ELSEVIER

Information Sciences 136 (2001) 175–191

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Combining GP operators with SA search to evolve fuzzy rule based classifiers

Luciano Sánchez [a,*], Inés Couso [b], J.A. Corrales [a]

[a] *Dep. Informática, Universidad de Oviedo, Calvo Sotelo s/n, 33007 Oviedo, Spain*
[b] *Dep. Estadística, I.O. y D.M., Universidad de Oviedo, Oviedo, Spain*

Abstract

The genotype–phenotype encoding of fuzzy rule bases in GA, along with their corresponding crossover and mutation operators, can be used by other search schemes, improving the behavior of these last ones. As a practical consequence of this, a simulated annealing-based method for inducting both parameters and structure of a fuzzy classifier has been developed. The adjacency operator in SA has been replaced with a macromutation taken from tree-shaped genotype GAs. We will show that results of SA search are similar to those of GP in both the efficiency of the learned classifiers and in its linguistic interpretability, while the memory consumption of the learning process is lower.

Keywords: Fuzzy classification; Simulated annealing; Genetic algorithms; Genetic programming

1. Introduction

A linguistically understandable fuzzy classifier system is defined by a base of fuzzy rules. Following Zadeh [17], a fuzzy rule base can be described in two levels of detail. The *surface structure* of the base, so-called "linguistic rule base", comprises the linguistic rules, and the *deep structure* comprises the set of rules plus the definition of linguistic partitions of all features used in the classification system. Every linguistic term is tied to a fuzzy set defined over one of the features.

* Corresponding author.
E-mail address: luciano@lsi.uniovi.es (L. Sánchez).

0020-0255/01/$ - see front matter
PII: S0020-0255(01)00146-3

The algorithms used for inducting a fuzzy classifier from a classified sample can operate in two different manners. Some of them separately learn the fuzzy partitions and the surface structure, but others integrate both parts in the same process. In the first case it is common to assume an equidistant partition and learn iteratively the rule base [8,13,15] or by means of genetic algorithms [4,16]. The memberships can also be approximated with clustering techniques, followed by projections [1,2]. The latter case relies on optimization or search techniques [6], in many cases genetic algorithms, with linear [3] or tree genotype [14].

Two different codifications of the base have been used when applying GA to this problem: linear genotype and tree-shaped genotype algorithms, so-called *genetic programming*. Linear genotype-based methods almost always use a kind of rules for which antecedents comprise linguistic terms, linguistic modifiers and the logical connective "and". With tree-shaped genotypes, whole bases of rules are regarded as single chains in a context-free grammar and are represented by means of their parse tree [4] or by a pair, composed by a syntactic tree and a chain of parameters [14]. We will refer to both approaches as "grammar-based GP". These genotypes should allow us to represent rule bases more compactly than linear representation can, hence we are mainly concerned with these representations.

In grammar-based GP, only subtrees generated by the same production rule can be interchanged, to preserve the syntactic correctness of the offspring. Mutation can be defined in different ways. We will cross the individual being mutated with a randomly generated individual (which is a concept similar to that of "headless chicken" crossover [9]) and select one of the offsprings. We will show that this mutation operator can be used within an SA-based search to learn simultaneously the fuzzy partitions and the surface structure of a rule base. By comparing GP to SA solutions for a fixed number of fitness evaluations, we will also show that SA is not worse than GP for this problem, thus SA can be a good algorithm when the sizes of the rule bases are large. It suffices to keep two individuals in memory at a time, while GA may need hundreds or thousands of them.

This paper is organized as follows: first we will define how we will represent a fuzzy classification system and define the objective of the learning algorithm. Then we will propose an SA algorithm able to search for a solution in the space of all tree shaped genotypes of fuzzy rule bases. Next, numerical results are shown in which the behaviors of SA and GA are compared.

2. Representation of an individual. Grammar of a valid fuzzy rule base

A classifier system is a decision rule that assigns a class to every point in the feature space [5]. We will regard fuzzy classifiers as linguistic representations of

these decision rules, and say that two different linguistic expressions that represent the same decision surfaces are two *equivalent* fuzzy classifiers.

Not all decision surfaces can be represented by linguistic rules. The goals of the linguistic classifier induction are two: (1) finding the most precise *representable* decision surface for a problem, given a sample (i.e. minimizing the expected classification error) and (2) finding the shortest individual in the equivalence class defined by this surface (i.e., minimizing the complexity of the linguistic description).

2.1. Rule-based and relation-based classifiers

There exists a correspondence between fuzzy relations and linguistic rules under the standard fuzzy reasoning. Given the fuzzy partitions, a fuzzy rule based classifier is ultimately a fuzzy relation defined over the Cartesian product of the linguistic terms set and the class marks sets. For example, to decide whether a piece of fruit is a banana or a pear, by examining its weight and color, we can use some rules like this one:

```
if weight is low and color is yellow
 then the fruit is a banana with confidence 0.8.
```

This rule give us the following information:

$$
\begin{aligned}
R(\texttt{high, yellow, banana}) &= 0,\\
R(\texttt{high, yellow, pear}) &= 0,\\
R(\texttt{low, yellow, banana}) &= 0.8,\\
R(\texttt{low, yellow, pear}) &= 0,\\
R(\texttt{high, green, banana}) &= 0,\\
R(\texttt{high, green, pear}) &= 0,\\
R(\texttt{low, green, banana}) &= 0,\\
R(\texttt{low, green, pear}) &= 0.
\end{aligned}
$$

Observe that $R(\cdot) = 0$ means "we know nothing". We will combine fuzzy rules to form a base by t-conorm by "adding" fuzzy relations associated to every rule. The inference will consist in projecting over the linguistic output space the intersection between the cylindrical extension of the input and the fuzzy relation assigned to the base. We will propose later a grammar to define which linguistic expressions are valid fuzzy rules (see Section 2.2) and their semantics by means of fuzzy relations (see Section 2.4).

The linguistic expression of a fuzzy relation by means of a fuzzy rule base is not unique: many different rule bases define the same fuzzy relation. Moreover, there can be different fuzzy relations that are also equivalent, as shown in the following example:

Example. Let us measure the diameter and weight of some pieces of fruit: these can be pears (class C_0) or bananas (class C_1). Measurements can take real values ranking from 0 to 1, and we define for every measurement the labels "low", with $\mathrm{low}(x) = 1 - x$ and "high", with $\mathrm{high}(x) = x$. Let us define now the following rule base:

$$\texttt{if X is low and Y is low then } (\mathtt{C}_0, \mathtt{C}_1) = (\mathtt{0}, \mathtt{1})$$
$$\texttt{if X is low and Y is high then } (\mathtt{C}_0, \mathtt{C}_1) = (\alpha_1, \alpha_2)$$
$$\texttt{if X is high and Y is low then } (\mathtt{C}_0, \mathtt{C}_1) = (\alpha_1, \alpha_2)$$
$$\texttt{if X is high and Y is high then } (\mathtt{C}_0, \mathtt{C}_1) = (\mathtt{1}, \mathtt{0})$$

with $\alpha_1, \alpha_2 \in [0, 1]$. Observe that, when using the standard fuzzy reasoning method (see Fig. 1), for every α_1 all bases for which $\alpha_2 \leqslant \alpha_1$ will produce the same partition, and vice versa.

2.2. Grammar of the deep structure of a fuzzy classifier and genotype of a rule base

Stating the grammar of a rule based classifier is necessary in order to define the genotype we have decided to use. For our purpose, a fuzzy classifier is a valid chain in the context free grammar defined by the production rules shown in Fig. 2. N_c is the number of classes, $x_1, \ldots, x_m$ are the features and n_i the number of linguistic terms in feature i, $i = 1, \ldots, m$.

Observe that we allow using the logical connective "or" in antecedents. For example, we would replace the pair of rules "if color is yellow and weight is high then the fruit is a pear" and "if color is green and weight is high then the fruit is a pear" with "if color is yellow or color is green and weight is high then the fruit is a pear". We also allow that some features are absent from the antecedent. For example, the rule "if color is yellow then the fruit is a pear" is

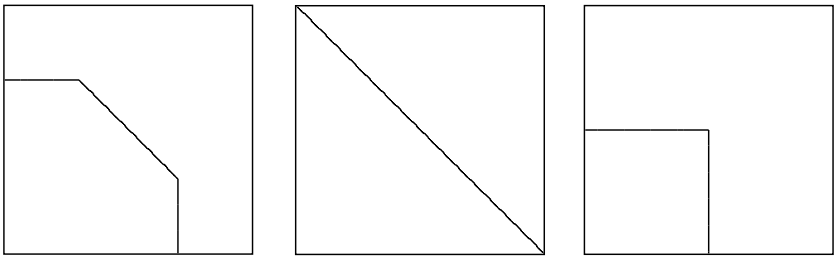

Fig. 1. Given a fuzzy partition of the universe of discourse and a fuzzy reasoning method, different fuzzy relations can produce the same classification system. In this figure we show the decision surfaces arising from the classifier defined in Section 2.1 for the values $\alpha_1 = 0.3$, $\alpha_2 \in [0, 0.3]$, (left) $\alpha_1 = 0$, $\alpha_2 = 0$ (center) and $\alpha_1 = 0.5$, $\alpha_2 \in [0, 0.5]$ (right).

```
CLASSIFIER → if CONDITION then class is c_1
             if CONDITION then class is c_2
             ...
             if CONDITION then class is c_Nc
             LOGICAL-CONS
             PARTITION-CONS
CONDITION → ASSERT_1 | ASSERT_2 | ... | ASSERT_m
             | (CONDITION ∨ CONDITION)
             | (CONDITION ∧ CONDITION)
             | K_1 | K_2 | ... | K_Nc
ASSERT_1 → left-trapezium(x_1,K_11,K_12 ) |
             triangle(x_1,K_11,K_12,K_13) |
             ...
             right-trapezium(x_1,K_1n_1−1,K_1n_1)
...
ASSERT_m → left-trapezium(x_m,K_m1,K_m2) |
             triangle(x_m,K_m1,K_m2,K_m3) |
             ...
             right-trapezium(x_m,K_mn_m−1,K_mn_m)
PARTITION-CONS → K_11 ... K_1n_1 ... K_m1 ... K_mn_m
LOGICAL-CONS → K_1 K_2 ... K_Nc
```

Fig. 2. Grammar defining the phenotype of a fuzzy rule based classifier. The genotype will consist in an acyclic graph built from the syntactic tree of the classifier. Observe that the linguistic expression comprises so many rules as classes, plus two chains of real numbers.

valid and has a different meaning than "if color is yellow and (weight is high or weight is low) then the fruit is a pear", because it may happen that "weight is high or weight is low" is not true, for the properties of fuzzy partitions. `left-trapezium`, `triangle` and `right-trapezium` are trapezoidal or triangular fuzzy memberships defined by two or three parameters (see Fig. 3). All K terminal symbols are real numbers. Since we allow using "or" in the antecedents, there are so many rules as classes. Observe that we admit "logical constants" in the expression of the antecedent; their meaning will be explained

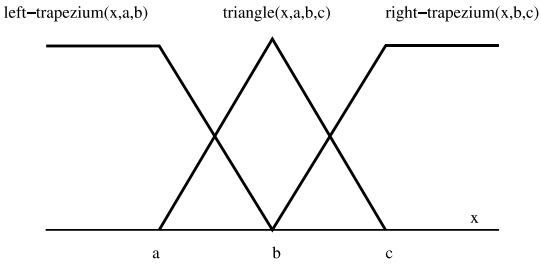

Fig. 3. Trapezoidal and triangular membership functions and the meaning of their arguments.

in the next section. There are two semantic restrictions over the values of the constants. All GA/SA operators must preserve them:

(1) Constants $K_1\ K_2 \ldots K_{N_c}$ take values between 0 and 1.

(2) The lists $[K_{11} \ldots K_{1n_1}] \ldots [K_{m1} \ldots K_{mn_m}]$, are ordered and their values must be inside the range defined for every one of the features.

For example, the surface structure of a linguistic classifier for which $N_c = 2$, $m = 2$, $n_1 = 3$, $n_2 = 3$ and all features range from 0 to 1 is as follows:

```
if xl is MEDIUM − l and x2 is MEDIUM − 2
  then class is cl with conf. l
if x2 is LOW − 2 then class is cl with conf. l
if xl is HIGH − l then class is c2 with conf. 0.2
```

Its deep structure is defined by its surface structure plus the definition of the fuzzy partitions of both features. For instance:

```
LOW − l(xl) = trapezium − left(xl, 0.l, 0.2)
MEDIUM − l(xl) = triangle(xl, 0.l, 0.2, 0.5)
HIGH − l(xl) = trapezium − right(xl, 0.2, 0.5)
LOW − 2(x2) = trapezium − left(x2, 0.0, 0.5)
MEDIUM − 2(x2) = triangle(x2, 0.0, 0.5, 0.8)
HIGH − 2(x2) = trapezium − right(x2, 0.5, 0.8)
```

We represent the deep structure of this fuzzy classifier by means of the chain that follows. The structure of the second rule will be explained in Section 2.4.2.

```
if triangle(xl, kll, kl2, kl3)
  and triangle (x2, k2l, k22, k23)
  or trapezium-left(x2, k2l, k22) then class is cl
if trapezium-right(xl, kl2, kl3) and k2 then class is c2
0.7 0.2 0.9
0.l 0.2 0.5 0.0 0.5 0.8
```

and the genotype is an acyclic directed graph formed by the syntactic tree of the chain and an array with the values of all parameters, shown in Fig. 4. Not all parameters need to be used; in this example, `kl` = 0.7 and `k2` = 0.9 are not referenced by any rule.

This graph must be labeled with the names of the production rules that originate every subtree, which is a particular case of strongly typed GP [10]. Other authors [4] use the parse tree of the chain. In the following section we will define the genetic operators.

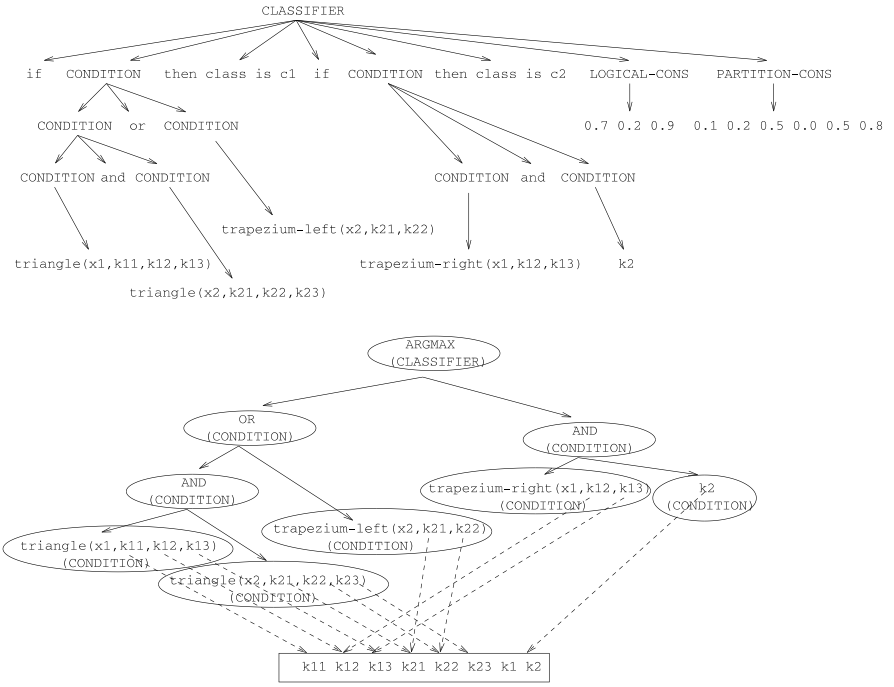

Fig. 4. Phenotype–genotype representation of the chain shown in Section 2.2 following Geyer Schulz's approach (upper part) and ours (lower part). In Geyer's the chain is the frontier of its parse tree and genotype crossover takes place between subtrees labeled with the same symbol in the root. In our method an acyclic graph derived from the syntactic tree of chain and GA-P crossover is used. The names of the production rule that originate each subtree are written in parentheses and act as types in strongly typed GP.

2.3. Crossover and macromutation operators

As in strongly typed GP crossover, we only interchange subtrees that are evaluated to values of the same type. Since we have defined the type of a node as the name of the production rule which originates it, typed crossover preserves grammatical correctness (i.e., any crossover of two trees is the syntactic tree of a valid chain in the grammar). Observe that this crossover can be seen as an extension of the two-point crossover to tree shaped genotype.

There are not numbers in the terminal nodes of the trees but pointers to an array (see Fig. 4). The macromutation operation we will adapt to SA is derived from GA-P algorithm crossover [7]. GA-P algorithms merge two kinds of crossovers at random: the usual subtree interchange crossover (see Fig. 5) and crossover between the arrays of real numbers that we have mentioned (see

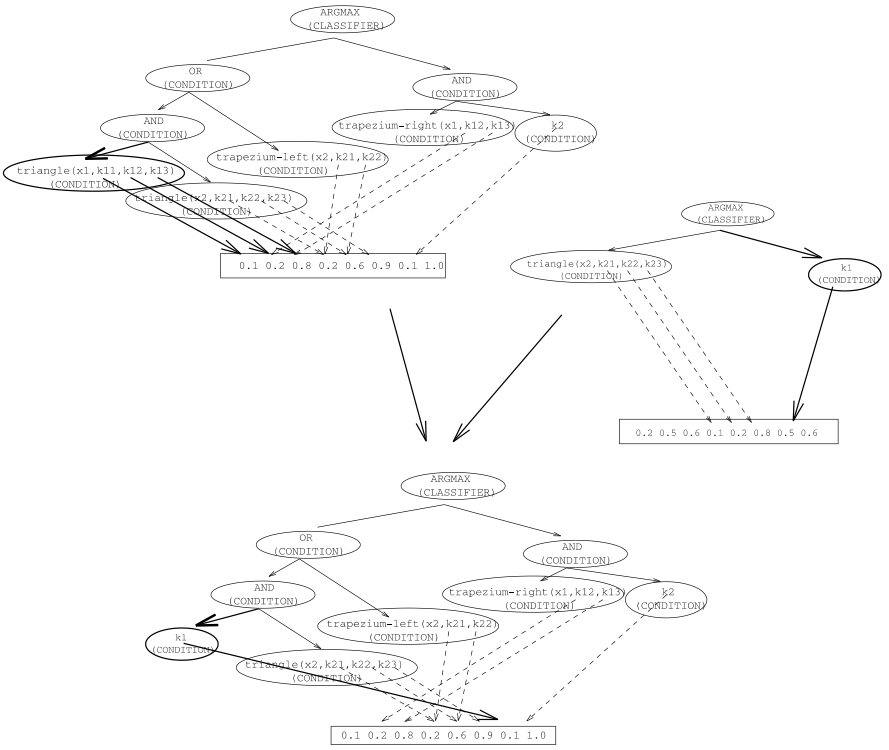

Fig. 5. Subtree macromutation: A subtree of the syntactic tree of the individual being mutated (upper part, left) is interchanged with a subtree of the same type in a random individual (upper part, right.) The array of parameters is not altered.

Fig. 6). Hence, we will alternate random subtree replacement and random subchain replacement in SA, as we will show in Section 3.

2.4. Semantic of a fuzzy rule base

We have defined all valid linguistic fuzzy classifiers by means of a grammar, their codification and the genetic operators. It lasts to state the meaning of a fuzzy rule in terms of a fuzzy relation, and how logical connectives and rule concatenation are interpreted. These semantics are defined by recursively applying the conditions that follow. Recall that a fuzzy classifier is defined by a fuzzy relation that assigns values between 0 and 1 to tuples of $m+1$ values. Each tuple reflects the confidence we have on a combination of linguistic values of all features and corresponds to one of the classes; in other words: the first value is a linguistic label defined over the first feature, the second value is a label in the second feature, and so on,

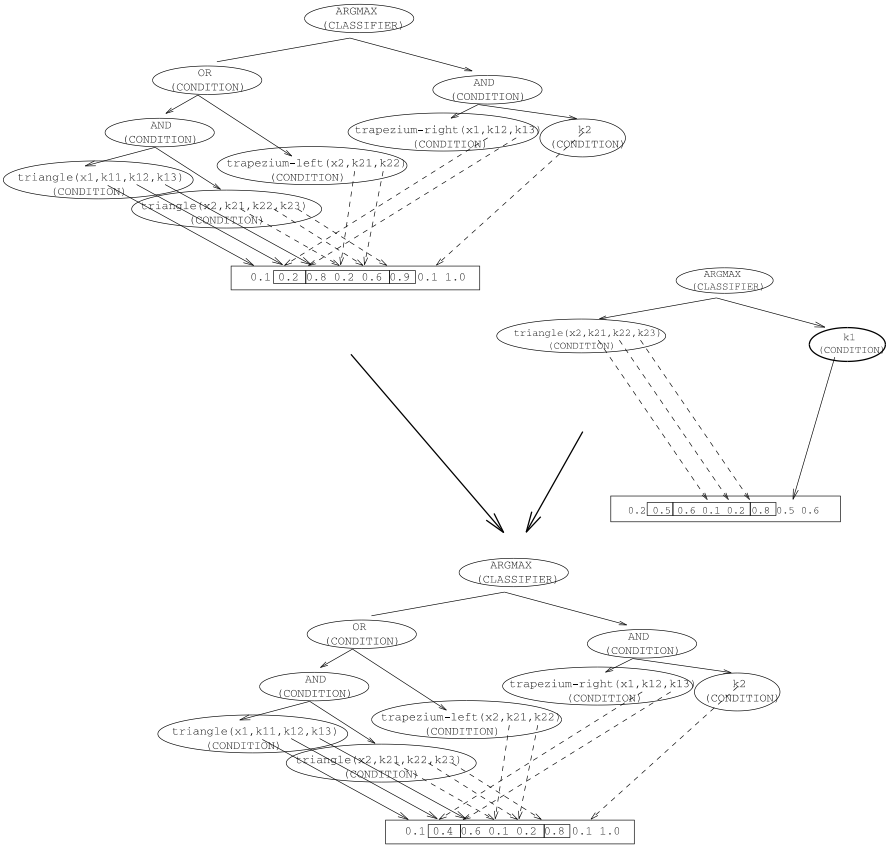

Fig. 6. Chain macromutation: The array of parameters in the individual being mutated (upper part, left) is crossed with the array of parameters of a random individual (upper part, right). The linguistic expression of the rule base is not modified (lower part.) We have used two-point crossover in the figure for clarity, but intermediate recombination in the real algorithm.

until the value number $m+1$, which is the class number (see the example in Section 2.1.)

2.4.1. Semantics of fuzzy rules

Let $\widetilde{l}_{ij}$ be the linguistic label number j in the feature number i. The rule

"`if` x_i `is` $\widetilde{l}_{ij}$ `then class is` c_k `with conf.` α"

defines the relation that follows:

$$R(x_1,\ldots,x_m,c)=\begin{cases}\alpha & \text{if } x_i=\widetilde{l}_{ij} \text{ and } c=c_k,\\ 0 & \text{otherwise.}\end{cases}$$

The chain

"if α then class is c_k"

is valid in our grammar. It means "all points belong to class c_k with truth α". Besides it should not appear in the final base, its definition is useful to make clear the meaning of logical constants in antecedents, as will be shown in Section 2.4.2. This rule defines the following relation:

$$R(x_1,\ldots,x_m,c)=\begin{cases}\alpha & \text{if } c=c_k,\\ 0 & \text{otherwise.}\end{cases}$$

2.4.2. Semantics of logical expressions in antecedents

If the rules

"if A then class is c_k"

"if B then class is c_k"

define the relations R_A and R_B, then the rules

if $A\wedge B$ then class is c_k

if $A\vee B$ then class is c_k

define the relations

$$R_{A\wedge B}(x)=\min(R_A(x),R_B(x)),$$
$$R_{A\vee B}(x)=\max(R_A(x),R_B(x)),$$

respectively. As a consequence of this, the rule

"if $A\wedge\alpha$ then class is c_k"

describes the same relation as the rule

"if A then class is c_k with confidence α".

Therefore, if we allow the presence of logical constants in the antecedents, all relations can be described by rules with confidence 1. For this reason, we do not use degrees of confidence in the linguistic expression of the rules, as was shown in the example in Section 2.1.

2.4.3. Semantics of a concatenation of rules

The concatenation of N rules defined by relations $R_1,R_2,\ldots,R_N$ defines the relation $R(x)=\max\{R_1(x),\ldots,R_N(x)\}$. Two rules with the same consequent can be combined in one. The construction

if A then class is c_k

if B then class is c_k

is identical to

if $A \vee B$ then class is c_k.

3. Genetic programming and simulated annealing

In this section we will incorporate the macromutation operator, and the representation of a fuzzy rule base we developed before, to the SA Metropolis algorithm. There are small differences between GA mutation and the operation needed in SA. The "adjacent" operation in SA produces a random individual which is near the individual being mutated in some sense. This is immediate in real optimization, but not so evident when optimizing a function defined over the chains of a grammar. We have used an edition distance to measure similarity between surface structures of classifiers. This is the minimum number of insertions, deletions and replacements of terminal symbols needed to convert one classifier into the other one.

Let `C` be the genotype of the individual, `param(C)` its array of parameters and `expr(C)` the syntactic tree of the chain. `T` is the current temperature in SA. `p` defines the balance between subtree mutation and chain mutation, as defined in the previous section. `p=1` means "only chain mutation", `p=0` means "only subtree mutation". Chain mutation consists in applying intermediate recombination [11] between the individual and a randomly generated one with an amplitude parameter that depends on the current temperature by a constant K_1. Subtree mutation is inside a loop that finishes when the distance between the result and the individual is lower than a limit that also depends on the current temperature by a factor K_2.

```
algorithm macromutation
needs: C, p, T, K_1, K_2
produces: C1
 if U(0,1)<p then
  A=random real vector
  param(C)=
    param(C)*(T/K_1)+(1-(T/K_1))*A
 else
  repeat
   E=expr(C)
   select a subtree of E
   replace it by a randomly generated subtree
  until edition_distance(E,expr(C))<T/K_2
  expr(C)=E
 end if
```

The macromutation operator is called from the algorithm that follows. It depends on an initial temperature and a final temperature, that serve to stop the learning process; alternatively one can stop when the number of fitness evaluations is high enough.

```
algorithm sa
needs: cooling factor, Tinitial, Tfinal, p, K_1, K_2
produces: Cbest

T = Tinitial
C = Cbest = random individual
while T > Tfinal do
 Cl = macromutation(C, p, T, K_1, K_2)
 delta = error(Cl)-error(C)
 v = random value with uniform distribution U(0, 1)
 if delta < 0 or v<exp(-delta/T) then
  C = Cl
  if C < Cbest then Cbest = C end if
 end if
 T = T*cooling factor
end while
```

4. Numerical results

All values defining SA algorithm execution parameters are displayed in Table 1. Every experiment has been repeated 10 times starting from different, random candidates. SA was rather more sensible to initial temperature and cooling pattern than GA is to its own execution parameters, so tuning the learning was more difficult in SA than it was in GA. As a rule of thumb, we

Table 1
Execution parameters of SA. Distance in subtree mutation is not limited, $K_2 = 0$, distance in intermediate recombination is limited to a 20%, $K_1 = 5$

Parameter	Meaning	Value
Cooling rate	Temperature decrease/iteration	0.9999
T_0	Initial temperature	5
Iterations	Maximum number of iterations	50 000
K_1	Inverse of maximum (per unit) distance in intermediate recombination	5
K_2	Inverse of maximum jump in subtree mutation	0
p	Probability of mutation in the chain of numbers	0.5

obtained good results with an initial temperature approximately equal to the expected per cent error of the classifier (i.e., $T_0 = 5$ if we expect 5% error in the final classifier).

We used 10 subpopulations of 100 individuals each in all genetic programming experiments. A niching scheme was combined with every subpopulation (10 niches of constant size), and 1% of the offspring of crossovers was inserted into a subpopulation different than their parents'. Steady-state and tournament selection of size 3 was used. Two tournament losers were replaced by the offspring in every crossover. One per cent of the offspring is mutated. Chain crossover is always applied when both parents belong to the same niche, and 50%/50% subtree/chain crossover otherwise. Experiments were repeated 10 times from random populations. Evolution finishes after 50 000 fitness evaluations in both SA and GA-P algorithms.

In Table 2 we can see that SA is not different from genetic programming in all datasets. In Table 3 the results of applying k-NN, linear classifiers and multilayer perceptrons to the same datasets are included. "Not different" means that differences are not statistically significant if 5×2cv test is applied (95%).

Table 2
Comparison of results between genetic programming and simulated annealing when inducting fuzzy rule bases in classification problems

Dataset	GA-P		Rules	Variables uses/tot	SA		Rules	Variables uses/tot
	Mean	Dev.						
Cancer-1	2.58	0.78	5	5.1/9	2.93	1.07	4.9	4.9/9
Cancer-2	5.91	1.20	5.1	4.7/9	5.74	1.28	5.4	5.2/9
Cancer-3	5.34	1.33	5.4	4.3/9	5.28	0.56	5	5.3/9
Mean	4.62			4.7/9	4.65			5.1/9
Thyroid-1	5.18	0.67	5.3	4.8/21	4.51	1.08	5.5	5/21
Thyroid-2	4.51	0.80	5.9	3.9/21	4.16	1.34	5.2	5.2/21
Thyroid-3	4.90	0.70	5.6	4.3/21	4.38	1.66	4.8	5.1/21
Mean	4.87			4.3/21	4.35			
Pima-1	25.57	1.12	4.7	4.5/8	25.41	1.86	3.9	4.1/8
Pima-2	28.07	2.10	4.7	4.2/8	27.29	1.66	3.7	4.1/8
Pima-3	24.06	1.66	4.7	4/8	22.86	0.94	3.4	4.2/8
Mean	25.90			4.2/8	25.19			4.1/8
Glass-1	43.96	7.91	9.1	6.2/9	41.88	4.97	7.8	5.5/9
Glass-2	41.13	2.03	8.7	5.9/9	42.07	5.34	9.1	6.6/9
Glass-3	44.52	7.55	10.2	7.2/9	42.45	7.41	8.6	5.8/9
Mean	43.20			6.4/9	42.13			6/9

In this table, means and standard deviation of the test results obtained after repeating 10 times every experiment are shown. The equivalent "only and" number of rules and the mean number of variables used are also displayed.

Table 3
Results obtained with k-NN, linear and multilayer perceptron classifiers over the same datasets used in the fuzzy rule induction problem

Problem	k-NN	Linear		MLP	
		Mean	Dev.	Mean	Dev.
Cancer-1	1.7	2.93	0.18	1.38	0.49
Cancer-2	4.0	5.00	0.61	2.38	0.35
Cancer-3	4.5	5.17	0.00	3.70	0.52
Media	3.4	4.36		3.28	
Thyroid-1	5.95	6.56	0.00	2.38	0.35
Thyroid-2	6.00	6.56	0.00	1.91	0.24
Thyroid-3	6.50	7.23	0.02	2.27	0.32
Media	6.15	6.78		2.18	
Pima-1	25.5	25.83	0.56	24.10	1.91
Pima-2	27.6	24.69	0.61	26.42	2.26
Pima-3	23.5	22.92	0.35	22.59	2.23
Media	25.5	24.48		24.37	
Glass-1	35.8	46.04	2.21	32.70	5.34
Glass-2	33.9	55.28	1.27	55.57	3.70
Glass-3	35.0	60.57	3.82	58.40	7.82
Media	34.9	53.96		48.89	

MLP and linear classifier results are taken from [12].

In Table 4 the results of applying SA to PIMA dataset are tabulated for train and test partitions for different limits in the jumps in the subtree mutation. Best results are obtained when no restrictions are imposed over these jumps. This result suggests to us that the edition distance is not correlated with the fitness landscape: two similar individuals do not always have small differences in their classification error.

Finally, in Table 5, the most relevant characteristics, as found with statistical methods, and the most frequently used variables in GP and SA are displayed. The statistical method is a greedy algorithm that selects characteristics by querying mutual information between the class and every variable. It is clear that there exists a relation between the order of the variables as determined by their mutual information and the order defined by their relative frequency of use. This frequency is calculated repeating SA or GP induction 10 times from random starting values and counting how many times each feature appears in the final classifier. This shows that allowing a variable number of features in the antecedents causes both GP and SA to perform a meaningful feature selection in the learning process.

5. Concluding remarks and future work

In this work we have shown that the research devoted to new representations of fuzzy rule bases and to application-dependent crossover operators can

Table 4
Comparative results: SA with subtree mutation limited to different degrees. Best results are obtained when limits are not applied

K_2	Pima-1		Pima-2		Pima-3		Pima-1		Pima-2		Pima-3		μ TRA	μ TST
5	39.28	11.68	39.06	12.54	41.61	14.43	40.63	10.17	41.15	8.69	41.03	14.46	36.65	40.93
1	21.65	0.59	21.15	0.35	22.38	0.44	25.88	1.42	28.22	1.41	22.60	1.53	21.72	25.55
0.5	21.14	0.46	20.83	0.47	22.25	0.38	25.15	1.51	28.95	1.45	23.54	1.37	21.40	25.86
0.2	21.28	0.41	20.87	0.44	21.86	0.46	25.78	2.00	27.66	1.87	23.39	1.39	21.34	25.61
0	21.18	0.41	20.63	0.43	21.89	0.42	25.42	1.81	27.29	1.66	22.86	0.94	21.23	25.19

Table 5
There exists a relation between the order of the variables as determined by their mutual information and the order defined by their relative frequency of use when SA or GP induction is repeated a number of times from random starting values

	Variables (IM)	Variables (GA)	Variables (SA)
Cancer	0 1 2 5 6	0 1 2 5 7	0 1 2 5 7
Pima	0 1 5 7	1 5 6 7	1 5 6 7
Glass	1 2 3 4 5 6	2 3 4 5 6 7	1 2 3 5 6 7
Thyroid	2 16 17 18 20	0 2 16 19 20	2 16 17 18 19

The numbers of the most relevant features (the first feature has number 0) are shown for statistical methods (mutual information), Genetic Algorithms and Simulated Annealing.

be applied to different search schemes, allowing them to be applied to new fields. It remains to be studied whether this technique is also useful for inducting rules in other problems. Preliminary results in fuzzy modeling go in this direction. The ability of this method to perform a feature selection is also interesting and it should be studied in depth whether the quality of this selection is high enough to be used in combination with other learning algorithms (i.e., using GP or SA to do the feature selection and then applying a different learning method over the reduced dataset).

With respect to the practical usefulness of this method, SA was more difficult to adjust than GP, but the memory requirements of this algorithm are very reduced and this can be an advantage when the size of the individuals (the number of rules) is large.

References

[1] S. Abe, M.S. Lan, A method for fuzzy rule extraction directly from numerical data and its application to pattern classification, IEEE Transactions on Fuzzy Systems 3 (1) (1995) 18–18.
[2] S. Abe, R. Thawonmas, A fuzzy classifier with ellipsoidal regions, IEEE Transactions on Fuzzy Systems 5 (3) (1997) 358–368.
[3] O. Cordón, M.J. Del Jesus, F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, International Journal of Approximate Reasoning 20 (1) (1991) 21–45.
[4] A. Geyer-Schulz, Fuzzy Rule-Based Expert Systems and Genetic Machine Learning, second ed., Physica-Verlag, Wurzburg, 1997.
[5] D.J. Hand, Discrimination and Classification, Wiley, New York, 1981.
[6] T.P. Hong, C.Y. Lee, Induction of fuzzy rules and membership functions from training examples, Fuzzy Sets and Systems 84 (1996) 33–47.
[7] L. Howard, D. D'Angelo, The GA-P: a genetic algorithm and genetic programming hybrid, IEEE Expert (1995) 11–15.
[8] H. Ishibuchi, Distributed representation of fuzzy rules and its application to pattern classification, Fuzzy Sets and Systems 52 (1992) 21–32.
[9] T. Jones, Crossover, macromutation, and population-based search, in: Proceedings of the Sixth International Conference on Genetic Algorithms, Pittsburgh, PA, 15–19, 1995, pp. 73–80.

[10] D. Montana, Strongly typed genetic programming, Evolutionary Computation 3 (1995) 199–230.
[11] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization, Evolutionary Computation 1 (1) (1993) 25–49.
[12] L. Prechelt, PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
[13] S.K. Pal, D.P. Mandal, Linguistic recognition system based in approximate reasoning, Information Sciences 61 (1992) 135–161.
[14] L. Sánchez, S. García Carbajal, Fuzzy classifier induction with GA-P algorithms, ESTYLF-EUSFLAT Joint Conference, Palma de Mallorca, 1999.
[15] L.X. Wang, J. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems, Man and Cybernetics 25 (2) (1992) 353–361.
[16] Y. Yuan, H. Zhuang, A genetic algorithm for generating fuzzy classification rules, Fuzzy Sets and Systems 84 (1996) 1–19.
[17] L. Zadeh, Fuzzy logic, neural networks and soft computing, Communications of the ACM 37 (3) (1994) 77–84.