

Toward Evolving Consistent, Complete, and Compact Fuzzy Rule Sets for Classification Problems

Jorge Casillas, Albert Orriols-Puig, and Ester Bernadó-Mansilla

Abstract—This paper proposes Pitts-DNF-C, a multi-objective Pittsburgh-style Learning Classifier System that evolves a set of DNF-type fuzzy rules for classification tasks. The system is explicitly designed to only explore solutions that lead to consistent, complete, and compact rule sets without redundancies and inconsistencies. The behavior of the system is analyzed on a collection of real-world data sets, showing its competitiveness in terms of performance and interpretability with respect to three other fuzzy learners.

I. INTRODUCTION

Genetic fuzzy systems [5] are learning techniques that use genetic algorithms to optimize different components of fuzzy rule-based systems. These types of learning methodologies aim at obtaining highly accurate and understandable models. These two objectives are usually contradictory since more accurate models tend to be less interpretable. A lot of research has been conducted on this topic, searching for a good trade-off between the interpretability and the accuracy of fuzzy models [2], [3].

In this paper, we address this issue and propose Pitts-DNF-C, a multi-objective Pittsburgh-style fuzzy rule-based classification system that evolves a set of fuzzy rules whose antecedent is in *conjunctive normal form*. That is to say, the antecedent is the conjunction of a set of propositions, and each of these propositions use a disjunction to connect a set of linguistic terms. These types of generalized rules are usually addressed as DNF-type fuzzy rules [8]. Moreover, the system is explicitly designed to create *consistent* (where each input subspace has only one possible class), *complete* (where every training example fires at least one fuzzy classification rule), and *compact* (without redundant rules) fuzzy rule sets without *over-general rules* (thus avoiding covering input areas without data). For this purpose, new genetic operators are designed to guarantee that all the individuals in the population satisfy these four conditions.

Although these conditions are relatively easy to satisfy with simple fuzzy rules (see for example [11], where measures of incompleteness and inconsistency are used as penalty in the rule's fitness), it becomes more complex in the case of DNF-type fuzzy rules. Most of methods that deal with some kind of generalization of the antecedent of the fuzzy

rule (e.g. DNF-type rules or rules with “don't care”) do not address properly the problem [8], [9], [10], [12], [13]. Indeed, some of these proposals use penalty fitness to correct these deficiencies, others infer a default output when no rules are fired, others tend to generate a high number of rules, some others simply do not prevent the system from generating inconsistencies or redundancies...

There are few proposals that explicitly try to evolve consistent, complete, and compact fuzzy rule sets with generalization of the antecedent. For example, Wang *et al.* [14] use the same functions defined in [11] to detect conflicts with an agent-based evolutionary approach in which the agents were multi-objective Pittsburgh-style genetic fuzzy systems. However, they use simple crossover and mutation operators and an *a posteriori* reparation to solve inconsistencies and redundancies.

We take a completely different approach from the above methods and propose an algorithm that intrinsically explores feasible solutions, thus avoiding the use of penalties or reparations. We have already designed an approach that deals with these properties for regression problems [4]. In this work, we address the classification case by particularizing the design of the algorithm to the special features of the field.

The remainder of this paper is organized as follows. Section II briefly discusses the difficulties that appear when using DNF-type fuzzy classification rules. Section III describes Pitts-DNF-C. Section IV analyzes the behavior of Pitts-DNF-C on several real-world problems and compares the system with three other fuzzy learners. Finally, Sect. V concludes and suggests further work.

II. DNF-TYPE FUZZY RULES

In order to obtain high interpretability degrees, we opted for a compact description based on DNF-type fuzzy rules where the antecedent is described in conjunctive normal form [8]. This kind of fuzzy rule has the following structure:

IF x_1 is \widetilde{A}_1 and \dots and x_n is \widetilde{A}_n **THEN** $Class = c$ (1)

where each input variable X_i is represented by a set of linguistic terms $\widetilde{A}_i = \{A_{i1} \vee \dots \vee A_{in_i}\}$, whose members are joined by a disjunctive (T -conorm) operator, whilst the output variable contains the class c predicted by the rule. The structure allows the absence of some input variables in each rule by making \widetilde{A}_i to be the whole set of linguistic terms available.

When a set of these rules is learned, two types of conflicts can easily appear: *inconsistency* and *redundancy*.

This work was supported by *Ministerio de Educación y Ciencia* under projects TIN2005-08386-C05-01 and TIN2005-08386-C05-04, and *Generalitat de Catalunya* under grants 2005FI-00252 and 2005SGR-00302

J. Casillas is with the Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain (e-mail: casillas@decsai.ugr.es)

A. Orriols-Puig and E. Bernadó-Mansilla are with the Grup de Recerca en Sistemes Intel·ligents, Universitat Ramon Llull, 08022 Barcelona, Catalonia, Spain (e-mails: {aorriols, esterb}@salle.url.edu)

A rule is inconsistent with another if the antecedent of the two rules overlap. That is, the two rules predict different classes and (i) the antecedents of the two rules are the same; or (ii) they coincide in some labels for each input variable; or (iii) one antecedent is subsumed by the other (i.e., the antecedent of one rule is completely contained in the antecedent of the other rule, which is larger and more comprehensible). For instance, the following two fuzzy classification rules are inconsistent as they have equal antecedent but different class:

```
IF X1 is A1 and X2 is A2 THEN Class=c1
IF X1 is A1 and X2 is A2 THEN Class=c2
```

Similarly, the two rules below are inconsistent since their antecedents are partially overlapped:

```
IF X1 is {A1 or B1} and X2 is A2 THEN Class=c1
IF X1 is A1 and X2 is {A2 or B2} THEN Class=c2
```

And in the below case, the inconsistency is because the first rule subsumes the second rule:

```
IF X1 is A1 and X2 is {A2 or B2} THEN Class=c1
IF X1 is A1 and X2 is A2 THEN Class=c2
```

All these cases of inconsistency cause a linguistic contradiction that should be avoided.

A second, less serious problem is when the antecedents of two rules are overlapped, as in any of the above examples, but the consequent is the same. In this case, we have a *redundancy*. For example:

```
IF X1 is A1 and X2 is {A2 or B2} THEN Class=c1
IF X1 is A1 and X2 is A2 THEN Class=c1
```

Having redundant rules may unnecessarily increase the rule set size. Moreover, redundant rules may provoke some undesirable interaction effects with some inference engines. When both rules have the same antecedent or one subsumes the other, the fuzzy rule set can be easily fixed by removing the repeated or the most specific rule.

The use of this flexible structure to represent the antecedent of the fuzzy rule can also lead the algorithm to generate over-general rules. This fact appears when a DNF-type fuzzy rule includes a higher number of linguistic terms per variable than the ones strictly necessary. It makes the fuzzy rule to cover input regions without training data. Although it is arguable, this effect may be undesirable since there is not any evidence that indicates that this region should be covered by the given rule. So, our system avoids creating rules for regions of the feature space which are not represented by any example. In this way, the learning algorithm that we propose in this paper has been designed to avoid generating solutions with inconsistencies, redundancies, or over-generality.

III. DESCRIPTION OF PITTS-DNF-C

Figure 1 outlines the algorithm of Pitts-DNF-C for classification tasks. The algorithm is a multi-objective Pittsburgh-style Learning Classifier System which is guided by two objectives: individual's accuracy and individual's size. In the next sections, a deep description of the system is provided.

```
function Pitts-DNF-C ( d is Dataset )
  P := Initialize ( P )
  CH := coveringHypermatrix ( d )
  Evaluate ( P, dataset )
  while ( not stop condition )
    P1 := multiObjectiveSelection ( P )
    P2 := crossover ( P1 )
    P3 := antecedentMutation ( P2, CH )
    P4 := consequentMutation ( P3 )
    P5 := completenessOperator ( P4, d )
    P5 := evaluate ( P5, d )
    P := multiObjectiveReplacement ( P5, P )
  endWhile
endFunction
```

Fig. 1. Scheme of Pitts-DNF-C

A. Representation

Pitts-DNF-C consists of a population of individuals. Each individual is represented by a set of rules of variable size. The rules, whose condition is in *conjunctive normal form*, follow the structure presented in Eq. 1. The antecedent of each rule is encoded as a binary string of length equal to the sum of the number of linguistic terms per variable; the consequent is represented by an integer that identifies the predicted class. A one-valued allele indicates that the corresponding linguistic term is used in the variable.

For example, assuming that we have three linguistic terms (S [small], M [medium], and L [large]) and three classes (c_0 , c_1 , c_2), the fuzzy rule [IF X_1 is S and X_2 is {M or L} THEN $Class=c_1$] is encoded as [100|011||1]. A chromosome is the concatenation of an arbitrary number of these fuzzy rules, e.g., [100|011||1 010|111||0 001|101||3] for a set of three rules.

Variables with all labels set to '1' are allowed (which means that the variable is not considered in the corresponding rule). However, variables with all labels set to '0' are forbidden. That is, we do not consider that a variable with all labels set to '0' equals to a variable with all labels set to '1', as sometimes done in the literature. We take the former approach since the latter one implies that solutions that are genotypically far would be closer phenotypically, which is not recommended.

B. Initialization

The initialization procedure has to guarantee that the initial individuals cover all the input examples, that is, that the rule set satisfies the completeness condition. For this purpose, we use a variation of the Wang-Mendel algorithm [15] for classification to create the first individual, which works as follows. First, a rule is generated for each training example; the linguistic term that maximizes the matching with the input value is assigned to each variable and the class of the rule is set to the class of the input example. Then, redundant and inconsistent rules are removed. That is, rules that have the same antecedent are removed from the rule set, regardless of the class they predict. Instead of them, a new rule with

the same antecedent and the majority class among this group of rules is introduced into the first individual.

The remaining individuals are initialized similarly. For each new individual, the first individual is copied, and the class of each rule is randomly chosen among all the classes of the training examples with which the rule has a matching degree greater than zero. Thus, the length of all the individuals in the initial population is the same, and the rules variables only have one linguistic term. So, note that the initial generality is really low. The multi-objective approach and the antecedent mutation operator will be responsible for introducing more generality to these rules.

C. Covering Hypermatrix Computation

The *covering hypermatrix* is the structure responsible for avoiding overgenerality in the rule sets. This structure is created as follows. For each training example, we create all the rules' antecedents which match with the example and whose variables contain a single linguistic term. Then, all these rules are inserted into the covering hypermatrix. For example, let us assume that we have a problem with two input variables ranging from 0 to 1. Moreover, each variable can take three linguistic terms (S [small], M [medium], and L [large]), which are defined by means of Ruspini's strong fuzzy partitions by three uniformly distributed triangular-shaped membership functions. Then, for the training example (0.25, 0.75), we will insert the following four combinations of labels into the hypermatrix: (X_1 is S and X_2 is M), (X_1 is S and X_2 is L), (X_1 is M and X_2 is M), and (X_1 is M and X_2 is L). This structure has been implemented with a hash table, since it provides an efficient implementation for key searching.

D. Crossover Operator

The crossover operator interchanges rules between the two parents, but it does not modify them. Furthermore, it guarantees that the offspring present neither inconsistencies nor redundancies. The pseudo code for this operator is shown in Fig. 2.

E. Antecedent Mutation Operator

This operator, jointly with the consequent mutation operator, are responsible for the creation of new fuzzy rules. Figure 3 shows the pseudo code for the operator. It randomly selects a variable to undergo mutation. Mutation permits two options: (i) expand the variable, i.e., add a new linguistic term to the variable, or (ii) contract the variable, i.e., remove a linguistic term from the variable. The operator analyzes all the possible mutations that ensure consistency and non-overgenerality of the resulting rule set. Consistency after mutation is checked by analyzing the collision of the new rule with the remaining rules of the individual. Non-overgenerality is checked by means of the covering hypermatrix. Then, an option among the possible mutations is chosen as follows. A mutation by expansion is chosen if it is possible. Otherwise, mutation by contraction is selected. It is worth highlighting that the antecedent mutation operator

Function: Crossover Operator.
Input: Two individuals (parents).
Output: Two new individuals (children).
Preconditions: The parents have not internal inconsistencies.
Postconditions: The children have neither inconsistencies nor redundancies by subsumption, but it is possible that they have redundancies by partial overlapping. Lack of completeness can also appear.

- 1) Put all the rules of the two parents into a set S .
- 2) Analyze the inconsistent rules in S (which have to come from two different 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 parent.
- 3) Divide every group of inconsistent rules into two subsets depending on the parent from which the rule came, and take these rules out of S . Assign each subset to a different children.
- 4) Take a random number $r \sim U[1, |S|]$, which will give the number of rules that will be assigned to the first child. Consequently, the remaining $(|S| - r)$ rules will be assigned to the second child.
- 5) Choose r rules from S and assign them to the first child. Each rule has a probability proportional to its generality degree to be selected. Thus, the most general rules will be placed to the first child with higher probability.
- 6) Assign the remaining rules 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 most specific rules.

Fig. 2. Crossover operator

Function: Antecedent Mutation Operator.
Input: One individual and the covering hypermatrix.
Output: The input individual after being mutated.
Preconditions: The received individual has not internal inconsistencies.
Postconditions: The generated individual has neither inconsistencies nor subsumed rules, but it is possible that it has redundancies by partial overlapping. Lack of completeness can also appear.

- 1) Randomly choose an specific input variable to be mutated.
- 2) If expansion is possible, apply it. Otherwise, apply contraction.

Fig. 3. Antecedent Mutation Operator

only explores feasible solutions; thus, it constrains the search space to ensure a better exploration.

1) *Contraction operation:* It converts the mutated rule into a more specific one by choosing a gene of the selected variable with value '1' and flipping it to '0'. This operator can only be applied to variables that have more than one linguistic term. This operator will never cause inconsistency, redundancy or over-generality since it generates a more specific rule, avoiding any conflict with other rules. However, it can cause lack of completeness in the offspring. In this case, the completeness operator will repair the offspring.

2) *Expansion operation:* Oppositely to contraction, the expansion operator generalizes the rule. It chooses a gene

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

- 1) Extract the examples of the data set that are not covered by any fuzzy rule encoded of the individual.
- 2) Apply Wang-Mendel for classification algorithm over this example subset to generate the minimum number of simple fuzzy classification rules that represent them.
- 3) Add these fuzzy rules to the individual. Since these rules come from data that were not covered by any previous rule, neither consistency nor redundancy problems arise.

Fig. 4. Completeness Operator

with allele ‘0’ and flips it to ‘1’. This operation can only be applied when the variable does not have all linguistic terms.

This operator can either cause collision with other rules or generate over-general rules. Therefore, firstly the set of expansion movements that can be applied to the selected variable without causing inconsistencies or over-generality (this latter case is checked using the covering hypermatrix) are generated, and then, one of them is randomly chosen. In case that no expansions can be applied, and that the variable contains, at least, two linguistic terms, the contraction operation is applied.

If after expansion the mutated rule subsumes other rules, the most specific ones are removed. With this operation it is not possible to get lack of completeness.

F. Consequent Mutation Operator

This operator creates new rules by changing the consequent. It simply consists in randomly selecting one 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, a new class is randomly chosen and assigned to the consequent of the rule. This operation does not cause over-generality or lack of completeness.

G. Completeness Operator

The crossover operator and the antecedent mutation by contraction can produce fuzzy rule sets that do not cover some specific training examples. In this case, the completeness operator adds new rules to patch uncovered input sub-spaces. Thus, the completeness operator can be considered as a reparation operator with low incidence since it does not change the generated rules (it only adds new ones). Fig. 4 shows its operation mode.

H. Reasoning Method

We consider the winner rule reasoning method. It consists in returning the class predicted by the fuzzy rule with the highest matching degree. Minimum and maximum are used as T-norm and T-conorm operators. In test phase, if there

does not exist any rule matching a new instance, we search for the closer rule to the example and return the class it predicts. The closer rule is the rule that maximizes the sum of the matching degrees with each input variable.

I. Objective Functions

We consider two objective functions to assess the quality of the individuals: (i) the training error as a measure of their performance, and (ii) the rule set size as a measure of their complexity.

- *Performance:* It is measured with the train accuracy, i.e., the proportion of instances correctly classified by the rule set (for maximization):

$$F_1(S) = \frac{\# \text{ correct classifications}}{\# \text{ examples}} \quad (2)$$

- *Complexity:* As complexity measure, we use the number of DNF-type fuzzy rules (for minimization):

$$F_2(S) = |S| \quad (3)$$

Since the algorithm is designed to ensure optimal covering, i.e., without lack of completeness or with over-generality, we do not care about the linguistic complexity (i.e., number of linguistic terms) of each fuzzy rule. In a natural way, the most general the fuzzy rules, the fewer the number of rules.

This design supposes an advantage since we simplify the interpretability-based objective function. For example, in [10] the authors need to consider a third objective to evaluate the generality degree of “don’t care” attributes in fuzzy rules since their algorithm does not hold this correspondence between number of rules and generality.

J. Multiobjective Approach

A generational approach with the multiobjective elitist replacement strategy of NSGA-II [6] is used. 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 with the complexity objective according to the extreme values of the solutions contained in the analyzed front. For the error objective, the interval $[0, 1]$ is considered when computing the crowding distance.

IV. EXPERIMENTAL RESULTS

This section analyzes the performance of Pitts-DNF-C in several real-world problems. To evaluate its competitiveness, we compare the results with those created by other fuzzy rule-based learners. In the following, we first present the experimental methodology and then show the results.

A. Methodology

We selected a collection of six data sets, whose characteristics are summarized in Table I. All them were extracted from the UCI repository, except for *tao*, which was selected from a local repository.

TABLE I

PROPERTIES OF THE DATASETS. THE COLUMNS DESCRIBE: THE IDENTIFIER OF THE DATASET (ID.), THE NAME OF THE DATASET (DATASET), THE NUMBER OF INSTANCES (#INST), THE TOTAL NUMBER OF FEATURES (#FEA), AND THE NUMBER OF CLASSES (#CL).

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

The quality of the rule sets evolved by Pitts-DNF-C were compared with those created by three fuzzy rule-based classification systems: a variation of the Wang-Mendel algorithm [15] for classification, Fuzzy GP [13], and Fuzzy MaxLogitBoost [12]. We use the version of the Wang-Mendel algorithm for classification described in Sect. III-B. Note that the length of the rule set created by Wang Mendel will be equal to the length of the initial individuals of Pitts-DNF-C. Thus, Pitts-DNF-C is expected to evolve more reduced rule sets. Fuzzy GP is a genetic programming method that builds a fuzzy classifier for each class of the domain by searching for a tree that represents an analytic expression which relates the input and the output variables as accurately as possible. Fuzzy MaxLogitBoost is a boosting algorithm that iteratively invokes a genetic algorithm to extract simple fuzzy rules that are combined to decide the output of new examples. These two methods were run using KEEL [1]. We configured these methods as recommended in the documentation. We only changed the maximum population size of Fuzzy MaxLogitBoost, which was set to a higher value to improve the accuracy of the models (i.e., $N=50$).

The models created by the different methods were evaluated in terms of performance and interpretability. To measure the performance of the models, we used the test accuracy, that is, the proportion of correct classifications on previously unseen examples. Ten-fold cross validation was used to obtain reliable estimates.

Pitts-DNF-C creates fuzzy classification rule sets without overlaps, redundancies, or inconsistencies. The condition of each rule is in *conjunctive normal form* (see Eq. 1). Similar rule sets are created by the Wang-Mendel algorithm, which is described in Sect. III-B. On the other hand, rule sets evolved by Fuzzy GP and Fuzzy LogitBoost can have overlaps, redundancies, and inconsistencies. Fuzzy GP evolves fuzzy rules which are directly mapped from a tree. These rules have the following structure:

$$\begin{aligned} &\mathbf{IF} (x_1 \text{ is } \widetilde{A}_1^1 \text{ and } x_2 \text{ is } \widetilde{A}_2^1) \text{ or} \\ &(x_1 \text{ is } \widetilde{A}_1^2 \text{ and } x_2 \text{ is } \widetilde{A}_2^2) \mathbf{THEN} \text{Class}=\text{c} \end{aligned} \quad (4)$$

That is, each variable can appear in different parts of the antecedent, and the logical operators *and* and *or* are used to join the conditions over variables. Thus, these rules are

much more flexible than the rules of Pitts-DNF-C. However, their interpretability degree is lower.

Fuzzy LogitBoost evolves linguistic fuzzy rules with the following form:

$$\mathbf{IF} x_1 \text{ is } \widetilde{A}_1 \text{ and } \dots \text{ and } x_n \text{ is } \widetilde{A}_n \mathbf{THEN} \text{Class is } c_1 \mathbf{with} w_1, \dots, \text{ and } c_m \mathbf{with} w_m \quad (5)$$

where each variable x_i is represented by a linguistic term $\widetilde{A}_i = \{ A_{i1} \vee \dots \vee A_{in_i} \}$. The consequent maintains one weight w_m per each class, which indicates the soundness with which the weight predicts this class.

Consequently, the comparison of the model's readability among the four learners is complicated. In our analysis, for Pitts-DNF-C, Wang-Mendel, and Fuzzy LogitBoost, we consider the number of rules in the final model. For Fuzzy GP, we collect the number of *and*, *or*, and *is* that appear in the rules.

B. Results

Table II provides the test accuracies obtained with Pitts-DNF-C, Wang-Mendel, Fuzzy-LB, and Fuzzy-GP. As Pitts-DNF-C performs multi-objective optimization, a set of alternative solutions (those of the Pareto set) with different interpretability-accuracy trade-offs is returned in each run. Thus, we report three representative solutions of Pitts-DNF-C for each data set: (i) the test accuracy of the individual with maximum train accuracy, (ii) the test accuracy of the median solution in the Pareto set, and (iii) the test accuracy of the individual with the smallest number of rules. Analogously, Table III shows the model sizes.

In the following, we draw several conclusions from the results:

- Models of Pitts-DNF-C with maximum train accuracy present a higher test accuracy and smaller model size than those created by the Wang-Mendel algorithm, according to a Wilcoxon signed-ranks test at $\alpha = 0.05$ (as recommended in [7]). Thus, Pitts-DNF-C is able to evolve more reduced and accurate rule sets than the ones evolved by the Wang-Mendel algorithm, also assuring that these rule sets are consistent, complete, and non-redundant.
- Models of Pitts-DNF-C with maximum train accuracy do not perform statistically differently than those evolved by Fuzzy MaxLogitBoost and Fuzzy GP. Results also denote that the rule sets evolved by Pitts-DNF-C are larger than those created by Fuzzy MaxLogitBoost. Nonetheless, note that the rule sets of Pitts-DNF-C do not contain any inconsistencies or redundancies. In this case, more rules may be needed in the final populations to avoid it.
- A more detailed analysis of the results obtained with Pitts-DNF-C indicates that it can achieve accurate and compact models for *irs*, *tao*, and *thy*. It achieves models of moderate size with respect to the other learners for *bpa* and *gls*. However, it evolves large models for *wbcd*. That is, the most compact models created by Pitts-DNF-C in the *wbcd* problem contain approximately 232 rules.

TABLE II

COMPARISON OF THE PERFORMANCE OF PITTS-DNF-C WITH WANG-MENDEL FOR CLASSIFICATION, FUZZY MAXLOGITBOOST, AND FUZZY GP.

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

TABLE III

COMPARISON OF THE SIZES OF THE MODELS CREATED BY PITTS-DNF-C WITH THOSE BUILT BY WANG-MENDEL FOR CLASSIFICATION, FUZZY MAXLOGITBOOST, AND FUZZY GP.

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

This measure is by far larger than the 13 rules returned by Fuzzy MaxLogitBoost and the trees evolved by Fuzzy GP. This would indicate that the given problem has a distribution that is difficult to represent with a set of rules without inconsistencies and redundancies. Further research will be made to find more compact ways to express the knowledge in these cases.

V. SUMMARY, CONCLUSIONS, AND FURTHER WORK

This paper proposed Pitts-DNF-C, a multi-objective Pittsburgh-style fuzzy rule-based classification system that evolves consistent, complete, non-redundant, and compact fuzzy rule sets. The proposed learning methodology re-designs typical Pittsburgh-style operators, i.e., initialization, crossover, and mutation, and proposes new ones, i.e., the completeness operator, to ensure that new individuals introduced into the population satisfy these conditions. Some of these operators rely on the covering hypermatrix, a new structure designed with the aim of avoiding over-generality.

Experimental results show the suitability of this method on six real-world data sets. Pitts-DNF-C created rule sets that are smaller and more accurate than those generated by the Wang-Mendel algorithm, and statistically equivalent to the ones evolved by Fuzzy MaxLogitBoost and Fuzzy GP. A more detailed analysis of the results denoted that Pitts-DNF-C created rules sets of the *wbcd* problem that were considerably larger than those built by the other learners. This indicates that a higher number of rules may be needed to evolve complete rule sets without consistencies and redundancies. Further research will be conducted on several parts of the algorithm to analyze in more detail the generalization and scalability capabilities of Pitts-DNF-C.

REFERENCES

[1] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández,

and F. Herrera. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, In press.

[2] J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors. *Accuracy improvements in linguistic fuzzy modeling*. Springer, Heidelberg, Germany, 2003.

[3] J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors. *Interpretability issues in fuzzy modeling*. Springer, Heidelberg, Germany, 2003.

[4] J. Casillas and P. Martínez. Consistent, complete and compact generation of DNF-type fuzzy rules by a Pittsburgh-style genetic algorithm. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pages 1745–1750, 2007.

[5] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*. World Scientific, 2001.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarevian. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[7] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[8] A. González and R. Pérez. Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems*, 96(1):37–51, 1998.

[9] A. González and R. Pérez. SLAVE: a genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.

[10] H. Ishibuchi and Y. Nojima. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, 44(1):4–31, 2007.

[11] Y. Jin, W. von Seelen, and B. Sendhoff. On generating FC³ fuzzy rule systems from data using evolution strategies. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 29(4):829–845, 1999.

[12] J. Otero and L. Sánchez. Induction of descriptive fuzzy classifiers with the logitboost algorithm. *Soft Computing*, 10(9):825–835, 2006.

[13] L. Sánchez, I. Couso, and J. A. Corrales. Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences*, 136(1–4):175–191, 2001.

[14] 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*, 35(2):143–155, 2005.

[15] L.-X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, 1992.