Advanced Topics in Computational Intelligence

Master AI

GENETIC FUZZY SYSTEMS

Àngela Nebot

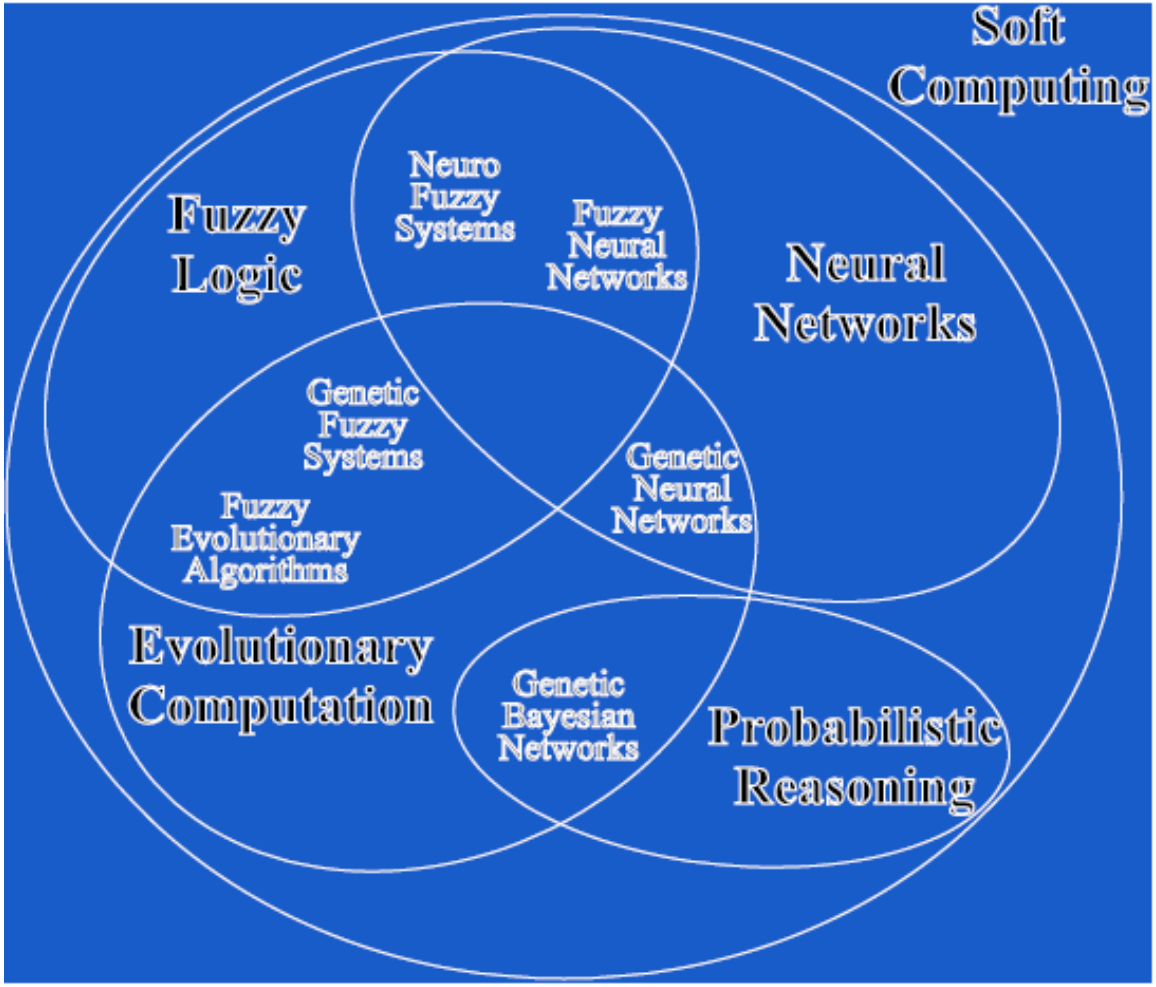
Introduction to Genetic Fuzzy Systems

- The use of genetic/evolutionary algorithms (GAs) to design fuzzy systems constitutes one of the branches of the Soft Computing paradigm: genetic fuzzy systems (GFSs)

- The most known approach is that of genetic fuzzy rule-based systems, where some components of a fuzzy rule-based system (FRBS) are derived (adapted or learnt) using a GA

- Some other approaches include genetic fuzzy neural networks and genetic fuzzy clustering, among others

Genetic Fuzzy Systems and Soft Computing



Introduction to Genetic Fuzzy Systems

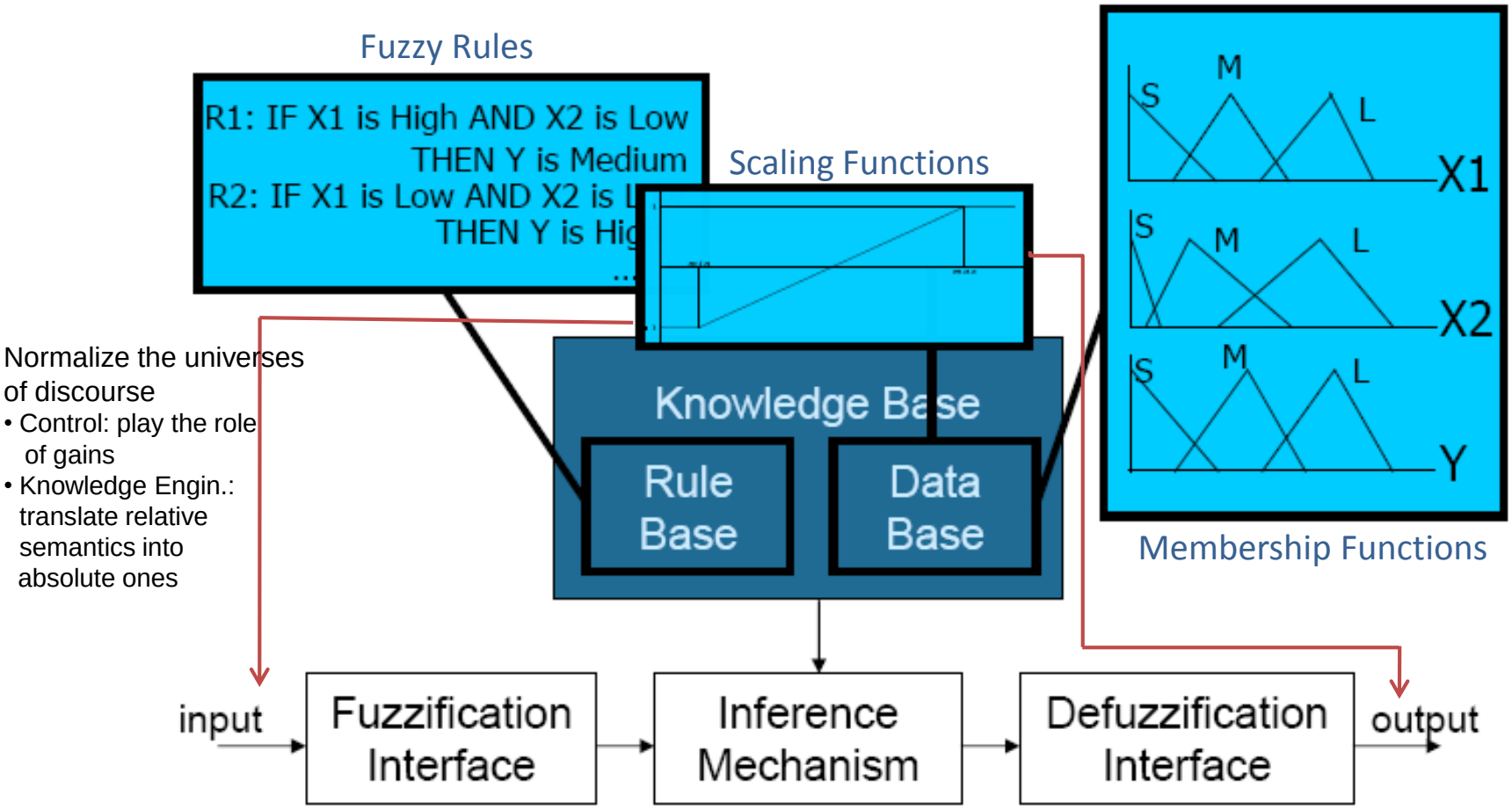
Design of fuzzy rule-based systems:

- An FRBS (regardless it is a fuzzy model, a fuzzy logic controller or a fuzzy classifier), is comprised by two main components:
  - The Knowledge Base (KB), storing the available problem knowledge in the form of fuzzy rules
  - The Inference System, applying a fuzzy reasoning method on the inputs and the KB rules to give a system output

- Both must be designed to build an FRBS for a specific application:
  - The KB is obtained from expert knowledge or by machine learning methods
  - The Inference System is set up by choosing the fuzzy operator for each component (conjunction, implication, defuzzifier, etc.)

    Sometimes, the latter operators are also parametric and can be tuned using automatic methods

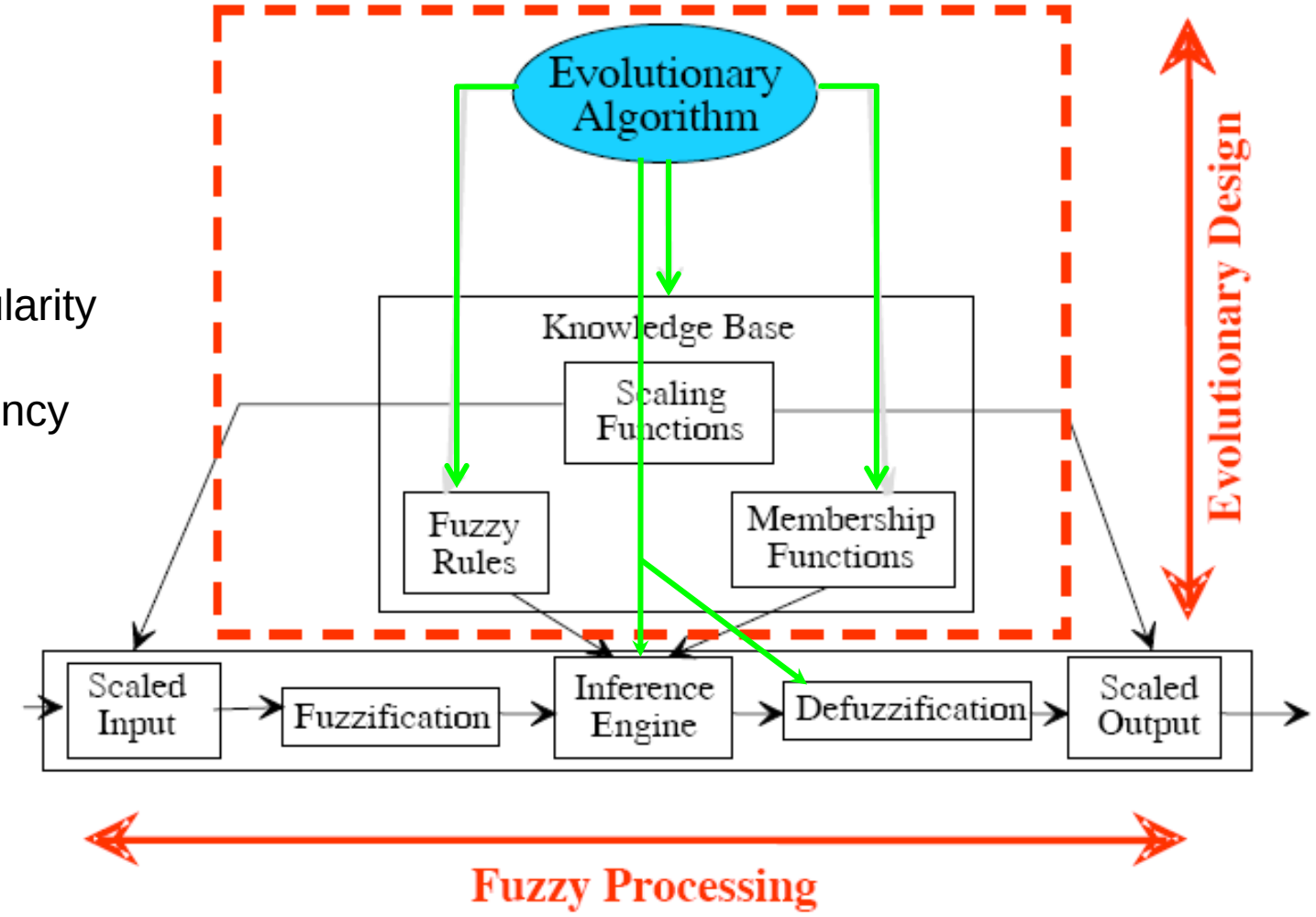
Introduction to Genetic Fuzzy Systems



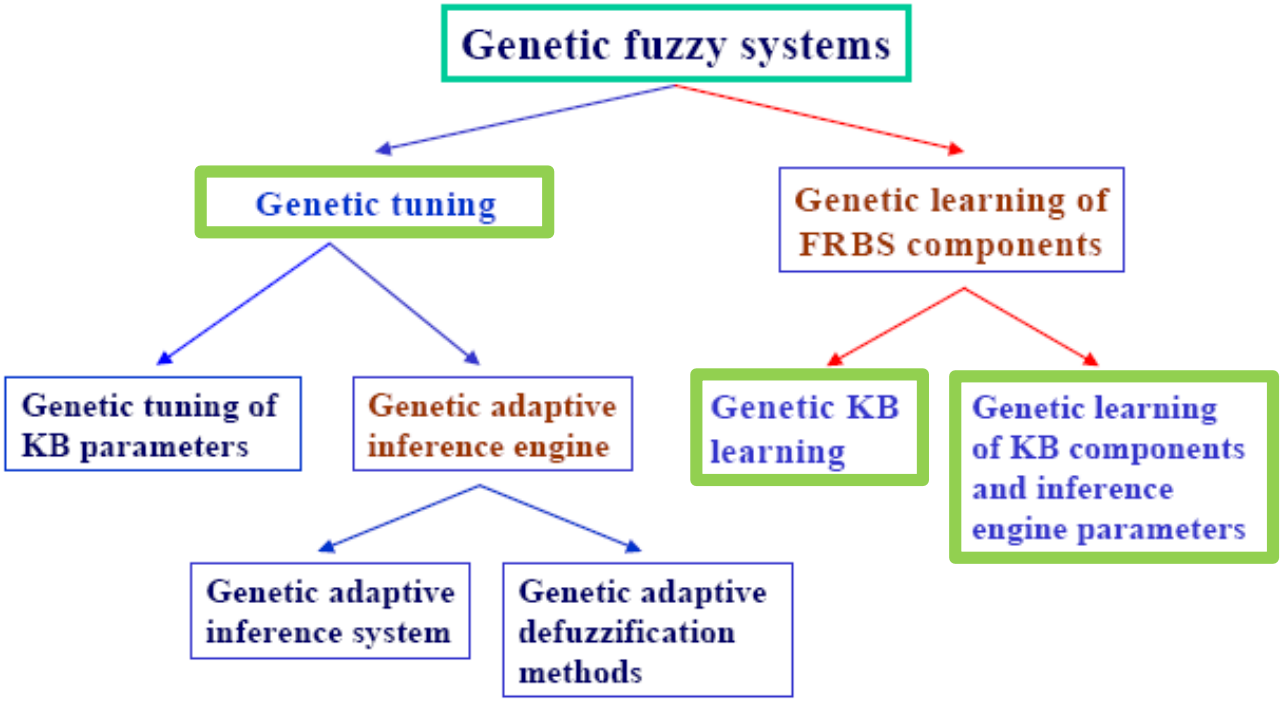
Fuzzy rule-based system

Introduction to Genetic Fuzzy Systems

Granularity
vs.
Efficiency



Taxonomy of Genetic Fuzzy Systems



Concerned with optimisation of an existing FRBS

Automated design method that does not depend on a predefined set of rules

Classical GFS learning approaches

- **Genetic derivation of the FRBS Rule Base**
  - Pittsburgh learning approach
  - Michigan learning approach
  - Iterative Rule learning approach (SLAVE)

The cooperation versus competition problem

There is a major problem that appears when designing FRBSs by means of EAs which has to be solved adequately in order to obtain accurate FRBS:

• On the one hand, GFRBSs take an interesting feature of the FRBSs into account, the interpolative reasoning they develop, which is a consequence of *cooperation among the fuzzy rules composing the KB*.

• On the other hand, the main feature of an EA is considered, which is *the competition induced among the population members representing possible solutions to the problem being solved*, which allows us to obtain better ones.

Since a GFRBS combines both said features, it works by inducing competition to get the best possible cooperation. The problem is to find the best possible way to put this into effect.

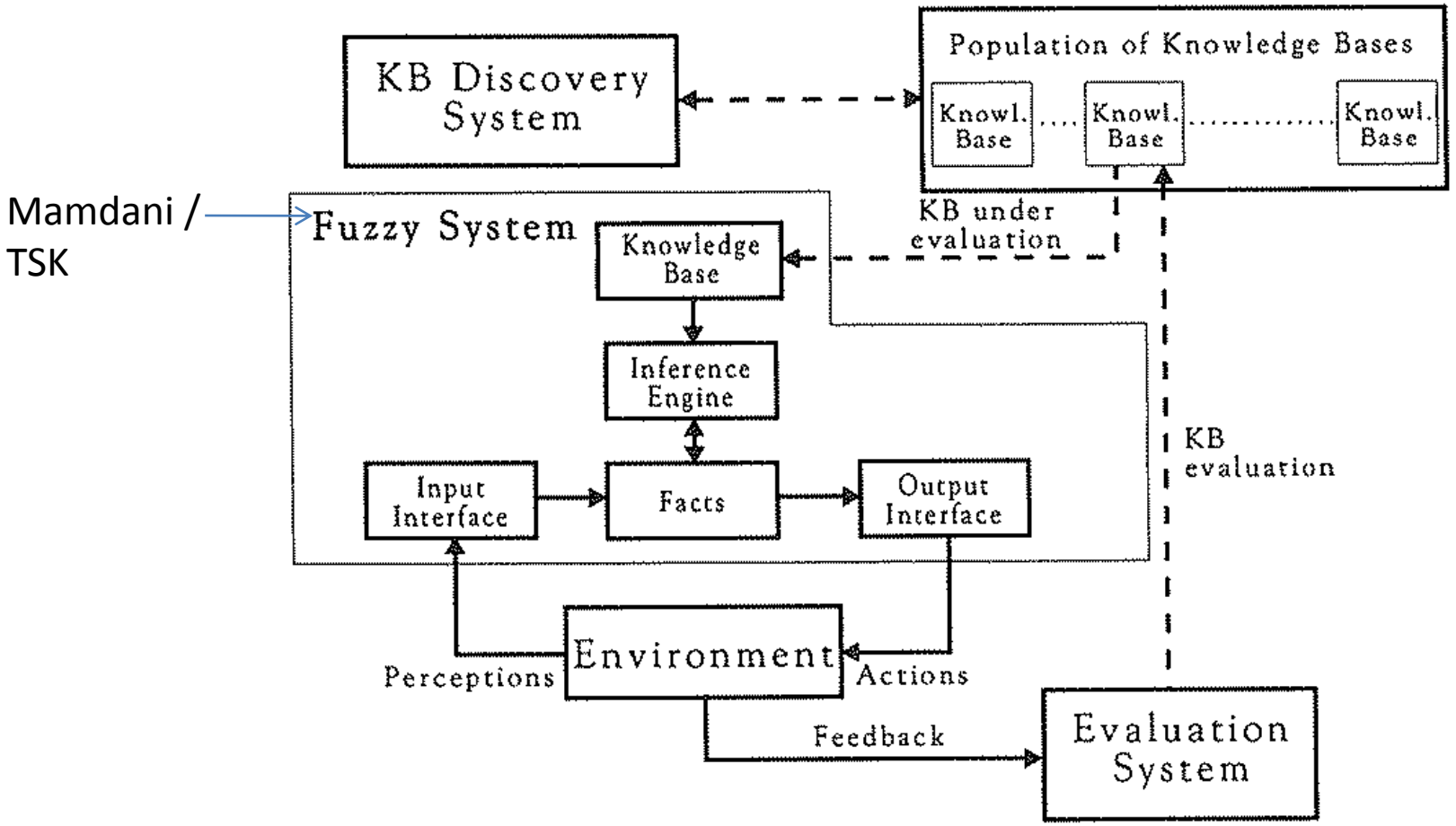
Genetic derivation of the FRBS Rule Base

Pittsburgh Learning Approach:

- Each chromosome encodes a whole fuzzy rule set and the derived RB is the best individual of the last population
- The fitness function evaluates the performance at the complete RB level, so the CCP is easy to solve
- However, the search space is huge, thus making difficult the problem solving and requiring sophisticated GFS designs
- Mainly used in off-line learning (fuzzy modeling and classification applications)

Genetic derivation of the FRBS Rule Base

Pittsburgh Learning Approach:



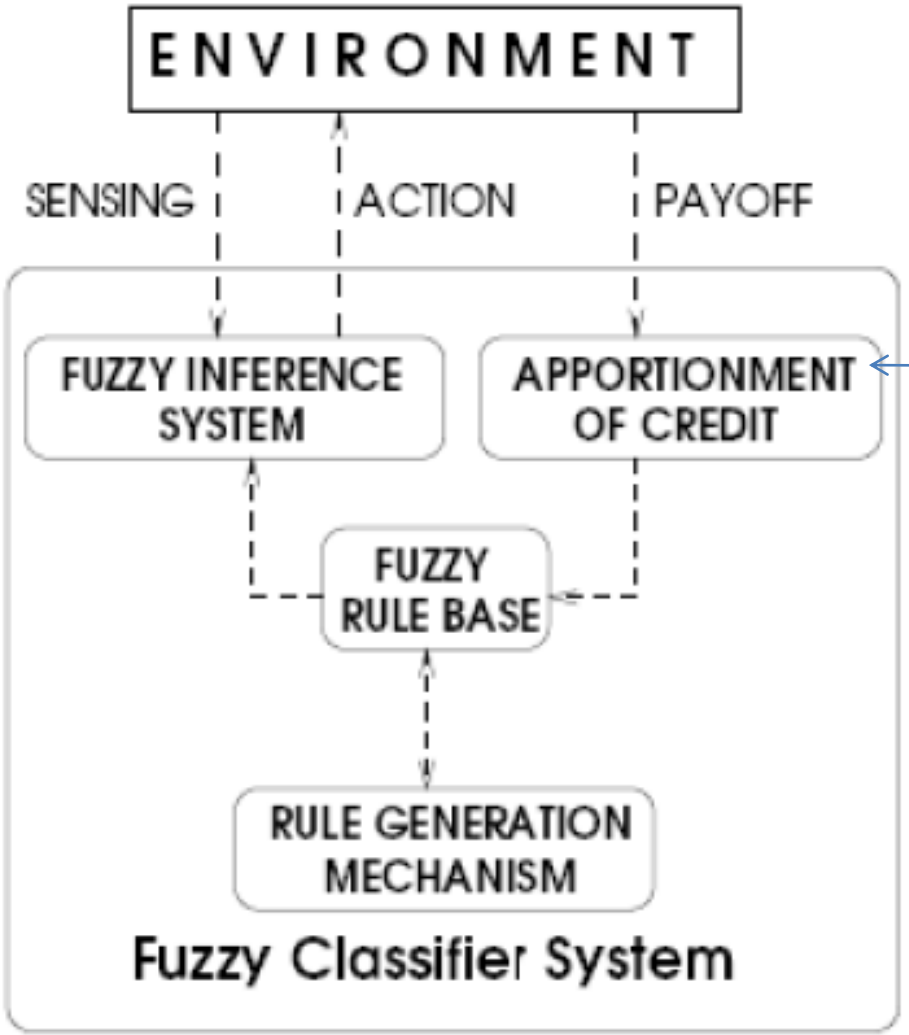
Genetic derivation of the FRBS Rule Base

Michigan Learning Approach:

- Each chromosome encodes a single fuzzy rule and the derived RB is composed of the whole population
- Reinforcement mechanisms (reward (credit apportion) and weight penalization) are considered to adapt the rules through a GA
- Low weight (bad performing) rules are substituted by new rules generated by the GA
- The key question is to induce collaboration in the derived RB as the evaluation procedure is at single rule level (cooperation vs. competition problem (CCP))
- Mainly used in on-line learning (fuzzy control applications)

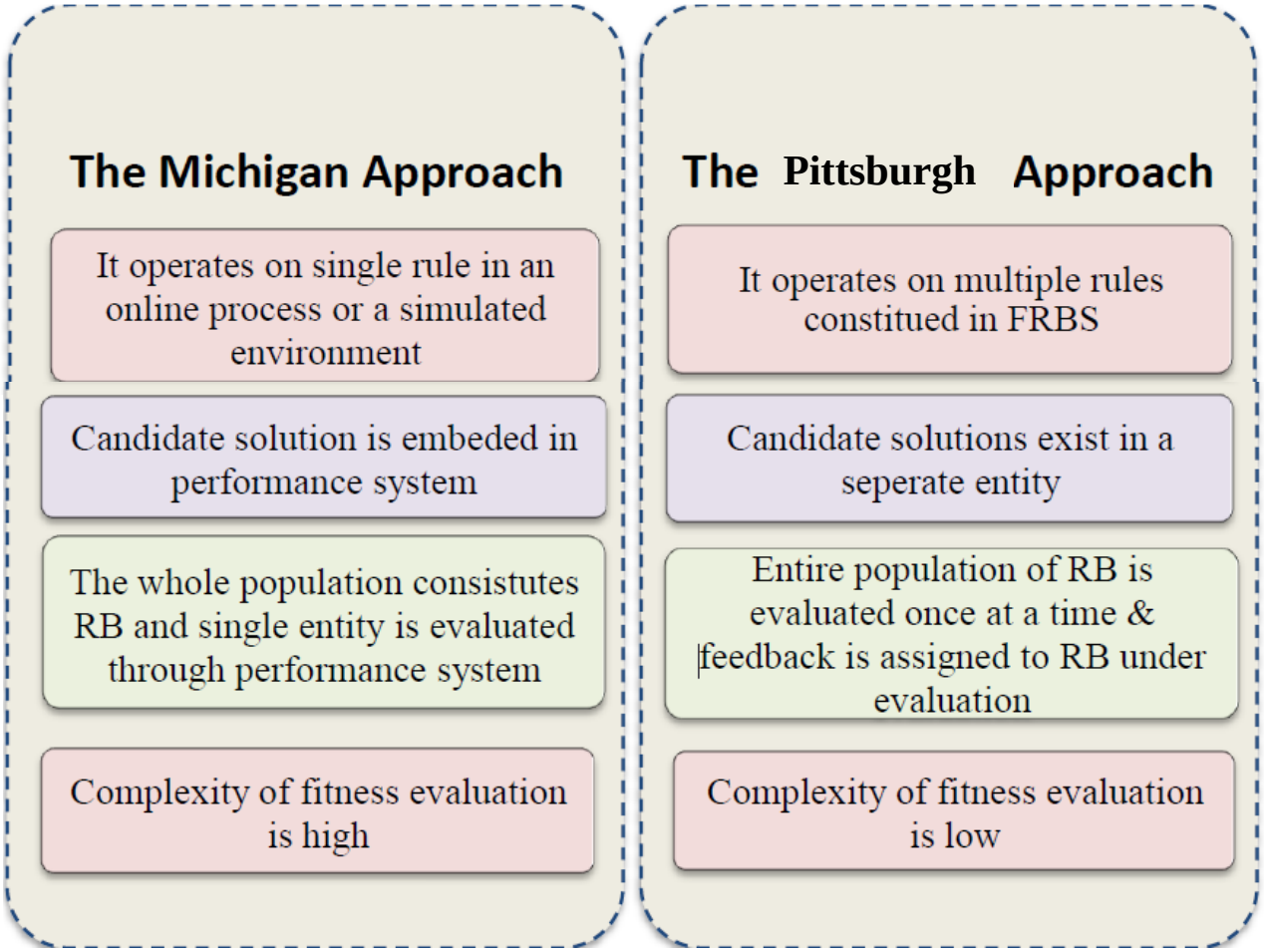
Genetic derivation of the FRBS Rule Base

Michigan Learning Approach:



Reward or penalise individual rules based on their contribution to the overall success or failure of the entire fuzzy classifier system

Pittsburgh vs Michigan Approaches



Genetic derivation of the FRBS Rule Base

Iterative Rule Learning Approach:

- Intermediate approach between the Michigan and Pittsburgh ones, based on partitioning the learning problem into several stages and leading to the design of multi-stage GFSs

- As in the Michigan approach, each chromosome encodes a single rule, but a new rule is learnt by an iterative fuzzy rule generation stage and added to the derived RB, in an iterative fashion, in independent and successive runs of the GA
- The evolution is guided by data covering criteria (rule competition). Some of them are considered to penalize the generation of rules covering examples already covered by the previously generated fuzzy rules (soft cooperation)

Generation Process: derives a preliminary set of rules representing the knowledge existing in the data set

Genetic derivation of the FRBS Rule Base

Iterative Rule Learning Approach:

- A second post-processing stage is considered to refine the derived RB by selecting the most cooperative rule set and/or tuning the membership functions (cooperation induction)

- Hence, the CCP is solved taking the advantages of both the Michigan and Pittsburgh approaches (small search space and good chances to induce cooperation)

- Mainly used in off-line learning (fuzzy modeling and classification applications).

Post-processing Process: refines the previous rule set in order to remove the redundant rules and select those fuzzy rules that cooperate in an optimal way

SLAVE generation process

Generation process:

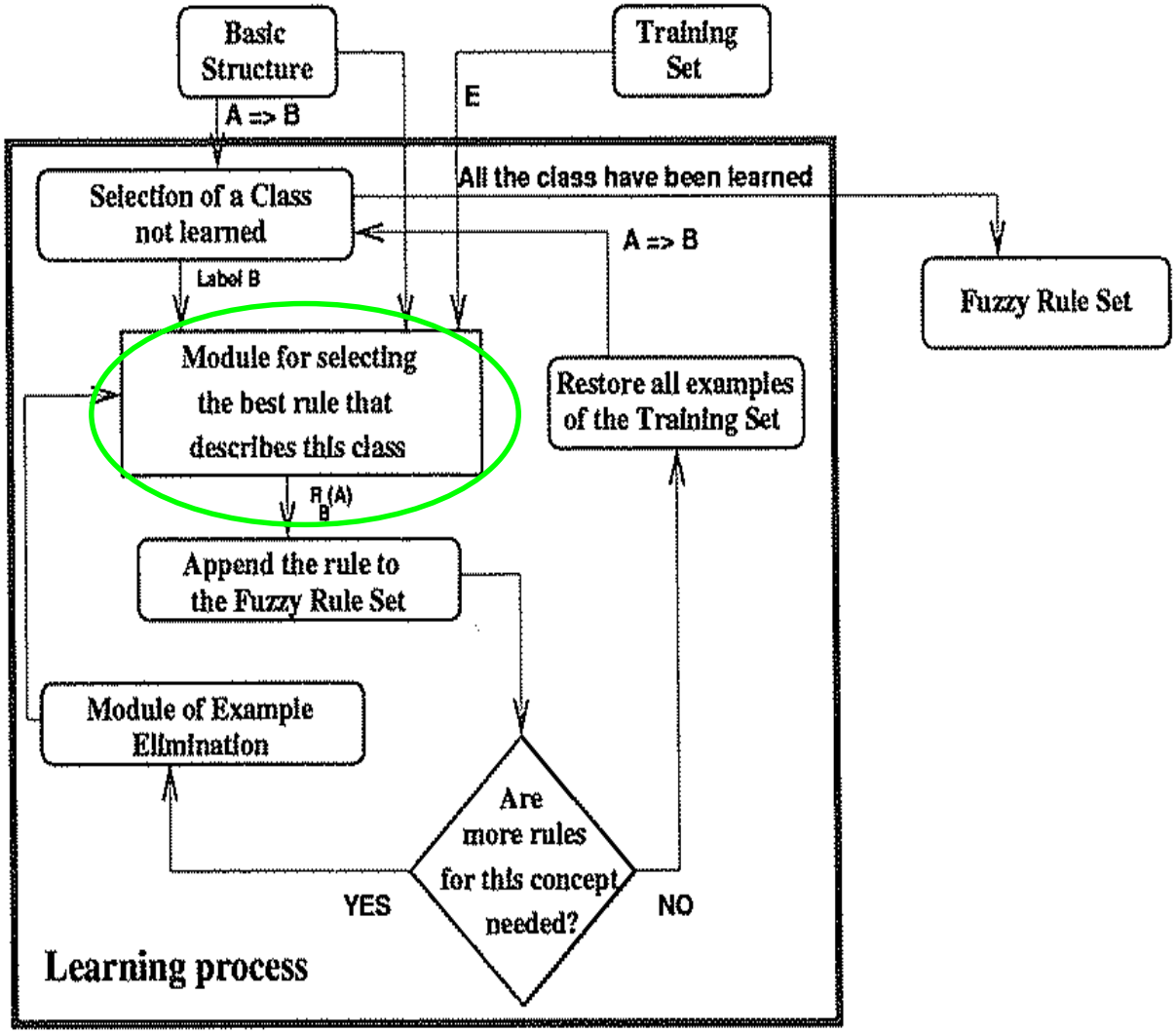
1. Use a GA to obtain a rule for the system

2. Incorporate the rule into the final set of rules

3. Eliminate from the data set all those examples that are covered by this rule to a sufficient degree λ (λ-covered)

4. If the set of rules obtained is sufficient to represent the examples in the training set, the system ends up returning the set of rules as the solution. Otherwise return to step 1.

SLAVE generation process



The rule model in SLAVE

- A common approach to code individual rules is the use of the disjunctive normal form (DNF) represented in the form of a fixed length binary string

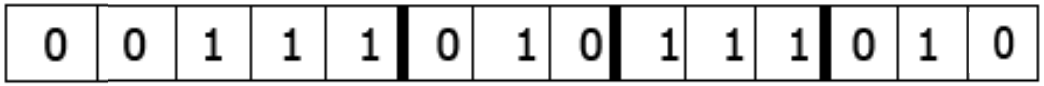
$$\ldots \text{ and } X_i = \{L_4, L_5, L_6\} \text{ and } \ldots$$

$$\ldots \text{ and } \{X_i \text{ is } L_4 \text{ or } X_i \text{ is } L_5 \text{ or } X_i \text{ is } L_6\} \text{ and } \ldots$$

The rule model in SLAVE

- A DNF fuzzy rule allows an antecedent variable to take a disjunction of linguistic terms from its domain as a value:

IF Femur_length is (medium or big-medium or big) and Head_diameter is (medium) and Foetus_sex is (male or female or unknown) THEN Foetus_weight is normal



Femur_length ={small,small-medium, medium, big-medium,big}

Head_diameter ={small,medium,big}

Foetus_sex ={male,female,unknown}

Foetus_weight = {low, normal, high}

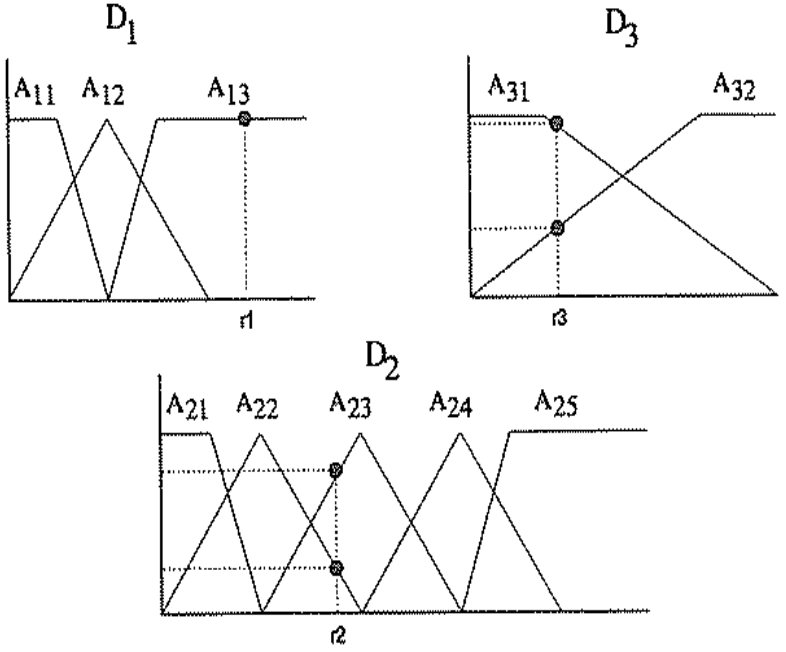
IF Femur_length is (medium or big-medium or big) and Head_diameter is (medium) and ~~Foetus_sex is (male or female or unknown)~~ THEN Foetus_weight is normal

SLAVE parameters

- ***Genetic operators:*** two-points crossover, uniform mutation. Can use proportional, linear ranking and elitist selection schemes.

- ***Generation of the initial population:*** The initial population is generated by randomly selecting examples from the class that must be learned (i.e. matches de rule consequent value) and obtaining for each one the most specific rule antecedent that best describes the example.

Crisp input vector of the example: $(r1, r2, r3)$



X_1 *is* A_{13} *and* X_2 *is* A_{23} *and* X_3 *is* A_{31}

C= (0010010010)

Note: the consequent part is specified by the iterative covering method

SLAVE parameters

•***Fitness function:*** to generate fuzzy classification rules that give the consistency criterion priority over the positive examples.

$$F(R_i) = \begin{cases} n^+(R_i), & \text{if } R_i \text{ is } k\text{-consistent} \\ 0, & \text{otherwise} \end{cases}$$

- *Positive example* for R_i: it matches its antecedent as well as its consequent; *Negative example:* it matches de antecedent and not the consequent.

- $n^+(R_i)$, number of positive examples.

- R_i is *k*-consistent when its associated number of negative examples is less than a percentage *100.k* of the number of the positive examples.

SLAVE parameters

•***Termination criteria:*** the GA ends returning the best rule of the last population if:

- the number of iterations is greater than the fixed limit.

- the fitness function of the best rule in the population does not increase its value within a fixed number of iterations and rules with this consequent have already been obtained.

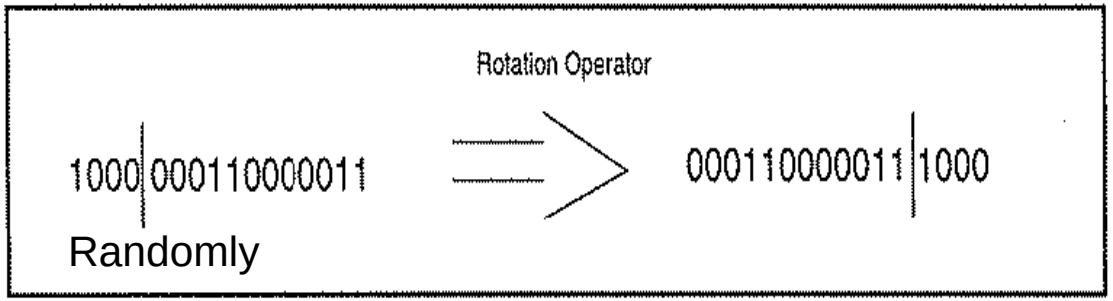
- when no rule has been obtained for this concept yet, the generating process continues, until the current best rule eliminates at least one example from the training set.

SLAVE new genetic operators

Introducing new genetic operators with the intention of:

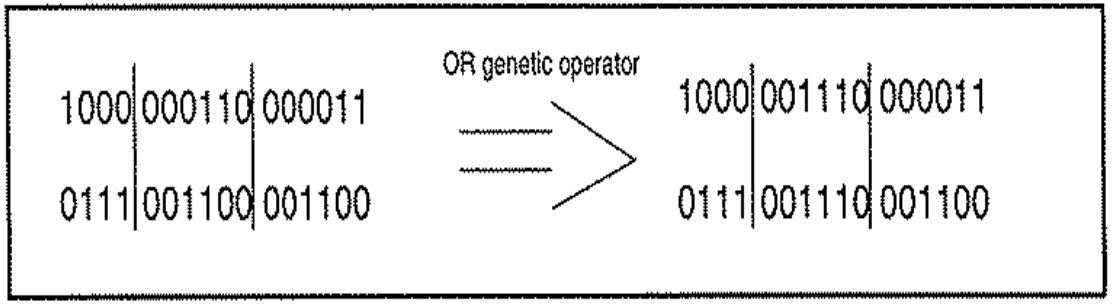
1. Improving the diversity in the genetic population
2. Increasing the understanding of the rules that are obtained

Rotation operator (1):



OR operator (1):

(crossover)



SLAVE new genetic operators

Generalization operator (2):

Antecedents of variable *X2*: **00111/00101; fitness(00111)=fitness(00101)**

$\ldots$ *and* X_2 *is* $\{A_{23}, A_{24}, A_{25}\}$ *and* $\ldots$

.. *and* X_2 *is higher than or equal to* A_{23} *and* ..

$\ldots$ *and* X_2 *is* $\{A_{23}, A_{25}\}$ *and* $\ldots$

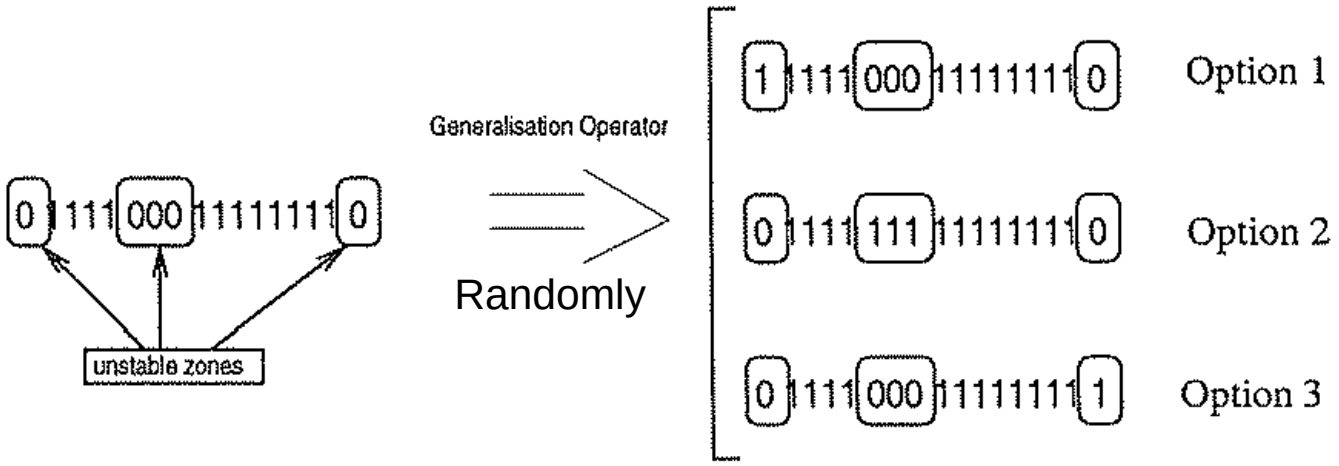
Red more understandable description from a descriptive point of view but the GA selects randomly one of them.

This operator tries to maximize the number of stable variables in the rule, in the sense that a variable is considered stable if the terms associated to it form a unique, consecutive sequence without gaps.

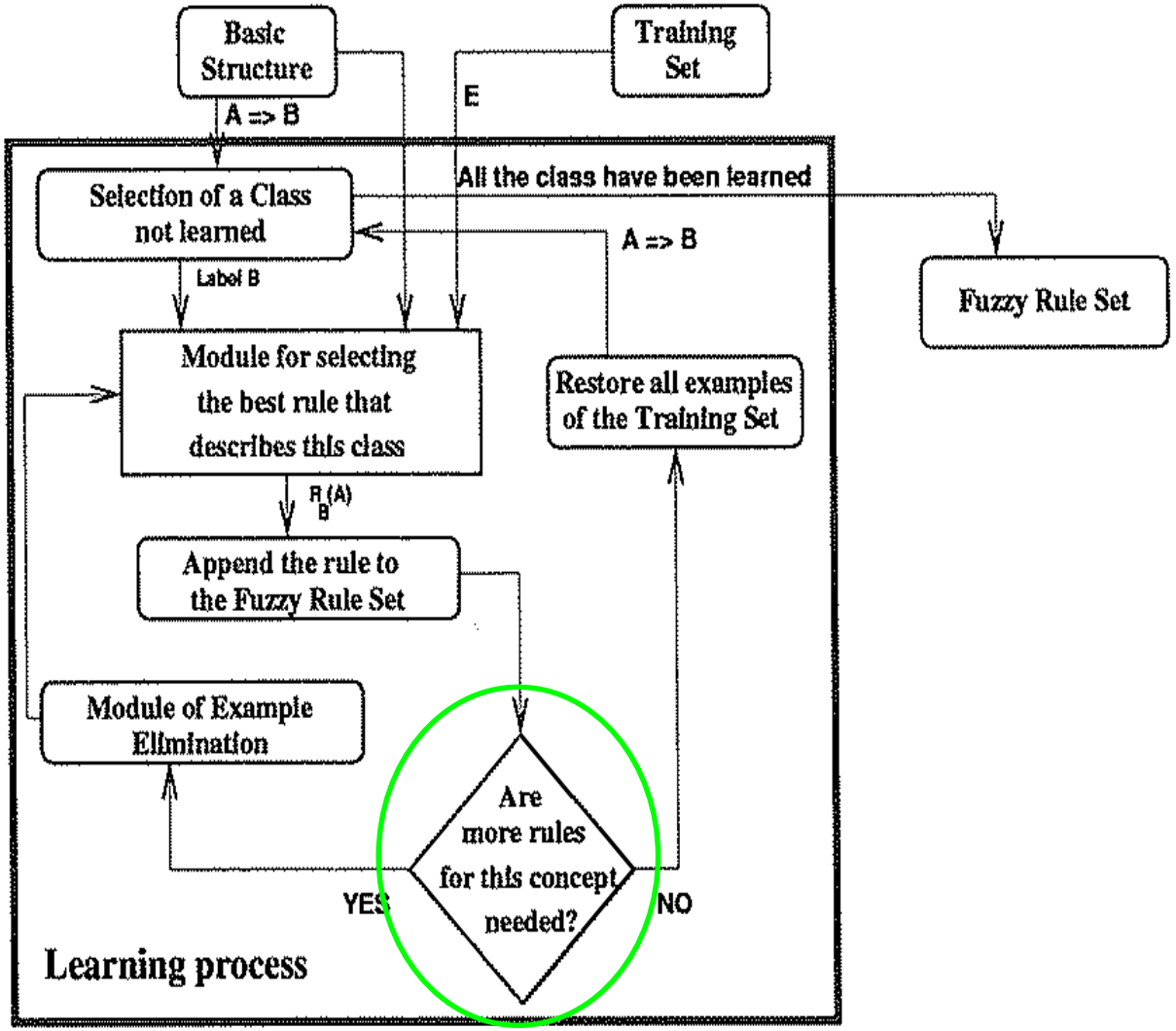
SLAVE new genetic operators

The generalization operator works as follows:

1. *A variable of a DNF rule antecedent encoded in the individual is selected at random.*
2. *If this variable is unstable, do the following:*
   - *2.1 Detect its unstable regions and randomly select one of them.*
   - *2.2 Replace all the '0'-bits in that region with a consecutive sequence of '1'-bits.*
   - *2.3 If the fitness value of the original chromosome is smaller than or equal to that of the new chromosome, substitute the former by the latter*



SLAVE iterative covering method



SLAVE iterative covering method

• The termination criteria in the SLAVE covering method is based on the *weak completeness condition for a class.*

• The SLAVE covering method might terminate generating fuzzy rules for a concept (associated to the output) even though there remain class examples not yet covered by the current fuzzy rule set.

• A set of rules $\{R_1, R_2, \ldots, R_s, R_{s+1}\}$, with output class C satisfies the weak completeness condition if and only if,

$$\begin{cases} R_{s+1} \text{ is not } k\text{-consistent} \\ or \\ R_{s+1} \text{ is } k\text{-consistent and } E_\lambda(R_{s+1}) = \emptyset. \end{cases} \longleftarrow \begin{array}{c} \text{examples } \lambda\text{-covered} \\ \text{by } R_{s+1}. \end{array}$$

• The iterative covering method terminates upon the first rule R_{S+1} that satisfies the weak completeness condition, without adding it to the final rule set.

SLAVE post-processing process

- The post-processing algorithm tries to promote the cooperation among the fuzzy rules in order to improve the accuracy of the FRBs and to simplify the fuzzy rule set.

- The post-processing algorithm is an heuristic algorithm that uses the ***hill-climbing*** optimization strategy (local search) and it is composed of operations of generalization, addition and elimination of rules that are repeated until the global error and the number of rules cannot be decreased.

- It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

SLAVE Examples

- The IRIS database [11]. In this database we try to classify three different classes of plants using four predictive variables, using 150 examples. All the predictive variables have a continuous domain and we have considered five uniformly distributed linguistic labels for discretizing each range.
- The INFARCTION database [22, 15]. This database is made up by 14 different variables for determining the nominal diagnostic variable. There are 2 nominal variables and the rest of them are continuous. These variables are divided into two different type of tests, morphological methods and biochemical analysis. These tests are accepted as the best markers for the postmortem diagnosis of myocardial infarction. The databases has 78 examples and contains missing values. We have considered seven uniformly distributed linguistic labels for discretizing the continous variables.
- WINE recognition data. These data are the results of a chemical analysis of wines from the same region but different types of grapes, using 13 continuous variables and 178 examples. We have used seven uniformly distributed linguistic labels for discretizing the range of each variable.

SLAVE Examples

		IRIS	INFARCTION	WINE
C4.5	accuracy	92.7	81.4	93.2
C4.5rules	accuracy	94.4	82.2	93.5
the original SLAVE	accuracy	95.72	83.22	88.54
	rules	4.2	11.8	21.8
	time	1.00	1.00	1.00
only OR	accuracy	95.72	83.41	89.13
	rules	4.2	8.6	18.3
	time	0.94	0.36	0.48
OR and Generalization	accuracy	95.72	86.4	90.44
	rules	4.4	6.8	16.6
	time	0.46	0.16	0.29
OR and Rotation	accuracy	95.72	89.04	93.32
	rules	4.2	8.4	16
	time	0.46	0.17	0.28
OR, Generalization, Rotation	accuracy	95.72	90.03	93.83
	rules	4.4	7.4	16.2
	time	0.45	0.16	0.27

References

- Genetic fuzzy systems. O. Cordón, F. Herrera, F. Hoffman, L. Magdalena, 2001

- SLAVE: A genetic learning system based on an iterative apporach. A. Gonzalez, R. Perez. IEEE Transactions on Fuzzy Systems, Volume: 7, Issue: 2, 1999