I Workshop on Knowledge Extraction based on Evolutionary Learning

Toward Evolving Consistent, Complete, and Compact Fuzzy Rule Sets: A Genetic Multiobjective Approach

Jorge Casillas
Dept. Computer Science and Artificial Intelligence
University of Granada
http://decsai.ugr.es/~casillas

In collaboration with

Regression: Pedro Martínez and Alicia D. Benítez

Classification: Albert Orriols-Puig and Ester Bernadó-Mansilla

Granada, May 16, 2008

Outline

Introduction

- Predictive induction: it involves generating legible models that describe with the highest reliability the whole data set that represents the analyzed system

- In predictive induction, we look for a rule set rather than individual patterns

- With the aim of generating legible enough models (which, no doubt, is one of the fundamental premises in any knowledge extraction process), we propose to use fuzzy rule-based systems as knowledge representation

- Automatic extraction of rules: greedy algorithms, artificial neural networks, evolutionary computation, etc.

Introduction

- Why we bet genetic algorithms (GAs)?
  - They have a powerful search capacity that allows us to work with multiobjective optimization $\Rightarrow$ interpretability-accuracy balance
  - They can manage flexible representation structures mixing coding schemes or including restrictions $\Rightarrow$ DNF-type fuzzy rules that provide a high degree of compactness and knowledge synthesis

- To do predictive induction (rule sets) with GAs, the Pittsburgh style (where each individual encodes a complete set of rules) seems to be the best approach to properly assess the interaction among the different fuzzy rules

DNF-type fuzzy rule

- DNF-type fuzzy rule:

$$\text{Regression: } \mathbf{IF}\ X_1 \text{ is } \tilde{A}_1 \text{ and } \ldots \text{and } X_n \text{ is } \tilde{A}_n\ \mathbf{THEN}\ Y \text{ is } B$$

$$\text{Classification: } \mathbf{IF}\ X_1 \text{ is } \tilde{A}_1 \text{ and } \ldots \text{and } X_n \text{ is } \tilde{A}_n\ \mathbf{THEN}\ \mathit{Class} = c_j$$

$$\tilde{A}_i = \{A_{i1} \text{ or} \ldots \text{or } A_{il_i}\}$$

- Advantages:
  - Compact knowledge representation
  - Absence of input variables in the rule is allowed
  - Scalability to higher number of variables and/or linguistic terms
- Drawbacks:
  - When a set of DNF-type fuzzy rules is simultaneously learnt (as Pittsburgh-style GA does), several difficulties arise

DNF-type fuzzy rule

- Difficulties of {DNF-type fuzzy rule + Pittsburgh-style GA}:
  - **Consistency**: each combination of antecedent (one label per input variable) should have only one possible class
  - **Completeness**: every training data example should fires at least one fuzzy classification rule
  - **Compactness**: the lowest number of rules to accurately represent input-output relationships should be obtained. Among other issues, it involves to avoid redundant rules
  - **Non-overgeneral rules**: a DNF-type fuzzy rule should be general enough as to represent in a compact way the input space but specific enough as to avoid covering input areas without data
- Although we focus our study in fuzzy rules, these concepts may be extrapolated to interval rules when generality is considered

Consistency in DNF-type fuzzy rules

IF X_1 is $\{A_2$ or $A_3\}$ and X_2 is $\{B_1$ or B_2 or $B_3\}$ **THEN** Y is C_1

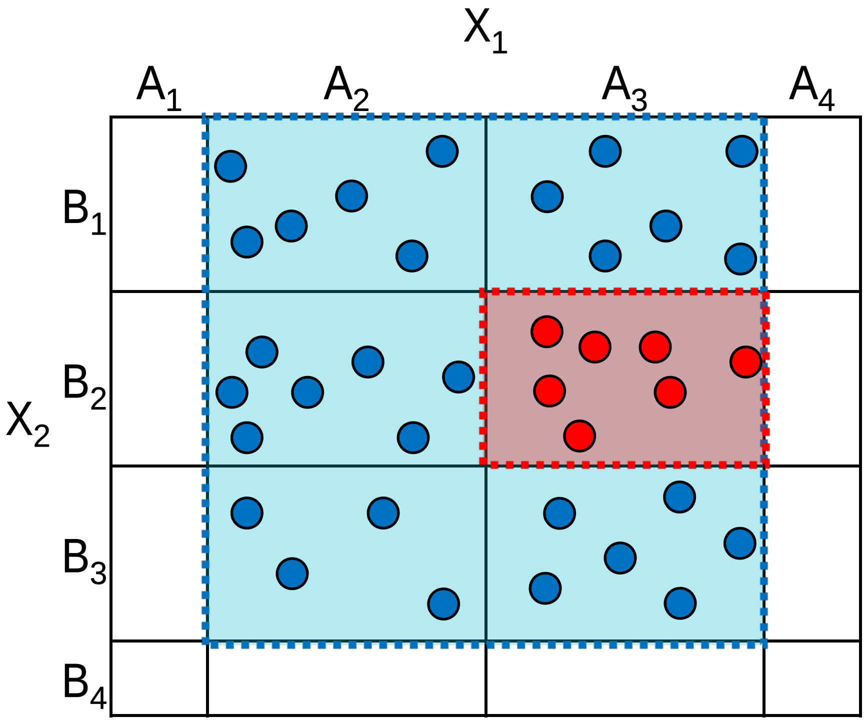
IF X_1 is A_3 and X_2 is B_2 **THEN** Y is C_2

IF X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_1 **THEN** Y is C_1

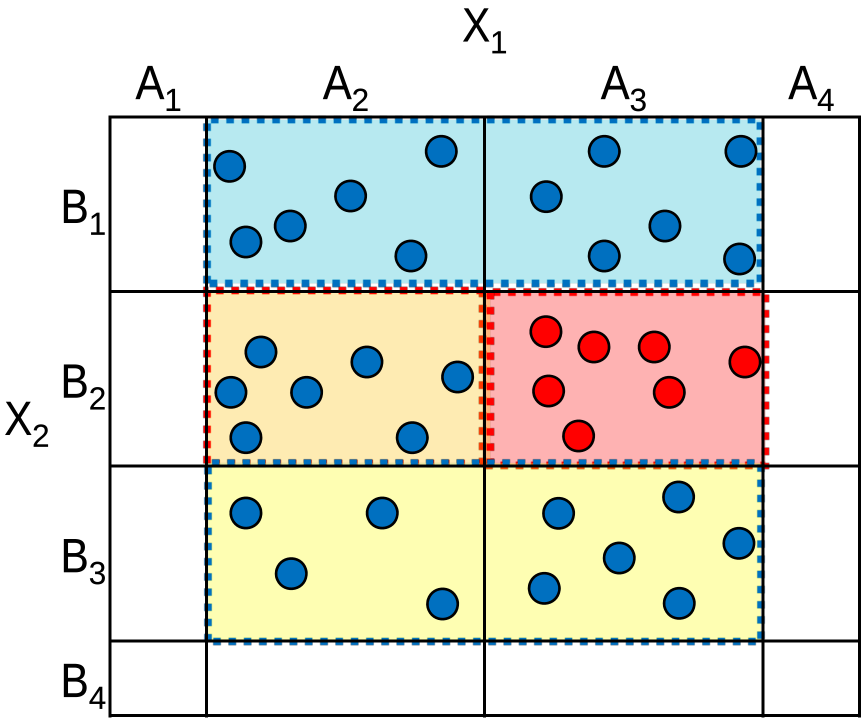
IF X_1 is A_2 and X_2 is B_2 **THEN** Y is C_1

IF X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_3 **THEN** Y is C_1

IF X_1 is A_3 and X_2 is B_2 **THEN** Y is C_2



(a) Inconsistent solution (2 rules)



(b) Consistent solution (4 rules)

Redundancy in DNF-type fuzzy rules

- Redundancy:
  - IF X_1 is A_1 and X_2 is $\{B_1$ or $B_2\}$ THEN Y is C_1
    IF X_1 is A_1 and X_2 is B_1 THEN Y is C_1

    If a rule subsumes another one, just remove the more specific one

  - IF X_1 is A_1 and X_2 is $\{B_1$ or $B_2\}$ THEN Y is C_1
    IF X_1 is $\{A_1$ or $A_2\}$ and X_2 is B_1 THEN Y is C_1

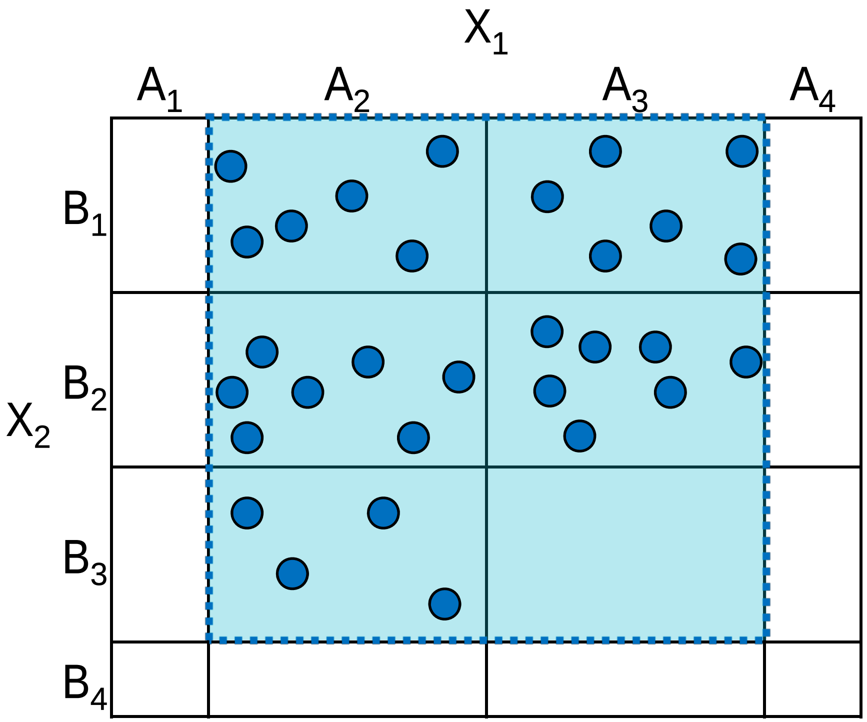
    This case is more difficult to be solved, but its effect is not very disruptive for the final fuzzy rule set

Over-generality in DNF-type fuzzy rules

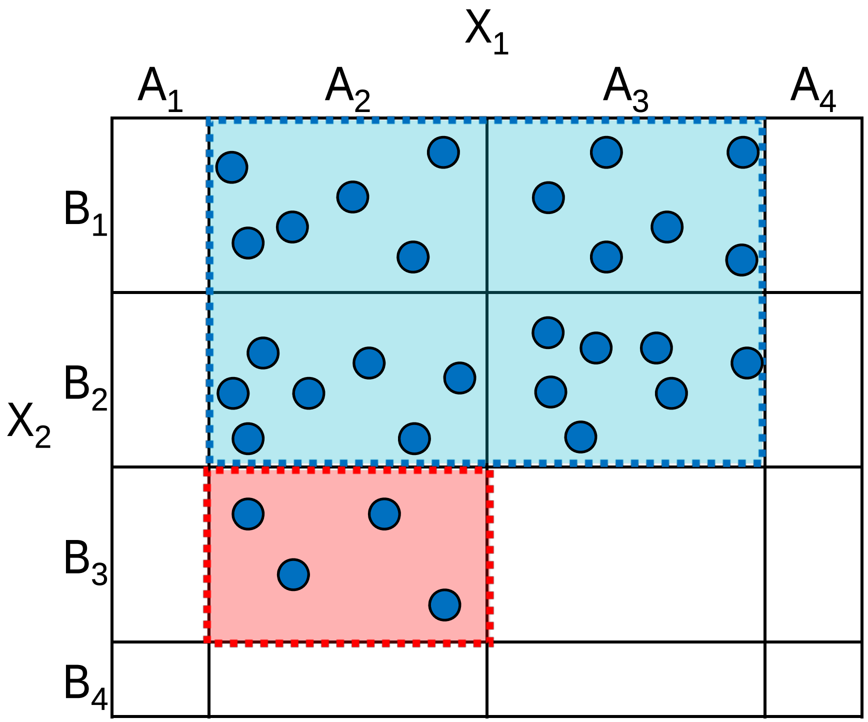
IF X_1 is $\{A_2$ or $A_3\}$ and X_2 is $\{B_1$ or B_2 or $B_3\}$ **THEN** Y is C_1

IF X_1 is $\{A_2$ or $A_3\}$ and X_2 is $\{B_1$ or $B_2\}$ **THEN** Y is C_1

IF X_1 is A_2 and X_2 is B_3 **THEN** Y is C_1



(a) Over-general solution (1 rule)



(b) Solution with optimal generality (2 rules)

How to Address These Difficulties?

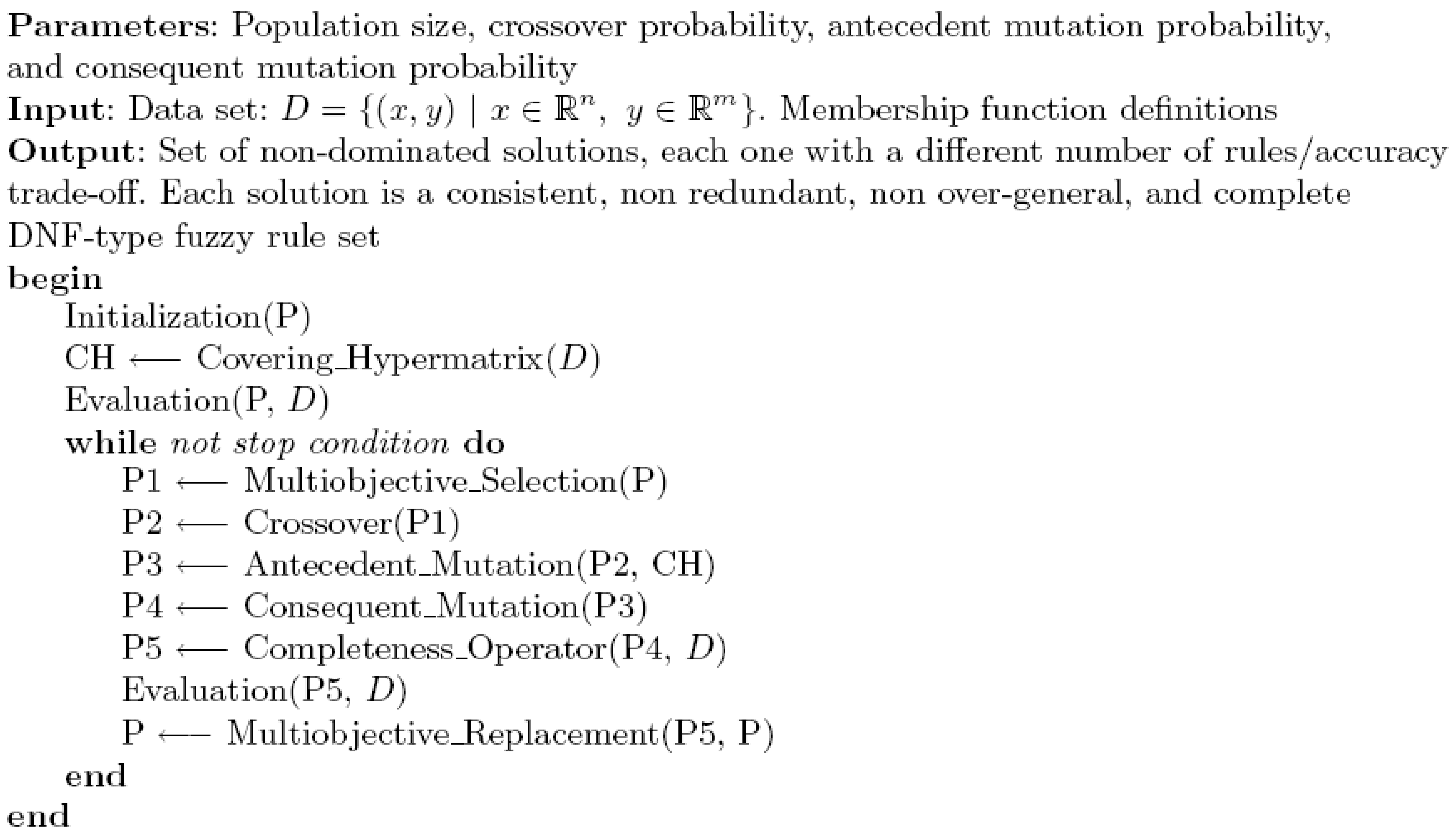
- Solutions to ensure consistency, completeness, compactness, and non over-generality:
  - Ignore bad solutions
  - Penalize fitness of bad solutions [Wang et al., 2005]
  - *A posteriori* repair bad solutions [Wang et al., 2005]
  - Constrain the search space to avoid generating bad solutions

[Wang et al., 2005] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F Man. Agent-based evolutionary approach for interpretable rule-based knowledge extraction. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 35(2):143-155, 2005

- We opt for the latter option by designing genetic operators that only explore correct solutions

J. Casillas, P. Martínez, A.D. Benítez, Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems, Soft Computing (2008). In press

Pitts-DNF Scheme



Parameters: Population size, crossover probability, antecedent mutation probability, and consequent mutation probability
Input: Data set: $D = \{(x, y) \mid x \in \mathbb{R}^n,\ y \in \mathbb{R}^m\}$. Membership function definitions
Output: Set of non-dominated solutions, each one with a different number of rules/accuracy trade-off. Each solution is a consistent, non redundant, non over-general, and complete DNF-type fuzzy rule set

```
begin
    Initialization(P)
    CH $\longleftarrow$ Covering_Hypermatrix($D$)
    Evaluation(P, $D$)
    while not stop condition do
        P1 $\longleftarrow$ Multiobjective_Selection(P)
        P2 $\longleftarrow$ Crossover(P1)
        P3 $\longleftarrow$ Antecedent_Mutation(P2, CH)
        P4 $\longleftarrow$ Consequent_Mutation(P3)
        P5 $\longleftarrow$ Completeness_Operator(P4, $D$)
        Evaluation(P5, $D$)
        P $\longleftarrow$ Multiobjective_Replacement(P5, P)
    end
end
```

Coding Scheme

- Binary coding scheme in the antecedent (allele ‘1’ stands for the corresponding label is used) and integer coding in the consequent (the allele indicates the index of the corresponding class)
- Example: Let us assume we have two input variables with three linguistic terms (Small [S], Medium [M], Large [L]) each of them

$$\begin{tikzcd}
\boxed{\text{IF } X_1 \text{ is S and } X_2 \text{ is } \{\text{M or L}\} \text{ THEN } (\text{Y is } C_2 \text{ , Class}=c_2)} \arrow[d] \\
\boxed{[100\,|\,011\,|\,2]}
\end{tikzcd}$$

- The chromosome will be a set of such a rules, i.e., variable-length concatenation of their strings

$$\boxed{[100\,|\,011\,|\,2][010\,|\,011\,|\,1][001\,|\,010\,|\,3][011\,|\,100\,|\,2]\ldots}$$

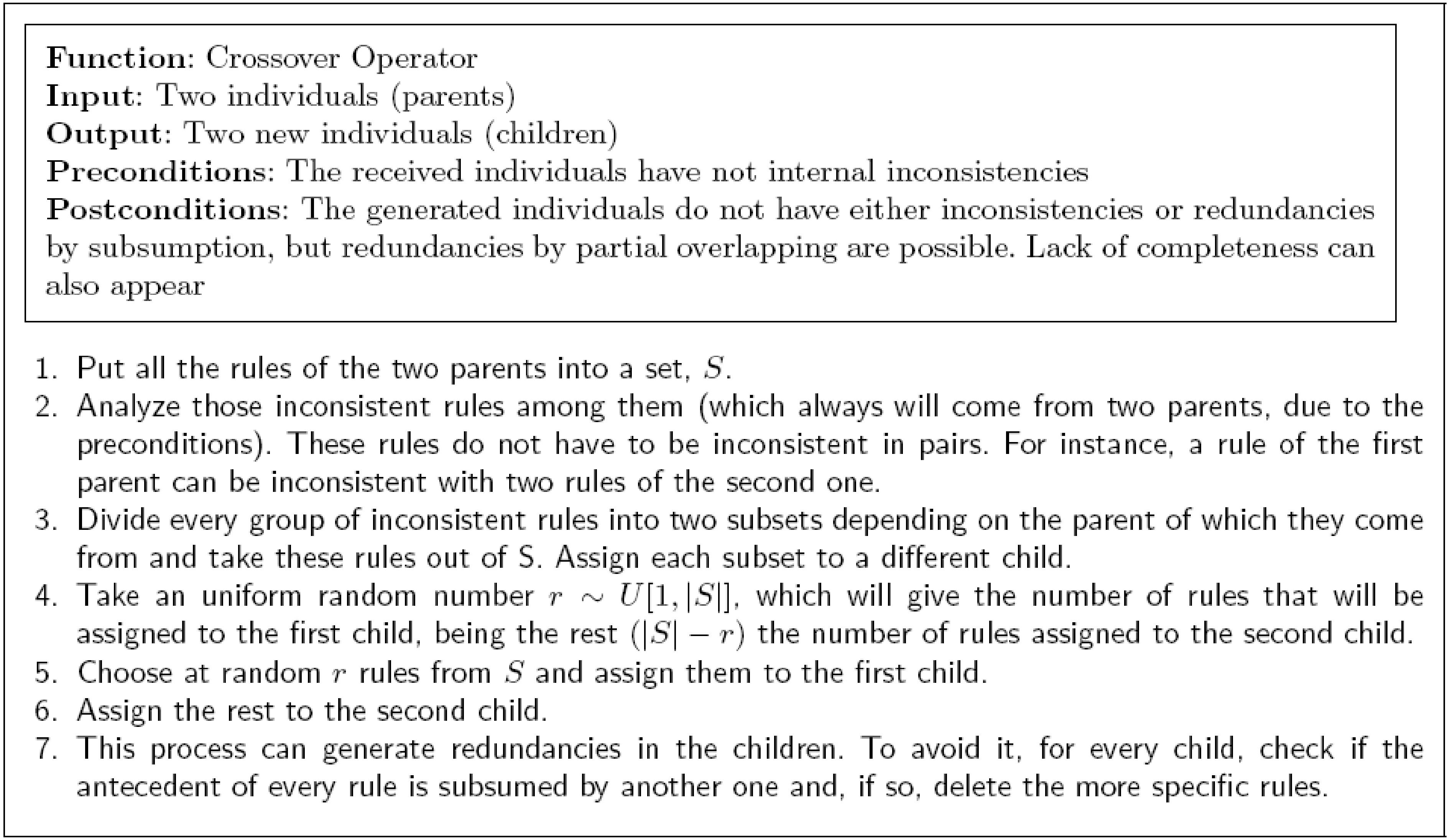
Initialization

- Since we are looking for optimal completeness, we need starting with rules which cover all (and only) the examples
- Because of that, we generate the antecedent structure by the Wang-Mendel algorithm
- Specifically, every chromosome is generated with the minimum number of rules that cover the examples according to this algorithm and with a random class for every rule
- Except one chromosome that uses the class of the most highly covered example for each rule (greedy solution)
- All chromosomes start with the same number of rules, being those so specific as possible (i.e., with a simple conjunction structure instead the conjunctive normal form)

Covering Hypermatrix Computation

- The objective of this step is to generate a data structure which will be used when generating new rules to efficiently avoid over-generality
- The structure stores the label combinations of the antecedent that cover everyone of the examples in the data set
- Notice that the hypermatrix represents the highest allowed input covering, but it does not show whether a lower number of rules would completely cover the training data set or not, so it can not be used to ensure completeness
- A hash table is used for efficient key searching

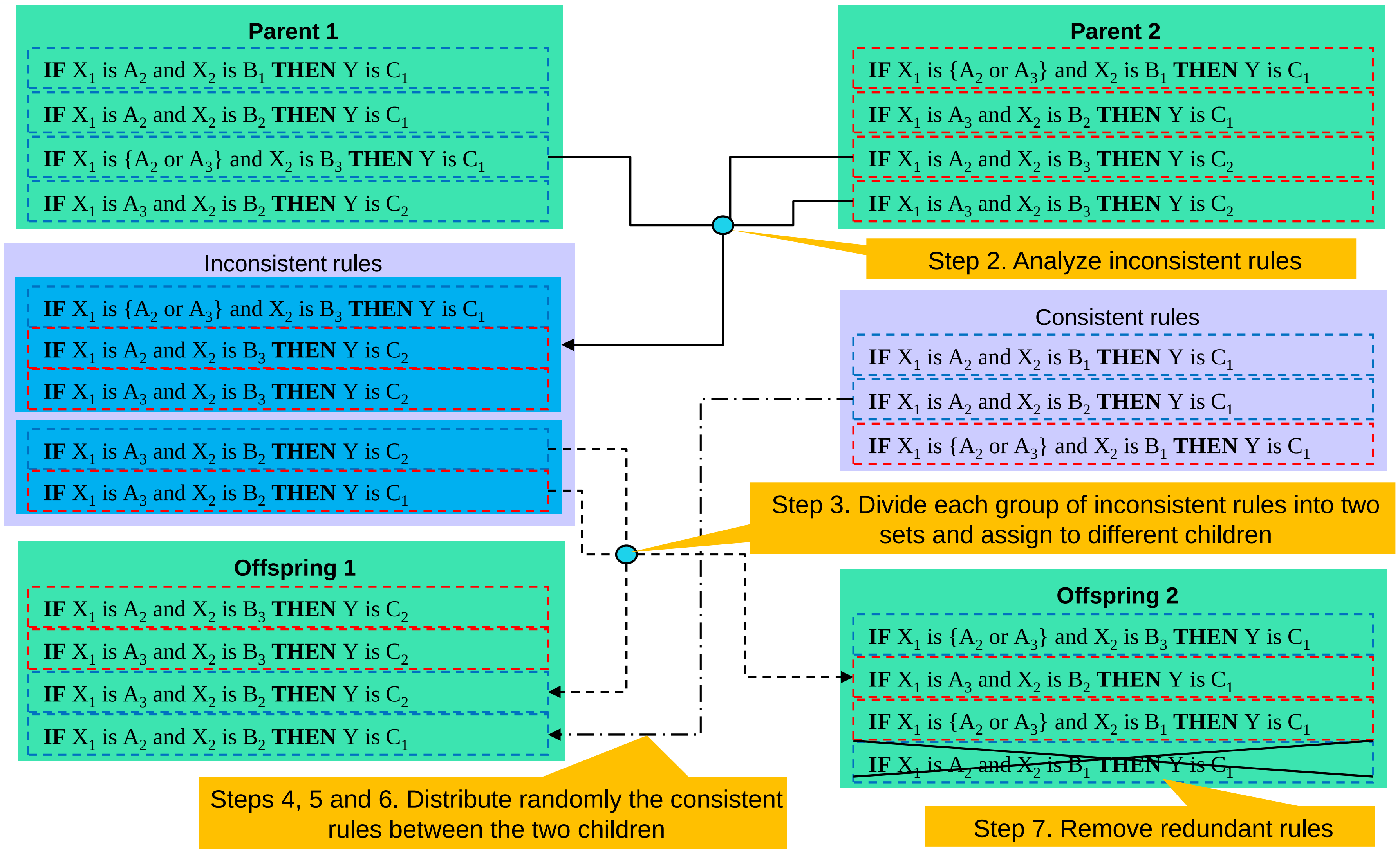
Crossover Operator



Function: Crossover Operator
Input: Two individuals (parents)
Output: Two new individuals (children)
Preconditions: The received individuals have not internal inconsistencies
Postconditions: The generated individuals do not have either inconsistencies or redundancies by subsumption, but redundancies by partial overlapping are possible. Lack of completeness can also appear

1. Put all the rules of the two parents into a set, S.
2. Analyze those inconsistent rules among them (which always will come from two parents, due to the preconditions). These rules do not have to be inconsistent in pairs. For instance, a rule of the first parent can be inconsistent with two rules of the second one.
3. Divide every group of inconsistent rules into two subsets depending on the parent of which they come from and take these rules out of S. Assign each subset to a different child.
4. Take an uniform random number $r \sim U[1, |S|]$, which will give the number of rules that will be assigned to the first child, being the rest $(|S| - r)$ the number of rules assigned to the second child.
5. Choose at random r rules from S and assign them to the first child.
6. Assign the rest to the second child.
7. This process can generate redundancies in the children. To avoid it, for every child, check if the antecedent of every rule is subsumed by another one and, if so, delete the more specific rules.

Crossover Operator



Parent 1

- **IF** X_1 is A_2 and X_2 is B_1 **THEN** Y is C_1
- **IF** X_1 is A_2 and X_2 is B_2 **THEN** Y is C_1
- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_3 **THEN** Y is C_1
- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_2

Parent 2

- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_1 **THEN** Y is C_1
- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_1
- **IF** X_1 is A_2 and X_2 is B_3 **THEN** Y is C_2
- **IF** X_1 is A_3 and X_2 is B_3 **THEN** Y is C_2

Step 2. Analyze inconsistent rules

Inconsistent rules

- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_3 **THEN** Y is C_1
- **IF** X_1 is A_2 and X_2 is B_3 **THEN** Y is C_2
- **IF** X_1 is A_3 and X_2 is B_3 **THEN** Y is C_2

- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_2
- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_1

Consistent rules

- **IF** X_1 is A_2 and X_2 is B_1 **THEN** Y is C_1
- **IF** X_1 is A_2 and X_2 is B_2 **THEN** Y is C_1
- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_1 **THEN** Y is C_1

Step 3. Divide each group of inconsistent rules into two sets and assign to different children

Offspring 1

- **IF** X_1 is A_2 and X_2 is B_3 **THEN** Y is C_2
- **IF** X_1 is A_3 and X_2 is B_3 **THEN** Y is C_2
- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_2
- **IF** X_1 is A_2 and X_2 is B_2 **THEN** Y is C_1

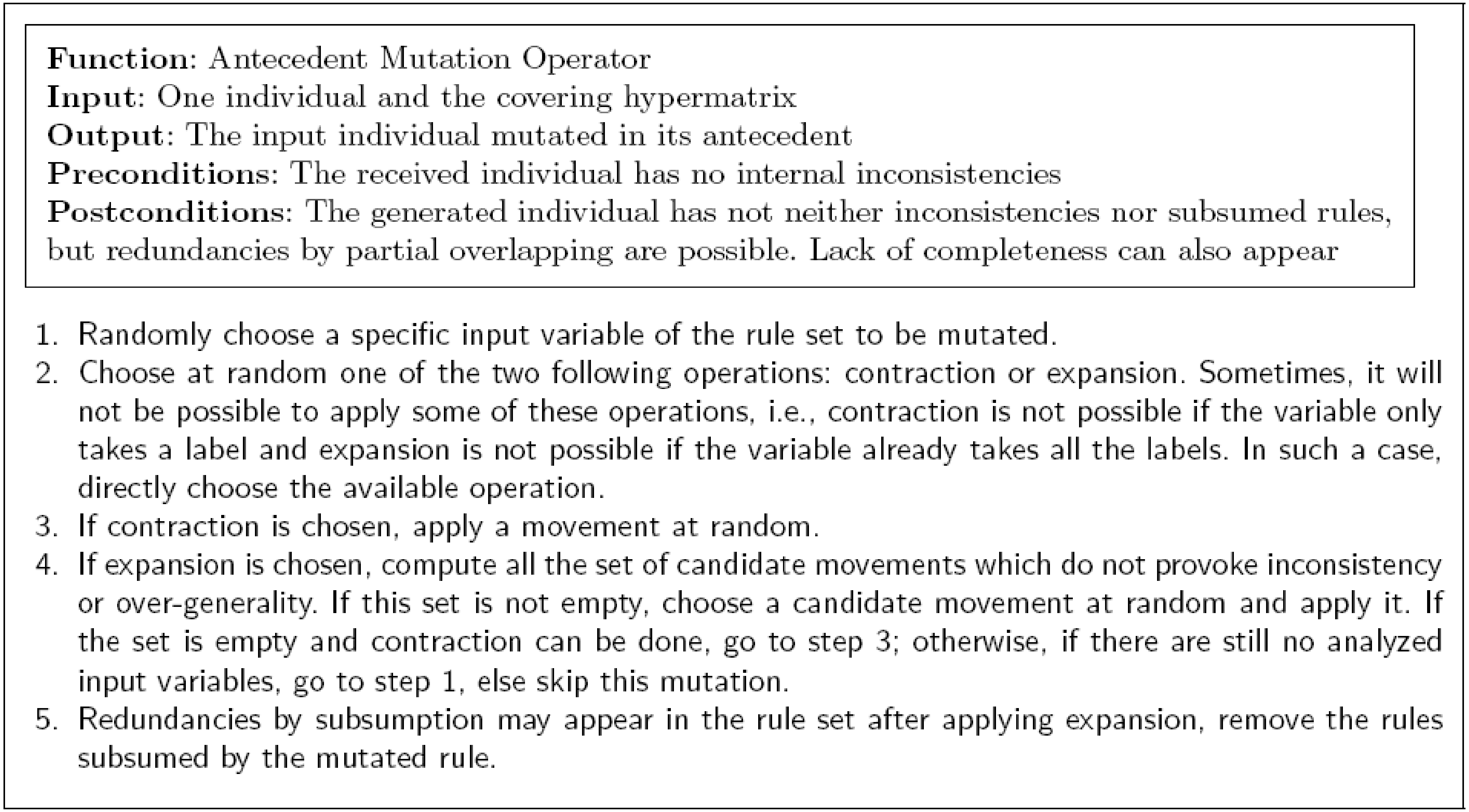
Offspring 2

- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_3 **THEN** Y is C_1
- **IF** X_1 is A_3 and X_2 is B_2 **THEN** Y is C_1
- **IF** X_1 is $\{A_2$ or $A_3\}$ and X_2 is B_1 **THEN** Y is C_1
- ~~**IF** X_1 is A_2 and X_2 is B_1 **THEN** Y is C_1~~

Steps 4, 5 and 6. Distribute randomly the consistent rules between the two children

Step 7. Remove redundant rules

Antecedent Mutation Operator



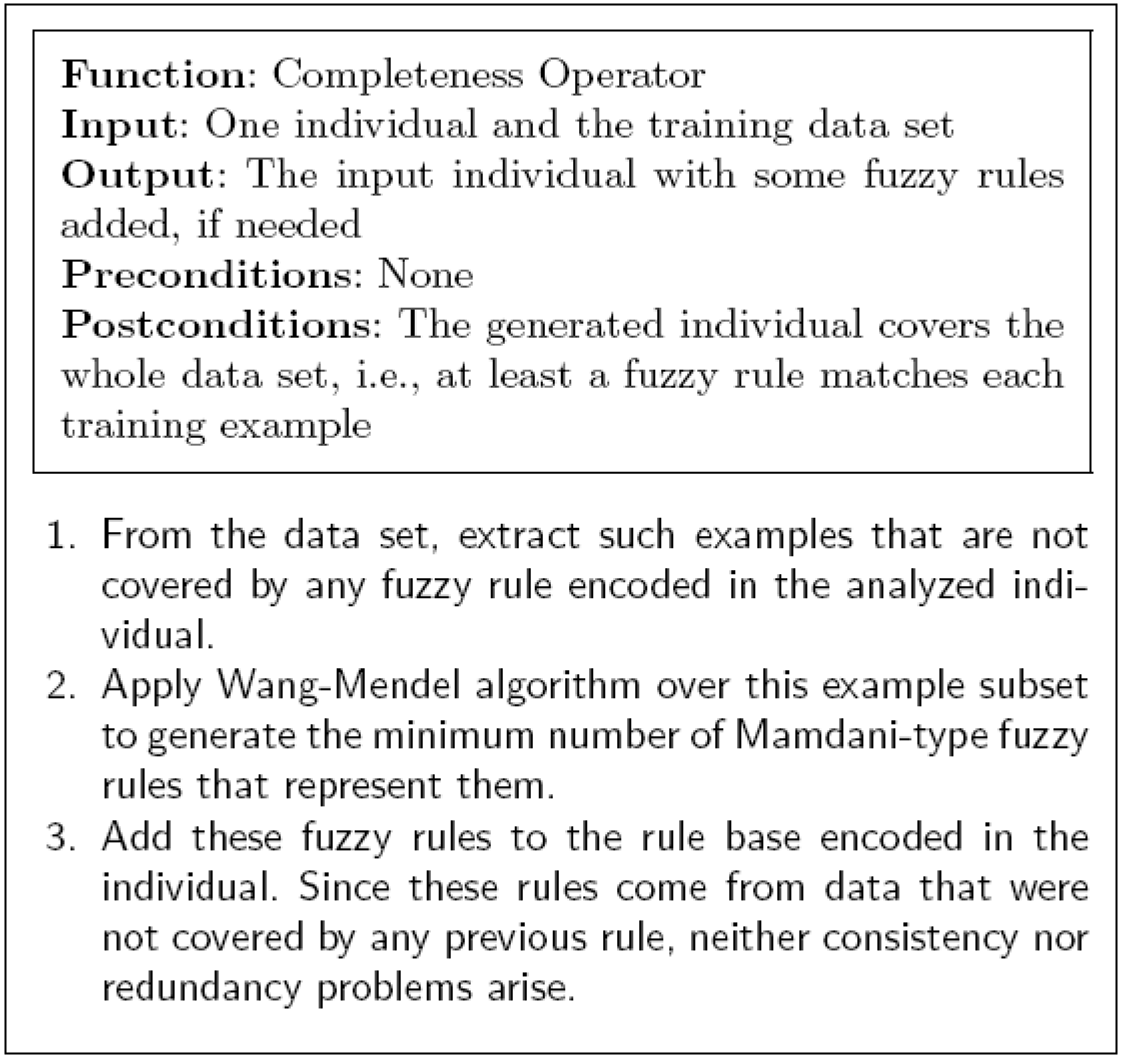
Function: Antecedent Mutation Operator
Input: One individual and the covering hypermatrix
Output: The input individual mutated in its antecedent
Preconditions: The received individual has no internal inconsistencies
Postconditions: The generated individual has not neither inconsistencies nor subsumed rules, but redundancies by partial overlapping are possible. Lack of completeness can also appear

1. Randomly choose a specific input variable of the rule set to be mutated.
2. Choose at random one of the two following operations: contraction or expansion. Sometimes, it will not be possible to apply some of these operations, i.e., contraction is not possible if the variable only takes a label and expansion is not possible if the variable already takes all the labels. In such a case, directly choose the available operation.
3. If contraction is chosen, apply a movement at random.
4. If expansion is chosen, compute all the set of candidate movements which do not provoke inconsistency or over-generality. If this set is not empty, choose a candidate movement at random and apply it. If the set is empty and contraction can be done, go to step 3; otherwise, if there are still no analyzed input variables, go to step 1, else skip this mutation.
5. Redundancies by subsumption may appear in the rule set after applying expansion, remove the rules subsumed by the mutated rule.

Consequent Mutation Operator

- It creates new rules by changing the class
- It consists on randomly selecting a rule that is not partially overlapped with other rules (it would be the only problematic case since the consequent mutation operator receives consistent and non-subsumed rules)
- Then, the class is randomly changed
- The operation does not cause over-generality or lack of completeness

Completeness Operator



Function: Completeness Operator
Input: One individual and the training data set
Output: The input individual with some fuzzy rules added, if needed
Preconditions: None
Postconditions: The generated individual covers the whole data set, i.e., at least a fuzzy rule matches each training example

1. From the data set, extract such examples that are not covered by any fuzzy rule encoded in the analyzed individual.
2. Apply Wang-Mendel algorithm over this example subset to generate the minimum number of Mamdani-type fuzzy rules that represent them.
3. Add these fuzzy rules to the rule base encoded in the individual. Since these rules come from data that were not covered by any previous rule, neither consistency nor redundancy problems arise.

Multiobjective Approach

- Multiobjective optimization is performed with the elitist replacement strategy of NSGA-II
- Crowding distance in the objective function space is considered
- Binary tournament selection based on the nondomination rank (or the crowding distance when both solutions belong to the same front) is applied
- The crowding distance is normalized for each objective according to the extreme values of the solutions contained in the analyzed front

Objective Functions

- Performance (for maximization): In regression, mean squared error; in classification, proportion of instances correctly classified by the rule set

$$\boxed{F_1(S) = \frac{1}{N}\sum_{i=1}^{N}(S(x^i) - y^i)^2}$$

$$\boxed{F_1(S) = \frac{\#correct\ classifications}{\#examples}}$$

- Complexity (for minimization): Number of DNF-type fuzzy rules

$$\boxed{F_2(S) = |S|}$$

  - The algorithm is designed to ensure optimal covering, so we do not care on the linguistic complexity of each single fuzzy rule. In a natural way, the most general the fuzzy rules, the fewer the number of rules
  - It simplifies the design of an interpretability-based objective function. Other authors consider one or more additional objectives to evaluate the complexity of each fuzzy rule (rule length, number of premises, etc.)

Experimental Setup for Regression

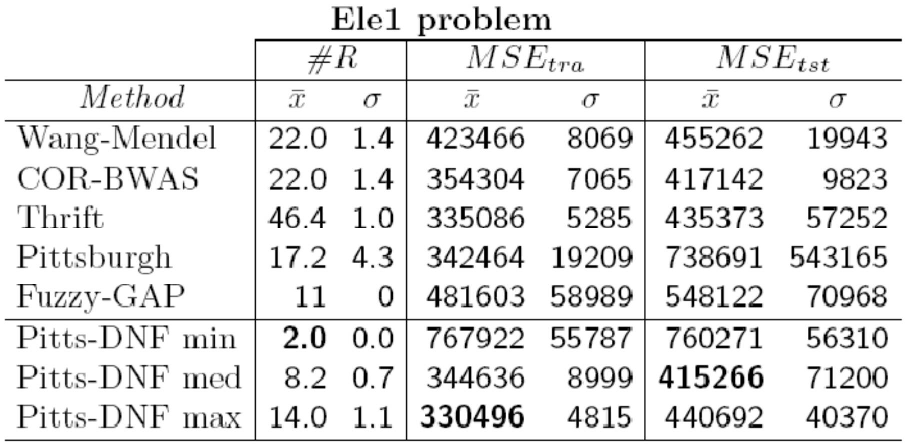
- Regression problems (5-fold cross validation)

Problem	*#InputVar*	*#Exam*	*#LingTerms*
Diabetes	2	43	7
Ele1	2	495	7
Laser	4	993	5
Ele2	4	1066	5
DEE	6	365	5

- Learning methods to compare

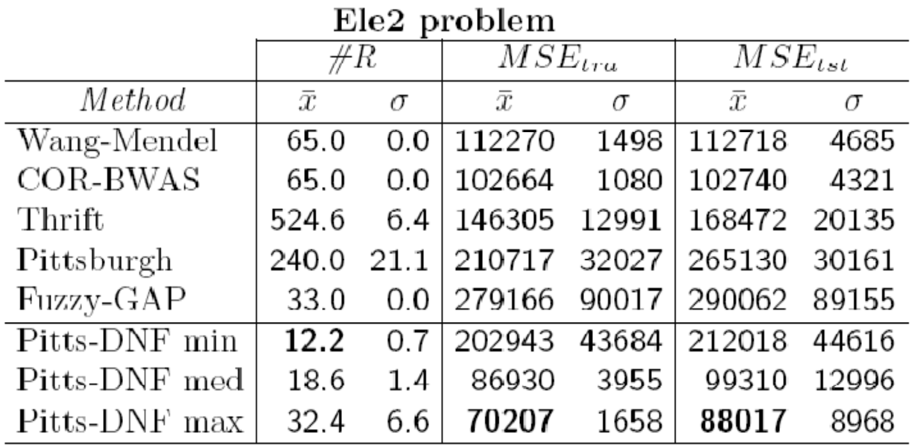
	Algorithm type	Representation	No. rules
Wang-Mendel	Greedy	Mamdani	Automatic
COR-BWAS	Ant colony	Mamdani	Automatic
Thrift	Pittsburgh	Mamdani	Automatic
Pittsburgh	Pittsburgh	DNF	Automatic
Fuzzy-GAP	GA-P	Conjunctions and disjunctions	Fixed

Experimental Results in Regression



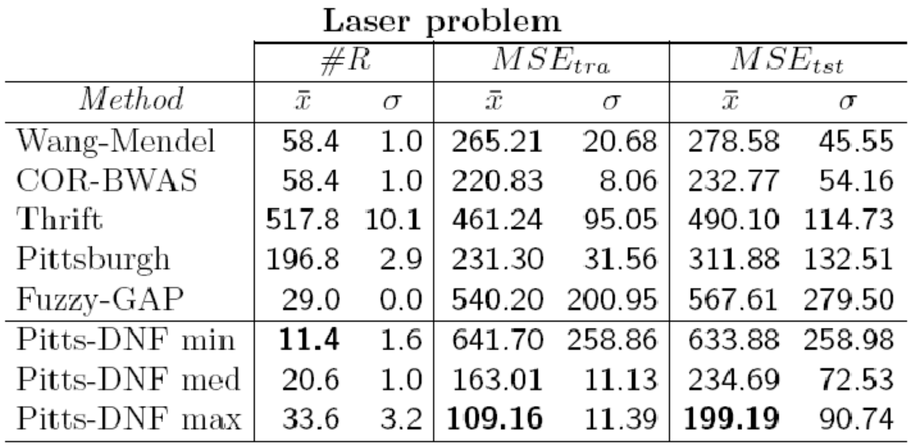
Ele1 problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	22.0	1.4	423466	8069	455262	19943
COR-BWAS	22.0	1.4	354304	7065	417142	9823
Thrift	46.4	1.0	335086	5285	435373	57252
Pittsburgh	17.2	4.3	342464	19209	738691	543165
Fuzzy-GAP	11	0	481603	58989	548122	70968
Pitts-DNF min	**2.0**	0.0	767922	55787	760271	56310
Pitts-DNF med	8.2	0.7	344636	8999	**415266**	71200
Pitts-DNF max	14.0	1.1	**330496**	4815	440692	40370



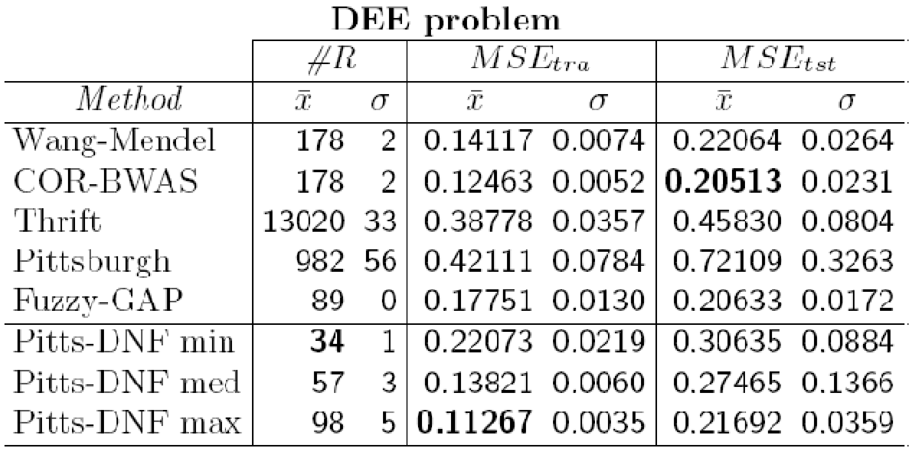
Ele2 problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	65.0	0.0	112270	1498	112718	4685
COR-BWAS	65.0	0.0	102664	1080	102740	4321
Thrift	524.6	6.4	146305	12991	168472	20135
Pittsburgh	240.0	21.1	210717	32027	265130	30161
Fuzzy-GAP	33.0	0.0	279166	90017	290062	89155
Pitts-DNF min	**12.2**	0.7	202943	43684	212018	44616
Pitts-DNF med	18.6	1.4	86930	3955	99310	12996
Pitts-DNF max	32.4	6.6	**70207**	1658	**88017**	8968



Laser problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	58.4	1.0	265.21	20.68	278.58	45.55
COR-BWAS	58.4	1.0	220.83	8.06	232.77	54.16
Thrift	517.8	10.1	461.24	95.05	490.10	114.73
Pittsburgh	196.8	2.9	231.30	31.56	311.88	132.51
Fuzzy-GAP	29.0	0.0	540.20	200.95	567.61	279.50
Pitts-DNF min	**11.4**	1.6	641.70	258.86	633.88	258.98
Pitts-DNF med	20.6	1.0	163.01	11.13	234.69	72.53
Pitts-DNF max	33.6	3.2	**109.16**	11.39	**199.19**	90.74



DEE problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	178	2	0.14117	0.0074	0.22064	0.0264
COR-BWAS	178	2	0.12463	0.0052	**0.20513**	0.0231
Thrift	13020	33	0.38778	0.0357	0.45830	0.0804
Pittsburgh	982	56	0.42111	0.0784	0.72109	0.3263
Fuzzy-GAP	89	0	0.17751	0.0130	0.20633	0.0172
Pitts-DNF min	**34**	1	0.22073	0.0219	0.30635	0.0884
Pitts-DNF med	57	3	0.13821	0.0060	0.27465	0.1366
Pitts-DNF max	98	5	**0.11267**	0.0035	0.21692	0.0359

Experimental Results in Regression

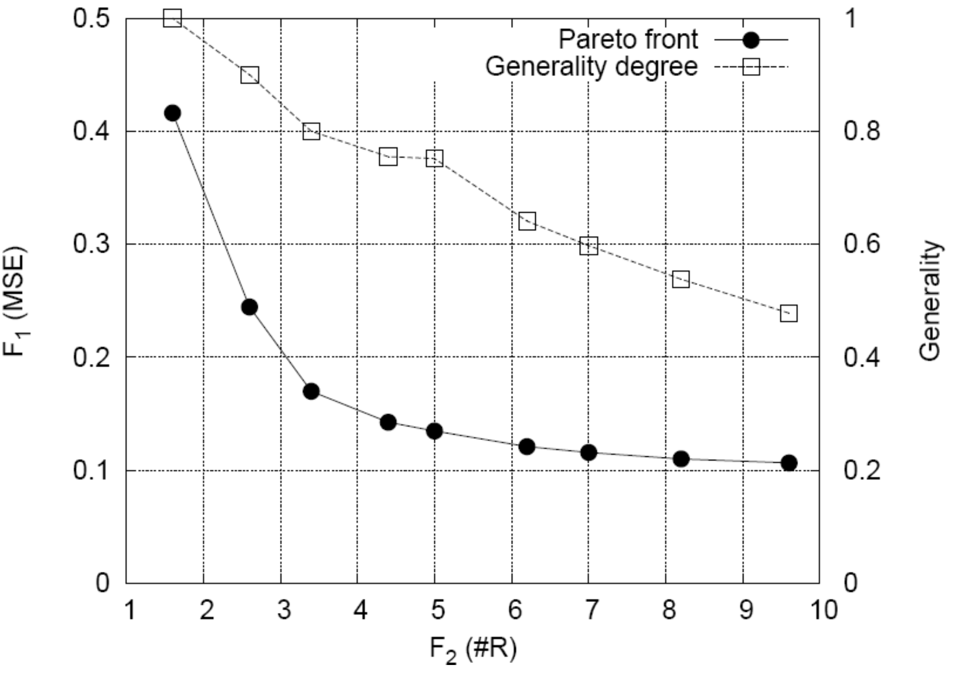
- COR-BWAS ($MSE_{tra/tst}$ = 356137 / 370626, #R=22)

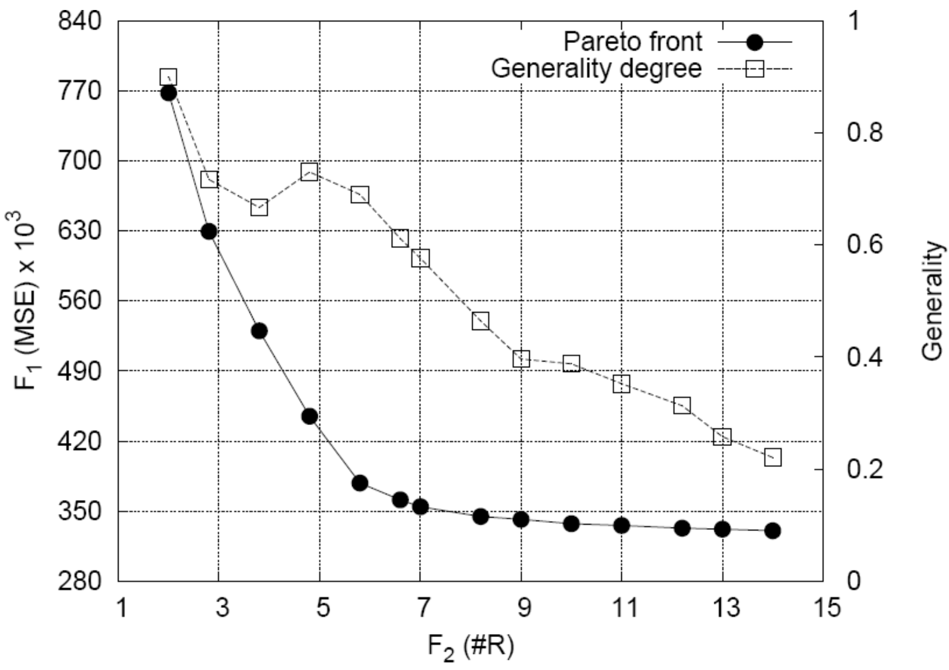
		X₁						
		XS	*VS*	*S*	*M*	*L*	*VL*	*XL*
	XS	XS	VS					
	VS	XS	VS	VS	VS			
	S	VS	S	S	S	M		
X₂	*M*	VS	M	S	VL			
	L		M	S	XL			
	VL	VS		L	M			
	XL	M						

- Pitts-DNF ($MSE_{tra/tst}$ = 348914 / 390556, #R=7, $\#R_{mam}$=16)

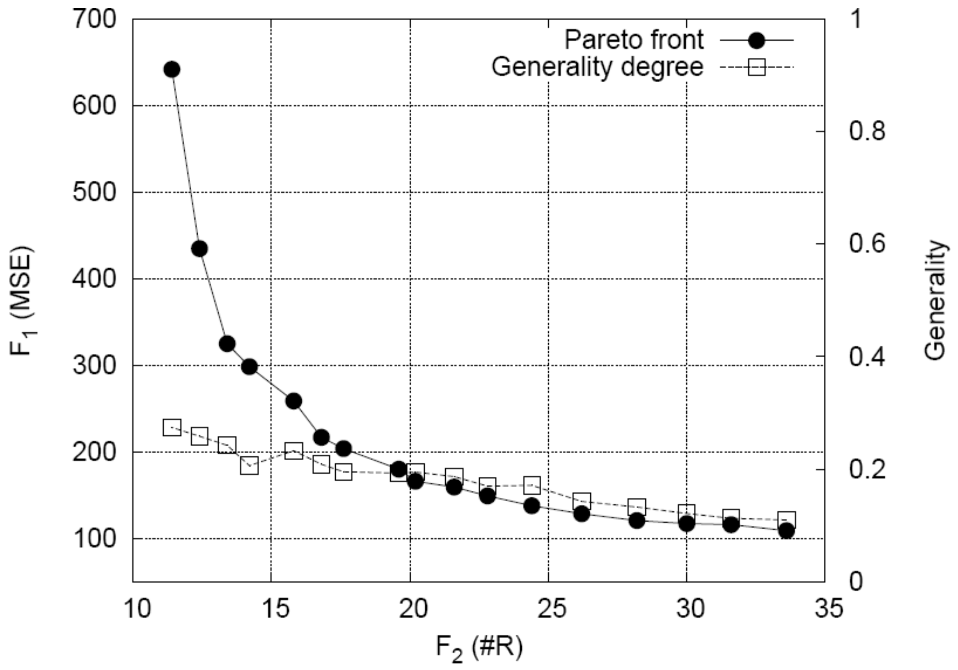
		X₁						
		XS	*VS*	*S*	*M*	*L*	*VL*	*XL*
	XS	XS_1			VL_7			
	VS	XS_1	VS_3	VS_3	VS_3			
	S	VS_2	S_5	S_5	S_5	S_5		
X₂	*M*	VS_2	M_6					
	L				VL_7			
	VL		M_6		VS_4			
	XL							

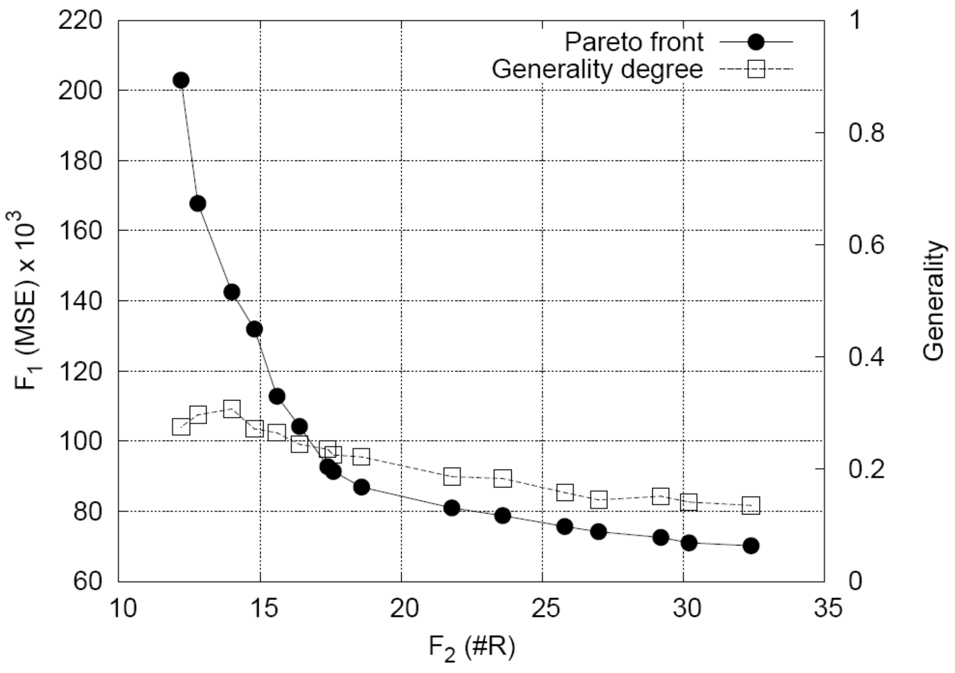
Pareto Fronts





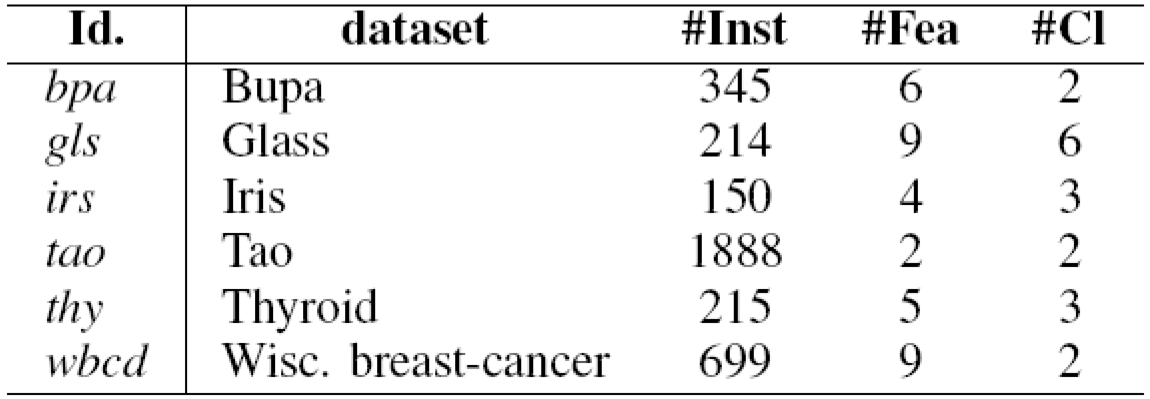
Pareto Fronts





Experimental Setup for Classification

- Classification problems (10-fold cross validation)



Id.	**dataset**	**#Inst**	**#Fea**	**#Cl**
bpa	Bupa	345	6	2
gls	Glass	214	9	6
irs	Iris	150	4	3
tao	Tao	1888	2	2
thy	Thyroid	215	5	3
wbcd	Wisc. breast-cancer	699	9	2

- Learning methods to compare

	Algorithm type	Rule representation
Wang-Mendel	Greedy	Conjunction
Fuzzy GP	Genetic programming	Antecedent in disjunctive normal form
Fuzzy MaxLogitBoost	Boosting + GA	Weighted rules with antecedent in conjunctive normal form

Experimental Results in Classification

- Performance

	Pitts-DNF-C			Wang-Mendel	Fuzzy MLB	Fuzzy GP
	Best acc.	**Median**	**Best size**			
bpa	53.91%	53.35%	46.67%	54.16%	56.53%	56.62%
gls	61.98%	58.79%	44.44%	55.35%	62.18%	48.89%
irs	92.67%	92.00%	88.00%	87.33%	92.00%	94.47%
tao	89.56%	89.56%	87.61%	76.80%	84.52%	80.36%
thy	91.64%	92.10%	84.17%	89.74%	95.33%	86.98%
wbcd	96.24%	94.81%	66.00%	96.00%	91.83%	93.31%

Pitts-DNF-C outperforms Wang-Mendel (statistically significant)

Pitts-DNF-C does not perform statistically differently than Fuzzy MLB and Fuzzy GP

- Complexity

	Pitts-DNF-C			Wang-Mendel	Fuzzy MLB	Fuzzy GP		
	Best acc.	**Median**	**Best size**			**and**	**or**	**is**
bpa	37.7	25.4	19.9	120.9	13.3	17.6	37.3	54.9
gls	42.6	34.1	27.8	88.6	22.8	28.8	32.9	61.7
irs	8.7	8.3	5.4	44.6	4	18.7	21.6	40.4
tao	5.4	5.4	4.7	21.0	18	19.0	20.3	39.2
thy	12.2	10.2	7.2	48.0	8.8	18.2	20.0	38.1
wbcd	295.5	264.25	231.5	301.8	13.1	20.4	40.1	60.5

Conclusion: Self-analysis

- SWOT analysis:
  - **S**trengths: main advantages of Pitts-DNF
  - **W**eaknesses: drawbacks of Pitts-DNF
  - **O**pportunities: further works on Pitts-DNF
  - **T**hreats: optional approaches considered by other methods than can compete with Pitts-DNF

	Positive	Negative
Internal	Strengths	Weaknesses
External	Opportunities	Threats

Conclusion: Self-analysis

Strengths	Weaknesses
• Its performance, both in interpretability and accuracy, is competitive compared with other approaches • It uses a flexible fuzzy rule structure for a better knowledge synthesis • It generates consistent, complete, compact, and non over-general fuzzy rule sets • It performs multiobjective optimization to return solutions with different interpretability-accuracy trade-offs	• It only works in data-driven problems • The considered properties (consistency, completeness and optimal generality) may involve to generate a higher number of rules in some problems • Although it is better prepared for dealing with high dimensional problems than other approaches, it still needs to be improved to properly generalize the fuzzy rules

Conclusion: Self-analysis

Opportunities	Threats
• To adapt the algorithm to learn Takagi-Sugeno fuzzy rules • To combine Pitts-DNF with a membership function parameter learning/tuning process • To study more complex solutions for avoiding over-generality without leaving uncovered regions • To analyze other fuzzy rule structures even more flexible than DNF-type for a more compact knowledge representation	• Solutions based on guiding the search process by objective functions that penalize the lack of some of the analyzed properties may obtain good solutions in the practice due to the search flexibility • Consistency may be obtained by applying a two-stage sequential approach: generation of Mamdani-type fuzzy rules plus an *a posteriori* rule grouping process