot \frac{\#\text{ rules with } c_{rule} = c_{max}}{\#\text{ rules in the population}} \,. \tag{13}$$

With Def. 13, it is possible, that PSR/PLR exceeds one. However, as PSR and PLR only serves to indicate a lack of short or long rules, the values can be restricted to one, by using $\min(1, PSR)$ and $\min(1, PLR)$.

As the normalised average rule length $\bar{c}_{NRL}$ is only used for the detection of the extreme search progresses, it is sufficient to define the three linguistic values *small*, *normal* and *big* (Figure 10). Based on our experiments the five linguistic values extreme small (xxs), very small (xs), small (s) and medium (m) are defined for PSR and PLR in Figure 10.

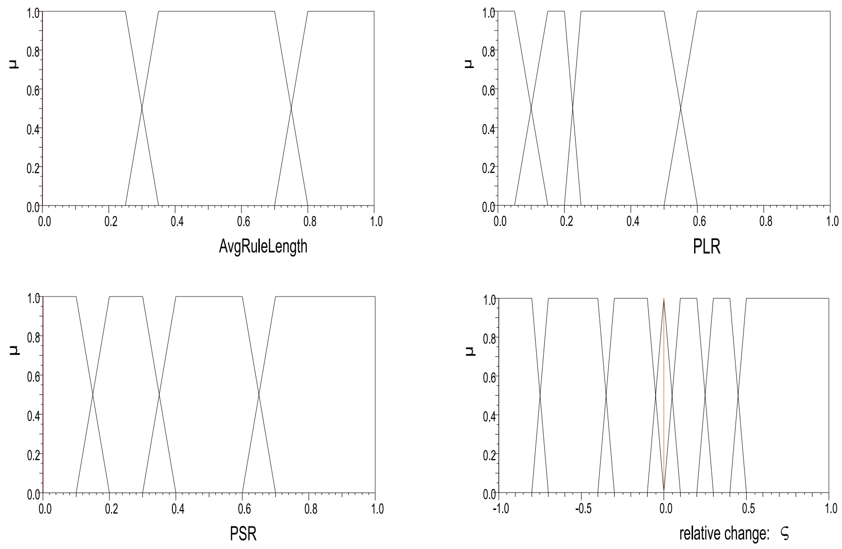


Figure 10: Membershipfunctions for nomalised average rule length $\bar{c}_{NRL}$, part of long rules PLR, part of short rules PSR and the relative change ζ.

In our experiments it could be observed, that a lack of short or long rules is unfavourable for the search process. If the population only consists of short rules, usually after a short time almost none new relevant rules are generated. The reason is, that in most applications the number of short relevant rules is very much smaller than the number of long relevant rules, due to the exponential growth of the number of possible rules depending on the combination depth c_{rule}. Additionally, short rules can easily be found by a complete search or a Monte Carlo search. On the other hand, if there are only a few short rules in the population, it can also be observed, that only few new rules are generated. A possible explanation is, that complex rules are most often generated by recombination of short rules.

Figures 11 and 12 show the difference in the search progress in relation to the indicators discussed above for two runs of the evolutionary algorithm without adaptation of the strategy parameters. The learning data were taken from the classification example discussed in section 4.2.

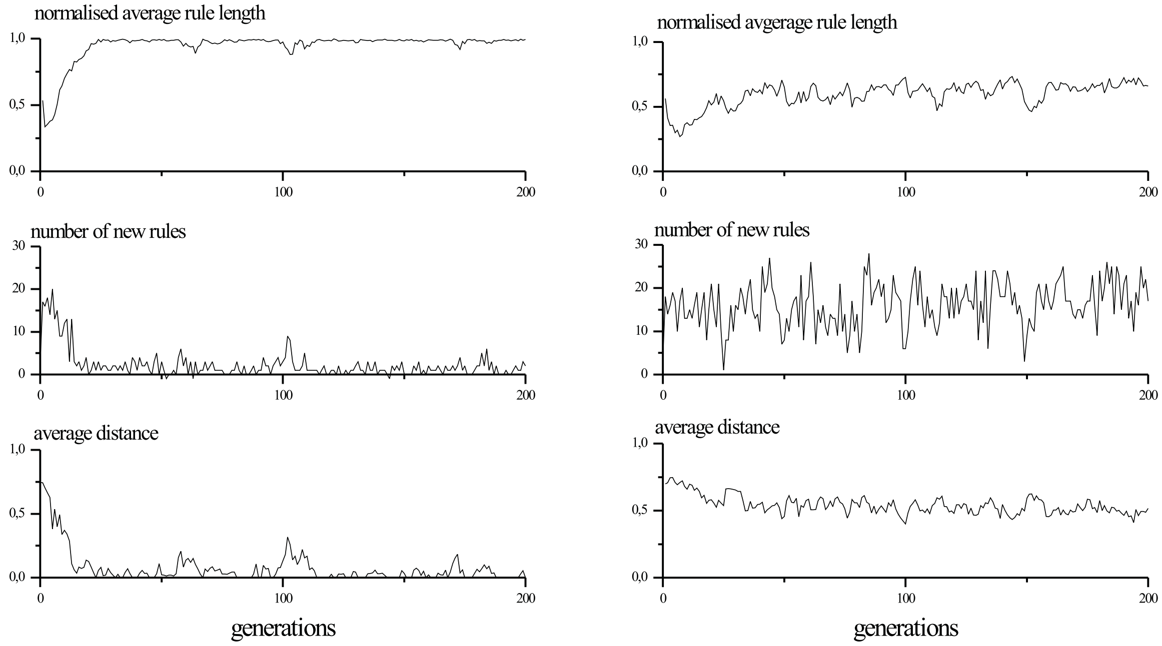


Figure 11: Progress of the indicators during 200 generations for the classification in quality control (Left charts for $\delta_{red} = 0.9$ and right charts for $\delta_{red} = 0.5$)

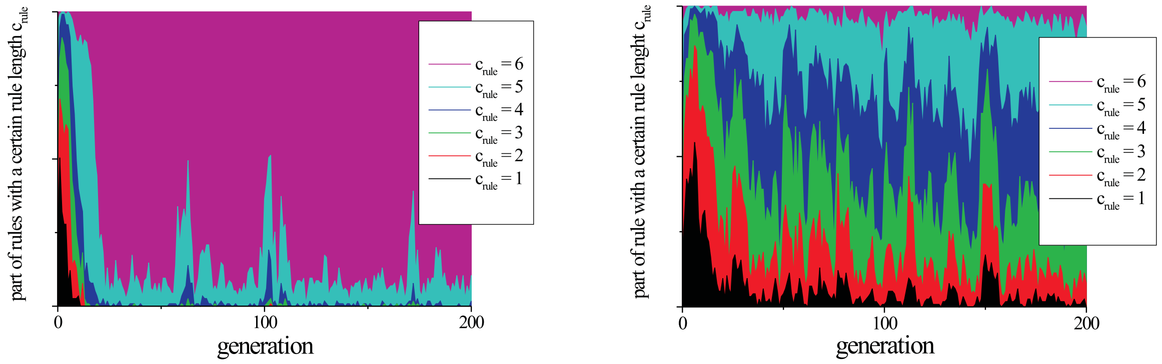


Figure 12: Part of rules with a certain combination depth c_{rule} in the same searches as in Figure 11 (Left charts for $\delta_{red} = 0.9$ and right charts for $\delta_{red} = 0.5$)

A fixed penalty factor for covered rules $\delta_{red} = 0.9$ was chosen for the first run and $\delta_{red} = 0.5$ for the second run. In this application more complex rules are often covered by generalising rules and therefore are devaluated. Consequently, a higher value for δ_{red} leads to a better fitness for more complex rules in the average case. Due to this better performance for $\delta_{red} = 0.9$, in the first search the population is inundated with rules of the maximum combination depth (left side Figure 11 and 12). This is indicated by a normalised average rule length $\bar{c}_{NRL}$ close to one. As discussed above, it can be observed, that only few additional rules are generated in each generation. This premature stagnation results also in a loss of diversity in the population. Therefore the rules are very similiar and consequently a small average distance $\bar{\delta}_R$ can be observed.

A more desireable search progress can be reached for a $\delta_{red} = 0.5$. A normalised average rule length $\bar{c}_{NRL}$ of approximately 0.5 indicates a balanced mix of rules with different combination depths c_{rule}. The distribution of c_{rule} in the population is shown more precisely in Figure 12. In comparison to the first search no premature stagnation and loss of diversity can be observed and finally significantly a larger number of new rules are found during the 200 generations.

In order to avoid the undesirable states of the search process discussed above a rule base for the adaptation of the fitness function was developed (table 2). Considering only the normalised average rule length $\bar{c}_{NRL}$, the control strategy is to preserve an average rule length in the population. Additionally the indicators *PSR* and *PLR* are used to avoid a lack of short or long rules.

If	Then ζ=
AvgRuleLength=small	ps
AvgRuleLength=normal	z
AvgRuleLength=big	ns
PSR=xxs and AvgRuleLength=big	nb
PSR=xxs and AvgRuleLength=normal	n
PSR=xs and AvgRuleLength=big	n
PSR=xs and AvgRuleLength=normal	n
PSR=s and AvgRuleLength=big	ns
PSR=m	z
PLR=xxs and PSR=m and AvgRuleLength=small	pb
PLR=xxs and PSR=m and AvgRuleLength=normal	ps
PLR=xxs and PSR=s and AvgRuleLength=small	p
PLR=xxs and PSR=s and AvgRuleLength=normal	ps
PLR=xs and PSR=m	ps

Table 2: Rule Base for adapting the fitness function

3.2.3 Rule for Restart

In the case of stagnation or convergence of the search process, a new population with a higher diversity can probably be obtained, by a complete reintialisation (restart). In our experiments a stagnation or convergence is indicated, if the following strategy parameter exceed certain limits

- number of mutation $n_{mut} > 0.9$
- average rule length $\bar{c}_{NRL} > 0.8$
- penalty factor for covered rules $\delta_{red} \leq 0.05$ (5% of the original fitness value)

In a (conservative) approach a restart is performed, if each of the strategy parameter exceeds its limit during 20 generations.

3.3 Performance of the Hybrid Evolutionary Search Concept

In this Section the efficiency of the proposed hybrid evolutionary search concept—with and without adaptation—is compared to a simple Monte Carlo search as well as to an earlier approach proposed in [13], which is based on a simpler representation and lacks a mechanism for strategy parameter adaptation.

To assess the quantitative and qualitative search results, the following criteria were used: the number of generated relevant rules, the number of relevant rules with a minimum fitness, and the number of relevant rules with a minimum complexity.

The test suite consisted of three applications: (i) a synthetic benchmark problem consisting of 400 rules with a maximum combination depth of five in a search space of 50 input variables and seven linguistic values each, (ii) modelling a heat exchanger in a semi–batch process [4], and (iii) the classification of automatic gearboxes [7] discussed in Section 4.2.

The benchmark problem has the advantage that the global optimum (the original rule base) is known. All experiments are performed on the same hardware[5] and each search method is given the same computing time. The mean values of five experiments are taken for each configuration.

The results after a ten-minute search for problems (i) and (ii) are depicted in Figure 13 (a)–(c).

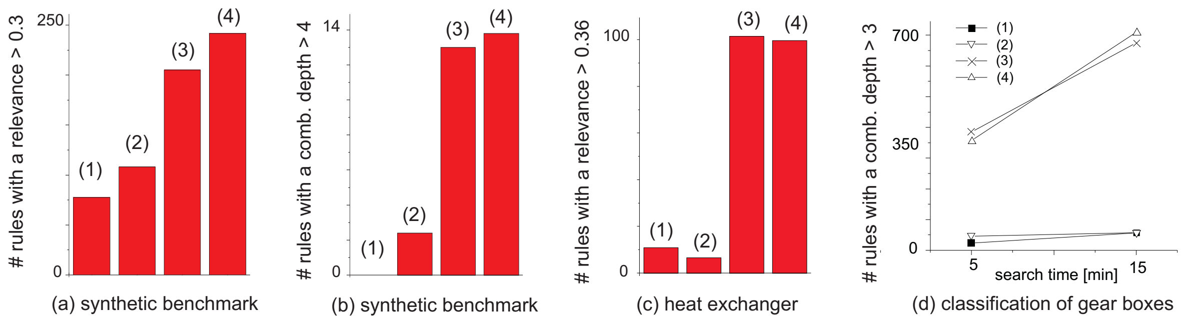


Figure 13: Results after a ten-minute search (a)–(c) and time dependency of search progress (d) for (1): Monte Carlo, (2): simple EA [13], (3): EA without adaptation, (4): EA with adaptation.

In all experiments, the hybrid evolutionary search concept—with or without adaptation—shows significantly better performance on quantitative and qualitative search results. In particular, the much greater ability to find complex rules is important for many applications (Figure 13 (c) and (d)). Figure 13 (c) shows that the additional computational effort for parameter adaptation does not pay off. This is because the density of relevant rules is very high in this example. In this sense, the problem turns out to be too easy.

In more complex cases, the advantage of parameter adaptation becomes noticeable, especially, in the long run, as in the example of classification of gearboxes 13 (d).

4 Applications of the Hybrid Search Concept

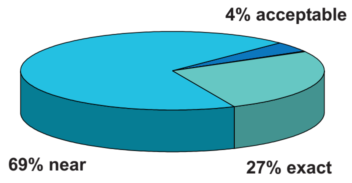
4.1 Prediction of Contract Durations

Insurance companies are interested in a long duration of contracts to avoid high administration costs. Therefore, statistical methods are used to analyse the dependency of contract durations on client profiles. Contract duration is the time between the start of the contract and the date of cancellation. Here, the profile of a client is characterized by the following seven sociodemographic characteristics: *social status, profession, sex, age, type of first contract, place of residence, federal state.*

Data–based rule generation of a fuzzy system to predict contract duration is a transparent approach, as the premise of a fuzzy rule can be interpreted as a profile of a client. The premise is a fuzzy conjunction of the sociodemographic characteristics.

First results are presented in Figure 14 . In a 24-hr hybrid evolutionary search, a rule base of 2297 rules with a certain minimal fitness was generated and used directly without any subsequent reduction for

[5]Pentium 120 MHz (without MMX), 40 MB EDO–RAM



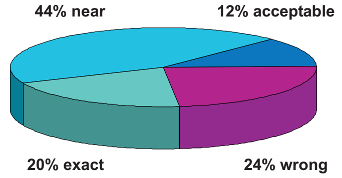


Figure 14: Left chart: Prediction for 4761 of 18399 learning data sets (26%), Right chart: Prediction for 251 of 808 validation data sets (31%).

prediction. Comparisons[6] between real and predicted contract durations leads to the conclusion that in principle it is possible to learn fuzzy rules for typical client profiles. The design of a fuzzy classifier is the subject of our present work.

4.2 Classification in Quality Control

In quality control, different parameters are inspected during a test to detect material faults, production faults, assembly faults or unusual sounds during operation. In our application, automatic gearboxes produced by an automobile manufacturer are tested acoustically by human specialists. With 149 acoustic input characteristics, the design of a fuzzy–classifier is a very complex problem (10^{17} possible rules with maximum combination depth of six).

As described in [7, 28] data–based generation of a fuzzy–classifier is possible using the Fuzzy–ROSA method based on 1060 data sets (1000 'o.k.' and 60 'not o.k.'). Five equidistant membership functions were chosen heuristically for each characteristic.

After a 20-hr hybrid evolutionary search and subsequent rule reduction the resulting rule base (with less than 100 rules and a maximum combination depth of six) achieved a 100% correct classification on the learning data. In order to test the systems capability of generalizing classification results learning was repeated on 90% of the data set. The remaining 10% were taken as test data. The classification error was 8% (5% for 'ok' and 30% for 'not ok').

5 Conclusions and Outlook

We have shown that the hybrid evolutionary search concept embedded in the Fuzzy–ROSA method is a promising approach for fuzzy modelling in high dimensional search spaces. It was successfully applied to a complex benchmark example as well as to different practical applications with up to 149 input variables.

Our future work will aim at improving the fuzzy–adaptation mechanism by using more or better indicators, parameters and control strategies. In particular, the specification of subspaces not covered by rules or with high modelling error seems to be a promising feedback for the hybrid evolutionary search

[6] If the prediction error is larger than four years, the result is called **wrong**, if it is between two and four years it is called **acceptable** and if it is between four months and two years it is called **near**. An **exact** prediction means the error is less than four months.

concept. From a closer adaptation of the MBEA concept we would expect an additional considerable improvement. However, a prerequisite is the development of an applicable metric for fuzzy rules with variable combination depth.

Acknowledgement:

This research was sponsored by the Deutsche Forschungsgemeinschaft (DFG), as part of the Collaborative Research Center 'Computational Intelligence' (531) of the University of Dortmund.

Part of the **references** can be **downloaded** from
http://esr.e-technik.uni-dortmund.de/winrosa/winrosa.htm.

References

[1] A. Krone and H. Kiendl. Automatic generation of positive and negative rules for two-way fuzzy controllers. In *Second European Congress on Intelligent Techniques and Soft Computing (EUFIT'94)*, pages 438–447, Aachen (Germany), 1994.

[2] M. Krabs and H. Kiendl. Anwendungsfelder der automatischen Regelgenerierung mit dem ROSA-Verfahren. *Automatisierungstechnik*, 43(6):269–276, 1995.

[3] H. Kiendl. Design of advanced fuzzy–systems. In *Symp. Tool Environments and Development Methods for Intelligent Systems (TOOLMET '99)*, pages 57–76, Oulu (Finland), 1999.

[4] A. Krone, C. Frenck, and O. Russak. Design of a fuzzy controller for an alkoxylation process using the ROSA method for automatic rule generation. In *Third European Congress on Intelligent Techniques and Soft Computing (EUFIT '95)*, pages 760–764, Aachen (Germany), 1995.

[5] A. Krone and U. Schwane. Generating fuzzy rules from contradictory data of different control strategies and control performances. In *Fifth IEEE International Conference on Fuzzy Systems. (FUZZ-IEEE '96)*, pages 492–497, New Orleans (USA), 1996.

[6] T. Slawinski, J. Praczyk, U. Schwane, and H. Kiendl. Data–based generation of fuzzy–rules for classification, prediction and control with the Fuzzy–ROSA method. In *European Control Conf. (ECC' 99)*, Karlsruhe (Germany), 1999. (in print).

[7] T. Slawinski, U. Schwane, J. Praczyk, A. Krone, H. Jessen, H. Kiendl, and D. Lieske. Application of WINROSA for controller adaptation in robotics and classification in quality control. In *Reihe Computational Intelligence*, Aachen, Germany, 1998. No CI-46/98, ISSN 1433-3325, Collaborative Research Center 531, University of Dortmund.

[8] H. Jessen. A mean-value based test and rating strategy for automatic fuzzy rule generation and application to load prediction. In *International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'99)*, pages 198–203, Vienna (Austria), 1999.

[9] A. Krone and H. Taeger. Relevance test for fuzzy rules. In *Reihe Computational Intelligence*. No CI-40/98, ISSN 1433-3325, Collaborative Research Center 531, University of Dortmund, 1998.

[10] H. Jessen and T. Slawinski. Test- and rating strategies for data-based rule generation. In *Reihe Computational Intelligence*. No CI-39/98, ISSN 1433-3325, Collaborative Research Center 531, University of Dortmund, 1998.

[11] H. Kiendl. *Fuzzy Control methodenorientiert*. Oldenbourg Verlag, Muenchen (Germany), 1997.

[12] A. Krone and T. Slawinski. Data-based extraction of unidimensional fuzzy sets for fuzzy rule generation. In *Seventh IEEE International Conf. on Fuzzy Systems. (FUZZ-IEEE '98)*, pages 1032–1037, Anchorage (Alaska), 1998.

[13] A. Krone and H. Kiendl. Evolutionary concept for generating relevant fuzzy rules from data. *International Journal of Knowledge-based Intelligent Engineering Systems*, 1(4):207–213, 1997.

[14] A. Krone. Advanced rule reduction concepts for optimising efficiency of knowledge extraction. In *Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96)*, pages 919–923, Aachen (Germany), 1996.

[15] C.L. Karr. Genetic algorithms for fuzzy controllers. *Al Expert*, pages 393–404, 1991.

[16] A. Bonarini. Evolutionary learning of general fuzzy rules with biased evaluation functions: competition and cooperation. In *IEEE Press*, pages 51–56, 1994.

[17] F. Herrera, M. Lozano, and J. L. Verdegay. Generating fuzzy rules from examples using genetic algoritms. Technical Report DECSAI-93115, Department of Computer Science and Artificial Intelligence, University of Granada, 1993.

[18] S.F. Smith. *A learning system based on genetic adaptive systems.* PhD thesis, University of Pittsburgh, 1980.

[19] J.H. Holland, K.J. Holyoak, and R.E. Nisbett. *Induction: Processes of Inference, Learning and Discovery.* MIT Press, 1986.

[20] Th. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):1–15, 1997.

[21] I. Rechenberg. *Evolutionsstrategie '94, Volume 1 of Werkstatt Bionik und Evolutionstechnik.* Frommann-Holzboog, Stuttgart (Germany), 1994.

[22] B. Sendhoff, M. Kreutz, and W. von Seelen. A condition for the genotype-phenotype mapping : Causality. In Th. Bäck, editor, *Seventh Int'l Conf. Genetic Algorithms (ICGA '97), East Lansing, MI, 19.-23. Juli 1997*, pages 73–80, San Francisco, CA, 1997. Morgan Kaufmann.

[23] S. Droste and D. Wiesmann. On representation and genetic operators in evolutionary algorithms. In *Reihe Computational Intelligence.* No CI-41/98, ISSN 1433-3325, Collaborative Research Center 531, University of Dortmund, 1998.

[24] L.T Koczy and K. Hirota. Approximate reasoning by linear rule interpolation and general approximation. *International Journal of Approximate Reasoning*, (9):197–225, 1993.

[25] Y.-P. Huang. Adaptive fuzzy control: A ga approach. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems. (FUZZ-IEEE '96)*, pages 1266–1272, 1996.

[26] R. Hinterding, Z. Michalewicz, and A.E. Eiben. Adaptation in evolutionary computation: A survey. In *Proceedings of the Fourth IEEE International Conference on Evolutionary Computation*, Piscataway, NJ, 1997. IEEE Press.

[27] F. Herrea and M. Lozano. *Adaptation of Genetic Algorithm parameters based on fuzzy–logic controllers.* Physica–Verlag, New York (USA), 1996.

[28] J. Praczyk, H. Kiendl, and T. Slawinski. Finding relevant process characteristics with a method for data–based complexity reduction. In *International Conference Computational Intelligence (Sixth Fuzzy–Days)*, pages 548–555, Dortmund (Germany), 1999.