Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 510167, 11 pages
http://dx.doi.org/10.1155/2013/510167

Research Article

Ant Colony Optimization with Three Stages for Independent Test Cost Attribute Reduction

Zilong Xu, Hong Zhao, Fan Min, and William Zhu

Lab of Granular Computing, Minnan Normal University, Zhangzhou 363000, China

Correspondence should be addressed to Fan Min; minfanphd@163.com

Received 3 January 2013; Revised 1 May 2013; Accepted 23 May 2013

Academic Editor: Lu Zhen

Minimal test cost attribute reduction is an important problem in cost-sensitive learning. Recently, heuristic algorithms including the information gain-based algorithm and the genetic algorithm have been designed for this problem. However, in many cases these algorithms cannot find the optimal solution. In this paper, we develop an ant colony optimization algorithm to tackle this problem. The attribute set is represented as a graph with each vertex corresponding to an attribute and weight of each edge to pheromone. Our algorithm contains three stages, namely, the addition stage, the deletion stage, and the filtration stage. In the addition stage, each ant starts from the initial position and traverses edges probabilistically until the stopping criterion is satisfied. The pheromone of the traveled path is also updated in this process. In the deletion stage, each ant deletes redundant attributes. Two strategies, called the centralized deletion strategy and the distributed deletion strategy, are proposed. Finally, the ant with minimal test cost is selected to construct the reduct in the filtration stage. Experimental results on UCI datasets indicate that the algorithm is significantly better than the information gain-based one. It also outperforms the genetic algorithm on medium-sized dataset Mushroom.

1. Introduction

Cost-sensitive attribute reduction has gained much research interest in rough sets. People have considered three types of costs, namely, test cost [1], misclassification cost [2], and delay cost [3, 4]. Test cost is the money and/or time that is required to obtain attribute values. In real applications, only part of tests is needed to maintain enough information for classification. We would like to choose an attribute reduct [5] that minimizes the total test cost. This issue is called the minimal test cost reduct (MTR) problem [1].

We explain the MTR problem through a simple example. There is a medical decision system with five attributes, Mcv, Alkphos, Sgpt, Sgot, and Gammagt. The test costs of these five attributes are \$55, \$10, \$15, \$20, and \$25, respectively. The decision system has three reducts, {Mcv, Alkphos}, {Alkphos, Sgpt, Gammagt}, and {Alkphos, Sgot, Gammagt}. Alkphos is the core attribute. The minimal reduct is {Mcv, Alkphos} including only two attributes. The minimal test cost reduct is {Alkphos, Sgpt, Gammagt} with only \$50.

The minimal test cost attribute reduction (MTR) is proposed for saving resources, and the problem is meaningful in applications. The MTR problem is more general than the minimal reduct problem which is NP-hard; hence the MTR problem is at least NP-hard. Consequently, heuristic algorithms are needed to deal with such problem. Heuristic algorithms including information gain-based λ-weighted reduction algorithm [1] and genetic algorithm [6] have been designed to deal with MTR problem. Unfortunately, they often do not find the optimal solution for medium-sized datasets. Although the competition approach [1] helps to improve the performance through constructing a population of reducts, the results are still unsatisfactory. There is still room to obtain more sophisticated algorithms through other techniques.

In this paper, we develop an algorithm based on ant colony optimization for the MTR problem. The attribute set is represented as a complete graph with each vertex corresponding to one attribute. Then batches of ants are generated for attribute subset selection. Our algorithm contains three stages, namely, the addition stage, the deletion stage, and the filtration stage. First, in the addition stage, core vertexes are compressed into one vertex as the initial position of all ants. From the initial position, each ant selects next

vertex according to test costs of each adjacent vertex and pheromone of each adjacent edge. If the attribute vertexes an ant has traveled satisfy the positive region condition, the ant stops, otherwise continues to add attributes. Second, in the deletion stage, redundant attributes are deleted from the obtained attribute subsets, and the pheromone of each edge is updated. Two strategies including centralized deletion and distributed deletion are designed for this stage. Third, the ant with the least test cost is selected, and the attribute subset corresponding to its path is output as the result in the filtration stage.

To evaluate the performance of algorithms, we adopt three measures, namely, finding optimal factor (FOF), maximal exceeding factor (MEF), and average exceeding factor (AEF) [1]. Experimental results indicate that our algorithm outperforms the information gain-based algorithm in most datasets with different test cost distributions. It can obtain better results than the genetic algorithm except some small datasets. One possible reason is that the ant colony optimization algorithm produces more diverse solutions than the existing ones. The distributed deletion strategy is superior to the centralized one on medium-sized dataset Mushroom.

The rest of the paper is organized as follows. Section 2 reviews the basic concepts in rough sets and decision system. Section 3 proposes the ant colony optimization to tackle the minimal test cost reduction problem. In Section 4, we present our experiment schemes and show the results. We also give a simple analysis of our experimental results. Finally, Section 5 presents the conclusion.

2. Preliminaries

This section reviews basic knowledge: test-cost-independent decision systems, relative reduct, minimal test cost reduct problem, genetic algorithm, and ant colony optimization.

2.1. Test-Cost-Independent Decision Systems. Most supervised learning approaches are based on decision systems. A decision system is often denoted by $S = (U, C, D, \{V_a \mid a \in C \cup D\}, \{I_a \mid a \in C \cup D\})$, where U is a finite set of objects called the universe, C is the set of conditional attributes, also called the set of tests, D is the set of decision attributes, also called the decision, V_a is the set of values for each $a \in C \cup D$, and $I_a : U \to V_a$ is an information function for each $a \in C \cup D$. We often denote $\{V_a \mid a \in C \cup D\}$ and $\{I_a \mid a \in C \cup D\}$ by V and I, respectively. A decision system is often represented by a decision table, as shown in Table 1.

We consider the simplest case though most widely used type of cost-sensitive decision systems as follows.

Definition 1 (see [7]). A test-cost-independent decision system (TCI-DS) S is the 6-tuple

$$S = (U, C, D, V, I, c), \tag{1}$$

where U, C, D, V, and I have the same meanings as in a decision system and $c : C \to \mathbb{R}^+ \cup \{0\}$ is the test cost function. Test costs are independent of one another; that is, $c(B) = \sum_{a \in B} c(a)$ for any $B \subseteq C$.

Table 1: An exemplary decision table.

Patient	Headache	Muscle	Temperature	Snivel	Flu
x_1	No	Yes	Normal	No	No
x_2	Yes	Yes	High	Yes	Yes
x_3	Yes	Yes	Very high	No	Yes
x_4	No	Yes	Normal	No	No
x_5	No	No	High	No	No
x_6	No	Yes	Very high	Yes	Yes

Table 2: The test cost vector.

a	Headache	Muscle pain	Temperature	Snivel
$c(a)$	50	25	40	20

We usually use a vector $c = [c(a_1), c(a_2), \ldots, c(a_{|C|})]$ to represent the cost function. An exemplary cost vector is shown in Table 2. In fact, cost-sensitive decision systems are more general than decision systems. If all elements in c are 0, a TCI-DS coincides with a DS. For simplicity, free tests are not considered in this work. This consideration is reasonable since we always need some cost to obtain data.

2.2. The Relative Reduct. The relative reduct is a crucial concept in rough sets, and there are many different definitions, such as positive region reducts [5], maximum distribution reducts [8], fuzzy reducts [9], and β-reduct [10]. These definitions are equivalent if the decision table is consistent. Furthermore, there are some extended concepts such as dynamic reducts [11], parallel reducts [12, 13], and M-relative reducts [14]. In some extensions of rough sets, that is, covering-based rough sets [15], there are other definitions of a reduct (see, e.g., [16–18]).

Most existing reduct problems aim at finding the minimal description of the data. And the objectives include finding attribute subsets with the minimal size [5, 19], the minimal space [20], or a covering with the minimal number of subsets [16]. Since the test cost issue is the focus of this paper, we are interested in reducts with the minimal test cost. This type of reducts is defined as follows.

Definition 2 (see [1]). Let S be a TCI-DS and $\mathrm{Red}(S)$ the set of all reducts of S. Any $R \in \mathrm{Red}(S)$, where $c(R) = \min\{c(R') \mid R' \in \mathrm{Red}(S)\}$, is called a minimal test cost reduct.

The set of all minimal test cost reducts is denoted by $\mathrm{MTR}(S)$. And the problem of constructing $\mathrm{MTR}(S)$ is called the *minimal test cost reduct* (MTR) problem. As indicated in [1], the time complexity of computing $\mathrm{MTR}(S)$ is the same as $\mathrm{Red}(S)$.

2.3. The Minimal Test Cost Reduct Problem. Attribute reduction is a key issue of the rough sets research. The classical [5], covering-based [16, 21], decision-theoretical [3], variable-precision [10], dominance-based [22], and neighborhood [23] rough sets models address the reduction problem from different perspectives. A number of definitions of relative

reducts exist [5, 19, 23, 24] for different rough sets models. This paper employs the definition based on the positive region.

2.4. The Genetic Algorithm. In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution [25]. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [25]. In [26], the genetic algorithm is employed to evolve the cost-sensitive decision trees. Recently, the genetic algorithm has been employed to tackle the minimal test cost reduction problem [6] and attribute reduction with test cost constraint [27].

2.5. The Ant Colony Optimization. Swarm intelligence is a relatively new approach to problem solving that takes inspiration from the social behaviors of insects and other animals [28]. The ant colony optimization (ACO) algorithm is a probabilistic technique for solving computational problems, which can be reduced to finding good paths through graphs. ACO algorithms are state-of-the-art for the sequential ordering problem [29], the vehicle routing problem with time window constraints [30], the quadratic assignment problem [31], the arc-weighted l-cardinality tree problem [32], and the shortest common supersequence problem [33]. In rough sets, the classical attribute reduction problem has been tackled by the ACO [34, 35].

2.6. Evaluation Measures. For evaluating the experiment results, we adopt evaluation measures proposed in [1]. The new algorithm is compared with the information gain-based heuristic algorithm on four UCI datasets [36].

We need a measure to evaluate the quality of one particular reduction. Since an algorithm can run on many datasets or one dataset with different test cost settings, we adopt three metrics from a statistical viewpoint. They are finding optimal factor (FOF), maximal exceeding factor (MEF), and average exceeding factor (AEF) [1].

2.6.1. Finding Optimal Factor. Let the number of experiments be K and the number of successful searches of an optimal reduct k. The finding optimal factor is defined as

$$\mathrm{op} = \frac{k}{K}. \tag{2}$$

2.6.2. Exceeding Factor. For a dataset with a particular test cost setting, let R' be an optimal reduct. The exceeding factor of a reduct R is

$$\mathrm{ef}(R) = \frac{c^*(R) - c^*(R')}{c^*(R')}. \tag{3}$$

The exceeding factor provides a quantitative metric to evaluate the performance of a reduct. It indicates the badness of a reduct when it is not optimal. Naturally, if R is an optimal reduct, the exceeding factor is 0. To demonstrate the performance of an algorithm, statistical metrics are needed. Let the number of experiments be K. In the ith experiment ($1 \le i \le K$), the reduct computed by the algorithm is denoted by R_i. The maximal exceeding factor is defined as

$$\max_{1 \le i \le K} \mathrm{ef}(R_i). \tag{4}$$

This shows the worst case of the algorithm given some dataset. Although it relates to the performance of one particular reduct, it should be viewed as a statistical rather than an individual metric. The average exceeding factor is defined as

$$\sum_{i=1}^{K} \frac{\mathrm{ef}(R_i)}{K}. \tag{5}$$

Since it is averaged on K different test-cost-sensitive decision systems, it shows the overall performance of the algorithm solely from a statistical perspective.

3. The Algorithm

In this section, one hand, we revise the classical ant colony optimization. On the other hand, we present our ant colony optimization with different techniques to tackle the minimal test cost reduct problem.

3.1. The Problem Representation and Algorithm Framework. In order to apply ant colony optimization to tackle the minimal test cost reduction problem, we adopt the following model.

Graph. The decision system is represented as a graph.

Vertex. An attribute is represented as a vertex. Each feature vertex has information about test cost.

Edge. There is an edge between any two vertexes. Each edge has information on pheromone density.

Adjacent Matrix. The values of the matrix represent pheromone density of each edge.

Because our objective is attribute reduction, not classification which produces rule sets, we do not adopt the tree structure such as the ant colony decision tree. In this section, we employ the first, simplified ant colony optimization—AS. We represent the general algorithm framework as follows.

Stage 1 (addition stage). Compute the core of the dataset. K batches of ants are created with each batch containing k ants, therefore giving a total of $K \times k$ ants. Each ant takes the core as the starting position. From the initial positions, each ant

traverses edges probabilistically until the stopping criterion is satisfied.

Stage 2 (deletion stage). Delete redundant attributes and update the pheromone of the traveled path.

Stage 3 (filtration stage). Gather the attribute subsets obtained by ants, and compute their test cost. Choose the attribute subset with minimal test cost, and output it as the result.

Throughout the paper, the stopping criterion is the positive region condition. The selecting probability depends on the test cost of each adjacent attribute vertex and the pheromone of each adjacent edge. The probabilistic transition rule is

$$p_{ij}^{k} = \frac{[\tau_j]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_l]^{\alpha}[\eta_{il}]^{\beta}} \quad j \in \mathcal{N}_i^k, \tag{6}$$

where k is the number of the ant, $\tau_j = 1000/tc(a_j)$ as the heuristic information, η_{ij} means the pheromone density of the edge (i, j), α is the exponent of τ_j, β is the exponent of η_{ij}, and $\mathcal{N}_i^k$ is the set of all unvisited adjacent vertexes of the attribute i.

The difference between the centralized deletion strategy and the distributed strategy is the time to delete attributes. The deletion of redundant attributes follows the finish of the journey of all ants in the case of the centralized deletion strategy. When using the distributed deletion strategy, the algorithm deletes redundant attributes after each ant travels a path, and then it adjusts the path. In this situation, the pheromone of edges adjoining to redundant attribute vertexes will not be updated. The adjusting method is illustrated in Figure 2. Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results that is by running simulations many times over in order to calculate those same probabilities heuristically just like actually playing and recording your results in a real casino situation: hence the name. We employ Monte Carlo method to simulate the process that an artificial ant selects the next attribute vertex probabilistically.

3.2. The ACO with Distributed Deletion. We represent the substantial algorithm of the ant colony optimization with distributed deletion strategy as Algorithm 1. The algorithm with centralized deletion strategy is similar.

We explain the algorithm in details as follows. Lines 1 through 8 correspond to Stage 1 in the algorithm framework. In lines 2 through 4, K batches of ants are created, with each batch containing k ants, therefore giving a total of $K \times k$ ants. Each ant takes the core, as indicated by line 5. Lines 9 through 21 represent Stage 2. Select vertexes until all ants in the batch meet the positive region condition. Whenever an ant stops, it removes the redundant attributes and adjusts the traveled path. In line 18, the ants release pheromone to the adjusted paths. After all ants finish their journey, select the best reduct as lines 22 through 27 have shown. Obviously, these lines relate to Stage 3. If we adopt the centralized deletion strategy, the algorithm deletes redundant attributes after all ants finish their journey.

3.3. A Running Example. The TCI-DS is given by Tables 1 and 2. We illustrate the process of an ant in Figure 1, where attributes `Headache`, `Muscle`, `Temperature`, and `Snivel` are represented by 0, 1, 2, and 3, respectively.

After an ant travels a path, the redundant attributes are removed. After deletion, the algorithm reconstructs the path. This adjusting process is represented in Figure 2.

Each ant is placed on an attribute randomly. For simplicity, assume $k = 1$; namely, there is one artificial ant in one batch; Consider Core(S) = $\emptyset$; hence any ant initially takes no vertex.

Stage 1 (addition stage)

Stage 1.1 (artificial ants start). Assume ant[0] starts from the attribute[0].

Stage 1.2 (artificial ants select attributes)

Iteration 1. Ant[0] is at the attribute Headache. Ant[0] has three attributes to select, and they are `Muscle pain`, `Temperature,` and `Snivel`.

According to (6), the denominator of selecting probability is $(1000/25)^2 \cdot 1^2 + (1000/40)^2 \cdot 1^2 + (1000/20)^2 \cdot 1^2 = 1600 + 625 + 2500 = 4725$.

We compute the probability of three selections:

$$\begin{aligned} p_{01} &= 1600/4725 = 0.339, \\ p_{02} &= \frac{625}{4725} = 0.132, \\ p_{03} &= \frac{2500}{4725} = 0.529. \end{aligned} \tag{7}$$

Because $p_{03} > p_{01} > p_{02}$, the ant will select this as next feature with high probability. That is to say, the ant does not select the attribute necessarily. In this case, we assume the ant choose the feature `Snivel`. At the same time, the ant will add the selected attribute to the visited attribute set. We find the selected attribute set the ant has visited does not satisfy the positive region condition, so it will continue to select next attribute.

Iteration 2. Ant[0] is at the attribute Snivel currently.

The candidate features are `Muscle pain` and `Temperature`. According to the probability function, the denominator of selecting probability is $(1000/25)^2 \cdot 1^2 + (1000/40)^2 \cdot 1^2 = 1600 + 625 = 2225$.

Consider

$$\begin{aligned} p_{31} &= \frac{1600}{2225} = 0.719, \\ p_{32} &= \frac{625}{2225} = 0.281. \end{aligned} \tag{8}$$

More over $p_{31} > p_{32}$; then the artificial ant[0] almost selects the attribute Muscle pain.

Input: $S = (U, C, D, V, I, c)$
Output: M, a reduct of S
Method: acomtr
(1) $M = \emptyset$;//the best reduct obtained by ants
(2) **for** ($i = 0$; $i < K$; i ++) **do**
(3) **for** ($j = 0$; $j < k$; j + +) **do**
(4) create ant[$i \times k + j$];
(5) $B_{i \times k + j}$ = core(S);//takes the set of core attributes
(6) **end for**
//Each ant selects attributes
(7) **for** (each $m \in [i \times k, (i + 1) \times k]$ where ant[m] has not stopped) **do**
(8) MonteCarlo(tc, ph);//select the next vertex
(9) **if** ($POS_{B_m}(D) == POS_C(D)$) **then**
(10) ant[m] stops; //Delete redundant vertexes.
(11) mtc = 10000;//minimal test cost
(12) **for** ($j = |B_i| - 1$; $j >= |\text{Core}(S)|$; j--) **do**
(13) **if** ($POS_{(B_i - \{a_j\})}(D) == POS_{B_i}(D)$) **then**
(14) $B_i = B_i - \{a_j\}$;
(15) **end if**
(16) **end for**
(17) Adjust the traveled path
(18) Update the pheromone of edges ant[m] traveled
(19) **end if**
(20) **end for**
(21) **end for**
//Choose the minimal test cost reduction from the last n ants
(22) **for** ($i = K \times k - n$; $i <= K \times k - 1$; i ++) **do**
(23) **if** $tc(B_i) \le mtc$ **then**
(24) $mtc = tc(B_i)$;
(25) $M = B_i$;
(26) **end if**
(27) **end for**
(28) return M;

Algorithm 1: ACO with distributed deletion to MTR problem.

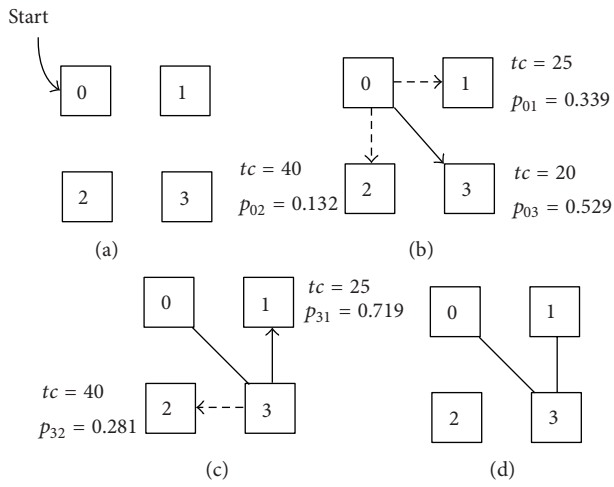

Figure 1: (a) Start from the attribute Headache. (b) Select second attribute. (c) Select third attribute. (d) Stop.

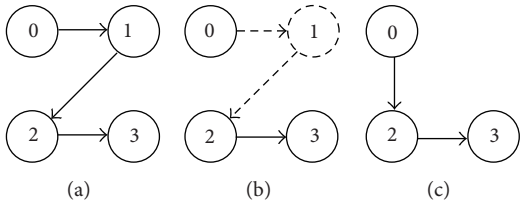

Figure 2: (a) The path which an ant traveled, (b) Delete the redundant attribute, (c) Reconstruct the path.

Now, the attribute set obtained by ant[0] has satisfied the positive region constraint. The ant[0] stops working and updates pheromone information.

Stage 1.3 (pheromone updating). After an artificial ant stops working, it will update the pheromone density of edges traveled by itself. In our algorithm, when one path is crossed by an ant, the pheromone diffusion will increase by one. Of course, the rule of pheromone updating may be designed in other methods. We adopt the way in this paper.

In this example, the sequence attribute selection of ant[0] is `Headache`, `Snivel,` and `Muscle pain`. The pheromone of edge (0, 3) and edge (3, 1) increases by one.

Each artificial ant runs in above three stages.

Stage 2 (deletion stage). After all ants have stopped working, the algorithm will delete redundant attributes from the selected attribute sets of last few ants using positive region. If an attribute does not contribute to the positive region, we remove it.

Stage 3 (filtration stage). After deletion, we choose the reduct obtained from last n ants with minimal test cost as the result.

In this example, we adopt the centralized deletion strategy. Figure 2 illustrates the distributed deletion strategy in an iteration. In the figure, the ant travels the paths 0, 1, 2, and 3. Suppose the attribute 1 is a redundant attribute, we remove it from the attribute subset and adjust the path. The adjusting process is shown in Figure 2(c).

4. Experiments

In this section, we try to answer the following questions by experimentation.

(1) How does the number of ants in the ant colony influence the result?

(2) How does the strategy of deleting redundant attributes influence the quality of the result?

(3) Does our ant colony optimization algorithm outperform the existing one?

The UCI datasets we used are Zoo, Voting, Tic-tac-toe, and Mushroom. Since these datasets have no test cost settings, for statistical purposes, we apply three common distributions to generate random test cost. The three distributions are uniform distribution, normal distribution, and Pareto distribution. In this paper, the test cost is a random integer ranging from 1 to 100. The exponent β in (6) is set to be 2, since under many circumstances this value is a good setup [28].

We do not design the parameter learning mechanism. The employed competition approach has the selection mechanism of the parameter. Different applications use the same range, and users do not specify the value of the parameter. The strategy is more straightforward than many other parameter tuning strategies. However, we may design other strategies to save time. Finding optimal factor, minimal exceeding factor, and average exceeding factor are employed to measure the effectiveness of the algorithms. We run each algorithm on datasets with 1000 times. The results have statistical characteristics.

4.1. The Influence of Ant Counts on Experiment Results. In order to find the relationship between the number of ants and the quality of the result, we conduct this experiment. We run our algorithm with 100 ants, 150 ants, 200 ants . . . 400 ants. In this section, the number of experiments is set to be 100. The exponent α in (6) is set to be 2. Results are shown in Tables 3 and 4.

4.2. Comparison with Existing Heuristic Algorithms. According to the result of the above experiments, we find 100 is the optimal setting of the number of the ants. We conduct an empirical study to examine the effect of our algorithm. To improve the performance, we use the competition approach [1] to enhance our algorithm. The exponent α in (6) is set as integers ranging from 1 to 4. We illustrate the results among the information gain-based algorithm, GA-1, GA-2, ACO with centralized deletion and the ACO with distributed deletion, in Table 5. For clarity, the results on dataset Mushroom are shown in Figure 5.

Our algorithm is tested on the UCI datasets 1000 times, respectively. The new algorithm with different techniques is compared with the information gain-based algorithm [1] and the genetic algorithm [6]. Experimental results are listed in Figures 3, 4, and 5 and Tables 3, 4, and 5.

4.3. Experimental Results. Now we can answer the questions proposed at the beginning of this section.

(1) Experiment results indicate that the effect of the algorithm becomes worse with the increment of the ant counts. We find that 100 is the optimal setting of the number of ants. So, we run our algorithm with 100 ants to compare with the existing heuristic algorithms.

(2) From Figures 3 and 4, we infer that the distributed deletion outperforms the centralized one on the medium-sized dataset Mushroom without the competition approach. When we use the competition approach, the centralized deletion strategy and the distributed strategy produce the similar quality of results shown in Table 5 and Figure 5.

Table 3: The finding optimal factor of ACO with different numbers of ants without competition method using centralized deletion.

Dataset	Distribution	Number of ants						
		100	150	200	250	300	350	400
Iris	Uniform	0.66	0.70	0.73	0.68	0.62	0.69	0.61
	Normal	0.53	0.45	0.44	0.43	0.37	0.40	0.43
	Pareto	0.95	0.93	0.94	0.96	0.87	0.93	0.94
Zoo	Uniform	0.75	0.67	0.70	0.68	0.70	0.68	0.63
	Normal	0.66	0.48	0.38	0.29	0.30	0.33	0.34
	Pareto	0.99	1.00	0.96	0.94	0.97	0.95	0.84
Voting	Uniform	0.90	0.94	0.84	0.80	0.86	0.81	0.83
	Normal	1.00	0.94	0.91	0.95	0.95	0.90	0.93
	Pareto	0.98	0.99	0.98	0.99	0.99	0.99	0.94
Tic-tac-toe	Uniform	0.96	0.91	0.89	0.78	0.68	0.69	0.68
	Normal	1.00	0.96	0.96	0.96	0.89	0.86	0.83
	Pareto	1.00	1.00	1.00	0.99	0.99	0.99	1.00
Mushroom	Uniform	0.75	0.72	0.69	0.64	0.59	0.60	0.49
	Normal	0.61	0.55	0.52	0.51	0.39	0.41	0.31
	Pareto	0.97	0.97	0.96	0.95	0.95	0.96	0.94

Table 4: The finding optimal factor of ACO with different numbers of ants without competition method using distributed deletion.

Dataset	Distribution	Number of ants						
		100	150	200	250	300	350	400
Iris	Uniform	0.72	0.70	0.68	0.67	0.72	0.63	0.72
	Normal	0.42	0.40	0.37	0.40	0.48	0.41	0.42
	Pareto	0.90	0.95	0.94	0.93	0.92	0.91	0.97
Zoo	Uniform	0.72	0.69	0.69	0.66	0.63	0.64	0.55
	Normal	0.59	0.50	0.39	0.33	0.33	0.36	0.27
	Pareto	0.99	0.95	0.93	0.93	0.97	0.91	0.94
Voting	Uniform	0.93	0.84	0.87	0.86	0.81	0.79	0.83
	Normal	1.00	1.00	0.97	0.98	0.96	0.96	0.97
	Pareto	1.00	0.98	0.99	0.97	0.97	0.97	0.98
Tic-tac-toe	Uniform	0.90	0.78	0.74	0.77	0.71	0.74	0.66
	Normal	0.99	0.99	0.92	0.90	0.90	0.91	0.77
	Pareto	1.00	1.00	1.00	0.99	1.00	1.00	0.99
Mushroom	Uniform	0.80	0.71	0.70	0.62	0.53	0.59	0.58
	Normal	0.62	0.54	0.53	0.34	0.33	0.27	0.29
	Pareto	0.99	0.99	0.96	0.95	0.95	0.94	0.94

Table 5: Results of λ-information gain algorithm and the test cost based ant colony optimization with centralized deletion and distributed deletion.

Dataset	Distribution	Information gain			GA-1			GA-2			ACO centralized			ACO distributed		
		FOF	MEF	AEF	FOF	MEF	AEF	FOF	MEF	AEF	FOF	MEF	AEF	FOF	MEF	AEF
Zoo	Uniform	0.915	0.193	0.004	0.966	0.118	**0.001**	0.947	0.224	0.002	0.973	**0.177**	**0.001**	**0.975**	0.216	**0.001**
	Normal	0.833	**0.028**	**0.001**	**0.970**	0.109	**0.001**	0.924	0.101	0.002	0.849	0.040	0.002	0.844	0.048	0.002
	Pareto	0.999	0.167	**0.000**	0.998	0.167	**0.000**	0.993	0.200	0.001	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
Voting	Uniform	0.998	0.053	**0.000**	**1.000**	**0.000**	**0.000**	0.999	0.009	**0.000**	0.997	0.065	**0.000**	**1.000**	**0.000**	**0.000**
	Normal	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
	Pareto	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	0.995	0.067	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
Tic-tac-toe	Uniform	0.877	0.055	0.002	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
	Normal	0.408	0.018	0.003	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
	Pareto	0.986	0.125	0.002	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**
Mushroom	Uniform	0.762	0.523	0.023	0.448	1.000	0.067	0.766	0.543	0.019	**0.948**	**0.085**	**0.002**	0.946	0.144	**0.002**
	Normal	0.176	0.306	0.010	0.321	0.401	0.063	0.551	0.340	0.022	0.958	**0.020**	**0.000**	**0.966**	**0.020**	**0.000**
	Pareto	0.733	0.500	0.064	0.935	0.400	0.015	0.949	0.500	0.012	**1.000**	**0.000**	**0.000**	**1.000**	**0.000**	**0.000**

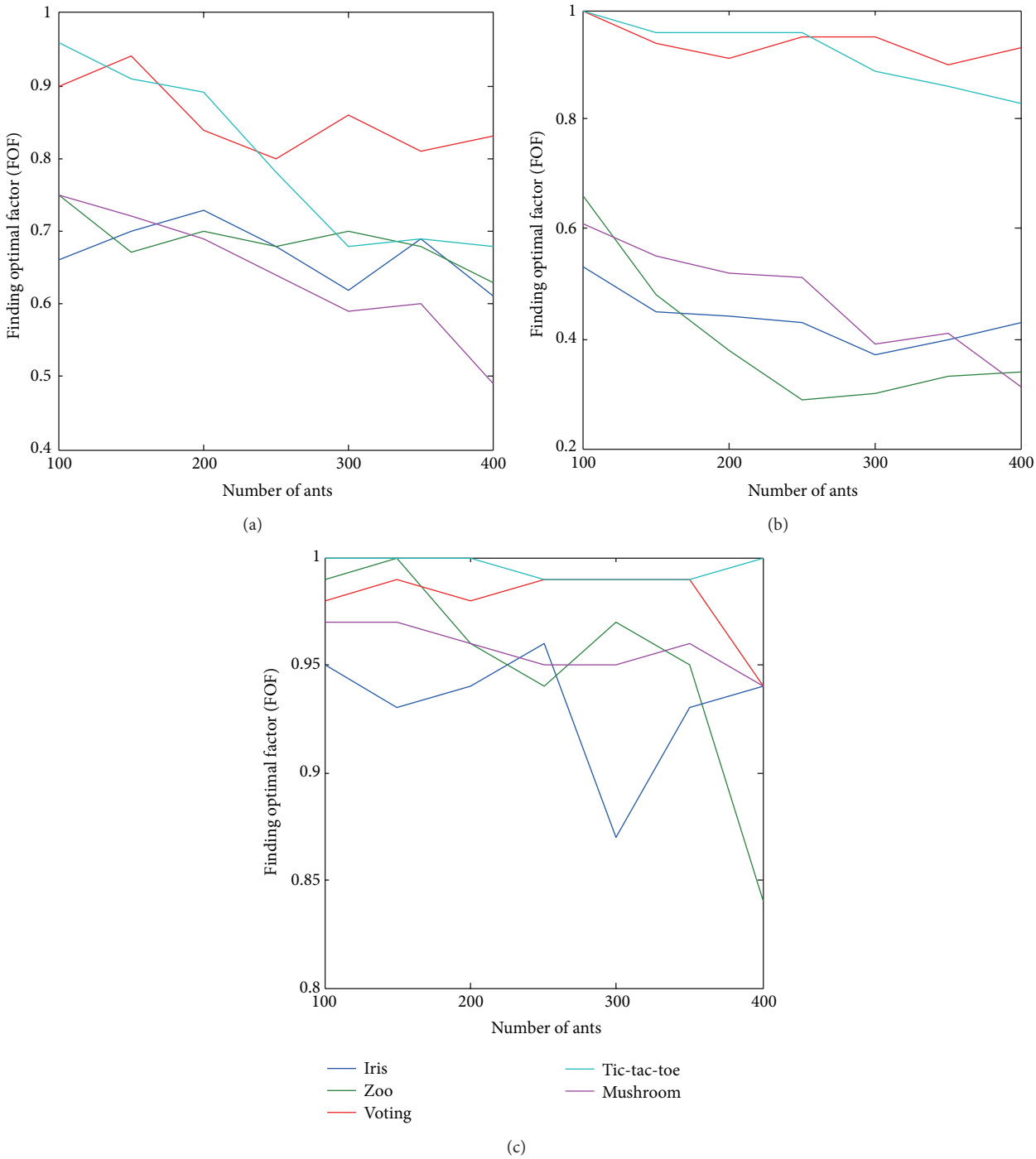

Figure 3: ACO with centralized deletion on different distributions: (a) uniform; (b) normal; (c) pareto.

(3) The ant colony optimization provides better performance than the information gain one and the genetic one. Our algorithm overcomes the shortcoming of the genetic algorithm which performs badly on the larger dataset. On medium-sized dataset Mushroom, the ACO outperforms the genetic one significantly. For example, when the test cost distribution is normal, the FOF of the ant colony optimization using centralized deletion and distributed deletion is 95.8%, 96.6%, respectively, outperforming the information gain one, GA-1, and GA-2 with 17.6%, 32.1%, and 52.1%, respectively. Figure 5 gives a more intuitive understanding of results on the dataset Mushroom. One possible reason is that the ant colony optimization algorithm produces more diverse solutions than the existing one.

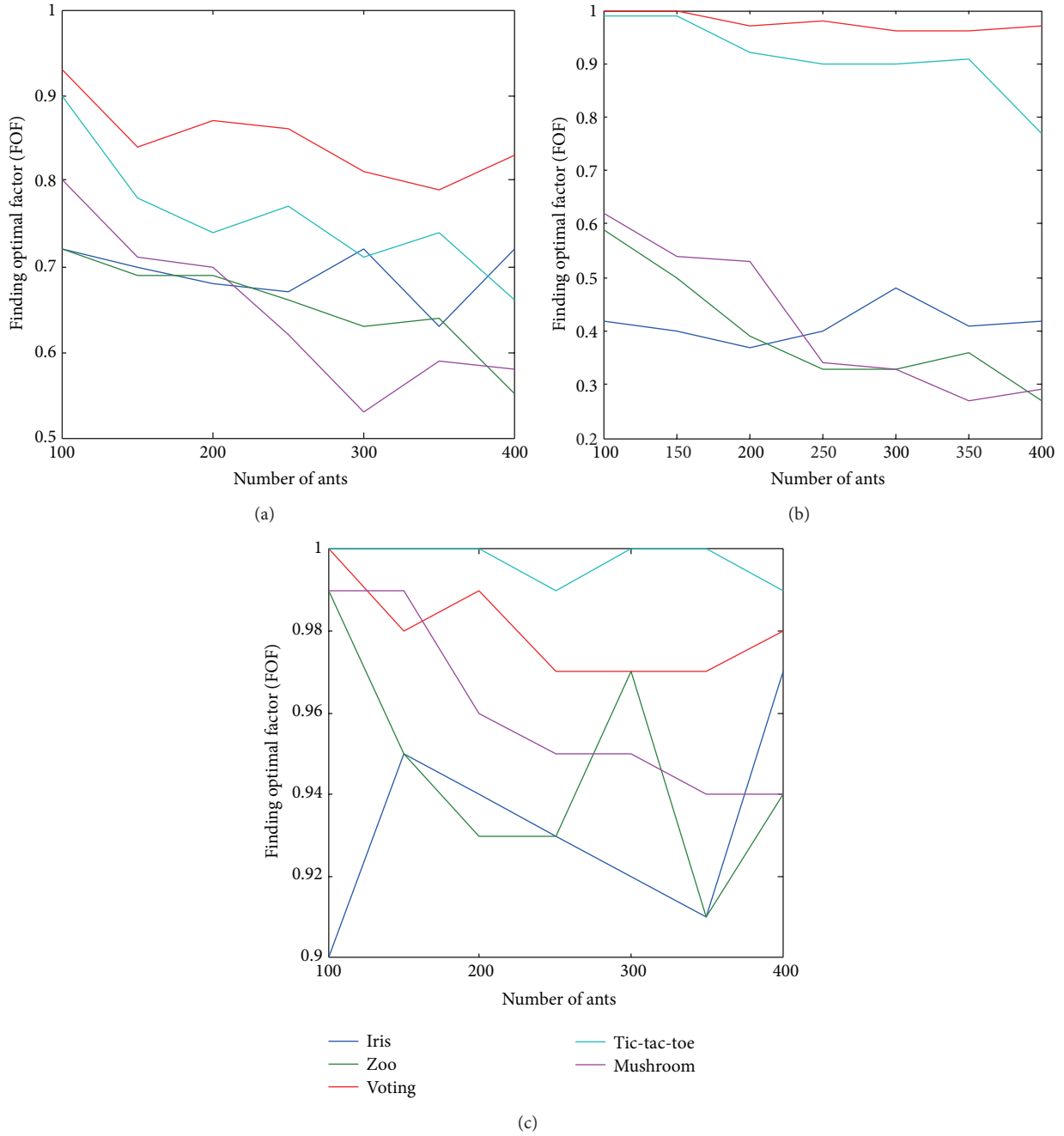

Figure 4: ACO with distributed deletion on different distributions: (a) uniform; (b) normal; (c) Pareto.

5. Conclusions

In this paper, we have pointed out the shortcoming of the existing heuristic algorithms including the information gain-based one and the genetic one. We have designed a new algorithm based on ant colony optimization to tackle the MTR problem. In deletion stage, our algorithm contains centralized deletion strategy and distributed deletion strategy. We have tested the new one with three representative test cost distributions on four UCI datasets. Experimental results show that the new algorithm outperforms the existing ones significantly, especially 1‘on medium-sized dataset such as Mushroom. Moreover, when distribution is normal, the distributed deletion strategy obtains better results than the centralized one on Mushroom dataset.

Acknowledgments

This work is in part supported by the National Science Foundation of China under Grant no. 61170128, the Natural Science Foundation of Fujian Province, China, under Grant no. 2012J01294, State Key Laboratory of Management and

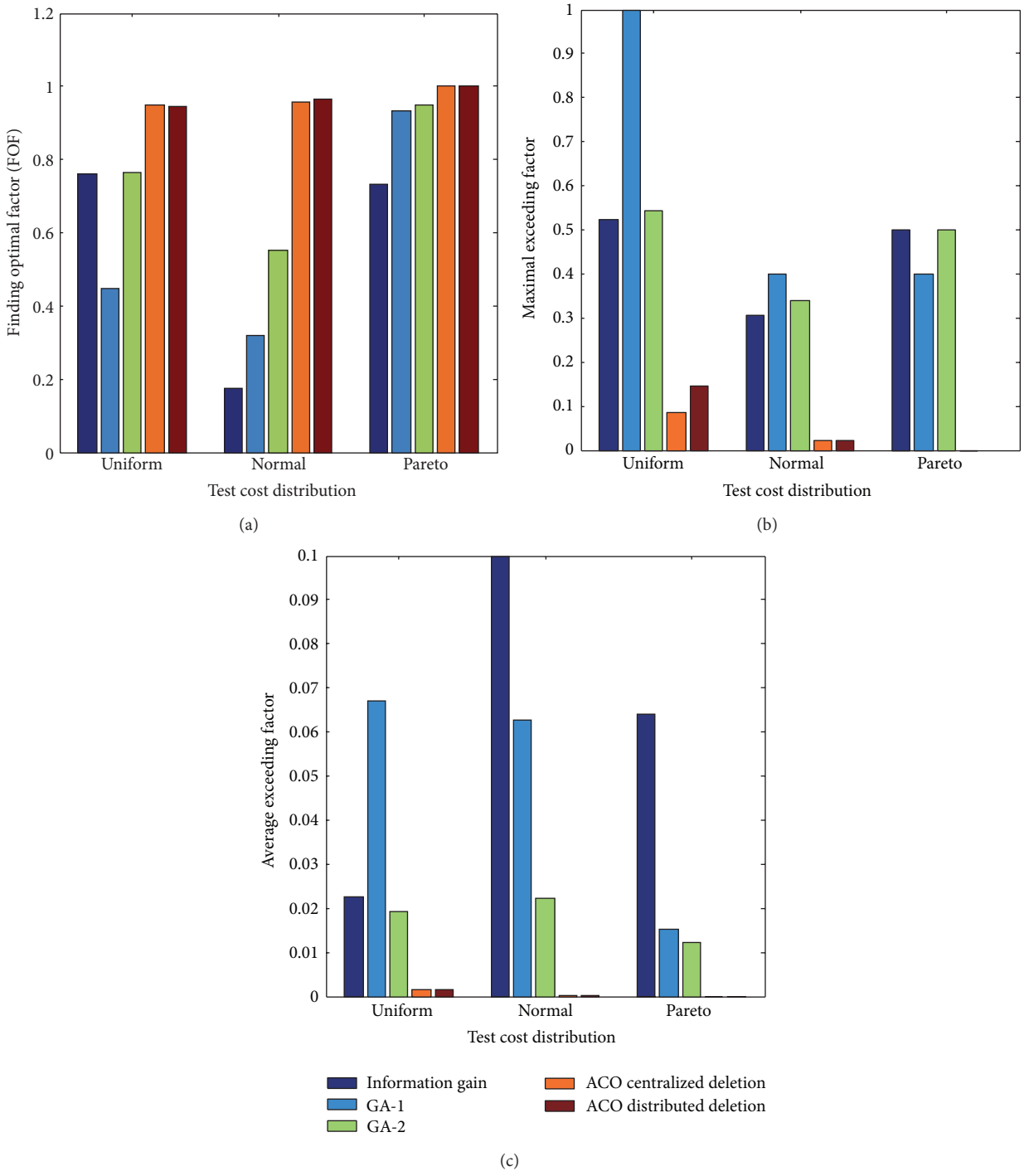

Figure 5: Result on Mushroom: (a) finding optimal factor; (b) maximal exceeding factor; (c) average exceeding factor.

Control for Complex Systems Open Project under Grant no. 20110106, and Fujian Province Foundation of Higher Education under Grant no. JK2012028.

References

[1] F. Min, H. He, Y. Qian, and W. Zhu, "Test-cost-sensitive attribute reduction," *Information Sciences*, vol. 181, no. 22, pp. 4928–4942, 2011.

[2] F. Min and W. Zhu, "Minimal cost attribute reduction through backtracking," in *Proceedings of the International Conference on Database Theory and Application*, vol. 258 of *FGIT-DTA/BSBT*, pp. 100–107, CCIS, 2011.

[3] Y. Y. Yao and Y. Zhao, "Attribute reduction in decision-theoretic rough set models," *Information Sciences*, vol. 178, no. 17, pp. 3356–3373, 2008.

[4] X. Jia, W. Liao, Z. Tang, and L. Shang, "Minimum cost attribute reduction in decision-theoretic rough set models," *Information Sciences*, vol. 219, pp. 151–167, 2013.

[5] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[6] G. Pan, F. Min, and W. Zhu, "A genetic algorithm to the minimal test cost reduct problem," in *Proceedings of the IEEE International Conference on Granular Computing (GrC '11)*, pp. 539–544, Kaohsiung, Taiwan, November 2011.

[7] F. Min and Q. Liu, "A hierarchical model for test-cost-sensitive decision systems," *Information Sciences*, vol. 179, no. 14, pp. 2442–2452, 2009.

[8] W. X. Zhang, J. S. Mi, and W. Z. Wu, "Knowledge reductions in inconsistent information systems," *Chinese Journal of Computers*, vol. 26, no. 1, pp. 12–18, 2003.

[9] Q. Liu, L. Chen, J. Zhang, and F. Min, "Knowledge reduction in inconsistent decision tables," in *Proceedings of Advanced Data Mining and Applications*, vol. 4093 of *Lecture Notes in Computer Science*, pp. 626–635, 2006.

[10] W. Ziarko, "Variable precision rough set model," *Journal of Computer and System Sciences*, vol. 46, no. 1, pp. 39–59, 1993.

[11] J. G. Bazan and A. Skowron, "Dynamic reducts as a tool for extracting laws from decision tables," in *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, pp. 346–355, 1994.

[12] D. Deng, "Parallel reduct and its properties," in *Proceedings of the International Conference on Granular Computing (GRC '09)*, pp. 121–125, Nanchang, China, August 2009.

[13] D. Deng, J. Wang, and X. Li, "Parallel reducts in a series of decision subsystems," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, pp. 377–380, Hainan, China, April 2009.

[14] F. Min, Z. Bai, M. He, and Q. Liu, "The reduct problem with specified attributes," in *Proceedings of the Rough Sets and Soft Computing in Intelligent Agent and Web Technology, International Workshop at WI-IAT*, pp. 36–42, 2005.

[15] W. Zhu and F. Wang, "On three types of covering rough sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1131–1144, 2007.

[16] W. Zhu and F. Y. Wang, "Reduction and axiomization of covering generalized rough sets," *Information Sciences*, vol. 152, pp. 217–230, 2003.

[17] C. C. Aggarwal, "On density based transforms for uncertain data mining," in *Proceedings of the 23rd IEEE International Conference on Data Engineering*, pp. 866–875, 2007.

[18] H. Zhao, F. Min, and W. Zhu, "Test-cost-sensitive attribute reduction of data with normal distribution measurement errors," *Mathematical Problems in Engineering*, vol. 2013, Article ID 946070, 12 pages, 2013.

[19] D. Ślęzak, "Approximate entropy reducts," *Fundamenta Informaticae*, vol. 53, no. 3-4, pp. 365–390, 2002.

[20] F. Min, X. Du, H. Qiu, and Q. Liu, "Minimal attribute space bias for attribute reduction," in *Proceedings of the Rough Set and Knowledge Technology*, pp. 379–386, 2007.

[21] W. Zhu, "Topological approaches to covering rough sets," *Information Sciences*, vol. 177, no. 6, pp. 1499–1508, 2007.

[22] S. Greco, B. Matarazzo, R. Slowinski, and J. Stefanowski, "Variable consistency model of dominance-based rough sets approach," in *Proceedings of the Rough Sets and Current Trends in Computing*, vol. 2005 of *Lecture Notes in Computer Science*, pp. 170–181, 2000.

[23] Q. Hu, D. Yu, J. Liu, and C. Wu, "Neighborhood rough set based heterogeneous feature subset selection," *Information Sciences*, vol. 178, no. 18, pp. 3577–3594, 2008.

[24] Y. Qian, J. Liang, W. Pedrycz, and C. Dang, "Positive approximation: an accelerator for attribute reduction in rough set theory," *Artificial Intelligence*, vol. 174, no. 9-10, pp. 597–618, 2010.

[25] Wikipedia, "Genetic algorithm," http://en.wikipedia.org/wiki/Geneticalgorithm.

[26] P. D. Turney, "Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm," *Journal of Artificial Intelligence Research*, vol. 2, pp. 369–409, 1995.

[27] J. Liu, F. Min, S. Liao, and W. Zhu, "A genetic algorithm to attribute reduction with test cost constraint," in *Proceedings of the 6th IEEE International Conference on Computer Sciences and Convergence Information Technology (ICCIT '11)*, pp. 751–754, 2011.

[28] M. Dorigo and T. Stutzle, Eds., *Ant Colony Optimization*, MIT Press, Boston, Mass, USA, 2004.

[29] L. Gambardella and M. Dorigo, "Has-sop: hybrid ant system for the sequential ordering problem," Tech. Rep. (IDSIA-11-97), 1997.

[30] L. M. Gambardella, E. Taillard, G. Agazzi, I. D. Corne, M. Dorigo, and F. Glover, "Macs-vrptw: a multiple ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*, pp. 63–76, McGraw-Hill, New York, NY, USA, 1999.

[31] V. Maniezzo and A. Colorni, "The ant system applied to the quadratic assignment problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 5, pp. 769–778, 1999.

[32] C. Blum, M. Blesa, and U. P. de Catalunya, Departament de Llenguatges i Sistemes Informàtics, Metaheuristics for the edge-weighted k-cardinality tree problem, 2003.

[33] R. Michel and M. Middendorf, "An aco algorithm for the shortest common supersequence problem," in *New Ideas in Optimization*, pp. 51–62, McGraw-Hill, London, UK, 1999.

[34] Y. Chen, D. Miao, and R. Wang, "A rough set approach to feature selection based on ant colony optimization," *Pattern Recognition Letters*, vol. 31, no. 3, pp. 226–233, 2010.

[35] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1351–1357, 2008.

[36] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998, http://www.ics.uci.edu/~mlearn/mlrepository.html.