Information Sciences 181 (2011) 4928–4942

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Test-cost-sensitive attribute reduction

Fan Min [a,*], Huaping He [b], Yuhua Qian [c], William Zhu [a]

[a] Lab of Granular Computing, Zhangzhou Normal University, Zhangzhou 363000, China
[b] School of Computer Science, Sichuan University of Science and Engineering, Zigong 643000, China
[c] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan 030006, Shanxi, China

ARTICLE INFO

Article history:
Received 10 December 2010
Received in revised form 8 June 2011
Accepted 3 July 2011
Available online 14 July 2011

Keywords:
Cost-sensitive learning
Attribute reduction
Test cost
Heuristic algorithm

ABSTRACT

In many data mining and machine learning applications, there are two objectives in the task of classification; one is decreasing the test cost, the other is improving the classification accuracy. Most existing research work focuses on the latter, with attribute reduction serving as an *optional* pre-processing stage to remove redundant attributes. In this paper, we point out that when tests must be undertaken in parallel, attribute reduction is *mandatory* in dealing with the former objective. With this in mind, we posit the minimal test cost reduct problem which constitutes a new, but more general, difficulty than the classical reduct problem. We also define three metrics to evaluate the performance of reduction algorithms from a statistical viewpoint. A framework for a heuristic algorithm is proposed to deal with the new problem; specifically, an information gain-based λ-weighted reduction algorithm is designed, where weights are decided by test costs and a non-positive exponent λ, which is the only parameter set by the user. The algorithm is tested with three representative test cost distributions on four UCI (University of California – Irvine) datasets. Experimental results show that there is a trade-off while setting λ, and a competition approach can improve the quality of the result significantly. This study suggests potential application areas and new research trends concerning attribute reduction.

1. Introduction

Classification is one of the most important topics of data mining and machine learning research [43]. In some applications, data are stored in databases [13] or available without charge, and our objective is to build a classifier with high accuracy. In other applications, however, data are not free, and there is a test cost [19,36,25] for each data item. Therefore the classifier should also exhibit low test costs. For example, in a clinic system, a patient is often required to undertake a number of medical tests. With the results of these tests, the doctor obtains a diagnosis of whether or not the patient has a particular illness or disease. In this case, money and/or time required to perform these tests are test costs; the probability of obtaining a correct diagnosis is the classification accuracy.

Numerous classifier building approaches have been proposed to deal with the classification accuracy issues. Famous ones [43] include decision trees [32,48], support vector machines [37], artificial neural networks [17], Bayes networks [11], and k-nearest neighbor algorithms. Some researchers used the concept of misclassification cost [14,12,52] to extend classification accuracy, since “the cost of erroneously diagnosing a patient as healthy may be much bigger than that of mistakenly diagnosing a healthy person as sick” [52]. With this concept, researchers even proposed considering both issues together.

* Corresponding author. Tel.: +86 133 7690 8359.
E-mail addresses: minfanphd@163.com (F. Min), hehuaping_001@163.com (H. He), jinchengqyh@sxu.edu.cn (Y. Qian), williamfengzhu@gmail.com (W. Zhu).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.07.010

A representative approach is the so-called ICET (inexpensive classification with expensive tests) [36], which is a hybrid of the GENESIS (genetic search implementation system) [15] genetic algorithm and the C4.5 [32] decision tree algorithm. In the decision tree construction process, a test is selected only if the benefit, in terms of more accurate diagnosis and therefore less misclassification cost, justifies the test cost [36]. However, this approach among many others is most effective when tests can be undertaken sequentially, for then the choice of a test can take advantage of previous test results.

In this paper, we assume that all tests must be undertaken in parallel. This assumption, called the parallel test assumption, has areas of application despite its simplifications. For example, when a casualty arrives by ambulance with a life-threatening condition, time is very critical. The doctor cannot wait for test results to decide on a course of action or which tests to further undertake. Instead, she/he has to immediately select those decisive tests. With this assumption, the learning task is divided into two disjoint subtasks: choose a set of tests, and build the classifier. The first subtask deals with the test cost issue, and the second one deals with the classification accuracy issue.

We focus on the test cost issue, with the classification accuracy issue serving as a constraint. One may argue that there is a trade-off between these: the higher the costs, the more accurate is the diagnosis. This claim holds only if tests are insufficient. If existing tests can support the diagnosis, more tests are redundant. Therefore, in regard to the classification accuracy issue, we only need to meet the constraint that there is a sufficiency of tests. Detail over the classification accuracy issue, i.e., the classifier building approach, are out of the scope of this paper.

Our problem is stated as follows: select a set of tests satisfying a minimal test cost criterion, and this test set should suffice as the set of all available tests. The most correlative approach to this problem might be attribute reduction [28,20,21,46], which has been investigated by the rough set [28] community. An attribute reduct is a subset of attributes/tests that are jointly sufficient and individually necessary for preserving a particular property of the given information table [28,46]. According to this definition, an attribute reduct satisfies well our requirement of "sufficiency". Moreover, the optimization objectives of attribute reduction, that include minimizing the number attributes [33], minimizing the attribute space [24], etc., which can be easily revised to minimize the total test cost of the reduct.

Under the context of rough set theory, our problem is restated as follows: find a minimal test cost reduct. We argue that this problem is not a simple extension of existing attribute reduction problems; these approaches were employed as an optional pre-processing stage of other classifier building approaches mentioned earlier. Their objectives include improving the efficiency of learning and improving the classification accuracy of the classifier. The former objective can be achieved when the dimensionality of the data is very high [8]; however, the second objective cannot be achieved in a straight-forward manner. In fact, there may exist many optimal reducts of a decision system, not all of being effective in producing good classifiers. This is why dynamic reducts [3] are useful in obtaining stable reducts. In contrast, with the parallel test assumption, attribute reduction is a *mandatory* stage in dealing with the test cost issue alone. The performance of respective approaches is the total test cost, which is independent of the performance (e.g., coverage, accuracy, *F*-measure) of the classifier built afterwards.

Before solving the problem, we define it formally. The definition of the problem involves the data model and the optimization metric. As the latter has already been discussed above, we now focus on the former. In [25], we addressed the common-test-cost issue and the sequence issue, building a hierarchy of six test-cost-sensitive decision systems. Under this context we have assumed that tests are undertaken in parallel, and the sequence issue never arises. Hence the most general model on which the problem is defined is the sequence-independent test-cost-sensitive decision system. Because finding a minimal reduct is non-polynomial (NP)-hard [33], and finding a minimal reduct is a special case of finding a minimal test cost reduct, the problem is also at least NP-hard. Consequently, heuristic algorithms are needed to deal with such problems.

We propose a framework of reduction algorithms. Practical algorithms are distinguished by the attribute significance function. Since this work is the first step toward test-cost-sensitive attribute reduction, while designing substantial algorithms, we shall here only consider a simpler model where test costs are independent from each other. Specifically, we design an information gain based λ-weighted function, where weights are decided by test costs and a non-positive exponent λ; here λ is introduced to adjust the influence of the test cost. Weighted reduction [44,45] is a general approach where the settings of weights vary among applications. Naturally, setting λ is much easier than setting weights for every attribute. We also propose the competition approach; that is, run the algorithm with different λ values, and choose the best reduct. With this approach, the user does not have to know the best setting of λ.

Four UCI datasets are employed to study the performance of our algorithm and identify better weighting schemes. As there is no test cost setting in these datasets, we use three distribution functions to generate test costs for these. The three functions correspond with different applications. We propose a metric to evaluate the quality of a reduct, and three metrics to evaluate the performance of the algorithms. Experimental results show that our algorithm can generate a minimal test cost reduct in most cases. Even if the reduct constructed is not optimal, it is still acceptable from a statistical point of view. Experiments are undertaken using an open source software called COSER (cost-sensitive rough sets) [27].

The rest of the paper is organized as follows: Section 2 defines formally the minimal test cost reduct problem. Three different generators are designed to produce test costs subject to certain distributions. Four evaluation metrics are proposed, one to evaluate the quality of a reduct, and the other three to evaluate the performance of a heuristic algorithm. Section 3 presents the algorithm framework and a substantial attribute significance function. Section 4 presents the experiment process and lists some results. Finally, Section 5 presents conclusions and outlines further research trends.

2. Problem definition

In this section, we define the minimal test cost reduct problem. First, we present the data model on which the problem is defined. Then we illustrate a well-known definition of relative reducts. Finally we propose the optimization objective, that is, minimizing the test cost.

2.1. Test-cost-independent decision systems

There are a number of test-cost-sensitive decision systems. In [25], we considered two issues related to test cost, *viz.* common-test-costing and sequencing, and proposed a hierarchy of six models. The common-test-cost issue arises while two or more tests share a common test cost. For example, the cost incurred in collecting a blood sample from a patient is shared by a set of blood tests [36,25]. The sequence issue arises while different testing sequences incur different test cost. For example, if blood test B is performed after A, there is a common test cost associated with blood sample collection if test A requires the greater blood sample; if the reverse sequence is employed, collection would have to be done again, and there is no common test cost.

Since we have assumed that tests are undertaken in parallel, the sequence issue never arises in our problem. Hence we only need to consider sequence-independent test-cost-sensitive decision systems, as defined as follows.

Definition 1 [25]. A *sequence-independent test-cost-sensitive decision system* (SITC-DS) S is the 6-tuple:

$$S = (U, C, D, \{V_a | a \in C \cup D\},\ \{I_a | a \in C \cup D\}, c^*), \tag{1}$$

where U is a finite set of objects called the universe, C is the set of conditional attributes, D is the set of decision attributes, 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 $a \in C \cup D, c^* : 2^C \rightarrow \mathbb{R}^+ \cup \{0\}$ is the attribute subset test cost function, where $\mathbb{R}^+$ is the set of positive real numbers.

Terms *conditional attribute*, *attribute* and *test* are already employed in the literature, and these have the same meaning throughout this paper. U, C, D, $\{V_a\}$ and $\{I_a\}$ can be displayed in a classical decision table. Table 1 is an example where $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, C = {Headache, Temperature, Lymphocyte, Leukocyte, Eosinophil, Heartbeat}, and D = {Flu}.

The attribute subset test cost function c^* is central to test-cost-sensitive decision systems; for each $A \subseteq C$, one should specify the value of $c^*(A)$. Therefore the simplest representation of c^* is to employ a vector

$$c^* = [c^*(\emptyset), c^*(\{a_1\}), c^*(\{a_2\}), \ldots, c^*(\{a_1, a_2\}), \ldots, c^*(C)]. \tag{2}$$

The length of c^* is $2^{|C|}$, which in applications grows too fast with increasing $|C|$ that only simple problems can be handled using a computer. In [25] we developed an alternative representation of the test cost function that helps in making the model applicable to real-world problems.

The SITC-DS is the most general model pertinent to the minimal test cost reduct problem, and more specifically to the consideration of problem definition and algorithm framework design. It is, however, hard to deal with directly from an algorithmic perspective. Since this work is the first step towards a resolution of our problem, we do not address here the common-test-cost issue when designing a substantive algorithm. Instead, we consider the simplest though most widely used model [25] as follows.

Definition 2 [25]. A *test-cost-independent decision system* (TCI-DS) S is the 6-tuple:

$$S = (U, C, D, \{V_a | a \in C \cup D\},\ \{I_a | a \in C \cup D\}, c), \tag{3}$$

where U, C, D, $\{V_a\}$, and $\{I_a\}$ have the same meanings as in Definition 1, $c : C \rightarrow \mathbb{R}^+ \cup \{0\}$ is the test cost function and

$$c^*(A) = \sum_{a \in A} c^*(\{a\}) = \sum_{a \in A} c(a), \tag{4}$$

where c^* is the attribute subset test cost function as defined in Definition 1.

Table 1
An exemplary decision table.

Patient	Headache	Temperature	Lymphocyte	Leukocyte	Eosinophil	Heartbeat	Flu
x_1	Yes	High	High	High	High	Normal	Yes
x_2	Yes	High	Normal	High	High	Abnormal	Yes
x_3	Yes	High	High	High	Normal	Abnormal	Yes
x_4	No	High	Normal	Normal	Normal	Normal	No
x_5	Yes	Normal	Normal	Low	High	Abnormal	No
x_6	Yes	Normal	Low	High	Normal	Abnormal	No
x_7	Yes	Low	Low	High	Normal	Normal	Yes

Here we assume that the range of cost function c is non-negative ($\mathbb{R}^+ \cup \{0\}$), which in practice is a natural assumption. A test cost function can easily be represented by a vector $c = [c(a_1), c(a_2), \ldots, c(a_{|C|})]$. Table 2 presents a cost function. Tables 1 and 2 represent a TCI-DS.

Eq. (4) indicates that the various test costs are independent of one another. For example, if the doctor selects tests Temperature, Leukocyte and Heartbeat, the total test cost would be \$2 + \$20 + \$10 = \$32 rather than some reduced cost. This is why we call this type of decision system *cost-independent*.

2.2. Test cost setting

Most datasets from the UCI library [6] have no intrinsic test costs. For statistical purposes, we employ three different schemes to produce random test costs. These schemes comprise: uniform distribution, normal distribution, and Pareto distribution. For simplicity, test costs are integers ranging from M to N, and are evaluated independently. In the following we briefly discuss these distributions and present how to compute the respective random numbers.

2.2.1. Uniform distribution

A uniform distribution may be the most commonly used distribution. A discrete uniform distribution is a probability distribution whereby a finite number of equally-spaced values are equally likely to be observed [2]. Let c_u denote a test cost under the discrete uniform distribution.

$$P(c_u = n) = \frac{1}{N - M + 1}, \tag{5}$$

where n is an integer in $[M,N]$.

In many programming languages, it is straightforward to generate a uniformly distributed random number x on $(0,1)$. Then

$$c_u(M,N,x) = M + \lfloor (N - M + 1)x \rfloor \tag{6}$$

takes discrete and uniformly-distributed values on $[M,N]$.

2.2.2. Normal distribution

A normal distribution is often used to describe, at least approximately, any variable that tends to cluster around the mean [2]. For example, the cost in collecting precipitation data from different gauge stations in an area is roughly normally-distributed.

The normal distribution is described by the probability density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{7}$$

where parameters μ and σ^2 are the mean and the variance respectively. The standard normal distribution appears with $\mu = 0$ and $\sigma^2 = 1$.

Let x_1 and x_2 be uniformly distributed on $(0,1)$, then

$$y_1(x_1,x_2) = \sqrt{-2\ln x_1}\cos(2\pi x_2) \tag{8}$$

and

$$y_2(x_1,x_2) = \sqrt{-2\ln x_1}\sin(2\pi x_2) \tag{9}$$

are two random numbers which have a normal distribution with $\mu = 0$ mean and $\sigma^2 = 1$. A faster and more robust approach is given in [1].

Since we need a random number in $[M,N]$, we let

$$y(M,N,x_1,x_2) = \frac{M + N + 1}{2} + \alpha y_1(x_1,x_2), \tag{10}$$

where α is a non-positive number that permits adjustments in the distribution; we set $\alpha = 8$ in our experiments.

Table 2
An exemplary cost vector.

a	Headache	Temperature	Lymphocyte	Leukocyte	Eosinophil	Heartbeat
$c(a)$	\$2	\$2	\$15	\$20	\$20	\$10

Finally

$$c_n(M,N,x_1,x_2) = \begin{cases} M & \text{if } y < M; \\ N & \text{if } y > N; \\ \lfloor y(M,N,x_1,x_2) \rfloor & \text{otherwise} \end{cases} \tag{11}$$

is bounded by $[M,N]$ with a mean of approximately $\frac{M+N}{2}$.

2.2.3. Pareto distribution

The Pareto distribution is a powerful law probability distribution that coincides with social, scientific, geophysical, actuarial, and many other type of observable phenomena [2]. In a clinic system, most tests (e.g., headache, temperature, heartbeat and Lymphocyte) have relatively low cost, some tests (e.g., brain CT) have medium cost, and a few tests (e.g., arteriocerebral angiography) have very high cost. In a computer system, job sizes in terms of CPU requirement also have a similar property [16].

In practice, the bounded Pareto distribution is more applicable. Let x be uniformly distributed on $(0,1)$. Then

$$p(M,N,x) = \left(-\left(\frac{x(N+1)^{\alpha} - xM^{\alpha} - (N+1)^{\alpha}}{M^{\alpha}(N+1)^{\alpha}}\right)\right)^{-1/\alpha} \tag{12}$$

is bounded Pareto-distributed on $(M,N+1)$, where α determines the shape of the distribution; we set $\alpha = 2$ in our experiments. Finally,

$$c_p(M,N,x) = \lfloor p(M,N,x) \rfloor \tag{13}$$

is discrete, bounded, and Pareto-distributed on $[M,N]$.

For convenience, in the following context, the discrete uniform distribution, the discrete bounded normal distribution and the discrete bounded Pareto distribution are referred to as the Uniform distribution, the Normal distribution, and the Pareto distribution, respectively.

2.3. Relative reducts

Concepts of reduct and relative reduct have been thoroughly investigated by the rough set community. The concept of reduct is built on information systems, and is straightforward to apply. In contrast, the concept of relative reduct is built on decision systems, and there are many different definitions, such as positive region reducts [28], maximum distribution reducts [49], fuzzy reducts [20], β-reduct [56]. These definitions are equivalent if the decision table is consistent, but different if otherwise. Furthermore, there are some extended concepts such as *M*-relative reducts [26], dynamic reducts [3], and parallel reducts [9,10]. In some extensions of rough sets, i.e., covering-based rough sets [55], there are other definitions of a reduct (see, e.g., [54,7]). A general definition of a reduct is the focus of [51].

In this paper, we only consider the original concept, i.e., positive-region-based reducts [28]. To present the definition, a number of basic concepts of rough set theory are needed. Any $\emptyset \neq B \subseteq C \cup D$ determines an indiscernibility relation $I(B)$ on U. A partition determined by B is denoted by $U/I(B)$, or U/B for brevity. Let $\underline{B}\,\underline{X}$ denote the $B-$ *lower approximation* of X. The positive region of D with respect to $B \subseteq C$ is defined as $POS_B(D) = \bigcup_{X \in U/D} \underline{B}(X)$.

Definition 3. Any $B \subseteq C$ is called a *decision relative reduct* (or *a relative reduct* for brevity) of S iff:

(1) $POS_B(D) = POS_C(D)$, and
(2) $\forall a \in B$, $POS_{B-\{a\}}(D) \subset POS_C(D)$.

Conditions of the definition indicate that a reduct is (1) jointly sufficient and (2) individually necessary for preserving a particular property (positive region in this context) of the decision system [28,51,46]. The set of all relative reducts of S is denoted by $Red(S)$. The *core* of the system S is the intersection of these reducts. Namely, $core(S) = \cap Red(S)$. Core attributes are of great importance to the decision system and should never be removed, except when information loss is allowed [46].

2.4. Minimal test cost reducts

Sometimes a number of reducts are needed. For example, dynamic reducts [3] and parallel reducts [9,10] are based on a number of reducts and are more stable than any one of them. Genetic algorithms [42] are usually employed to produce base reducts. In [41], to preserve more information of the decision system, a few fuzzy reducts are also required to produce fuzzy decision trees.

In most applications, only one reduct is needed. Since there may exist many reducts, an optimization metric is needed. Existing metrics include the number of attributes [29,38], the attribute space [23], etc. Naturally, the metric employed in this work is the test cost of the reduct. In other words, we are interested in reducts with minimal test costs. We define reducts of this type as follows.

Definition 4. Let $Red(S)$ denote the set of all relative reducts of a sequence-independent test-cost-sensitive decision system S. Any $R \in Red(S)$ where $c^*(R) = \min\{c^*(R') | R' \in Red(S)\}$ is called a *minimal test cost reduct*.

There may exist a number of minimal test cost reducts. Our problem is to find any one of these; the new problem is called the *minimal test cost reduct problem*. A minimal test cost reduct is also called an *optimal reduct* in this context. Note that this problem is defined on SCITC-DS instead of more specific TCI-DS.

2.5. Evaluation metrics

One can develop many algorithms to deal with the minimal test cost reduct problem. Therefore, there is a need to define a number of evaluation metrics so that performances can be compared. First of all, we need a metric to evaluate the quality of one particular reduct. For example, if the test cost of the optimal reduct is \$100, a reduct with test cost \$110 is better than another with \$120. Since an algorithm can run on many datasets or one dataset with different test cost settings, we propose three metrics from a statistical viewpoint. These are *finding optimal* factor, *maximal exceeding* factor, and *average exceeding* factor.

2.5.1. Finding optimal factor

Let the number of experiments be K, and the number of successful searches of an optimal reduct be k. The *finding optimal factor* (FOF) is defined as

$$op = \frac{k}{K}. \tag{14}$$

This metric is both qualitative and quantitative. First, it only counts optimal solutions. Second, it is computed statistically. In our experiments, we generate different test cost settings for the same decision system. Therefore, we have as many test-cost-sensitive decision systems as we need. This allows us to have enough data to obtain the finding optimal factor for statistics purposes.

2.5.2. 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

$$ef(R) = \frac{c^*(R) - c^*(R')}{c^*(R')}. \tag{15}$$

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 ($1 \leqslant i \leqslant K$), the reduct computed by the algorithm is denoted R_i. The *maximal exceeding factor* (MEF) is defined as

$$\max_{1 \leqslant i \leqslant K} ef(R_i). \tag{16}$$

This shows the worst case of the algorithm given some data set. 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* (AEF) is defined as

$$\frac{\sum_{i=1}^{K} ef(R_i)}{K}. \tag{17}$$

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

3. Test-cost-sensitive reduction algorithms

To design test-cost-sensitive reduction algorithms, we need to review reduction algorithms for the minimal reduct problem first. There are generally two types of algorithms, namely, exhaustive algorithms and heuristic algorithms.

An exhaustive algorithm to the minimal reduct problem contains two main steps: Step 1, construct the set of all relative reducts using the discernibility matrix [33]; and Step 2, count the size of each reduct and at the same time, select a minimal reduct. Unfortunately, this algorithm is not scalable for the following reasons. First, Step 1 involve the DNF solution of CNF in the Boolean algebra, which is NP-hard [33]. Second, the number of reducts of a decision system is exponential with respect to the number of attributes. This is why heuristic algorithms have attracted much attention in this type of research.

Most heuristic algorithms to this problem have the same structure; their differences lie in the heuristic function, also called the fitness function [47] employed. From the viewpoint of heuristic functions, Qian et al. classified these methods into four categories [30]: positive region reduction, Shannon's entropy reduction [38], Liang's entropy reduction, and combination entropy reduction.

In case that the test cost of any attribute is identical, a TCI-DS coincides with a decision system, and the minimal test cost reduct problem coincides with the traditional relative reduct problem [25]. In other words, the latter problem can be viewed as a special case of the former. Therefore, we also propose two types of algorithms to our problem.

We revise the exhaustive algorithm mentioned above to deal with our problem more easily; that is, choose a minimal test cost reduct instead of a minimal reduct in Step 2. Computing the test cost of a reduct has the same complexity as counting the number of attributes; therefore this algorithm is also NP-hard. In this paper, it is employed to evaluate the performance of heuristic algorithms.

3.1. Information gain-based λ-weighted reduction

Now we revise the traditional addition-deletion attribute reduction algorithm framework [47] to deal with our problem. Algorithm 1 is a framework containing three main steps: Step 1, add core attributes to B; Step 2, add the current-best attribute a to B according to the heuristic function $f(B,a,c)$ until B becomes a super-reduct; and Step 3, delete redundant attributes from B. The usefulness of Step 3 will be shown through Example 9 in Section 4.2.1. To avoid the deletion phase, the addition method [47] can be revised in the same manner. However, since it requires the partial reduct check [47,50], which requires more analysis, we do not consider it in this paper.

Algorithm 1: A general test-cost-sensitive reduction algorithm

```
Input: $S = (U, C, D, \{V_a|a \in C \cup D\}, \{I_a|a \in C \cup D\}, c)$
Output: A reduct with sub-minimal test cost
Method: tcs-reduction
1:  $B = \emptyset$;
    //Core computing
2:  for ($i$=1 to $|C|$) do
3:    if ($POS_{C-\{a_i\}}(D) \neq POS_C(D)$) then
4:      $B \leftarrow B \cup \{a_i\}$;// $a_i$ is a core attribute
5:    end if
6:  end for
    //Addition
7:  $CA = C - B$;
8:  while ($POS_B(D) \neq POS_C(D)$) do
9:    For any $a \in CA$, compute $f(B,a,c)$;
10:   Select $a'$ with the maximal $f(B,a',c)$;
11:   $B = B \cup \{a'\}$; $CA = CA - \{a'\}$;
12: end while
    //Deletion
13: $CD = B$; sort attributes in $CD$ according to respective test cost in a descending order;
14: while ($CD \neq \emptyset$) do
15:   $CD = CD - \{a'\}$, where $a'$ is the first element of $CD$;
16:   if ($POS_{B-\{a'\}}(D) = POS_B(D)$) then
17:     $B = B - \{a'\}$;
18:   end if
19: end while
20: return $B$;
```

Lines 9 and 10 contain the key code of this framework. One can design different attribute significance functions $f(B,a,c)$ to obtain respective algorithms. Information gain [31] is often employed to represent the attribute significance [39,22]. To take into consideration test costs, we propose a weighting approach [44,45] based on that gain.

The concepts of information entropy and information gain are well known. Recently fuzzy entropy are employed to improve the generalization ability of rule sets [40]. We list basic concepts for completeness. In the following context, we let $S = (U, C, \{d\}, \{V_a\}, \{I_a\}, c)$ be a TCI-DS as defined in Definition 2; $P, Q \subseteq C \cup \{d\}$; partitions of the universe U introduced by P and Q be $X = \{X_1, X_2, \ldots, X_n\}$ and $Y = \{Y_1, Y_2, \ldots, Y_m\}$, respectively.

Any attribute subset P can be viewed as a random variable defined on the universe U, and the probability distribution can be determined as follows.

Definition 5 [39]. The σ-algebra distributions of P and Q on U are

$$[X:p] = \begin{bmatrix} X_1 & X_2 & \ldots & X_n \\ p(X_1) & p(X_2) & \ldots & p(X_n) \end{bmatrix} \tag{18}$$

and

$$[Y:p] = \begin{bmatrix} Y_1 & Y_2 & \dots & Y_n \\ p(Y_1) & p(Y_2) & \dots & p(Y_m) \end{bmatrix}, \tag{19}$$

respectively, where $p(X_i) = \frac{|X_i|}{|U|},\ i = 1, 2, \dots, n$, and $p(Y_j) = \frac{|Y_j|}{|U|},\ j = 1, 2, \dots, m$.

The uncertainty in the values of attributes in P is measured by the information entropy.

Definition 6 [39]. The information entropy of P is

$$H(P) = -\sum_{i=1}^{n} p(X_i) \log p(X_i). \tag{20}$$

Since we are dealing with the classification task, there is a need to consider conditional information entropy.

Definition 7 [39]. The conditional information entropy of Q wrt. P is

$$H(Q|P) = -\sum_{i=1}^{n} p(X_i) \sum_{j=1}^{m} p(Y_j|X_i) \log(p(Y_j|X_i)), \tag{21}$$

where $p(Y_j|X_i) = |Y_j \cap X_i|/|X_i|$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, m$.

In fact, for the classification task, we often let $Q = \{d\}$. That is, the entropy is conditional on the decision attribute d. We therefore arrive at the key concept.

Definition 8. Let $B \subset C$, $a_i \in C$ and $a_i \notin B$. The information gain of a_i respect to B is

$$f_e(B, a_i) = H(\{d\}|B) - H(\{d\}|B \cup \{a_i\}). \tag{22}$$

This concept was employed to design heuristic attribute reduction algorithms in [39].

Now we propose our λ-weighted function as follows.

$$f(B, a_i, c_i) = f_e(B, a_i) c_i^{\lambda}, \tag{23}$$

where λ is a non-positive number. If $\lambda = 0$, this function reduces to the traditional information gain; otherwise, this function has a tendency to low cost test. The setting of λ is the focus of our experiments in Section 4.2.

Note that the λ-weighted function requires test costs be non-zero. In applications, if a test has no cost at all, we still assign it a small value. This modification makes sense as there should be a certain cost in collecting each data item.

The time complexity of computing $f(B, a_i, c_i)$ is the same as that of $f_e(B, a_i)$. Consequently, the time and space complexities of the algorithm are the same that of the corresponding heuristic reduction algorithm. In our implementation, the time complexity is $O(|U|^2|C|^3)$ and the space complexity is $O(|U||C|)$. With the approach of [30], the time complexity can be reduced to $O\left(\sum_{i=1}^{|C|} |U_i|(|C| - i + 1)\right)$, where $U_i = U - POS_{\{a_1, a_2, \dots, a_i\}}(D)$. And the space complexity is not changed.

3.2. The competition approach

Different λ settings can result in different reducts. This claim will be validated in Section 4 by experimentation. Therefore, we can specify a set of λ values, obtain corresponding reducts using Algorithm 1, and choose a reduct with minimal test cost. In this approach, reducts compete against each other with only one winner, therefore it is called the competition approach.

Formally, let R_λ be the reduct constructed by Algorithm 1 using the exponential λ, with L the set of user-specified λ values.

$$c_L = \min_{\lambda \in L} c(R_\lambda) \tag{24}$$

is the minimal test cost that can be obtained using all λ values in L.

This process requires the algorithm to be run $|L|$ times, which is acceptable for relatively small $|L|$. Moreover, if we run this process on $|L|$ different computers in parallel, the time complexity is not changed. In many applications this approach is useful. As will be shown in Section 4.2.3, this simple approach will enhance the quality of the result significantly.

4. Experiments

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

(1) Is our algorithm appropriate for the minimal test-cost reduct problem?
(2) Is there an optimal setting of λ that is valid for any dataset?
(3) Does the test cost distribution influence the quality of the result?
(4) Can the competition approach improve the quality of the result?

4.1. Datasets

We deliberately select four datasets from the UCI Repository of Machine Learning Databases [6]. Their basic information are listed in Table 3, where $|C|$ is the number of attributes, $|U|$ is the number of instances, D is the name of the decision, and $|Red(S)|$ is the total number of reducts. $Red(S)$ is obtained using the exhaustive algorithm discussed in [33] and Section 2.3. One can also obtain it easily using RSES [4,5].

There are a number of observations that can be made in regard to all datasets.

(1) While counting the number of attributes, the decision attribute is not included.
(2) Missing values (e.g., those appearing in the Voting dataset) are treated as one particular value. That is,? is equal to itself, and unequal to any other value.

The "animal name" attribute is not useful in the Zoo dataset, and we simply remove it. There are 15 boolean attributes plus 1 enumerate attribute whose domain is $\{0,2,4,5,6,8\}$. There are 7 class values, which distinguish this dataset with from 2 class dataset.

The Iris dataset, found in the pattern recognition literature, is perhaps the best known. There are 4 numeric attributes. However, we do not employ discretization approaches to process it because the number of possible values is still limited.

The house-voting-84 dataset (Voting for short) has the same size of attributes as that of Zoo, but has more instances. However, it contains only 3 reducts.

The Tic-tac-toe dataset has the biggest size among the datasets tested. It also has a rational number of reducts.

Table 3
Dataset information.

Name	Domain	$\|C\|$	U	$D = \{d\}$	$\|Red(S)\|$
Zoo	Zoology	16	101	Type	33
Voting	Society	16	435	Vote	3
Tic-tac-toe	Game	9	958	Class	9
Mushroom	Botany	22	8124	Classes	292

Table 4
Test cost settings of Zoo.

ID	Distribution	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
1	Uniform	9	42	97	77	10	63	70	84	42	97	62	65	69	55	86	53
2		35	28	58	18	50	18	53	56	31	9	10	46	28	6	43	1
3		98	72	73	4	4	63	28	58	25	48	68	53	26	82	37	26
4		46	52	59	91	54	52	48	74	17	62	43	77	74	55	49	28
5		59	19	85	11	75	12	63	30	93	16	98	58	20	93	99	52
6		13	34	33	35	77	51	85	10	27	6	19	4	40	53	86	16
7		52	87	21	28	6	2	51	38	91	71	90	99	45	42	24	57
8		42	60	22	33	87	96	65	48	32	22	86	38	3	58	17	96
9		96	75	71	25	21	35	87	29	98	44	40	84	22	17	5	28
10		83	9	12	5	99	7	72	5	23	91	85	70	44	80	55	35
11	Normal	44	64	53	57	47	50	55	45	44	59	53	44	33	43	58	51
12		54	68	53	38	49	51	65	56	67	54	45	56	49	49	51	45
13		48	57	62	55	51	41	63	70	70	30	39	48	44	35	54	40
14		46	48	54	42	66	47	44	57	62	39	59	45	40	51	51	53
15		58	53	55	58	53	50	44	50	53	52	54	47	41	55	51	47
16		62	50	53	63	54	53	70	51	50	57	51	54	54	63	36	60
17		49	52	36	44	52	49	39	62	47	51	69	47	39	42	47	53
18		62	61	45	47	54	57	52	59	41	47	57	65	37	37	35	51
19		53	55	44	61	50	54	57	42	40	40	56	43	57	46	55	46
20		65	45	46	39	38	53	46	54	54	51	40	39	54	59	47	54
21	Pareto	1	2	84	4	42	1	1	1	1	1	30	2	1	1	19	3
22		13	1	1	1	1	3	1	10	54	19	4	1	1	1	3	2
23		33	14	2	6	1	11	8	1	1	2	3	1	3	1	1	62
24		10	11	3	1	5	62	4	1	1	11	1	1	2	1	13	1
25		2	5	1	3	1	6	1	3	1	1	4	2	1	37	1	1
26		1	1	2	1	37	12	1	2	19	2	12	8	2	1	1	13
27		3	21	1	1	1	1	3	2	30	7	27	84	2	2	1	4
28		5	4	2	6	2	1	33	62	1	1	1	11	23	3	8	1
29		4	1	18	1	9	1	5	1	37	1	71	4	1	37	84	3
30		21	3	8	5	6	1	1	2	2	47	1	1	12	2	1	6

4.2. Results

Optimal reducts are needed to evaluate the performance of the algorithm. These are obtained using the exhaustive approach implemented in RSES [4,5]. This approach takes much time for the Mushroom dataset. Fortunately, we need to obtain the reduct set one time only for each dataset; then, for each cost vector, the cost of each reduct is calculated and the optimal reduct is found.

We present both individual and statistical results. Individual results show more detail of the test cost setting, the reduct constructed by the algorithm, and the quality of the reduct. Two examples show the process of reduction. Statistical results are based on a large number of experiments on different datasets and/or test cost settings. These show the effectiveness of our algorithm, and indicate the appropriate setting of the exponent factor λ. We also compare three approaches mentioned in Section 3.

4.2.1. Individual results of Zoo

Since the Zoo has a rational number of attributes and instances, we focus on it for certain detailed analysis. 30 test cost settings are listed in Table 4; 10 for each distribution discussed in Section 2.2. Letters a through p are employed to represent attributes hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic, and catsize, respectively. This is partly because test cost settings for these attributes are rather artificial, and letters are more abstract and therefore more appropriate than real names.

The settings of each distribution is as follows: $M = 1$ and $N = 100$ for each distribution; for the normal distribution, $\alpha = 8$, and test costs as high as 70 and as low as 30 are often generated; for the bounded Pareto distribution, $\alpha = 2$, and test costs higher than 50 are often generated. These settings are valid through the rest of the paper.

Table 5
Results of Zoo.

ID	Optimal reduct	Minimal test cost	Constructed reduct	Test cost	Is optimal	Exceeding factor
1	{d,f,i,l,m}	316	{c,f,i,m,p}	324	No	0.025
2	{d,f,j,k,m,n,p}	90	{d,f,j,k,m,n,p}	90	Yes	0
3	{d,f,i,l,m}	171	{d,f,i,l,m}	171	Yes	0
4	{c,f,i,m,p}	230	{c,f,i,m,p}	230	Yes	0
5	{d,f,h,l,m}	131	{d,f,h,l,m}	131	Yes	0
6	{a,f,h,j,l,m}	124	{a,f,h,j,l,m}	124	Yes	0
7	{c,d,f,h,m}	134	{c,d,f,h,m}	134	Yes	0
8	{c,d,f,i,m}	186	{c,d,f,i,m}	186	Yes	0
9	{d,f,h,k,m,p}	179	{d,f,h,k,m,p}	179	Yes	0
10	{c,d,f,h,m}	73	{c,d,f,h,m}	73	Yes	0
11	{d,f,i,l,m}	228	{a,c,f,i,l,m}	268	No	0.175
12	{c,d,f,h,m}	247	{c,d,f,h,m}	247	Yes	0
13	{a,f,j,l,m,n}	246	{a,f,j,l,m,n}	246	Yes	0
14	{d,f,h,l,m}	231	{d,f,h,l,m}	231	Yes	0
15	{c,f,h,m,p}	243	{c,f,h,m,p}	243	Yes	0
16	{c,f,h,j,m}	268	{c,f,h,j,m}	268	Yes	0
17	{c,d,f,i,m}	215	{c,d,f,i,m}	215	Yes	0
18	{c,d,f,i,m}	227	{c,d,f,j,m,n}	270	No	0.189
19	{c,f,h,j,m}	237	{c,f,h,j,m}	237	Yes	0
20	{d,f,h,l,m}	239	{d,f,h,l,m}	237	Yes	0
21	{a,f,I,j,l,m}	7	{a,f,I,j,l,m}	7	Yes	0
22	{d,f,h,l,m}	16	{d,f,h,l,m}	16	Yes	0
23	{c,f,h,l,m,n}	19	{c,f,h,l,m,n}	19	Yes	0
24	{d,f,h,l,m}	67	{d,f,h,l,m}	67	Yes	0
25	{c,f,i,m,p}	10	{c,f,i,m,p}	10	Yes	0
26	{c,d,f,h,m}	19	{c,d,f,j,m,n}	20	No	0.053
27	{c,d,f,h,m}	7	{c,d,f,h,m}	7	Yes	0
28	{c,f,i,m,p}	28	{c,f,i,m,p}	28	Yes	0
29	{d,f,h,l,m}	8	{d,f,h,l,m}	8	Yes	0
30	{d,f,h,l,m}	21	{d,f,h,l,m}	21	Yes	0

Table 6
Statistics of Table 5.

Distribution	Finding optimal factor	Maximal exceeding factor	Average exceeding factor
Uniform	0.9	0.025	0.0025
Normal	0.8	0.189	0.0364
Pareto	0.9	0.053	0.0053
Overall	0.867	0.189	0.0147

Let $\lambda = -1$ in Eq. (23). Detailed results are listed in Table 5. These are also digested in Table 6. The finding optimal factor is 86.7%, which is not very high. However, the maximal exceeding factor is 0.189, which is acceptable, while the average exceeding factor is 0.0147, which is very good.

The following example indicates that the second **for** loop of the algorithm is necessary.

Example 9. For the test cost setting with ID 5, c=[59, 19, 85, 11, 75, 12, 63, 30, 93, 16, 98, 58, 20, 93, 99, 52]. With Algorithm 1, the attribute set {d, f, h, j, l, m} is obtained after the **while** loop. It is, however, not a reduct. Then in the **for** loop, attribute j is removed and a reduct {d, f, h, l, m} is constructed. Table 5 shows that this reduct is optimal.

If we run the algorithm many times with different cost vectors, we obtain some reducts with rather high exceeding factors. The following example gives an intuitive understanding.

Example 10. Let c=[2, 1, 4, 1, 2, 2, 42, 18, 25, 1, 2, 1, 1, 7, 2, 6], the optimal reduct is {d, f, j, l, m, n}, and the optimal test cost is 1 + 2+1 + 1+1 + 7 = 13. The reduct process is as follows: f, m (core attributes) → d → l → h, and the reduct found is {d, f, h, l, m}. The test cost for the reduct is 1 + 2+18 + 1 + 1 = 23, making an exceeding factor (23 − 13)/13 ≈ 0.77.

In the reduct construction process, the algorithm chose attribute h (with cost 18) instead of j (with cost 1) or n (with cost 7). In fact, the weighted information gain of attribute h is 0.0151, only slightly larger than that of j (0.0133) or n (0.0132). Also, the optimal reduct has one more attribute than the constructed reduct. This phenomenon indicates that the optimal reduct may be hard to construct for certain test cost settings.

Unfortunately, in the framework of our algorithm, we have not found a setting to guarantee optimality of the reduct constructed. Such a setting seems not to exist. This will be discussed further in the next subsection.

4.2.2. Statistical results of all datasets

To find an appropriate setting for λ, we let $\lambda = 0, -0.25, -0.5, \ldots$. When $\lambda = 0$, according to Eq. (23),

$$f(B, a_i, c_i) = f_e(B, a_i)c_i^{\lambda} = f_e(B, a_i). \tag{25}$$

In other words, the algorithm degrades to an entropy-based reduction algorithm without taking into account test costs.

For each weight-setting scheme applied to a particular test cost setting scheme and dataset, we undertake 4000 experimental runs. There are 4 datasets; for each dataset, there are 3 distributions as discussed in Section 2.2 to generate test costs; for each distribution, we generate 2,000 sets of test costs; finally, for each test cost setting, there are 9 λ settings. Therefore, a total of $4 \times 3 \times 2{,}000 \times 9 = 216{,}000$ experimental results are generated, that are aggregated according to distribution and our factor analysis into Figs. 1–3.

Fig. 1 shows the results of finding optimal factors. For different test cost distributions, the performance of the algorithm is different. With the Pareto distribution, the algorithm has the best performance; the normal distribution produces the worst. A possible reason is that the Pareto distribution generates many small values and a few large values, and there are many attributes with both low test costs and large information gain. In contrast, the normal distribution generates many values close to the mean value, and attributes with both low test costs and large information gain often do not exist. Moreover, there is usually a trade-off between the test cost and the information gain of the attribute. Finally, with the uniform

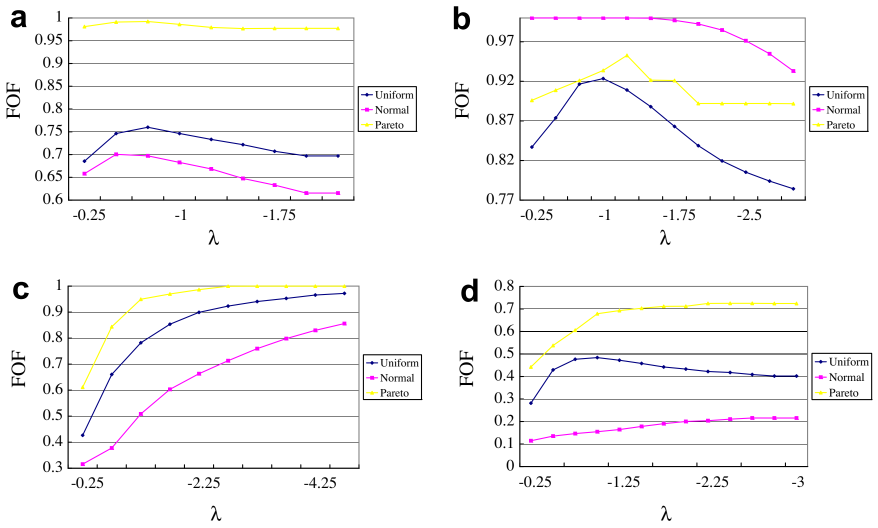

Fig. 1. Finding optimal factor: (a) Zoo; (b) Voting; (c) Tic-tac-toe; (d) Mushroom.

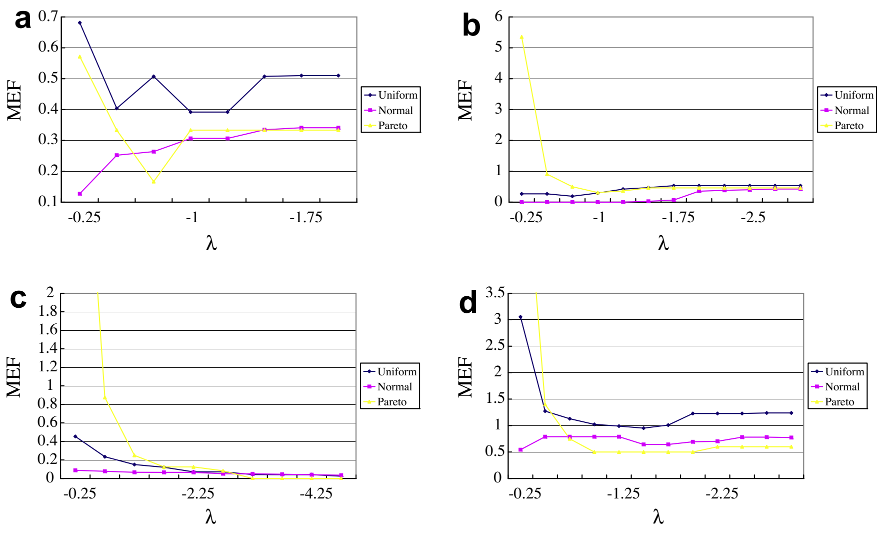

Fig. 2. Maximal exceeding factor: (a) Iris; (b) Zoo; (c) Tic-tac-toe; (d) Mushroom.

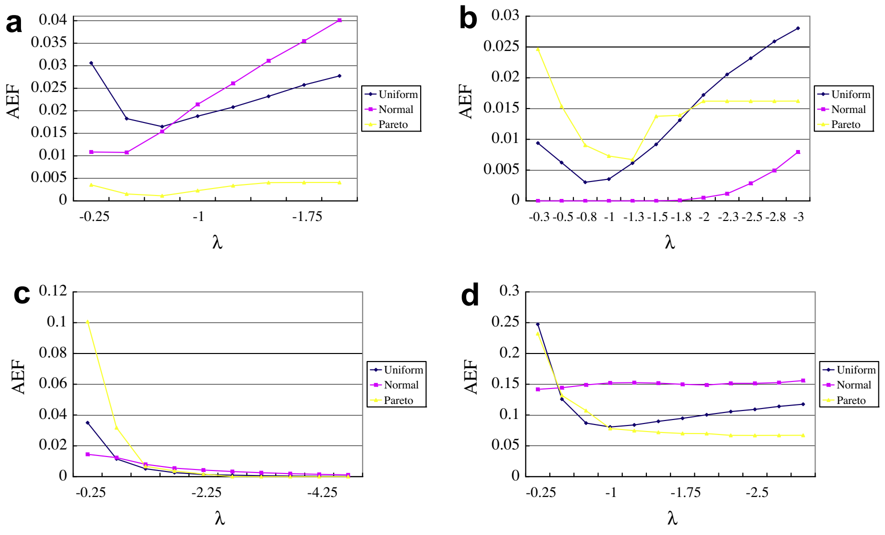

Fig. 3. Average exceeding factor: (a) Zoo; (b) Voting; (c) Tic-tac-toe; (d) Mushroom.

distribution, there are more cheap tests than that of the normal distribution, and less cheap tests than that of the Pareto distribution.

When $\lambda = 0$, the performance of the algorithm is very poor on Zoo, Iris and Tic-tac-toe. This is because test costs are not taken into account, and the reduct obtained is often a minimal reduct instead of a minimal test cost reduct. Furthermore, results here also indicate that traditional attribute reduction algorithms are inappropriate for the minimal test cost reduct problem. The reason why it always yields the optimal reduct on the Voting dataset for the normal distribution is that there are only 3 possible reducts, and a minimal reduct is also a minimal test cost reduct.

The optimal setting for λ varies between datasets but seldom between distributions; specifically, -0.5, -4, $-0.75 \sim -1$ and -6.5 for datasets Zoo, Iris, Voting and Tic-tac-toe, respectively. There is a tendency that with decreasing λ, the performance of the algorithm increases until a maximal is reached, after which it decreases. This phenomenon is obvious for Zoo, Voting, and Tic-tac-toe. In particular, FOF decreases suddenly for the uniform distribution after λ reaches -6.5.

With the best setting of λ, however, the performance of the algorithm still seems unsatisfactory. The finding optimal factor is only 0.67 for the normal distribution and 0.88 for the Pareto distribution. Although there is a need to design new algorithms to obtain better performance, we still argue that our algorithm is not so weak. Fig. 2 provides some evidence.

Note that $\lambda = 0$ is not illustrated. The associated data are listed in Table 7. This is because the performance of the algorithm is very poor, and their inclusion would make other data hard to distinguish. For example, in Fig. 2(a), the maximal exceeding factor is always less than 0.7. It is 9.67 when $\lambda = 0$.

Table 7
Results for $\lambda = 0$, λ with the optimal setting, and λ with a number of choices.

Dataset	Distribution	Finding optimal factor			Maximal exceeding factor			Average exceeding factor		
		$\lambda = 0$	$\lambda = \lambda^*$	$\lambda \in L$	$\lambda = 0$	$\lambda = \lambda^*$	$\lambda \in L$	$\lambda = 0$	$\lambda = \lambda^*$	$\lambda \in L$
Zoo	Uniform	0.077	0.760	0.903	2.593	0.391	0.391	0.30406	0.01649	0.00462
	Normal	0.095	0.701	0.856	0.275	0.127	0.107	0.05553	0.01076	0.00311
	Pareto	0.479	0.992	0.999	9.667	0.167	0.167	0.25798	0.00114	0.00008
Voting	Uniform	0.837	0.924	0.997	0.269	0.191	0.053	0.00940	0.00303	0.00006
	Normal	1.000	1.000	1.000	0.000	0.000	0.000	0.00000	0.00000	0.00000
	Pareto	0.896	0.953	0.999	5.353	0.313	0.071	0.02466	0.00671	0.00007
Tic-tac-toe	Uniform	0.121	0.972	0.976	0.601	0.121	0.025	0.12058	0.00019	0.00017
	Normal	0.119	0.856	0.864	0.128	0.066	0.036	0.03076	0.00113	0.00109
	Pareto	0.209	1.000	1.000	8.875	0.000	0.000	0.29494	0.00000	0.00000
Mushroom	Uniform	0.029	0.484	0.699	8.742	0.952	0.672	0.74323	0.08066	0.02974
	Normal	0.082	0.219	0.341	0.572	0.544	0.402	0.10014	0.14176	0.07522
	Pareto	0.365	0.712	0.750	19.752	0.500	0.500	0.47205	0.06983	0.06118

Fig. 2 shows the results of maximal exceeding factors. There the optimal settings of λ change. For example, for the Zoo dataset with the Pareto distribution, the MEF is minimal when $\lambda = -1$ instead of -0.5. If we set $\lambda = -0.5$, however, the maximal exceeding factor is only slightly greater than its minimal. It is about 0.47, that is, 47% more than that of the optimal reduct. Recall that this value is maximal for all 4000 test cost sets and remains acceptable.

For the Tic-tac-toe dataset with the uniform distribution, there is no obvious trend in the maximal distribution factor. This is due to some degree of randomness as this factor is based on one particular reduct.

Fig. 3 shows the average exceeding factor. Here the optimal setting for λ is very close, if not equal, to that for the finding optimal factor. This phenomenon indicates that from a statistical point of view, we only need to obtain the optimal setting for the finding optimal factor. The average exceeding factor is very low for the optimal setting. For example, it is about 0.015 for the Zoo dataset with the Pareto distribution. That is, constructed reducts have only 1% to 2% more test costs than that of the optimal reduct on average. This performance is satisfactory in applications.

In practice, however, there is seldom any additional information to indicate the optimal setting of λ. From Figs. 1–3, we know that $\lambda = -1$ is an acceptable setting for all datasets and test cost distributions tested.

4.2.3. Comparison of three approaches

Now we compare the performance of the three approaches mentioned in Section 3. All three are based on Algorithm 1. The first approach, called the non-weighting approach, is implemented by setting $\lambda = 0$. The second approach, called the best λ approach, is to choose the best λ value as depicted in Figs. 1–3. The third approach is the competition approach as discussed in Section 3.2. L contains λ values indicated in Figs. 1–3.

Table 7 lists results for all three approaches. We observe the following:

(1) The non-weighting approach only performs well on the Voting dataset, which has only 3 reducts. In all three other datasets, the results are unacceptable. Therefore the non-weighting approach is not suitable for the minimal test-cost reduct problem.
(2) The best λ approach obtains good results in most cases. If it fails to find the optimal reduct, the constructed reduct is still acceptable. Unfortunately, in applications we have no idea how to obtain the best λ. Therefore $\lambda = -1$ might be a rational setting, as depicted in Figs. 1–3.
(3) The competition approach significantly improves the quality of results, especially on datasets with many reducts. As an example, we consider the Zoo datasets with the Uniform distribution test cost. Compared with the best λ approach, the finding optimal factor increases 14.3%; and the average exceeding factor decreases to about 1/4. It is much easier to specify L than to guess the best λ. We can specify as many values as we want, as long as the running time is acceptable.

4.2.4. Discussions

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

(1) Algorithm 1 is better than the minimal reduct algorithm. Table 7 shows that the minimal reduct algorithm only obtains good results on the Voting dataset. However, this is partly due to the fact that the Voting dataset has only three reducts. With a good setting of λ, the quality of the result can be significantly improved. Therefore we find that Algorithm 1 is appropriate for the minimal test cost reduct problem.
(2) There is no optimal setting of λ that is valid for any dataset. There is, however, always a tradeoff for λ, but $\lambda = -1$ seems to be an acceptable setting for all datasets we tested.

(3) The test cost distribution influences the quality of the result. Specifically, when the test cost distribution is bounded Pareto, the algorithm has a greater probability of finding the optimal reduct.
(4) The competition approach can improve the quality of results. The improvement is significant, especially on datasets with many reducts. With this approach, λ values are easy to set. We set it to be -4.75 to 0 with step length 0.25 and obtain good results. In this way, $L = 20$ and Algorithm 1 always run 20 times. This is a good setting for all datasets we tested, and bigger L does not help improving the quality of results further.

A closely related problem was addressed by Susmaga [34] in 1999. Our work has a number of differences from his. First, we indicated the condition on which the problem is meaningful. Second, the problem in [34] is essentially restricted to TCI-DS, and our is for more general SICT-DS. Third, we designed a heuristic algorithm, instead of an exhaustive algorithm, to deal with the problem.

5. Conclusions and further works

This study has posited a new research theme in regard to attribute reduction. We formally defined the minimal test cost reduct problem, which is more general than the traditional reduct problem. The new problem has practical areas of application. In fact, when tests must be undertaken in parallel, test-cost-sensitive attribute reduction is mandatory in dealing independently with cost issues. Algorithm 1 is a framework and one can design different attribute significance functions to obtain a substantive algorithm. We also proposed an information gain based λ-weighted function. Experimental results indicate that the competition approach is a good choice even if the optimal setting of λ is known.

The following research topics deserve further investigation:

(1) Algorithm 1 can be enhanced to provide better performance. To improve the quality of results, one could design approaches other than the simple weighting indicated by Eq. (23). To improve the speed of the algorithm such that it can be employed in very large databases, one could use the accelerator proposed by Qian et al. [30], or efficient genetic algorithms [18,35,53].
(2) The minimal test cost reduct should be considered again in more complicated models such as the simple common-test-cost decision systems and the complex common-test-cost decision systems [25]. In these models, the algorithm may also be more complicated.
(3) Since this paper only considers test costing, one may also consider the misclassification cost under the framework of decision-theoretic rough set model [46].

We note that the major contribution of the paper is in the definition of the problem rather than the algorithm. As we known, usually the problem formulation is more important than the problem solving. We hope that this study opens a new door for rough sets research.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by the National Natural Science Foundation of China, Grant Nos. 60873077 and 60903110.

References

[1] Generating Gaussian random numbers. <http://www.taygeta.com/random/gaussian.html>.
[2] Wikipedia. <http://www.wkipedia.org>.
[3] J.G. Bazan, A. Skowron, Dynamic reducts as a tool for extracting laws from decision tables, in: ISMIS'94, Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems, 1994.
[4] J.G. Bazan, M. Szczuka, The RSES homepage, 1994–2005. <http://alfa.mimuw.edu.pl/~rses>.
[5] J.G. Bazan, M.S. Szczuka, RSES and RSESlib – a collection of tools for rough set computations., in: Rough Sets and Current Trends in Computing, LNCS, vol. 2005, 2000, pp. 106–113.
[6] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/mlrepository.html>.
[7] D. Chen, C. Wang, Q. Hu, A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets, Information Sciences 177 (17) (2007) 3500–3518.
[8] M. Dash, H. Liu, Consistency-based search in feature selection, Artificial Intelligence 151 (2003) 155–176.
[9] D. Deng, Parallel reduct and its properties, Granular Computing (2009) 121–125.
[10] D. Deng, J. Wang, X. Li, Parallel reducts in a series of decision subsystems, Computational Sciences and Optimization (2) (2009) 377–380.
[11] P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning 29 (1997) 103–130.
[12] W. Fan, S. Stolfo, J. Zhang, P. Chan, Adacost: Misclassification cost-sensitive boosting, in: Proceedings of the 16th International Conference on Machine Learning, 1999, pp. 97–105.
[13] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI Magazine 17 (1996) 37–54.
[14] J.H. Friedman, W. Stuetzle, Projection pursuit regression, Journal of the American Statistics Association 76 (1981) 817–823.
[15] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics (1986) 122–128.
[16] M. Harchol-Balter, Task assignment with unknown duration, Journal of the ACM 49 (2002) 260–288.
[17] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[18] Q. Hu, W. Pan, S. An, P. Ma, J. Wei, An efficient gene selection technique for cancer recognition based on neighborhood mutual information, International Journal of Machine Learning and Cybernetics 1 (1-4) (2010) 63–74.

[19] E.B. Hunt, J. Marin, P.J. Stone (Eds.), Experiments in Induction, Academic Press, New York, 1966.
[20] R. Jensen, Q. Shen, Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches, IEEE Transactions on Knowledge and Data Engineering 16 (12) (2004) 1457–1471.
[21] Q. Liu, L. Chen, J. Zhang, F. Min, Knowledge reduction in inconsistent decision tables, Advanced Data Mining and Applications 4093/2006 (2006) 626–635.
[22] Q. Liu, F. Li, F. Min, M. Ye, G. Yang, An efficient reduction algorithm based on new conditional information entropy, Control and Decision (in Chinese) 20 (8) (2005) 878–882.
[23] Z. Meng, Z. Shi, A fast approach to attribute reduction in incomplete decision systems with tolerance relation-based rough sets, Information Sciences 179 (16) (2009) 2774–2793.
[24] F. Min, X. Du, H. Qiu, Q. Liu, Minimal attribute space bias for attribute reduction, Rough Set and Knowledge Technology (2007) 379–386.
[25] F. Min, Q. Liu, A hierarchical model for test-cost-sensitive decision systems, Information Sciences 179 (14) (2009) 2442–2452.
[26] F. Min, Q. Liu, H. Tan, L. Chen, The *M*-relative reduct problem, Rough Set and Knowledge Technology (2006) 170–175.
[27] F. Min, W. Zhu, Coser: Cost-senstive rough sets, 2011. <http://grc.fjzs.edu.cn/~fmin/coser/index.html>.
[28] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences 11 (1982) 341–356.
[29] Z. Pawlak, Rough sets and intelligent data analysis, Information Sciences 147 (12) (2002) 1–12.
[30] Y. Qian, J. Liang, W. Pedrycz, C. Dang, Positive approximation: an accelerator for attribute reduction in rough set theory, Artificial Intelligence 174 (9-10) (2010) 597–618.
[31] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[32] J.R. Quinlan (Ed.), C4.5 Programs for Machine Learning, Morgan Kaufmann Publisher, San Mateo, California, 1993.
[33] A. Skowron, C. Rauszer, The discernibility matrics and functions in information systems, in: Intelligent Decision Support, 1992.
[34] R. Susmaga, Computation of minimal cost reducts, in: Foundations of Intelligent Systems, LNCS, vol. 1609, Springer, 1999, pp. 448–456.
[35] D. Tong, R. Mintram, Genetic algorithm-neural network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection, International Journal of Machine Learning and Cybernetics 1 (1–4) (2010) 75–87.
[36] P.D. Turney, Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal of Artificial Intelligence Research 2 (1995) 369–409.
[37] V. Vapnik, A. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and Its Application 16 (2) (1971) 264–280.
[38] G. Wang, Attribute core of decision table, in: Rough Sets and Current Trends in Computing, LNCS, vol. 2475, 2002, pp. 213–217.
[39] G. Wang, H. Yu, D. Yang, Decision table reduction based on conditional information entropy, Chinese Journal of Computers 2 (7) (2002) 759–766.
[40] X.-Z. Wang, C.-R. Dong, Improving generalization of fuzzy if–then rules by maximizing fuzzy entropy, IEEE Transactions on Fuzzy Systems 17 (3) (2009) 556–567.
[41] X.-Z. Wang, J.-H. Zhai, S.-X. Lu, Induction of multiple fuzzy decision trees based on rough set technique, Information Sciences 178 (16) (2008) 3188–3202.
[42] J. Wróblewski, Finding minimal reducts using genetic algorithms, in: International Workshop on Rough Sets Soft Computing at Second Annual Joint Conference on Information Sciences, 1995, pp. 186–189.
[43] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge Information Systems 14 (2008) 1–37.
[44] C. Xu, F. Min, Weighted reduction for decision tables, in: Fuzzy Systems and Knowledge Discovery, 2006, pp. 246–255.
[45] C. Xu, F. Min, Weighted reduction and its application in automatic correction of chinese part-of-speech tagging (in Chinese), Control and Decision 22 (7) (2007) 744–747.
[46] Y. Yao, Y. Zhao, Attribute reduction in decision-theoretic rough set models, Information Sciences 178 (17) (2008) 3356–3373.
[47] Y. Yao, Y. Zhao, J. Wang, On reduct construction algorithms, in: Rough Set and Knowledge Technology, 2006, pp. 297–304.
[48] W. Yi, M. Lu, Z. Liu, Multi-valued attribute and multi-labeled data decision tree algorithm, International Journal of Machine Learning and Cybernetics 2 (2) (2011) 67–74.
[49] W. Zhang, J. Mi, W. Wu, Knowledge reductions in inconsistent information systems, Chinese Journal of Computers 26 (1) (2003) 12–18.
[50] K. Zhao, J. Wang, A reduction algorithm meeting users' requirements, Journal of Computer Science and Technology 17 (2002) 578–593.
[51] Y. Zhao, F. Lou, S. Wong, Y. Yao, A general definition of an attribute reduct, in: Rough Sets and Knowledge Technology, LNAI, vol. 4481, 2007, pp. 101–108.
[52] Z. Zhou, X. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.
[53] J. Zhu, X. Li, W. Shen, Effective genetic algorithm for resource-constrained project scheduling with limited preemptions, International Journal of Machine Learning and Cybernetics (2011) 1–11.
[54] W. Zhu, F. Wang, Reduction and axiomization of covering generalized rough sets, Information Sciences 152 (1) (2003) 217–230.
[55] W. Zhu, F. Wang, On three types of covering rough sets, IEEE Transactions on Knowledge and Data Engineering 19 (8) (2007) 1131–1144.
[56] W. Ziarko, Variable precision rough set model, Journal of Computer and System Sciences 46 (1) (1993) 39–59.