Multi-Objective GA Rule extraction in a parallel framework

Passent M. Elkafrawy[o], Amr M. Sauber[o]

***Abstract*—Genetic algorithm (GA) has been used as a conventional method for classifiers to evolve solutions adaptively for classification problems. Multiobjective evolutionary algorithms (MOEAs) that use nondominated sorting and sharing have been criticized mainly for their: 1) O(MN^3) or O(MN^2) computational complexity (where M is the number of objectives and N is the population size); 2) nonelitism approach [?]; and 3) the need for specifying a sharing parameter. In this paper, a new simple yet efficient approach is proposed to improve the performance of Multi-objective GA-based classifiers; the computational complexity of the proposed technique is O(MN), we also used a class decomposition technique. A classification problem is fully partitioned into several small problems each of which is responsible for solving a fraction of the original problem. We experimentally evaluate our approach on three different datasets and demonstrate that our algorithm can improve classification rate compared with normal GA and nonpartioned techniques; our technique is optimized using OpenMP-like implementation to take advantage of multi-threads or multi-processors.**

***Index Terms*—Genetic Algorithm, Rule-based classification, divide and conquer, Multiobjective evolutionary Algorithms, muli-threading.**

I. Introduction

Data mining is a very active and rapidly growing research area in the field of computer science. The task of data mining is to extract useful knowledge for human users from a database. Evolutionary multiobjective optimization (EMO) has been applied to data mining in some studies; [8] used multi-objective association rule mining Pareto based GA for evaluating rules by defining three objectives; support count, comprehensibility and interestingness. Support count is the number of records, which satisfies all the conditions present in the rule, this objective gives the accuracy of the rules extracted from the database. Comprehensibility is measured by the number of attributes involved in the rule and tries to quantify the understandability of the rule. Interestingness measures how much interesting the rule is using defined formula. [14] replaced support count with predictive accuracy and presented a new Elitist multi-objective genetic algorithm (EMOGA) with a hybrid crossover operator for optimizing the objectives. [5] used a real coded MOGA for generating a set of optimized classification rules, where real-valued attribute ranges are encoded with real-valued genes and defined new suitable genetic operators, two objectives are used; confidence and coverage. Using three objectives confidence, coverage and attractiveness [6] used a Lexicographical approach to evaluate these objectives in a student database case study. The basic idea of Lexicographical approach is to assign different priorities to different objectives and then focus on optimizing the objectives in their order of priority, it treats each of the criteria separately, recognizing that each criterion measures a different aspect of quality of a candidate solution.

[15] and [12]both used NSGA-II to implement MOGA and association rule mining respectively. [13] proposed a genetic rule selection mechanism consists of two stages. In the first stage, a pre-specified number of candidate rules are extracted from numerical data using a data mining technique. In the second stage, an EMOGA is used for finding non-dominated rule sets with respect to three objectives: to maximize the number of correctly classified training patterns, to minimize the number of rules, and to minimize the total rule length. As far as we know there is no attempt to use MOGA along with class decomposition technique, our experiments show that this mechanism is efficient and scalable.

II. Multi-objective optimization

Contrary to single-objective optimization problem, multi-objective optimization problem deals with simultaneous optimization of several incommensurable and often competing objectives such as performance and cost. For example, when the design of a complex hardware is considered, it is required for the cost of such systems to be minimized while the maximum performance is expected. If there is more than one objective criterion as in the example mentioned above, some of them can be considered as constraints in the problem. For example, while trying to optimize a system for large performance in low cost, the size of the system must not exceed given dimensions as a separate optimization criterion. By this way, a multi-objective optimization problem can be formalized as follows [19]:

Definition 1: A multi-objective optimization problem includes, in general, a set of a parameters (called decision variables), a set of b objective functions, and a set of c constraints; objective functions and constraints are functions of the decision variables. The optimization goal is expressed as:

$$min/max\ y = F(x) = (f_1(x), f_2(x), ..., f_n(x))$$
$$constraint\ e(x) = (e_1(x), e_2(x), ..., e_n(x))$$
$$Where \quad x = (x_1, x_2, ..., x_n)\ \in X$$
$$y = (y_1, y_2, ..., y_n)\ \in Y$$

where x is the decision vector, y is the objective vector, X denotes the decision space, and Y is called the objective space; the constraints $e(x) \leq 0$ determine the set of feasible solutions.

Let us consider the above definition and assume that the two objectives performance (f_1) and cheapness (f_2), the

[o] Faculty of Science, Menoufiya University,Egypt

ISBN: 978-1-61804-019-0

inverse of cost, are to be maximized under size constraints (e_1). Then, an optimal design might be an architecture which achieves maximum performance at minimal cost and does not violate the size limitations. If such a solution exists, we actually only have to solve a single-objective optimization problem. The optimal solution for either objective is also the optimal for the other objective. However, what makes multi-objective optimization problems difficult is the common situation when individual optima corresponding to the distinct objective functions are sufficiently different.

We used a simple yet efficient Weighted sum approach: Transforming a multi-objective problem into a single objective function $F(x)$ by far the most commonly used approach in data mining literature. Normally, this can be done by a weighted sum of objective functions. That is $F(x)$ is the fitness function used by the GA of a given candidate rule is typically measured by the formula:$F(x) = w_1 f_1(x) + w_2 f_2(x) + ... + w_n f_n(x)$ where $f_i(x)$ is an objective and w_i is a weight represents the significance of the objective (to be let to the human expert to decide). The strength of this method is its simplicity and less computation complexity. Fortunately this simplicity did not affected the performance as our experiments emphasize.

III. Methodology

A classification problem is fully partitioned into C class modules; where each module is used for building a classifier for only one class. These modules are trained independently in parallel to find solutions for the C sub-problems. There are two general approaches for GA-based rule optimization and learning [2]. The Michigan approach uses GAs to evolve individual rules, a collection of which comprises the solution for the classification system[11]. Another approach is called the Pitt approach, where each chromosome represents a classifier (rule set), rule sets compete against each other with respect to performance on the domain task[4].

In this work, the Michigan approach is chosen, as it is straightforward for rule evaluation. Because each chromosome in the Michigan approach represents a candidate rule to that is used with other rules to construct a solution for target problem, efficient Rules are to compete freely without being influenced by other inefficient ones as in the Pitt approach. The number of rules in a classifier is given as a threshold.

After Selecting efficient rules for each class the final classifier is constructed by combining the C classifiers. Each classifier is represented in the final solution using the participating percentage of the corresponding class in the learning data. An integration algorithm is used to solve the conflicts between rules from the C different classifiers if exist.

We used a different Technique to implement our proposed method; instead of loading the data into memory as an array or a list we used a Relational database to manipulate data instances, and instead of using multiple if conditions to evaluate rules we used a T-SQL statement in order to enable scalability and implement a generic technique. Out technique also does not require data pre-processing as will be discussed in details in the following sub-sections

A. Individuals representation

An individual in our method is a classification rule where each gene represents the minimum and maximum values of intervals of each attribute that belongs to such rule. In our approach, we use fuzzy IF-THEN rules with continuous attributes. A rule set (classifier) consists of a - user determined - number of rules as a solution candidate for a classification problem. We encode rule R_i according to Figure 1.

Figure 1. An individual representation

where w_j is a real-valued variable taking values in the range [0,1] instead of either 0 or 1 as the related work [9], [16], [17], [18] . This variable indicates the fuzzy membership rate for the potential attribute presence in the corresponding classification rule. More precisely, when w_j is smaller than a user-defined threshold (called Limit) the attribute will be neglected in the related rule. Therefore, the greater the value of the threshold Limit, the smaller the probability that the corresponding attribute will be included in the rule. V_{jmin} and V_{jmax} are the limits of the intervals corresponding to the attribute A_i. Note that the above encoding is quite flexible with respect to the length of the rules. A traditional GA is very limited in this aspect, since it can only cope with fixed-length rule. In our approach, although each individual has a fixed length, the genes are interpreted (based on the value of the weight w_i) in such a way that the individual phenotype (the rule) has a variable length. The start of the first population consists of generating, arbitrarily, a fixed number of individuals during the evolution.

B. Pre-Processing

We implemented the individual representation 'as-is' such that no pre-processing is required to run our technique against any problem. Most of the related work encode V_{jmin}, V_{jmax} each as character by dividing the range of possible values to predefined intervals thus they encode the chromosomes as strings. Although this do simplify the implementation, it assumes that the data range is homogeneous or at least could be easily distributed. We used the given real numbers in order to accomplish two factors; 1) To implement a generic technique that is as independent as possible from the problem, 2) To minimize the overhead of pre-processing. Applying this representation along with using relational database as a back end for manipulating data, enabled us to implement a generic technique that takes a database name and a number of attributes as parameters. this technique can be implemented on any problem.

C. Genetic Operators

For the developed method, the usual one-point *crossover* operator is stochastically applied with a predefined probability, using two individuals of the selected pool. The crossover point is a percentage of the length of the individual that defines the

ISBN: 978-1-61804-019-0

starting point from where the crossover breaks the string. We use arithmetic crossover method [7]. The employed method works as follows:

Consider two chromosomes

$H_1 = (g1^1, ..., g_n^1)$ and $H_2 = (g_1^2, ..., g_n^2)$.

Applying the crossover operator on H_1 and H_2 generates two off-springs

$H_1' = (g_1^1, .., g_i^1, g_{i+1}^2, .., g_n^2)$ and $H_2' = (g_1^2, .., g_i^2, g_{i+1}^1, .., g_n^1)$.

where $g_l^j = (W_l, V_{lmin}, V_{lmin})$

The *mutation* operator is used to foster more exploration of the search space and to avoid unrecoverable loss of genetic material that leads to premature convergence to some local minima. In general, mutation is implemented by changing the value of a specific position of an individual with a given probability (0.8), denominated mutation probability. We gave a high mutation probability than given in common GAs to advance the internal change of the rule features. To allow all different possibilities that could be generated within the attribute representation. We developed two mutation operators tailored for our genome representation:

1) *Shift the starting location towards the right or the left. This operator changes the value in the starting location of a randomly selected gene is increased or decreased by 0.1 bounded by the minimum value for this attribute and the starting location.*
2) Shift the ending location towards the right or the left. This operator changes the value in the ending location of a randomly selected gene is increased or decreased by 0.1 bounded by the maximum value for this attribute and the ending location.

Using the lower and upper bounds of the attribute domain as bounds for shifts ensures that they are never exceeded.

D. Fitness Function

As each chromosome in our approach comprises one rule, the fitness function actually measures the collective behavior of the rule. The fitness function is a compound of two objectives: accuracy and score.

Definition 2: Accuracy of a certain rule is the ratio of correctly classified instances by this rule to the whole number of instance to which this rule satisfied

$$f_1 = \frac{C}{n} = \frac{Number\ of\ correctly\ classified\ instances\ by\ this\ rule}{Total\ number\ of\ instances\ satisfying\ this\ rule}$$

Definition 3: Score of a certain rule is the percentage of the instances satisfying the rule to the whole number of instance to represent the significance of this rule

$$f_2 = \frac{S}{N} = \frac{Total\ number\ of\ instances\ satisfy\ this\ rule}{Total\ number\ of\ instances}$$

Definition 4: Fitness Function $F = w_1 f_1 + w_2 f_2$where w_1, w_2 are weights assigned by the human expert.

E. Class Decomposition

Let us assume a classification problem has C classes in the n-dimensional attribute space. The task of classification is to assign instances to one out of the pre-defined C classes, by discovering certain relationship between the attributes. Then, the discovered rules can be evaluated by classification accuracy or error rate either on the training data or test data.

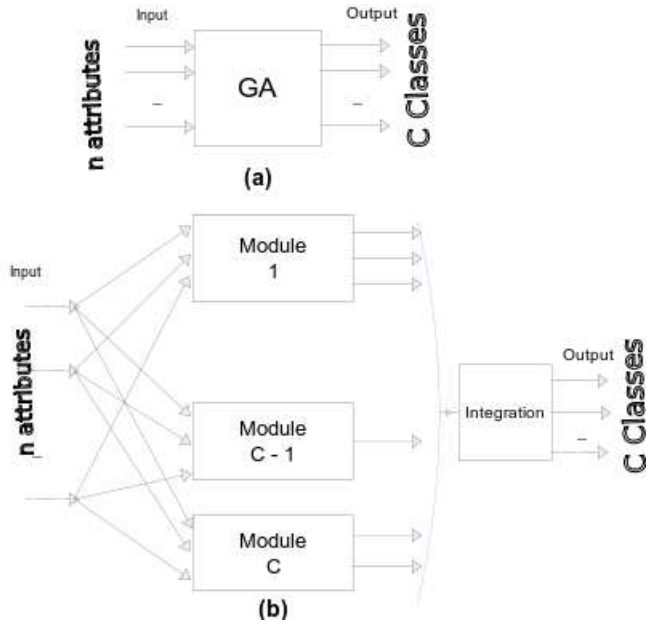

Figure 2. Class Decomposition

A traditional GA maps attributes to classes directly in a batch manner, which means all the attributes, classes, and training data are used together to train a group of GA chromosomes. Our evolutionary class decomposition-based approach is significantly different. As shown in Figure 2, it generally consists of three steps. First, the original problem is divided into C different sub-problems in terms of classes. Then, C GA modules are constructed for these sub-problems, where each module will be responsible for evolving a sub-solution. As input data can be inconsistent Finally, these sub-solutions are integrated to form the final solution of the original problem.

We used multi-threading to implement a portable machine-independent Parallelism using OpenMP like implementation. Two level Parallelism is used in the our technique; 1) the rule generation in each GA module is implemented using parallelism, 2) each GA module is trained in a different thread. It is to the operating system to map each thread to a working processors according to run-time implementation implementation.

F. Distributed Class Decomposition

Consider having c classes in the classification problem on hand, with n-dimensional pattern space, and p vectors where $X_i = (x_{i1}, x_{i2}, \ldots, x_{in}), i = 1, 2, \ldots, p, p \ll c$ are input instances. Accordingly, the given classification problem can be denoted as follows:

$$f : X \to T$$

where $X \in R^n$ is the set of instances with n attributes, and $T \in R^c$ is the set of output classes. Required a mapping f that maximizes *accuracy*. Assume that the c-class problem is

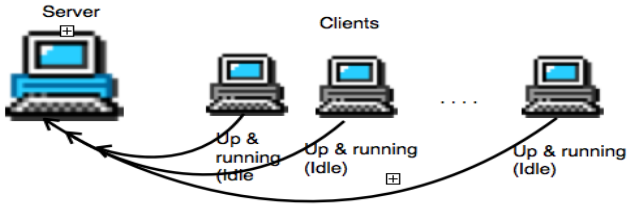

Figure 3. Server/Client setting

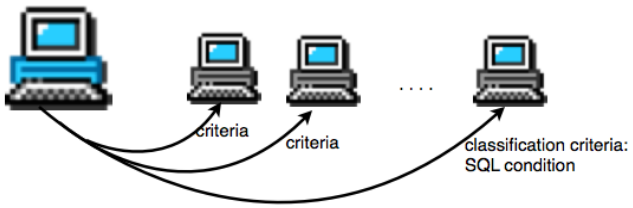

Figure 4. Server sends classification criteria to clients

divided into c sub-problems, each contains the vectors X_i that present T^j. Hence the class set can be denoted as

$$T = T^1 \bigcup T^2 \bigcup \ldots \bigcup T^c$$

each sub-problem can be formulated as required f_j with an optimal classification accuracy and score on T^j

$$f_j : X \rightarrow T^j$$

.

Having c sub-problems, c GA sub-modules are constructed and executed in parallel. As shown in figure 3, the server will send to each client the criteria that defines the subset of instances of each T^j as a SQL condition figure 4. The SQL condition is executed against a database of the classification problem. Then each client is busy working on its GA figure 5. After finishing, each module sends the result rule set to the server to be integrated with other sub-solutions to construct the final solution6.

G. Integration

Although each GA module has evolved a portion of the solution, we cannot just simply aggregate their sub-solutions as the final one, because each GA module only classifies only one class. Therefore, when the sub-solutions are combined together, there may still exist conflicts among the sub-solutions. For example, rules from different modules may

Figure 5. Clients busy executing its portion of GA module

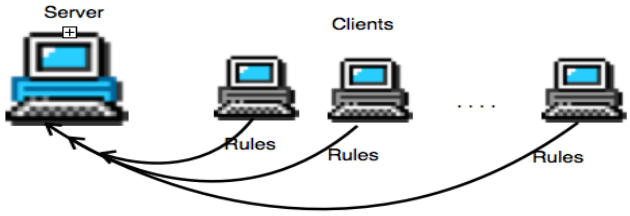

Figure 6. Rule Sets are sent to server

Algorithm 1 Conflects Resolving

- If an instance is classified into more than one class categories by the rule set, it will be classified into the one whose corresponding module achieves the highest classification rate in the training phase.
- If an instance is not classified into any class category by the rule set, it will be classified into the class whose corresponding module achieves the lowest classification rate in the training phase, if available.

classify an instance into several classes. In order to resolve these conflicts and further improve the classification rate, the classifier employs some intelligent decision rules. The detailed integration process is explained as follows.

The classifier constructs an overall rule set by aggregating all rules from C modules. Some decision rules are added to help solving the above-mentioned conflicts. The ending classification rate obtained from each module would be useful for this purpose. Currently, Algorithm 1 shows decision rules have been employed:

To sum up, the process employed can be summarized by algorithm 2

First, an initial Global Population GP is initialized and for each class C_i, an initial population P_i is generated in Step 2.a. Crossover and mutation operations are applied to each pair in P_i to generate the offspring population P_i^I in Step 2.b.i. The next population is constructed by choosing good solutions from the merged population $P_i \cup P_i^I$. We used elitist selection to select the best rules from the old generation to act as basis for the next generation, steps 2.b.i and 2.b.ii

Algorithm 2 Integration Algorithm

Input: Population size N; Maximum number of generations G; Crossover probability p_c; Mutation rate p_m.
Output: Classifier

1) GP = ϕ
2) for each Class C_i do
   a) P_i:=Initialize(P,C_i)
   b) while the termination criterion is not satisfied do
      i) P_i^I:= Genetic Operators(P_i)
      ii) P_i:= Rank and select fittest($P_i \cup P_i^I$)
   c) end while
   d) GP = $GP \cup P_i$
3) end for each
4) return (GP)

ISBN: 978-1-61804-019-0

are repeated to obtain the set of rules (classifier) of the current class, in step 2.d the rule set P_i is added to the GP. Step 2 is to be repeated for each class, finally GP should contain all rules needed to identify all classes.

IV. Experimental results

We have implemented several classifiers running on three benchmark data sets to evaluate our approach. The data sets chosen are the wine data, iris data and breast cancer data. They are all available in the UCI machine learning repository [3]. They all are real-world problems. We partition each data set into two parts with an equal number of instances. One half is for training, and the other half is for testing. We use the training data to train the rule set, and test the generalization power of resulting rule set with the test data. we used a robust and scalable implementation as we stored the training and testing data in MySQL database, and used SQL queries to evaluate the rules enabling further and more sophisticated application. we compared our technique with Normal GA [10] and[15], Class Decomposition single objective [1]. All experiments are completed on Intel 1.8GHz Dual Core, 1GB RAM PC running windows XP SP2 32bit. The results reported are averaged over five independent runs. The parameters, such as mutation rate, crossover rate, generation limits, are given under the results. We also noticed that the hybrid ending criteria made the execution very fast for certain databases some breast cancer tests took only 2 minutes with total accuracy of 0.90.

A. Parallel EMOGAR

1) The Wine Data: The wine data contains the chemical analysis of 178 wines from three different cultivars in the same region in Italy. The analysis determines the quantities of 13 constituents found in each of the three types of wines. In other words, it has 13 continuous attributes, 3 classes, 178 instances, and no missing values. The experimental results are shown in tableI

	Normal GA	Kaya's	Class Decomposition.	EMOGAR
1	0.27	0.82	0.84	0.85
2	0.25	0.88	0.86	0.86
3	0.29	0.73	0.90	0.88
4	0.41	0.83	0.85	0.88
5	0.25	0.79	0.83	0.92
Average	0.29	0.81	0.86	0.88

Table I
Comparison of the classifiers performance on Wine test data

2) The Iris data: The glass data set contains data of different flowers, the data set consists of samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample, they are the length and the width of sepal and petal. Based on the combination of the four features, Fisher developed a linear discriminant model to determine which species from these four measurements. This data set consists of 150 instances with 4 continuous attributes from 3 classes, and no missing values. The experimental results are shown in tableII

	Normal GA	Kaya's	Class Decomposition.	EMOGAR
1	0.92	0.96	0.95	0.97
2	0.90	0.95	0.96	0.96
3	0.84	0.92	0.92	0.96
4	0.92	0.90	0.96	0.97
5	0.88	0.92	0.95	0.96
Average	0.89	0.93	0.95	.97

Table II
Comparison of the classifiers performance on Iris test data

3) The Cancer data: The Breast Cancer Wisconsin contains features computed from digitized images of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image, these features are to be used in breast cancer diagnosis. This data set consists of 569 instances with 30 continuous attributes from 2 classes, and no missing values. The experimental results is shown in table III

	Normal GA	Kaya's	Class Decomposition.	D. EMOGAR
1	0.33	0.56	0.62	0.90
2	0.33	0.48	0.50	0.88
3	0.33	0.47	0.59	0.92
4	0.33	0.35	0.50	0.84
5	0.33	0.49	0.66	0.82
Average	0.33	0.47	0.58	.87

Table III
Comparison of the classifiers performance on Cancer test data

B. Distributed EMOGAR

	Normal GA	Kaya's	Class Decomposition.	D. EMOGAR
1	0.27	0.82	0.84	0.85
2	0.25	0.88	0.86	0.86
3	0.29	0.73	0.90	0.88
4	0.41	0.83	0.85	0.88
5	0.25	0.79	0.83	0.92
Average	0.29	0.81	0.86	0.88

Table IV
Comparison of the classifiers performance on Wine test data

1) The Wine Data:

	Normal GA	Kaya's	Class Decomposition.	EMOGAR
1	0.92	0.96	0.95	0.97
2	0.90	0.95	0.96	0.96
3	0.84	0.92	0.92	0.96
4	0.92	0.90	0.96	0.97
5	0.88	0.92	0.95	0.96
Average	0.89	0.93	0.95	.97

Table V
Comparison of the classifiers performance on Iris test data

2) The Iris data:

3) The Cancer data:

V. Conclusion

This chapter proposed a new multi-objective approach based on class decomposition for GA-based classifiers. A simple

ISBN: 978-1-61804-019-0

	Normal GA	Kaya's	Class Decomposition.	D. EMOGAR
1	0.33	0.56	0.61	0.91
2	0.33	0.48	0.50	0.87
3	0.33	0.47	0.59	0.92
4	0.33	0.35	0.50	0.84
5	0.33	0.49	0.66	0.83
Average	0.33	0.47	0.57	.86

Table VI
COMPARISON OF THE CLASSIFIERS PERFORMANCE ON CANCER TEST DATA

representation of two objective functions accuracy and score is used. A classification problem is decomposed into several modules and each module is responsible for solving a fraction of the original problem. We have two implementation; 1) Modules are trained in parallel (two level Parallelism inter-module Parallelism and Intra-module Parallelism), 2) Modules as trained in a distributed environment (parallelism within each module also implemented) . The sub-solutions obtained is either of the implementation are integrated to further obtain a final solution. To evaluate our method, we have conducted some experiments. The results have shown that our algorithm is efficient and robust . Our Experiments Also showed that there is a performance enhancement of about 30% for the distributed implementation without sacrificing the accuracy.

REFERENCES

[1] Passent El Kafrawy Amr M. Sauber. Using class decomposition for building ga with fuzzy rule-based classifiers. *Twenty Fourth International Conference on Industrial Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2011)*, 2011.

[2] E. Bernadó, X. Llora, and J.M. Garrell. Xcs and gale: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In *Proceedings of the 4th international workshop on learning classifier systems (IWLCS-2001)*, pages 337–341. Citeseer, 2001.

[3] C. L. Blake and C. J. Merz. Repository of machine learning databases, 1998.

[4] K. De Jong. Learning with genetic algorithms: An overview. *Machine learning*, 3(2):121–138, 1988.

[5] D. Dutta, G.N. Burdwan, W.B. Burdwan, and P. Dutta. Discovering classification rules by real coded moga. 2009.

[6] Dipankar Dutta. Discovering prediction rules for predicting quality of students using multi objective genetic algorithm from student database. *Proceedings of International Conference on recent trends in Computing and Communications (FACT 2009), KCG College of Technology, Chennai, India*, pages 19–24, 2009.

[7] Herrera F, Lozano M, and Verdegay JL. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[8] A. Ghosh and B. Nath. Multi-objective rule mining using genetic algorithms. In *Information Science*, volume 163, pages 123–133. 2004.

[9] S.U. Guan and F. Zhu. Class decomposition for ga-based classifier agents-a pitt approach. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):381–392, 2004.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[11] J. H Holland. Escaping brittleness: the possibilities of general- purpose learning algorithms applied to parallel rule-based systems. In *International Conference on Machine Learning*, 1986.

[12] H. Ishibuchi, I. Kuwajima, and Y. Nojima. Multiobjective classification rule mining. *Multiobjective problem solving from nature: from concepts to applications*, page 219, 2008.

[13] H. Ishibuchi and T. Yamamoto. Effects of three-objective genetic rule selection on the generalization ability of fuzzy rule-based systems. pages 69–69, 2003.

[14] Hisao Ishibuchi, Yusuke Nojima, and Isao Kuwajima. Finding simple fuzzy classification systems with high interpretability through multiobjective rule selection. pages 86–93, 2006.

[15] Mehmet Kaya. Autonomous classifiers with understandable rule using multi-objective genetic algorithms. *Expert Systems with Applications*, 37:3489–3494, 2010.

[16] F. Zhu and S. Guan. Feature selection for modular ga-based classification. *Applied Soft Computing*, 4(4):381–393, 2004.

[17] F. Zhu and S. Guan. Ordered incremental training for ga-based classifiers. *Pattern recognition letters*, 26(14):2135–2151, 2005.

[18] F. Zhu and S.U. Guan. Cooperative co-evolution of ga-based classifiers based on input decomposition. *Engineering Applications of Artificial Intelligence*, 21(8):1360–1369, 2008.

[19] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *evolutionary computation, IEEE transactions on*, 3(4):257–271, 1999.