The Fourteenth International Symposium on Artificial Life and Robotics 2009 (AROB 14th '09),
B-Con Plaza, Beppu, Oita, Japan, February 5 - 7, 2009

Pareto-optimal Fuzzy Rule Mining with EMO Algorithms and Its Improvement by Heuristic Initialization

Isao Kuwajima, Yusuke Nojima, and Hisao Ishibuchi

Gakuen-cho 1-1, Naka-ku, Sakai, Osaka 599-8531, Japan
(Tel : 81-072-254-9198; Fax : 81-072-254-9915)
(Email {kuwajima@ci., nojima@, hisaoi@}cs.osakafu-u.ac.jp)

Abstract: In this paper, we apply evolutionary multiobjective optimization (EMO) algorithms to Pareto-optimal fuzzy rule mining. Pareto-optimal rules, which are Pareto-optimal in confidence and support maximization, have an interesting feature that they maximize other various rule evaluation measures. In our method, we use MOEA/D and NSGA-II, which are simple and high-performance EMO algorithms, to efficiently discover Pareto-optimal fuzzy rules. Conventional data mining techniques such as Apriori need to set thresholds on confidence and support to reduce the search space. Our EMO-based method does not need those parameters because EMO algorithms make an efficient search toward confidence-support trade-off curve. Through computational experiments, we show that our EMO-based method can generate Pareto-optimal rules in a short time.

Keywords: Association rule mining, evolutionary multiobjective optimization, knowledge extraction.

I. INTRODUCTION

Association rule mining [1] is a popular and well-known method for discovering interesting relations. In its basic form, all association rules satisfying some constraints on rule evaluation criteria are extracted from a database. There are numerous proposals for the rule evaluation criteria that quantify the interestingness or goodness of a rule. There are two major rule evaluation criteria: confidence and support. Other rule evaluation criteria include gain, variance, chi-squared value, entropy gain, Gini, Laplace, lift, and conviction. It is shown in [2] that Pareto-optimal rules in terms of confidence and support maximization have an interesting feature that the best rule with respect to any of the above-mentioned criteria is included in the Pareto-optimal rules.

In this paper, we apply evolutionary multiobjective optimization (EMO) algorithms to discover Pareto-optimal fuzzy rules. Among EMO algorithms, we use MOEA/D [3] and NSGA-II [4], which are simple and effective EMO algorithms. Conventional data mining techniques such as the Apriori algorithm [1] need to set thresholds on confidence and support or need to set the maximum rule length to reduce the search space. Our EMO-based method does not need to set those parameters because EMO algorithms make an efficient search toward a confidence-support trade-off curve.

The performances of algorithms are examined on the data sets from the UCI machine learning repository. Through computational experiments, we show that our EMO-based method can generate Pareto-optimal rules in a short time compared to data mining techniques especially when we use MOEA/D. We also show that our EMO-based method can obtain better performance because there is no need to set thresholds on confidence and support or to set the maximum rule length.

In addition, we improve the performance of our EMO-based method using problem-specific initialization, which we call *heuristic initialization*. We show that the use of heuristic initialization can improve the performance of our EMO-based method at the early stage of evolution.

II. FUZZY RULES

Let $\mathbf{x} = (x_1, ..., x_n)$ be an n-dimensional pattern vector. Our task is to discover fuzzy rules of the following form:

$$\text{Rule } R: \text{If } x_1 \text{ is } A_1 \text{ and } ... \text{ and } x_n \text{ is } A_n \text{ then Class } C, \quad (1)$$

where $\mathbf{A} = (A_1, ..., A_n)$ are antecedent fuzzy sets and C is a consequent class. We denote the rule R in (1) as $\mathbf{A} \Rightarrow C$. Since we usually have no *a priori* information about appropriate fuzzy sets for each attribute, we use various fuzzy sets to extract candidate fuzzy rules. In computational experiments, we use 14 different triangular fuzzy sets in Fig. 1.

The membership value of the pattern $\mathbf{x}$ to the antecedent part $\mathbf{A}$ is calculated with the product operator as:

$$\mu_{\mathbf{A}}(\mathbf{x}) = \mu_{A_1}(x_1) \cdot ... \cdot \mu_{A_n}(x_n). \quad (2)$$

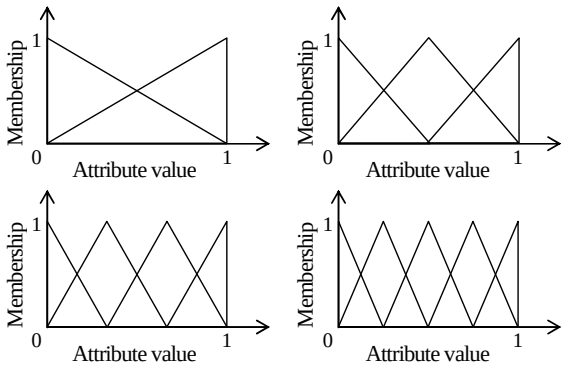


Fig. 1. Fuzzy sets used in the computational experiments.

In the field of data mining, two rule evaluation criteria called confidence and support are widely used to measure the goodness of rules.

Let us assume that we have m training patterns $\mathbf{x}_p$, (p = 1, 2, …, m). The fuzzy version of confidence is defined as follows [5]:

$$confidence(\mathbf{A} \Rightarrow C) = \frac{\sum_{\mathbf{x}_p \in C} \mu_{\mathbf{A}}(\mathbf{x}_p)}{\sum_{p=1}^{m} \mu_{\mathbf{A}}(\mathbf{x}_p)}. \quad (3)$$

In the same manner, support is defined as follows [5]:

$$support(\mathbf{A} \Rightarrow C) = \frac{\sum_{\mathbf{x}_p \in C} \mu_{\mathbf{A}}(\mathbf{x}_p)}{m}. \quad (4)$$

A rule evaluation criterion called coverage can be used instead of support. Coverage is defined as follows:

$$coverage(\mathbf{A} \Rightarrow C) = \frac{\sum_{\mathbf{x}_p \in C} \mu_{\mathbf{A}}(\mathbf{x}_p)}{sup(C)}, \quad (5)$$

where $sup(C)$ is consequent support, which is equal to the number of patterns whose class is C. This measure is often used for the problem of partial classification [6]. This problem is the search for the rules of a specified class. In partial classification, it is often convenient and intuitive to use coverage instead of support. Since the $sup(C)$ is constant in partial classification, the maximization of coverage is the same as that of support. We hereafter use coverage instead of support because the consequent class is fixed in our method.

We call the rules, which are Pareto-optimal in terms of the maximization of confidence and coverage, *Pareto-optimal rules*. Roughly speaking, a solution is Pareto optimal if it cannot result in further improvement of an objective without causing the degradation of another objective. It is shown in [2] that Pareto-optimal rules have the maximum value of other various rule evaluation criteria including gain, variance, chi-squared value, entropy gain, Gini, Laplace, lift, and conviction. Since the number of Pareto-optimal rules is conveniently small, Pareto-optimal rules can be discovered efficiently. In this paper, we discover Pareto-optimal rules using EMO algorithms.

III. PARETO-OPTIMAL RULE MINING WITH EMO ALGORITHMS

EMO algorithms are widely established and well developed for problems with multiple objectives. We apply EMO algorithms in the framework of genetics-based machine learning (GBML). GBML has two approaches: Michigan-style and Pittsburgh-style approach. In the Michigan-style approach, chromosomes are individual rules and a rule set is represented by the entire population. In the Pittsburgh-style approach, each chromosome represents a rule set (i.e., classifier). We adopt the former approach (i.e., Michigan-style approach) in our method where the antecedent part of a rule $\mathbf{A} = (A_1, ..., A_n)$ is encoded as a chromosome. The objectives of EMO algorithms are the maximization of confidence and coverage.

$$\text{maximize} \begin{cases} confidence(\mathbf{A} \Rightarrow C) \\ coverage(\mathbf{A} \Rightarrow C) \end{cases} \quad (6)$$

Among EMO algorithms, we use MOEA/D and NSGA-II. We briefly explain basic characteristics of these two algorithms in the following subsections.

1. MOEA/D

MOEA/D is an EMO algorithm proposed by Zhang [3]. Let P and EP be a current population and an external population, respectively. In MOEA/D, each individual has T neighbors to which one of the uniformly generated weight vectors is assigned. In MOEA/D, genetic operations for each individual are locally performed among its neighbors. The outline of MOEA/D can be written as follows:

```
1: Initialize P
2: while a termination condition is not satisfied do
3:     foreach individual x in P
4:         Select m, n from the T neighbors of x
5:         Generate y from m, n by genetic operations
6:         Update the neighbors of x with y
7:         Update EP
```

8: **end foreach**
9: **end while**
10: **return** *EP*

First an initial population is generated in line 1. In line 1, a set of uniformly distributed weight vectors is also generated. In line 4, a couple of parents **m** and **n** are randomly selected from the *T* neighbors of **x**. Then in line 5, an offspring **y** is generated from **m** and **n** by using genetic operators. In line 6, if **y** is better than some neighbors of **x**, they are replaced with **y**. The comparison is made by using scalarizing function (i.e., the weighted sum or the weighted Tchebycheff, we use the latter in our experiments). In line 7, all individuals dominated by **y** are removed from *EP*. These procedures are applied for each individual until a termination condition is satisfied.

2. NSGA-II

NSGA-II is an EMO algorithm proposed by Deb [4]. The outline of NSGA-II can be written as follows:

1: *P* = Initialize (*P*)
2: **while** a termination condition is not satisfied **do**
3: *P'* = Selection (*P*)
4: *P''* = Genetic Operations (*P'*)
5: *P* = Replace ($P \bigcup P''$)
6: **end while**
7: **return** non-dominated solutions (*P*)

First an initial population is generated in line 1. In line 3, parent individuals (i.e., *P'*) are selected from the current population *P*. The standard binary tournament selection is used to choose a pair of parent individuals. In line 4, an offspring population *P''* is generated from the parent population *P'* by genetic operations such as crossover and mutation. In line 5, the best individuals are chosen from the merged population ($P \bigcup P''$) to generate the next population *P*.

IV. COMPUTATIONAL EXPERIMENTS

Experiments were conducted by using the following seven datasets from the UCI machine learning repository: Breast W, Diabetes, Glass, Heart C, Iris, Sonar, and Wine. The parameter specifications in EMO algorithms are as follows:

Population size: 200 individuals,
Crossover probability: 1.0 (uniform crossover),
Mutation probability: $1/n$,
Termination condition: 1000 generations.

We compare the performance of our method with that of conventional data mining algorithms including the Apriori algorithm and the simple enumeration (SE). For Apriori and SE, we set minimum confidence and coverage as 0.6 and 0.01, respectively. We also set the maximum rule length as two for Sonar and three for the other datasets to alleviate a computational load.

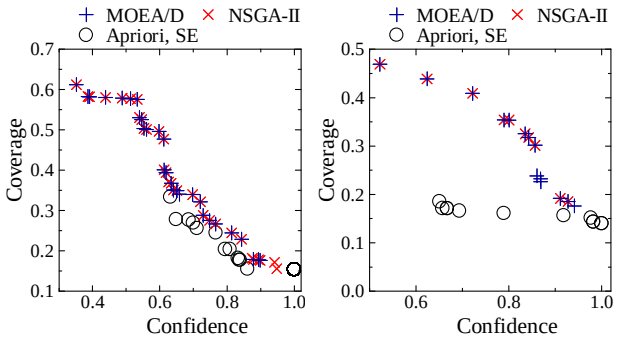
Table 1 shows the average CPU time of 100 runs. The experiment was conducted on Dual Core Xeon 3.6GHz, 4GB RAM workstations. The best result for each class is highlighted in bold. The values in parentheses will be mentioned later. From Table 1, we can see that MOEA/D is faster than the other algorithms for many datasets. While Apriori outperforms MOEA/D for some datasets, it is significantly slow for others.

Figure 2 shows generated fuzzy rules in confidence-coverage space for Class 4 and 5 of Glass. MOEA/D and NSGA-II clearly outperforms Apriori and SE. It should be noted that MOEA/D and NSGA-II can generate fuzzy rules that have four or more antecedent fuzzy sets. This allows MOEA/D and NSGA-II to generate fuzzy rules with high confidence and coverage.

One underlying problem with MOEA/D and NSGA-II is that they cannot discover useful rules when a dataset has a number of attributes. In the case of Sonar which has 60 attributes, all discovered rules by MOEA/D and NSGA-II were zero-confidence and zero-coverage. As the number of attributes increases, the search spaces of MOEA/D and NSGA-II exponentially increase. The size of the search space for Sonar is 15^{60},

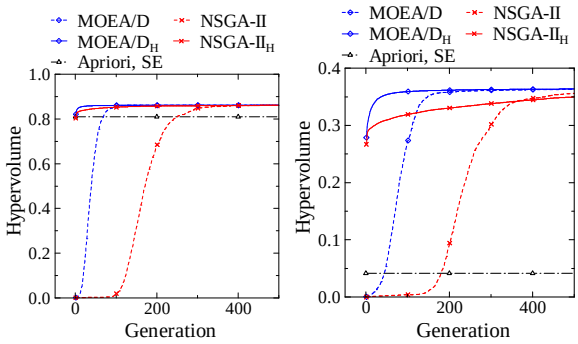
Table 1. CPU time (sec.)

Dataset	Class	MOEA/D		NSGA-II		Apriori	SE
Breast W	1	21.1	(21.1)	57.1	(57.6)	**8.1**	48.9
	2	**20.9**	(20.9)	53.6	(53.8)	63.4	49.0
Diabetes	1	21.8	(22.0)	60.3	(61.0)	**13.5**	33.5
	2	21.7	(21.9)	58.0	(58.5)	**15.0**	33.3
Glass	1	**7.4**	(7.4)	29.5	(30.2)	**7.4**	15.3
	2	**7.2**	(7.2)	29.6	(30.5)	13.4	15.4
	3	7.5	(7.6)	29.7	(30.5)	**7.4**	15.3
	4	**7.1**	(7.1)	28.3	(29.6)	14.8	15.4
	5	7.2	(7.2)	27.7	(28.7)	**6.1**	15.3
	6	**7.2**	(7.2)	28.4	(29.6)	13.3	15.4
Heart C	1	**12.2**	(12.1)	38.6	(39.1)	105.0	98.8
	2	**12.7**	(12.5)	38.9	(39.6)	105.9	97.4
	3	**12.6**	(12.5)	39.2	(39.5)	114.7	97.4
	4	**12.6**	(12.5)	39.1	(39.4)	118.5	97.4
	5	**13.0**	(12.9)	38.4	(39.9)	93.2	97.8
Iris	1	3.4	(3.5)	20.1	(20.1)	**0.1**	0.3
	2	3.4	(3.5)	20.2	(20.3)	**0.2**	0.3
	3	3.4	(3.5)	20.3	(20.3)	**0.2**	0.3
Sonar	1	**34.7**	(30.9)	76.4	(81.6)	201.5	82.3
	2	**34.7**	(30.9)	74.9	(81.6)	221.7	82.5
Wine	1	**7.5**	(7.6)	28.2	(28.4)	104.2	58.9
	2	**7.5**	(7.5)	28.3	(28.5)	164.2	58.9
	3	**7.5**	(7.6)	28.3	(28.8)	112.5	59.0



(a) Class 4 (b) Class 5

Fig. 2. Generated fuzzy rules (Glass).



(a) Class 1 (b) Class 2

Fig. 3. Hypervolume measure (Heart C).

which makes the search for Pareto-optimal rules impossible. One possible remedy for this problem is to use problem-specific initialization for improving the search ability.

V. HEURISTIC INITIALIZATION

In the design of fuzzy classifiers, the search ability of fuzzy classifiers can be improved by directly generating initial fuzzy rules from training patterns. Let N_{rule} be the population size. First we randomly select N_{rule} patterns. Then we choose the fuzzy set which has a high membership value. We probabilistically choose an antecedent fuzzy set from the 14 candidates B_k (k = 1, 2, …, 14) in Fig. 1 where each candidate B_k has the following selection probability for the attribute value x_i:

$$P(B_k) = \frac{\mu_{B_j}(x_i)}{\sum_{j=1}^{14} \mu_{B_j}(x_i)}. \quad (7)$$

We conducted the same experiment using heuristic initialization. By using heuristic initialization, MOEA/D and NSGA-II could discover as good rules as Apriori and SE for Sonar. To show the effects of heuristic initialization, we examined the hypervolume measure during evolution. The hypervolume measure, which indicates the size of the portion of objective space that is dominated by the solutions, is often used to assess the performance of EMO algorithms. Figure 3 shows the average value of the hypervolume measure at each generation for Heart C. In Fig. 3, MOEA/D_H and NSGA-II_H shows the results using heuristic initialization. We can see that MOEA/D_H and NSGA-II_H evolves faster than the original ones at the early stage.

The effect of using heuristic initialization on CPU time is not large. The values in parentheses in Table 1 show the CPU time with heuristic initialization. Heuristic initialization made 0.2-second delay for MOEA/D and 6.7-second delay for NSGA-II at most. In some cases (e.g., Heart C and Sonar), MOEA/D with heuristic initialization was faster than the original one.

VI. CONCLUSION

In this paper, we applied MOEA/D and NSGA-II to discover Pareto-optimal fuzzy rules. Through computational experiments, we showed that MOEA/D is faster than Apriori and SE for a number of datasets. Futhermore, the generated rules were better than Apriori and SE. We also showed that the search ability of MOEA/D and NSGA-II was improved by using heuristic initialization without increasing much computation time.

REFERENCES

[1] Agrawal R, Mannila H, Srikant H, Toivonen R, Verkamo AI (1996), Fast discovery of association rules, Advances in Knowledge Discovery and Data Mining: 307–328

[2] Bayardo RJ Jr., Agrawal R (1999), Mining the most interesting rules, Proc. of 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 145–153

[3] Zhang Q, Li H (2007), MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. on Evolutionary Computation, 11 (6): 712–731

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

[5] Hong TP (2001), Tradeoff between computation time and number of rules for fuzzy mining from quantitative data, International Journal of Uncertainty, 9 (6): 587–604

[6] Ali S, Manganaris S, Srikant R (1997), Partial classification using association rules, Proc. of the Third International Conference on Knowledge Discovery and Data Mining: 115–118