Attribute reduction of data with error ranges and test costs

Fan Min*, William Zhu

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

1. Introduction

Cost-sensitive learning is among the most challenging problems in data mining [62]. It has attracted much research interest from different data-mining societies working on topics such as decision trees (see, e.g., [13,30,55]), artificial neural networks (see, e.g., [24,68]), rough set (see, e.g., [37,38,65,66]), and Bayes networks (see, e.g., [6]). Most work addresses misclassification costs [6,24,32,65,66,68], but few address test costs [37,38,55], and even fewer address both [30,55].

Test cost is the time, money, or other resources one pays for obtaining a data item of an object. The topic has drawn our attention recently due to its broad applications. Based on data models discussed in [38], a number of situations have been identified, and respective problems have been defined. Specifically, test-cost-sensitive reduct problems were defined in [16,17,37]. These problems aim at finding a test set minimizing the test cost, while preserving the discernibility of the original decision system. The test cost constraint issue was introduced in [39,41] to address the situation where the total test cost one can afford is limited. The optimal sub-reduct problem was defined for this situation. Moreover, the problem was reconsidered in the dynamic environment, where both test costs and the constraint can change over time [40]. In these works, backtrack and heuristic algorithms were implemented and compared through an open source software Coser [42].
Observational error is the difference between a measured value of the data item and its true value. In applications, we do not know exactly the observation error; however, the error range is usually known. For example, we often use $37.7 \pm 0.1$ °C to indicate the body temperature of a person, where $37.7$ °C is the observed value, and $\pm0.1$ °C is the error range. Other forms of observational error exist for more complicated data. For example, we use $256 \times 256$ images to store the CT (computed tomography) information. We could use $\pm \frac{0.1}{256}$ to represent the error range; however, the physical meaning is quite different from the body temperature. Data with observational error is a form of uncertain data that has become a hot topic in data mining (see, e.g., [1,7]).

A number of measurement methods could exist to obtain a data item. For example, one can use a mercury thermometer, an electronic thermometer, or an infrared thermometer to test the body temperature. These methods require different test costs; specifically, the respective time requirements are 5 min, 1 min, and 2 s. As a result, the error ranges are also different. The infrared thermometer has the widest error range. Throughout this paper, we assume that the test cost of each data item, and the error range of each measurement method are known. We use the term “test” to indicate the process of obtaining a data item using one particular measurement method. It also refers to an attribute in an information system, or a conditional attribute in a decision system. Different measurement methods for the same data are viewed as different tests, and correspond to different attributes.

For classification purposes, we do not have to undertake all tests, nor do we have to choose the most sophisticated measurement method for each data item. Instead, we would like to find out the most economical test set which is sufficient to make the decision. This motivates us to define a new problem for this kind of test sets. In fact, the motivation of the new problem is similar to the work in [37]. However, the data under consideration are nominal ones from [37]. In contrast, this work deals with numerical data; hence, the new problem has different application areas.

We build a new model, called an error-range-based covering rough set, to address the observational error issue and formalize our problem. Note that the term “covering” instead of “neighborhood” is employed because a neighborhood system always generates a covering of the universe. The concept of reducts is employed to describe information preservation. Consequently, the new problem is referred to as the test-cost-sensitive attribute reduction through error-range-based covering rough set, or TARER for brevity. In fact, there exist some neighborhood systems [11,26,29], covering rough set frameworks [5,69,70,72], neighborhood rough set frameworks [18–20,25,60], near set [47], vague set [48], and generalized rough set over fuzzy lattices [10,12,21,33], that have gained much success in both theory and application. Specifically, attribute reduction in neighborhood rough set [14,18–20] can help to increase the prediction accuracy of the classifier. Although showing similarities, the existing work in [18–20] are essentially different from ours in three ways. First, the models constructed require a user-chosen distance function and a user-specified distance threshold $\delta$. In contrast, our model is based on error ranges that are intrinsic to data. Second, the objective of reduct algorithms for these models is to improve the classification accuracy. In contrast, our objective is to minimize the total test cost. Third, $\delta$ is used in these models to adjust the neighborhood size. In contrast, in our model different tests for the same data item produce not only different error ranges, but also different observed values on one data item. For example, with one thermometer we obtained the value $37.7 \pm 0.1$ °C, whereas with another we obtain $37.71 \pm 0.03$ °C.

The main contributions of the work are fourfold. First, we build the data model to formalize the situation mentioned above. Only numerical data are considered to simplify the discussion. Second, we build the computational model, namely the error-range-based covering rough set, and define key concepts such as lower and upper approximations, positive regions, and relative reducts. Third, we redefine the attribute reduction problem under the new model. Similar to the problem defined in [37], the objective is to minimize the total test cost. Fourth, we propose a backtrack algorithm to find an optimal reduct, a heuristic algorithm to find a sub-optimal reduct, and a competition approach to improve the performance of the heuristic algorithm. Experiments on ten datasets from the UCI library undertaken with open source software Coser [42] validate the effectiveness of these algorithms.

The rest of the paper is organized as follows: Section 2 presents the data models. Error ranges and test costs are considered one after another. Section 3 presents the computational model, namely error-range-based covering rough set model. The test-cost-sensitive attribute reduction problem under the new model is also defined in this section. Next, Section 4 presents two algorithms and the competition approach. Experiments settings and results are discussed in Section 5. Finally, Section 6 presents concluding remarks and points out further research trends.

2. Data models

This section studies data models. We start from basic information systems and decision systems. Next, we introduce observational errors to tests, and propose information and decision systems with error ranges. Finally we introduce test costs and define test-cost-sensitive decision systems with error ranges.

2.1. Information systems and decision systems

Information systems and decision systems are fundamental in machine learning and data mining. These are often stored in relational databases. For completeness, these are defined below.

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
Definition 1 [64]. An information system (IS) S is the 4-tuple:

\[ S = (U, A, V = \{ V_a | a \in A \}, I = \{ I_a | a \in A \}) \]

where U is a finite set of objects called the universe, A is the set of attributes (tests), V_a is the set of values for each a ∈ A, and I_a: U → V_a is an information function for each a ∈ A.

In many applications, we have a number of decision attributes. Therefore we can construct decision trees or generate decision rules to classify unseen objects.

Definition 2 [64]. A decision system (DS) S is the 5-tuple:

\[ S = (U, C, D, V = \{ V_a | a \in C \cup D \}, I = \{ I_a | a \in C \cup D \}) \]

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 with only discrete values, V_a is the set of values for each a ∈ C ∪ D, I_a: U → V_a is an information function for each a ∈ C ∪ D.

For brevity, V and I are sometimes dropped and these systems denoted by \( S = (U, A) \) and \( S = (U, C, D) \) [45].

In most applications, D = {d}; that is, we are given only one decision attribute. If |D| > 1, we can construct |D| decision systems, each with exactly one decision attribute. An example decision system is listed in Table 1, where D = {class}. It is a subtable of the Iris dataset. Conditional attributes are normalized to help processing and comparison. One possible normalization approach is to employ the linear function \( y = (x - \text{min})/(\text{max} - \text{min}) \), where x is the initial value, y is the normalized value, min and max are the minimal and maximal value of the attribute domain, respectively. Note that normalization is neither essential to our model nor critical to our algorithm. Table 1 can be also viewed as an information system where A = {sepal-length, sepal-width, petal-length, petal-width, class}.

### 2.2. Information systems and decision systems with error ranges

Attribute values of information systems and conditional attribute values of decision systems are often obtained through certain tests. Therefore, throughout this paper, attributes of information systems and conditional attributes of decision systems will also be called tests. Observational errors are intrinsic to tests especially when values are numerical. We include this issue in our model to make it applicable to more real data.

Definition 3. An information system with error ranges (IS-ER) is the 5-tuple:

\[ S = (U, A, V, I, e) \]

where U, A, V, and I have the same meaning as Definition 1, \( e: A \rightarrow \mathbb{R}^+ \cup \{0\} \) is the maximal observational error of a ∈ A, and ±e(a) is the error range of a.

Error free tests are tests with e(a) ≡ 0. In most cases, these are tests producing nominal outputs, which are transferred to numerical ones. If all tests are error free, an IS-ER degrades to an IS. Therefore IS-ER is a generalization of IS. Note that values of decision attributes are often specified by experts. Hence decision attributes are also prone to error. This issue involves the cost of these attributes [56] and will not be discussed further in this paper.

An example observation vector is listed in Table 2. We deliberately assign large values to the maximal observation errors to help explain some concepts through examples, where the number of objects is small. In applications these values should be small enough to make the data useful.

Similarly, we extend the concept of decision systems to consider observational errors.

Definition 4. A decision system with error ranges (DS-ER) S is the 6-tuple:

\[ S = (U, C, D, V, I, e) \]

<table>
<thead>
<tr>
<th>Plant</th>
<th>Sepal-length</th>
<th>Sepal-width</th>
<th>Petal-length</th>
<th>Petal-width</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>0.24</td>
<td>0.64</td>
<td>0.10</td>
<td>0.05</td>
<td>Setosa</td>
</tr>
<tr>
<td>x_2</td>
<td>0.18</td>
<td>0.41</td>
<td>0.10</td>
<td>0.05</td>
<td>Setosa</td>
</tr>
<tr>
<td>x_3</td>
<td>0.12</td>
<td>0.50</td>
<td>0.07</td>
<td>0.05</td>
<td>Setosa</td>
</tr>
<tr>
<td>x_4</td>
<td>0.76</td>
<td>0.45</td>
<td>0.60</td>
<td>0.52</td>
<td>Versicolor</td>
</tr>
<tr>
<td>x_5</td>
<td>0.35</td>
<td>0.09</td>
<td>0.38</td>
<td>0.43</td>
<td>Versicolor</td>
</tr>
<tr>
<td>x_6</td>
<td>0.65</td>
<td>0.32</td>
<td>0.52</td>
<td>0.52</td>
<td>Versicolor</td>
</tr>
<tr>
<td>x_7</td>
<td>0.59</td>
<td>0.35</td>
<td>0.86</td>
<td>1.00</td>
<td>Virginica</td>
</tr>
<tr>
<td>x_8</td>
<td>0.44</td>
<td>0.27</td>
<td>0.64</td>
<td>0.71</td>
<td>Virginica</td>
</tr>
<tr>
<td>x_9</td>
<td>0.82</td>
<td>0.41</td>
<td>0.83</td>
<td>0.81</td>
<td>Virginica</td>
</tr>
</tbody>
</table>

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
where $U, C, D, V$ and $I$ have the same meaning as Definition 2, $e : C \rightarrow \mathbb{R}^+ \cup \{0\}$ is the maximal observational error of $a \in C$, and $\pm e(a)$ is the error range of $a$.

Note that the concept of observational error is not applicable to decision attributes which are discrete. Tables 1 and 2 represent a DS-ER. Obviously, DS-ER is a generalization of DS.

In some applications, error ranges can be asymmetric. For example, let the true value be $v$, the observed value be $v'$, we may require $v' \in [v, v + e]$. In other words, the value should never be underestimated. Because it is straightforward to adjust the model to suit such kind of conditions, we only consider the one mentioned in Definition 4.

Another observation is that error ranges at the boundaries are different. For example, let the upper bound of a thermometer be 100 °C and the error range be ±0.1 °C. When the true value is 100 °C, the observed value is in the range [99.9 °C, 100 °C] instead of [99.9 °C, 100.1 °C]. Fortunately, the boundaries influence the data generation, but not the data model, nor the algorithms processing the data. Therefore this issue will not be addressed further.

2.3. Test-cost-independent decision system with error ranges

Test costs are also intrinsic to data. We introduce that next into the data model. For brevity, we only discuss models based on decision systems. One can follow the discussion for information systems.

**Definition 5.** A test-cost-independent decision system with error ranges (TCI-DS-ER) $S$ is the 7-tuple:

$$ S = (U, C, D, V, I, e, c), $$

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

Again we only consider the simplest case where test costs are independent. More complicated situations can be found in [38], and one can build more sophisticated models on these. An example TCI-DS-ER is given by Tables 1–3. Naturally, DS-ER can be viewed as a special case of TCI-DS-ER where $c(a) = c$ for all $a \in C$ where $c$ is a constant.

Note that in this model, tests costs are not applicable to decision attributes. In some applications one could consider the cost of teachers [56], and a decision attribute will be involved. However, both the motivation and the model are different.

3. Error-range-based covering rough set

In information systems, neighborhood [19,23,26,63] is an important concept when identifying a set of objects centered on a given one. From the viewpoint of granular computing [2,27,28], a neighborhood is also called an information granule. From the viewpoint of covering rough set [70,72], a neighborhood is also called a covering element, or a block.

In most existing work, two issues are employed to identify the neighborhood. The first is the distance function, which computes how far away two objects are. Well-known distance functions include the Manhattan distance, the Euclidean distance, and the Chebychev distance; these correspond to $1$-norm, $2$-norm, and $\infty$-norm, respectively. Fig. 1 [19,20] illustrates the neighborhoods of $x$ in a two-dimension real space. A neighborhood around $x$ based on the Manhattan distance is a square with diagonals parallel to the coordinate axes; one based on the Euclidean distance is a circle, whereas one based on the Chebychev distance is a square with edges parallel to the coordinate axes.

The second is a user-specified parameter. For the $k$-nearest neighbor algorithm ($k$-NN) [9], $k$ is the parameter and the top $k$ nearest objects will be viewed as neighbors. For the neighborhood rough set model [19,20], $\delta$ is a distance parameter and objects with a distance no further than $\delta$ are viewed as neighbors. All these approaches require the user to make certain choices.

In this section, we study error-range-based covering rough set. As mentioned earlier, error ranges are intrinsic to data rather than specified by users. The new model requires neither the distance function nor the distance upper bound $\delta$, hence the user is less involved in the model building process. Note that properties shared by neighborhood systems still hold for the new model. For example, the neighborhood relation induced by the new model can also be considered as a kind of

<table>
<thead>
<tr>
<th>a</th>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e(a)$</td>
<td>0.08</td>
<td>0.09</td>
<td>0.05</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
Let \( a_i \) and \( a_j \) be the tests for the same data item, Proposition 8. Unknown. The following proposition provides an estimation.

3.1. Error-range-based covering

With Definition 3, a new type of neighborhood can be defined as follows.

**Definition 6.** Let \( S = (U, A, V, I, e) \) be an IS-ER. Given \( x_i \in U \) and \( B \subseteq C \), the neighborhood of \( x_i \) with respect to error ranges on test set \( B \) is defined as

\[
e_B(x_i) = \{ x_j \in U | \forall a \in B, |a(x_i) - a(x_j)| \leq 2e(a) \}.
\]  

(6)

Naturally, \( e_B(x_i) = U \) for any \( x_i \in U \). We explain why \( 2e(a) \) instead of \( e(a) \) was employed in Eq. (6) as the maximal distance. Let the true value of \( x \in U \) be \( a'(x) \) for \( a \in B \). Due to the observational error, \( a'(x) - e(a) \leq a(x_i) \leq a'(x) + e(a) \). In extreme cases, \( a'(x) - e(a) \) and \( a'(x) + e(a) \) can be the observed value of the same object. We have \( (a'(x) + e(a)) - (a'(x) - e(a)) = 2e(a) \); therefore, observed values with no more than a difference of \( 2e(a) \) should be viewed the same.

An observed value \( a(x) \) should have its real value \( a'(x) \) in the range \( [a(x) - e(a), a(x) + e(a)] \). Suppose that \( x_i \) falls in the neighborhood of \( x_i \), from Definition 6, we obtain

\[
|a'(x_i) - a'(x_j)| \leq 4e(a).
\]  

(7)

In other words, more objects than we expect are included in the error range. The following example gives an intuitive understanding.

**Example 7.** Suppose there are two plants with normalized sepal lengths 0.68 and 0.72, and the error range is \( \pm 0.01 \). In the worst case, observed values could be 0.69 and 0.71, respectively. Since a true sepal length of 0.70 can be read as either 0.69 or 0.71, the sepal lengths of these two plants have to be viewed the same. Here \( 0.72 - 0.68 = 0.04 = 4 \times 0.01 \). Fig. 2 gives an illustration, where neighborhoods of \( x_2 \) have observed sepal lengths in [0.69, 0.73]. This example indicates that the observational error is amplified.

From Definition 6 we also know that

\[
e_B(x_i) = \bigcap_{a \in B} e_{a}(x_i).
\]  

(8)

That is, the neighborhood \( e_B(x_i) \) is the intersection of a number of basic neighborhoods.

Sometimes we have a number of tests to obtain the same data item. Suppose some error ranges are known and others are unknown. The following proposition provides an estimation.

**Proposition 8.** Let \( a_i \) and \( a_j \) be the tests for the same data item, \( |a_j(x) - a_i(x)| \leq e' \) for any \( x \in U \). We have

\[
e(a_j) \leq e(a_i) + e'.
\]  

(9)

**Proof.** Let \( a'(x) \) be the true value of \( x \) on the test. \( a_i(x) \leq a_j(x) + e' \leq a'(x) + (e(a_i) + e') \); \( a_j(x) \geq a_i(x) - e' \geq a'(x) - (e(a_i) + e') \). Therefore \( e(a_j) \leq e(a_i) + e' \). \( \square \)
Theorem 12. Let \( S = (U, A, V, I, e) \) be an IS-ER, \( NB \) be a neighborhood relation induced by \( B \) indicates such a relationship. 

One cannot decide the error range solely from the observed data. Moreover, given two tests for the same data item, one cannot tell which test has a narrower error range.

Remark 9. Let \( a_i, a_j \in A \) be two tests for the same data item. For any \( x \in U \), the relationship between \( e(a_i) \) and \( e(a_j) \) cannot be deduced from the test results.

Nevertheless, two tests for the same data item still meet certain constraints. One such constraint is given by the following proposition.

Proposition 10. Let \( a_i, a_j \in A \) be two tests for the same data item, for any \( x \in U \),

\[
|a_i(x) - a_j(x)| \leq e(a_i) + e(a_j)
\]  

(10)

Proof. Let \( a'(x) \) be the true value of \( x \). \( a'(x) - e(a_i) \leq a_i(x) \leq a'(x) + e(a_i) \), \( a'(x) - e(a_j) \leq a_j(x) \leq a'(x) + e(a_j) \). Therefore Eq. (10) holds. □

However, the reverse of Proposition 10 does not hold. According to Proposition 8, even if two tests satisfy Eq. (10), they cannot be viewed as tests for the same data item.

The shapes of the neighborhoods are a line segment, a rectangle, and a cuboid for one-, two-, and three-dimensional spaces, respectively. One- and two-dimensional blocks are depicted in Figs. 3 and 4. Naturally, the size of the neighborhood depends on error ranges of tests, and more objects fall into the neighborhood of \( x \) for wider error ranges.

According to Definition 3, every object belongs to its own neighborhood. Consequently, for any \( B \subseteq U \), we have

1. \( \forall x \in U, e_B(x) \neq \emptyset \);
2. \( \cup_{x \in U} e_B(x) = U \).

In other words, the family of neighborhoods \( \{e_B(x) | x \in U\} \) forms a covering of the universe. This is formally given by the following theorem.

Theorem 11. Let \( S = (U, A, V, I, e) \) be an IS-ER and \( B \subseteq A \). The set \( \{e_B(x) | x \in U\} \) is a covering of \( U \).

This is why we call the model error-range-based covering rough set.

A neighborhood relation \( N_B \) induced by \( B \subseteq A \) on the universe \( U \) can be written as a relation matrix \( M(N_B) = (r_{ij})_{n \times n} \), where

\[
r_{ij} = \begin{cases} 
1, & x_i \in e_B(x_j); \\
0, & \text{otherwise.} 
\end{cases}
\]  

(11)

From Definition 6, we know that neighborhood relations are a kind of tolerance relations, which satisfy the properties of reflexivity and symmetry. Neighborhood relations draw the objects together for tolerance or indistinguishability from the viewpoint of observational error. Correlated test sets can induce correlated neighborhood relations, the following theorem indicates such a relationship.

Theorem 12. Let \( S = (U, A, V, I, e) \) be an IS-ER, \( N_B \) be a neighborhood relation induced by \( B \subseteq A \). Given \( B_1, B_2 \subseteq A \), we have

\[
N_{B_1 \cup B_2} = N_{B_1} \cap N_{B_2}.
\]  

(12)

Proof. Given \( x_1, x_2 \in U \),

\[
N_{B_1 \cup B_2}(x_1, x_2) = 1 \\
\Leftrightarrow \forall a \in B_1 \cup B_2, |a(x_1) - a(x_2)| \leq 2e(a) \\
\Leftrightarrow \forall a \in B_1, |a(x_1) - a(x_2)| \leq 2e(a) \text{ and } \forall a \in B_2, |a(x_1) - a(x_2)| \leq 2e(a) \\
\Leftrightarrow N_{B_1}(x_1, x_2) = 1 \text{ and } N_{B_2}(x_1, x_2) = 1.
\]

This completes the proof. □

\[
\begin{array}{cccc}
& a(x) - 2e(a) & a(x) & a(x) + 2e(a) \\
\end{array}
\]

Fig. 3. One-dimensional neighborhood.
3.2. Error-range-based covering rough set

In the following context, an error-range-based neighborhood will be called a neighborhood for simplicity. If all tests are error free, namely, \( e(a) = 0 \) for any \( a \in A \), a neighborhood block degrades to an equivalence class. In this case, the objects in a neighborhood are equivalent to each other and the covering rough set model degenerates to that described by Pawlak. Therefore, the error-range-based covering rough set is a natural generalization of Pawlak’s rough set.

Note that some properties of this model look the same as those appearing in [19,20]. This is due to the fact that both models deal with neighborhood. However, since the data models are different, these properties should be restudied. As will be pointed out later, some properties in [19] are no longer valid in the new model. Now we discuss some fundamental issues of rough set in the new model.

**Definition 13.** Let \( S = (U,A,V,I,e) \) be an IS-ER, \( N_B \) be a neighborhood relation induced by \( B \subseteq A \). We call \( \langle U, N_B \rangle \) a neighborhood approximation space. For any \( X \subseteq U \), two subsets of objects, called lower and upper approximations of \( X \) in \( \langle U, N_B \rangle \), are defined as

\[
N_BX = \{ x_i \in U | e_{B}(x_i) \subseteq X \}; \tag{13}
\]

\[
\overline{N}_BX = \{ x_i \in U | e_{B}(x_i) \cap X \neq \emptyset \}. \tag{14}
\]

Obviously, \( N_BX \subseteq X \subseteq \overline{N}_BX \). The boundary region of \( X \) in the approximation space is defined as \( BNX = \overline{N}_BX - N_BX \). Note that the neighborhood relation is always pertinent to the test set \( B \). It cannot be independent as the one discussed in [19].

Generally, a finer covering (i.e., a covering with smaller blocks) is produced by a narrower error range, but a narrower error range does not necessarily produce a finer covering. Here, the corresponding statement similar to Theorem 2 in [19] does not hold. A counterexample is given below.

**Example 14.** Let \( a_1, a_2 \in A \) be two tests for sepal length, \( e(a_1) = 0.01 \), and \( e(a_2) = 0.02 \), therefore \( a_1 \) has a narrower error range; \( a \) be an error-free test for sepal length, its existence is simply to indicate the true value of the data item; \( x_1, x_2 \in U \) be two plants of interest. \( a(x_1) = 0.70, a_1(x_1) = 0.69, a_2(x_1) = 0.72, a(x_2) = a_1(x_2) = a_2(x_2) = 0.67 \).

According to **Definition 6**, \( e_{\{a_1\}}(x_i) = \{ x \in U | a_1(x) \in [0.67, 0.71] \} \), \( e_{\{a_2\}}(x_1) = \{ x \in U | a_2(x) \in [0.68, 0.76] \} \); \( x_2 \in e_{\{a_1\}}(x_1) \) but \( x_2 \notin e_{\{a_2\}}(x_1) \). In other words, with test \( a_1 \), \( x_2 \) is a neighbor of \( x_1 \); while with test \( a_2 \) it is not.

To produce a finer covering, we need a much narrower error range. We have the following theorem instead of Theorem 2 in [19].

**Theorem 15.** Let \( S = (U,A,V,I,e) \) be an information system with error ranges, \( a_i, a_j \in A \) be two tests for the same data item where \( e(a_i) \leq \frac{1}{2} e(a_j) \). We have

1. \( \forall x \in U, e_{\{a_i\}}(x) \subseteq e_{\{a_j\}}(x) \).
2. \( \forall X \subseteq U, \overline{N}_{\{a_i\}}(X) \supseteq \overline{N}_{\{a_j\}}(X) \), \( N_{\{a_i\}}(X) \subseteq N_{\{a_j\}}(X) \).

**Proof.** (1) Let the true value of \( x \) on the test be \( a(x) \). \( e_{\{a_i\}}(x) = \{ x' \in U | a_i(x') \in [a_i(x) - 2e(a_i), a_i(x) + 2e(a_i)] \} \). Since \( a(x) - e(a_i) \leq a_i(x) \leq a(x) + e(a_i) \),

\[
[a_i(x) - 2e(a_i), a_i(x) + 2e(a_i)] \subseteq [a(x) - 3e(a_i), a(x) + 3e(a_i)]. \tag{15}
\]

Alternatively, \( e_{\{a_j\}}(x) = \{ x' \in U | a_j(x') \in [a_j(x) - 2e(a_j), a_j(x) + 2e(a_j)] \} \). Since \( a(x) - e(a_j) \leq a_j(x) \leq a(x) + e(a_j) \),

\[
[a(x) - e(a_j), a(x) + e(a_j)] \subseteq [a_j(x) - 2e(a_j), a_j(x) + 2e(a_j)]. \tag{16}
\]

Because \( e(a_i) \leq \frac{1}{2} e(a_j) \), from Eqs. (15) and (16) we know that

\[
[a_i(x) - 2e(a_i), a_i(x) + 2e(a_i)] \subseteq [a_j(x) - 2e(a_j), a_j(x) + 2e(a_j)]. \tag{17}
\]
According to Definition 6, \(e_{\{a_i\}}(X) \subseteq e_{\{a_j\}}(X)\).

(2) Assuming \(e_{\{a_i\}}(X) \subseteq X\), we have \(e_{\{a_j\}}(X) \subseteq X\). Therefore, we have \(x \in N_{\{a_j\}}(X)\) if \(x \in N_{\{a_i\}}(X)\). However, \(x\) is not necessarily in \(N_{\{a_j\}}(X)\) if we have \(x \in N_{\{a_i\}}(X)\). Hence \(N_{\{a_i\}}(X) \supseteq N_{\{a_j\}}(X)\). Similarly, we have \(N_{\{a_j\}}(X) \supseteq N_{\{a_i\}}(X)\). □

For the purpose of decision making, we are often interested in each class of the decision system.

**Definition 16.** Let \(S = (U,C,D,V,I,e)\) be a DS-ER, and \(X_1, X_2, \ldots, X_K\) be the object subsets with decisions 1 through \(K\). The lower and upper approximations of decision \(D\) with respect to \(B \subseteq C\) are defined as

\[
N_{\bar{B}}D = \bigcup_{i=1}^K N_{\bar{B}}(X_i);
\]

\[
\overline{N}_{\bar{B}}D = \bigcap_{i=1}^K \overline{N}_{\bar{B}}(X_i).
\]

The decision boundary region of \(D\) with respect to attributes \(B\) is defined as

\[
BN_{\bar{B}}D = \overline{N}_{\bar{B}}D - N_{\bar{B}}D.
\]

The lower approximation \(N_{\bar{B}}D\) is also denoted by \(POS_{\bar{B}}(D)\). One can prove that

1. \(\overline{N}_{\bar{B}}D = U\);
2. \(POS_{\bar{B}}(D) \cup BN_{\bar{B}}D = U\);
3. \(POS_{\bar{B}}(D) \cap BN_{\bar{B}}D = \emptyset\).

In other words, the positive region contains objects that can be certainly classified into one class. The boundary region contains objects that can be classified into two or more classes. Given a test set \(B\), the size of the positive region reflects the recognition power or characterizing power in the classification task. Therefore, the power, or more often called the significance, of \(B\) to approximate \(D\) is defined as follows:

**Definition 17.** Let \(S = (U,C,D,V,I,e)\) be a DS-ER and \(B \subseteq C\). The dependency degree of \(D\) to \(B\) is defined as

\[
\gamma_{\bar{B}}(D) = \frac{|POS_{\bar{B}}(D)|}{|U|},
\]

where \(|\cdot|\) is the cardinality of a set.

We say \(D\) completely depends on \(C\) and the DS-ER is consistent if \(\gamma_{\bar{C}}(D) = 1\); otherwise, we say \(D\) depends on \(C\) in the degree of \(\gamma_{\bar{C}}(D)\). Next, we discuss the issue in a finer granularity. In one particular neighborhood of an object, there may exist some objects with different decision from the object. We propose the following definition to address this issue.

**Definition 18.** Let \(S = (U,C,D,V,I,e)\) be a DS-ER, \(B \subseteq C\) and \(x \in U\). Any \(y \in e_{\bar{B}}(x)\) is called an inconsistent object in \(e_{\bar{B}}(x)\) if \(D(y) \neq D(x)\). The set of inconsistent objects in \(e_{\bar{B}}(x)\) is

\[
\mathcal{I}_{\bar{B}}(x) = \{y \in e_{\bar{B}}(x) | D(y) \neq D(x)\}.
\]

The number of inconsistent objects, namely \(|\mathcal{I}_{\bar{B}}(x)|\), is important in evaluating the characteristics of the neighborhood block. It also influences the quality of rule induced by the block. From Definition 18 and Eq. (8) we know that

\[
|\mathcal{I}_{\bar{B}}(x_i)| = \bigcap_{a \in \bar{B}} |\mathcal{I}_{\{a\}}(x_i)|.
\]

Now that we have presented key concepts of the new model, the following example can help to explain these.

**Table 4** The neighborhood of objects on different test sets.

<table>
<thead>
<tr>
<th>(x)</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_1,a_2)</th>
<th>(a_1,a_2,a_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>(x_1,x_2,x_3,x_4)</td>
<td>(x_1)</td>
<td>(x_1)</td>
<td>(x_1)</td>
</tr>
<tr>
<td>(x_2)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_2)</td>
<td>(x_2)</td>
<td>(x_2)</td>
</tr>
<tr>
<td>(x_3)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_1,x_2)</td>
<td>(x_1)</td>
<td>(x_1)</td>
</tr>
<tr>
<td>(x_4)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_4)</td>
<td>(x_4)</td>
<td>(x_4)</td>
</tr>
<tr>
<td>(x_5)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_5)</td>
<td>(x_5)</td>
<td>(x_5)</td>
</tr>
<tr>
<td>(x_6)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_6)</td>
<td>(x_6)</td>
<td>(x_6)</td>
</tr>
<tr>
<td>(x_7)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_7)</td>
<td>(x_7)</td>
<td>(x_7)</td>
</tr>
<tr>
<td>(x_8)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_8)</td>
<td>(x_8)</td>
<td>(x_8)</td>
</tr>
<tr>
<td>(x_9)</td>
<td>(x_1,x_2,x_3)</td>
<td>(x_9)</td>
<td>(x_9)</td>
<td>(x_9)</td>
</tr>
</tbody>
</table>

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
Example 19. Given the decision system with error ranges as indicated by Tables 1 and 2. Let $a_1 = \text{sepal-length}$, $a_2 = \text{sepal-width}$, $a_3 = \text{petal-length}$, $a_4 = \text{petal-width}$, and $D = \{d\} = \{\text{class}\}$. $c_0(x)$ is listed in Table 4, where $B$ takes values listed as column headers, and $X$ takes values listed in each row.

Furthermore, $U$ is divided into a set of equivalence classes by $d$. $U/d = \{x_1, x_2, x_3\}$, $\{x_4, x_5, x_6\}$, $\{x_7, x_8, x_9\}$). In other words, based on the decision attribute, the objects are grouped into three subsets: $X_1 = \{x_1, x_2, x_3\}$, $X_2 = \{x_4, x_5, x_6\}$, and $X_3 = \{x_7, x_8, x_9\}$. $N_B(x)$ is listed in the first part of Table 5, where $B$ takes values listed as column headers, and $X$ takes values listed in each row. Similarly, $N_B(x)$ is listed in the second part of Table 5.

From Table 5, the positive regions and boundary regions of $U$ on different test sets are $\text{POS}_{\{a_1\}}(\{d\}) = \{x_2, x_3\}$, $\text{BNE}_{\{a_1\}}(\{d\}) = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9\}$, $\text{POS}_{\{a_1, a_2\}}(\{d\}) = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $\text{BNE}_{\{a_1, a_2\}}(\{d\}) = \emptyset$, $\gamma_{\{a_1\}}(\{d\}) = \frac{2}{3}$, $\gamma_{\{a_1, a_2\}}(\{d\}) = \frac{5}{6}$. Therefore $\{a_1, a_2, a_3\}$ has the same approximating power as $C$, and $\{d\}$ depends on $\{a_1, a_2, a_3\}$ completely.

### 3.3. The minimal-test-cost reduct problem

Attribute reduction has been intensively studied by the rough set society. There are many extensions of the classical rough set model [46], such as covering [71,72], decision-theoretical [32,66], variable-precision [73], and dominance-based [15,21] rough set models. In [59], to preserve more information of the decision system, a few fuzzy reducts are also required to produce fuzzy decision trees. A number of definitions of relative reducts exist [4,19,44,49] for different rough set models. Their relationships on consistent and inconsistent decision systems are closely studied [36]. Similar to the classical definition [44], we propose a new one based on the positive region in the new model.

**Definition 20.** Let $S = (U,C,D,V,I,e)$ be a DS-ER. Any $R \subseteq C$ is a decision-relative reduct iff:

1. $\text{POS}_{\{1\}}(D) = \text{POS}_{\{C\}}(D)$, and
2. $\forall a \in R, \text{POS}_{\{R-a\}}(D) \subseteq \text{POS}_{\{a\}}(D)$.

The first condition guarantees that the information in terms of the positive region is preserved, and the second condition guarantees that no superfluous test is included. Therefore, a decision-relative reduct is a minimal test subset with the same approximating power as the whole test set. With this concept, decision-relative core is naturally defined as follows.

**Definition 21.** Let $\text{Red}(S)$ denote the set of all decision-relative reducts of a DS-ER $S$. $\text{Core}(S) = \cap \text{Red}(S)$ is called the core of $S$.

In other words, $\text{Core}(S)$ contains those tests appearing in all decision-relative reducts. A decision-relative reduct (core) is also called a reduct (core) for brevity. Definitions 20 and 21 have the same form as the classical one [43,44]. However, the data model has changed and the definition of positive region is different; hence, the computation is totally different.

Super-reducts [41,67] are important in the reduct constructing process. These are also useful in the new model and are therefore defined as follows.

**Definition 22.** Any $R' \subseteq C$ is a decision-relative super-reduct iff $\text{POS}_{\{R'\}}(D) = \text{POS}_{\{C\}}(D)$.

In other words, we can remove the second condition in Definition 20 to obtain this concept. We also have the following proposition, which could serve as an alternative definition of super-reducts.

**Proposition 23.** Any $R' \subseteq C$ is a decision-relative super-reduct iff $\exists R \in \text{Red}(S)$ st. $R \subseteq R'$.

From Definition 18 we know immediately that given $B \subseteq C$, $x \in \text{POS}_{\{C\}}(D)$ if and only if $\text{ic}_D(x) = \emptyset$. Consequently, we have the following proposition, which can be employed as an alternative definition of a reduct.

**Proposition 24.** Let $S = (U, C, D, V, I, e)$ be a DS-ER. Any $R \subseteq C$ is a decision-relative reduct iff:

1. $\forall x \in \text{POS}_{\{C\}}(D)$, $\text{ic}_D(x) = \emptyset$, and

<table>
<thead>
<tr>
<th>$X$</th>
<th>${a_1}$</th>
<th>${a_1, a_2}$</th>
<th>${a_1, a_2, a_3}$</th>
<th>${a_1, a_2, a_3, a_4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_B(X)$</td>
<td>$x_1$</td>
<td>${x_2, x_3}$</td>
<td>${x_1, x_2, x_3}$</td>
<td>${x_1, x_2, x_3}$</td>
</tr>
<tr>
<td></td>
<td>$x_2$</td>
<td>$\emptyset$</td>
<td>${x_1}$</td>
<td>${x_4, x_5, x_6}$</td>
</tr>
<tr>
<td></td>
<td>$x_3$</td>
<td>$\emptyset$</td>
<td>${x_1}$</td>
<td>${x_7, x_8, x_9}$</td>
</tr>
<tr>
<td>$N_B(X)$</td>
<td>$x_4$</td>
<td>${x_1, x_2, x_3}$</td>
<td>${x_1, x_2, x_3}$</td>
<td>${x_1, x_2, x_3}$</td>
</tr>
<tr>
<td></td>
<td>$x_5$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
</tr>
<tr>
<td></td>
<td>$x_6$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
<td>${x_1, x_2, x_3, x_4, x_5, x_6}$</td>
</tr>
</tbody>
</table>

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
Theorem 27 shows that dependency increases monotonically with tests, which means that by adding new tests the dependency never decreases. Therefore we can employ the “depth-first-like” exhaustive algorithm [51] to construct optimal reducts. We can also adopt the addition, deletion, or addition-deletion strategies [67] to design our heuristic reduction algorithms.

3.4. Evaluation metrics

For brevity, the minimal-test-cost reduct will be called from hereon the optimal reduct. We can design many algorithms to deal with the minimal-test-cost reduct problem. Consequently, we need a number of metrics to evaluate the performance of these algorithms. We adopt the four metrics proposed in [37] for this purpose. These are finding optimal factor, exceeding factor, maximal exceeding factor, and average exceeding factor.

Given K datasets (TCS-DS-ERS), an attribute reduction algorithm A produces exactly one reduct for each dataset. Suppose that k out of K reducts are optimal for respective datasets. The finding optimal factor (FOF) is defined as

\[ \text{op} = \frac{k}{K}. \]  

(24)

This metric is both qualitative and quantitative. First, it only counts optimal solutions. Second, it is computed statistically on K datasets. In our experiments, we generated different test costs for the same DS-ER to produce different TCS-DS-ERS. Therefore we have enough data to obtain the finding optimal factor for statistics purposes.

For a TCS-DS-ER, let R’ be an optimal reduct. The exceeding factor of a reduct R is

\[ \text{ef}(R) = \frac{c(R) - c(R')}{c(R')}. \]  

(25)

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.
Suppose again the algorithm $A$ is run on $K$ datasets. On the $i$th dataset ($1 \leq i \leq K$), the reduct produced by the algorithm is denoted $R_i$. The maximal exceeding factor (MEF) is defined as

$$\max_{i \in [1, K]} \text{ef}(R_i).$$

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

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

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.

4. Algorithms

As mentioned in the last section, the minimal-test-cost reduct problem is more complex than the traditional reduct problem [43, 52]. Hence heuristic algorithms are needed to find sub-optimal reducts for large datasets. To evaluate the performance of a heuristic algorithm in terms of the quality of the solution, we should find an optimal reduct first. Consequently, exhaustive algorithms are also needed.

This section presents for the new problem both exhaustive and heuristic algorithms: the exhaustive is based on backtracking where pruning techniques are crucial in reducing computation, whereas the heuristic has a framework similar to that proposed in [37], though the algorithm is totally different due to the new data model. We also present the competition approach [37], which is still valid for the new environment to enhance heuristic algorithms.

4.1. The backtrack reduction algorithm

A general algorithm for finding all (or some) solutions to some computational problem is backtracking, which we employ to deal with minimal-test-cost reduct problems. As a general approach, we start from the empty set, and add tests one by one until a super-reduct is obtained. Then we backtrack to a former step and obtain other super-reducts. This process can be illustrated through a state space tree with exactly $2^{|C|}$ nodes, each corresponding to a test subset. Therefore one can traverse the tree in preorder and find the optimal solution in $2^{|C|}$ steps. Example 28 illustrates the case for 4 tests.

Example 28. Let $C = \{a_1, a_2, a_3, a_4\}$. The state space tree is depicted in Fig. 5. The traversal of the tree produces test subsets in the following sequence: $\emptyset, \{a_1\}, \{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\}, \{a_1, a_3\}, \{a_1, a_3, a_4\}, \{a_1, a_4\}, \{a_2\}, \{a_2, a_3\}, \{a_2, a_3, a_4\}, \{a_2, a_4\}, \{a_3\}, \{a_3, a_4\}, \{a_4\}.

This approach is, however, rather time consuming and unacceptable in most applications. The key issue in performing this algorithm is how to prune the state space tree, or in other words, knowing when to stop searching a subtree if no better solution exists inside. This section discusses this issue and presents three techniques for pruning.

If a test is redundant, we need not choose it in the process of reduct constructing. This consideration alludes to the first technique in pruning the state space tree. Formally, given a test set $B \subseteq C$ and a test $a \in C - B$, if $a$ is redundant with respect to $B$, there does not exist a reduct $R$ such that $B \cup \{a\} \subseteq R$. It is, however, not straightforward to tell whether or not a test is redundant. Intuitively, one may expect that if $\text{POS}_{B \cup \{a\}}(D) = \text{POS}_B(D)$, $a$ is redundant with respect to $B$. The following example shows that this condition is insufficient.

Example 29. A DS-ER is given by Table 6 and $e(a_1) = e(a_2) = 0.1 \text{POS}_{B \cup \{a_1\}}(D) = \text{POS}_B(D) = \emptyset$. However, $a_1$ should not be viewed redundant with respect to $\emptyset$. This is because $\text{POS}_{\{a_1, a_2\}}(D) = U$, and $\{a_1, a_2\}$ is a reduct.

Fig. 5. The state space tree for $C = \{a_1, a_2, a_3, a_4\}$.
From Example 29, we observe the following. When the test set is \( \emptyset \), all objects should be viewed neighbors. Therefore \( \epsilon_0(x_1) = U \). However, when \( a_i \) is added to the test set, we have \( \epsilon_{a_i}(x_1) = \{x_1, x_2\} \subseteq U \). In other words, \( a_i \) is useful in decreasing the neighborhood. In fact, if a test does not help decrease any neighborhood, it is definitely redundant. This is given by the following lemma.

**Lemma 30.** Let \( S = (U, C, D, V, I, e) \) be a DS-ER, \( B \subset C \) and \( a \in C - B \). If \( \forall x \in \text{POSC}(D) - \text{POSB}(D) \),

\[
\epsilon_a(x) = \epsilon_{B \cup \{a\}}(x),
\]

any \( B' \) where \( B \cup \{a\} \subseteq B' \subseteq C \) is not a reduct.

In the lemma, we only consider neighborhoods of \( x \in \text{POSC}(D) - \text{POSB}(D) \), and it is required that \( \epsilon_a(x) = \epsilon_{B \cup \{a\}}(x) \). We suppose \( \text{POS}(D) \neq \text{POS}(D) \), \( B' \) is not a reduct. Now suppose \( \text{POS}(D) = \text{POS}(D) \) and let \( x \in \text{POS}(D) \). From Eqs. (23) and (29) we know that \( \epsilon_{B \cup \{a\}}(x) = \epsilon_{B \cup \{a\}}(x) = \epsilon_{B \cup \{a\}}(x) = \epsilon_{B \cup \{a\}}(x) = \emptyset \). According to Proposition 24, \( B' \) is not a reduct. \( \Box \)

Note that, unlike Lemma 30, we do not require \( x \in \text{POSC}(D) - \text{POSB}(D) \). This is because \( \forall x \in \text{POSB}(D) \), \( \epsilon_a(x) = \epsilon_{B \cup \{a\}}(x) = \emptyset \). Since \( \epsilon_a(x) = \epsilon_{B \cup \{a\}}(x) \) gives \( \epsilon_a(x) = \epsilon_{B \cup \{a\}}(x) \), Lemma 30 is also proved.

**Theorem 31.** Let \( S = (U, C, D, V, I, e) \) be a DS-ER, \( B \subset C \) and \( a \in C - B \). If \( \forall x \in \text{POSC}(D), \)

\[
\begin{align*}
\text{lessInconsistent} &= \text{false}; /\text{Less inconsistent objects induced by } a_i? \\
U' &= \emptyset; /\text{Current level boundary objects} \\
\text{for} (\text{each } x \in U') \text{ do} \\
\text{if} (\epsilon_{B \cup \{a_i\}}(x) \subseteq \epsilon_a(x)) \text{ then} \\
\text{lessInconsistent} &= \text{true}; \\
\text{if} (\epsilon_{B \cup \{a_i\}}(x) \neq \emptyset) \text{ then} \\
U' &= U' \cup \{x\}; \\
\text{end if} \\
\text{end if} \\
\text{end if} \\
\text{end for} \\
\text{if} (\text{lessInconsistent} == \text{false}) \text{ then} \\
\text{continue} ;/\text{ail is not useful, prune} \\
\text{else} \\
\text{if} (U' = \emptyset) \text{ then} \\
R = B \cup \{a_i\}; /\text{A better super-reduct} \\
\text{continue}; \\
\text{else} \\
\text{backtrack} (U', B \cup \{a_i\}, i + 1); /\text{Next level} \\
\text{end if} \\
\text{end if} \\
\text{end if}
\end{align*}
\]

A backtrack algorithm to obtain one optimal reduct

**Algorithm 1.**

**Input:** \( S = (U, C, D, V, I, e) \), selected tests \( B \), last level boundary objects \( U' \), current level test index lower bound \( i \).  

**Output:** An optimal reduct is stored in the global variable \( R \).

**Method:** backtrack

```
1: for (i = 1; i < |C|; ++i) do
2:   if (c(B \cup \{a_i\}) \geq c(R)) then
3:     continue; //Not cheaper, no need to check, prune
4:   end if
5:   lessInconsistent = false; //Less inconsistent objects induced by \( a_i \)?
6:   U' = \emptyset; //Current level boundary objects
7:   for (each x \in U') do
8:     if (\epsilon_{B \cup \{a_i\}}(x) \subseteq \epsilon_a(x)) then
9:       lessInconsistent = true;
10:      if (\epsilon_{B \cup \{a_i\}}(x) \neq \emptyset) then
11:         U' = U' \cup \{x\};
12:       end if
13:     end if
14:   end for
15:   if (lessInconsistent == false) then
16:     continue; //\( a_i \) is not useful, prune
17:   else
18:     if (U' = \emptyset) then
19:       R = B \cup \{a_i\}; //A better super-reduct
20:       continue;
21:     else
22:       backtrack (U', B \cup \{a_i\}, i + 1); //Next level
23:     end if
24:   end if
25: end if
```

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
The first prune technique is implemented through the variable ‘lessInconsistent’ in the algorithm. Now we discuss the other two techniques to prune the state space tree. \( R \) is employed to record currently best super-reduct. If another set \( R_0 \) does not have less test cost, there is no need to check \( R_0 \). This technique is implemented in lines 2 through 4 of the algorithm.

Finally, tests are chosen in a bottom-up manner. That is, after including \( a_i \) in \( B \), the algorithm will not include any \( a_j \), where \( j < i \), in a deeper level. This technique cannot only prune the tree, but also guarantees the correctness of the algorithm.

In the process of running the algorithm, \( R \) might not be a reduct. Because \( U_0 = \emptyset \), \( R \) must be a super-reduct. That is, it could be a reduct, or a superset of a reduct. However, when the algorithm finishes \( R \) must be a reduct. This is because the cost of each test is non-negative, hence a super-reduct will generally be replaced by a reduct. The reduct might be a subset of the super-reduct, or another reduct with lower test cost.

4.2. The \( \lambda \)-weighted heuristic reduction algorithm

To design a heuristic algorithm, we employ an algorithm framework very similar to the one proposed in [37]. The algorithm is listed in Algorithm 2. It follows the typical addition-deletion strategies [67]. It constructs a super-reduct, then reduces it to obtain a reduct. Core attributes are not computed since it is time consuming to obtain them in the new model. The algorithm is essentially different from the one in [37] for the following reasons. First, the input \( S \) is a TCI-DS-ER instead of a TCI-DS, and test results are numerical rather than nominal. Second, the computation of positive regions is totally different.

**Algorithm 2.** A general test-cost-sensitive reduction algorithm

| Input: \( S = (U,C,D,V,I,e,c) \) |
| Output: A reduct with sub-minimal test cost |
| Method: tcs-reduction |
| 1: \( B = \emptyset; \)
| //Addition |
| 2: \( CA = C; \)
| 3: while \((POS_B(D) \neq POS_E(D))\) do |
| 4: For any \( a \in CA \), compute \( f(B,a,c) \); |
| 5: Select \( a' \) with the maximal \( f(B,a',c) \); |
| 6: \( B = B \cup \{a'\}; \) \( CA = CA - \{a'\}; \)
| 7: end while |
| //Deletion |
| 8: \( CD = B; \) sort attributes in \( CD \) according to respective test cost in a descending order; |
| 9: while \((CD \neq \emptyset)\) do |
| 10: \( CD = CD - \{a'\}, \) where \( a' \) is the first element of \( CD; \)
| 11: if \((POS_{B-\{a'\}}(D) = POS_B(D))\) then |
| 12: \( B = B - \{a'\}; \)
| 13: end if |
| 14: end while |
| 15: return \( B; \)

Lines 4 and 5 contain the key code of this framework. One can design different attribute significance functions to obtain respective algorithms. As discussed through Example 29, a positive region is not good heuristic information to the new problem; nor is information gain [50], which has been often employed as the heuristic information for attribute reduction (see, e.g., [37,58]), directly applicable.

According to Proposition 24 and Theorem 31, we know that \( |\text{ic}_B(x)| \) is useful in evaluating the quality of a neighborhood block. Therefore, we propose the following concepts.

---

**Table 6**

<table>
<thead>
<tr>
<th>( U )</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>0.41</td>
<td>0.81</td>
<td>Y</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0.42</td>
<td>0.42</td>
<td>N</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0.72</td>
<td>0.82</td>
<td>N</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>0.73</td>
<td>0.41</td>
<td>Y</td>
</tr>
</tbody>
</table>

---

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
in Section 5.4, this simple approach can enhance the quality of the result significantly.

For relatively small datasets, the generation of the data can be obtained using all possible settings. Constraints mentioned above are met in this process. Tests with narrower error ranges should be more expensive. In this subsection, we will discuss both the process and substantiate. The trade-off between accuracy and cost needs to be considered. Simplified models, data should not be randomly generated, but meet certain constraints. For example, for the same data item, different data distributions can be employed. Consequently, the reduction algorithm can be extensively studied. Unlike in particular applications, we will create some data for experimentation. In this way, different parameters can be specified and different results can be obtained.

5.1. Data generation

The main objective of this work is to study the performance of the reduction algorithm, rather than analyze the data for one particular application. We will create some data for experimentation. In this way, different parameters can be specified and different results can be obtained. Consequently, the reduction algorithm can be extensively studied. Unlike in simpler models, data should not be randomly generated, but meet certain constraints. For example, for the same data item, tests with narrower error ranges should be more expensive. In this subsection, we will discuss both the process and substantial settings the generation of the data. Constraints mentioned above are met in this process.

First, we choose ten datasets from the UCI library, as listed in Table 7. In this way, the datasets are not totally artificial, having certain application domains. Each dataset should contain exactly one decision attribute, and no missing value should exist. To make the data easier to handle, data items are normalized onto \([0, 1]\). Missing values are directly set to 0.5.

Second, we produce the number of additional tests for one particular data item. We use the uniform distribution generator [37] to produce integers between 0 and \(k\). That is, for each data item, we have 1 to \((k+1)\) measurement methods to obtain it; \(k\) is set to less than 5 in our experiments. In real applications, one may find data that can be obtained through more than 5 different approaches. However, commonly used ones are seldom more than 5. The number of tests for our experiments is \(|C|\) in Table 7.
Third, we produce the error ranges for each test. Since data are normalized, we set the error ranges to be ±0.005\(i\) for the \(i\)th method. That is, we assume the error range of the original data is ±0.005. This assumption may not hold for the data. However, unless specified by the original creator, the true error range is never known. The specified error ranges are reasonable in applications, a data with an error more than 0.025 (the maximal observational error of the 5th method) seems not quite useful. For such data, from Example 7, we know that two real values with 0.1 difference could be viewed the same.

Fourth, we produce “new” data subject to error ranges. Let \(a_1\) be the original (first) test, according to Proposition 8, we can add a random number in \([-0.005(i-1), -0.005(i-1)]\) to \(a_i(x)\) to produce \(a_i(x)\), where \(x \in U\). The number is generated by the uniform distribution generator [37]. In this way, \(a_i\) is the new test with error rage ±0.005\(i\). Note that if a datum smaller than 0 (or larger than 1) is produced, it is set to 0 (or 1) directly so as not to exceed the boundary.

Fifth, we produce test costs, which are always represented by positive integers. Let \(a_1\) be the original (first) test and \(a_l\) is the last test for one particular data item. \(c(a_i)\) is set to a random number in \([1,100]\) subject to the uniform distribution. \(c(a_i)\) where \(1 \leq i < l\) is set to \(2 \times c(a_{i+1})\). This setting guarantees that tests with narrower error ranges are more expensive.

An example dataset generated by this approach is listed in Tables 8 and 9. We generate one DS-ER from each dataset, and many TCS-DS-ERs from each DS-ER by setting different test costs. Note that by employing this approach we can generate as many datasets as we need.

5.2. Efficiency of the backtrack algorithm

We study the efficiency of Algorithm 1 using two metrics. One is the number of steps the program performs on the state space tree, i.e., the number of times the backtrack method is invoked. This metric is used to study the effectiveness of the pruning techniques. The other is the run-time compared with heuristic algorithms. Specifically, we employ Algorithm 2 for the comparison, where \(j\) is set to -1.

For each dataset listed in Table 7, experiments are undertaken with 100 different test cost settings. The state space tree size and the number of steps for Algorithm 1 are listed in Table 10. The average and maximal run time for both algorithms are depicted in Fig. 6, where the unit of run-time is 1 ms.

From the results we note the following:

1. With the pruning techniques, the number of steps is very small compared with the state space tree size. Therefore the pruning techniques are very effective.

| No. | Name | Domain | | \(U\) | | \(C\) | | \(C'\) | | \(D = \{d\}\) |
|-----|------|--------|---|----|---|---|---|---|
| 1   | Iris | Zoology | 150 | 4 | 10 | Class |
| 2   | Glass | Manufacture | 214 | 9 | 29 | Type |
| 3   | Wine | Agriculture | 178 | 13 | 36 | Class |
| 4   | Wgpbc | Clinic | 198 | 33 | 50 | Outcome |
| 5   | Wdbc | Clinic | 569 | 30 | 56 | Diagnosis |
| 6   | Credit | Commerce | 690 | 15 | 26 | Class |
| 7   | Image | Graphics | 210 | 19 | 34 | Class |
| 8   | Iono | Physics | 351 | 34 | 62 | Class |
| 9   | Liver | Clinic | 345 | 6 | 17 | Selector |
| 10  | Diab | Clinic | 768 | 8 | 15 | Class |

Table 7
Dataset information.

Table 8
A decision system generated from Iris.

<table>
<thead>
<tr>
<th>Plant</th>
<th>SL</th>
<th>SL-1</th>
<th>SL-2</th>
<th>SW</th>
<th>PL</th>
<th>PL-1</th>
<th>PL-2</th>
<th>PW</th>
<th>PW-1</th>
<th>PW-2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_0)</td>
<td>0.23529</td>
<td>0.23274</td>
<td>0.22699</td>
<td>0.63636</td>
<td>0.09524</td>
<td>0.09969</td>
<td>0.10144</td>
<td>0.04762</td>
<td>0.04472</td>
<td>0.04852</td>
<td>Setosa</td>
</tr>
<tr>
<td>(x_1)</td>
<td>0.17647</td>
<td>0.17657</td>
<td>0.17267</td>
<td>0.40909</td>
<td>0.09524</td>
<td>0.09599</td>
<td>0.10214</td>
<td>0.04762</td>
<td>0.04782</td>
<td>0.04662</td>
<td>Setosa</td>
</tr>
<tr>
<td>(x_2)</td>
<td>0.11765</td>
<td>0.11945</td>
<td>0.12345</td>
<td>0.50000</td>
<td>0.07143</td>
<td>0.06763</td>
<td>0.07303</td>
<td>0.04762</td>
<td>0.04327</td>
<td>0.05402</td>
<td>Setosa</td>
</tr>
<tr>
<td>(x_3)</td>
<td>0.79412</td>
<td>0.79747</td>
<td>0.79382</td>
<td>0.50000</td>
<td>0.54762</td>
<td>0.54777</td>
<td>0.55512</td>
<td>0.47619</td>
<td>0.47620</td>
<td>0.47620</td>
<td>Versicolor</td>
</tr>
<tr>
<td>(x_{51})</td>
<td>0.61765</td>
<td>0.61405</td>
<td>0.61985</td>
<td>0.50000</td>
<td>0.50000</td>
<td>0.50350</td>
<td>0.49020</td>
<td>0.52381</td>
<td>0.52381</td>
<td>0.52381</td>
<td>Versicolor</td>
</tr>
<tr>
<td>(x_{52})</td>
<td>0.76471</td>
<td>0.76861</td>
<td>0.76181</td>
<td>0.45455</td>
<td>0.59524</td>
<td>0.59209</td>
<td>0.60464</td>
<td>0.52381</td>
<td>0.52301</td>
<td>0.53351</td>
<td>Versicolor</td>
</tr>
<tr>
<td>(x_{53})</td>
<td>0.64706</td>
<td>0.64616</td>
<td>0.65396</td>
<td>0.40909</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.66667</td>
<td>0.76190</td>
<td>0.76645</td>
<td>0.7583</td>
<td>Virginica</td>
</tr>
<tr>
<td>(x_{54})</td>
<td>0.55882</td>
<td>0.56102</td>
<td>0.56172</td>
<td>0.50901</td>
<td>0.71429</td>
<td>0.71079</td>
<td>0.71079</td>
<td>0.90476</td>
<td>0.90316</td>
<td>0.91366</td>
<td>Virginica</td>
</tr>
<tr>
<td>(x_{55})</td>
<td>0.47059</td>
<td>0.47314</td>
<td>0.46729</td>
<td>0.40909</td>
<td>0.64286</td>
<td>0.64006</td>
<td>0.64186</td>
<td>0.66667</td>
<td>0.67037</td>
<td>0.66467</td>
<td>Virginica</td>
</tr>
</tbody>
</table>

SL stands for sepal length, SW stands for sepal width, PL stands for petal length, and PW stands for petal width.

1 and 2 after SL, PL, and PW indicate different revision of the original data.

There is only one method to obtain SW.
2. In about half the datasets tested, the backtrack algorithm is comparable to, or at least not significantly worse than, the heuristic algorithm. This is partly due to the fact that these datasets are not big.

3. The time expenditure does not simply rely on the dataset size. Glass and Credit are much smaller than Wdbc; however, the run-times are longer.

4. The heuristic algorithm is more stable in terms of run-time. For example, on the Credit dataset, the maximal run-time is only 1.7 times of the average one for Algorithm 2. In contrast, it is 7.3 times for Algorithm 1.

In summary, for small or medium sized datasets, the backtrack algorithm is a good choice to obtain the optimal reduct. For large datasets, however, the heuristic algorithm must be employed.

5.3. Effectiveness of the heuristic algorithm

We let \( \lambda = 0, -0.25, -0.5, \ldots, -2 \). The algorithm runs 4000 times with different test cost settings with each \( \lambda \) setting on all datasets except Glass. The backtrack algorithm takes rather a long time on Glass; hence, we only perform it 500 times. Results are depicted in Figs. 7–9. Data for \( \lambda = 0 \) are not included in figures because respective results are incomparable to others. We will discuss this issue in more detail in Section 5.4.

From the results we observe the following:

1. The quality of the results varies for different datasets. It is not simply related to the size of the dataset.
2. The quality of the results is not satisfactory in terms of the finding optimal factor. However, the average exceeding factor is less than 0.1 in most cases. In other words, the results are acceptable.
3. There is no universally optimal setting of \( \lambda \). \( \lambda = -0.5 \) might be a rational setting if no further information is available.

Although the results are generally acceptable, the performance of the algorithm should be improved. This issue will be discussed further in Section 5.4.

---

**Table 9**

A generated observation error vector and a generated test cost vector.

<table>
<thead>
<tr>
<th>( a )</th>
<th>SL</th>
<th>SL-1</th>
<th>SL-2</th>
<th>SW</th>
<th>PL</th>
<th>PL-1</th>
<th>PL-2</th>
<th>PW</th>
<th>PW-1</th>
<th>PW-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c(a) )</td>
<td>0.005</td>
<td>0.010</td>
<td>0.015</td>
<td>0.005</td>
<td>0.005</td>
<td>0.010</td>
<td>0.015</td>
<td>0.005</td>
<td>0.010</td>
<td>0.015</td>
</tr>
<tr>
<td>( e(a) )</td>
<td>52</td>
<td>26</td>
<td>13</td>
<td>80</td>
<td>372</td>
<td>186</td>
<td>93</td>
<td>360</td>
<td>180</td>
<td>90</td>
</tr>
</tbody>
</table>

**Table 10**

Number of steps for Algorithm 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree size</th>
<th>Minimal steps</th>
<th>Maximal steps</th>
<th>Average steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>( 2^{10} )</td>
<td>16</td>
<td>58</td>
<td>34</td>
</tr>
<tr>
<td>Glass</td>
<td>( 2^{29} )</td>
<td>7702</td>
<td>218,972</td>
<td>56,591</td>
</tr>
<tr>
<td>Wine</td>
<td>( 2^{36} )</td>
<td>10</td>
<td>446</td>
<td>82</td>
</tr>
<tr>
<td>Wpbc</td>
<td>( 2^{50} )</td>
<td>19</td>
<td>345</td>
<td>106</td>
</tr>
<tr>
<td>Wdbc</td>
<td>( 2^{56} )</td>
<td>63</td>
<td>3604</td>
<td>484</td>
</tr>
<tr>
<td>Credit</td>
<td>( 2^{26} )</td>
<td>1004</td>
<td>141,868</td>
<td>16,274</td>
</tr>
<tr>
<td>Image</td>
<td>( 2^{34} )</td>
<td>44</td>
<td>7551</td>
<td>907</td>
</tr>
<tr>
<td>Iono</td>
<td>( 2^{62} )</td>
<td>129</td>
<td>6471</td>
<td>1023</td>
</tr>
<tr>
<td>Liver</td>
<td>( 2^{17} )</td>
<td>194</td>
<td>1965</td>
<td>855</td>
</tr>
<tr>
<td>Diab</td>
<td>( 2^{15} )</td>
<td>129</td>
<td>1146</td>
<td>462</td>
</tr>
</tbody>
</table>

**Fig. 6.** Run time comparison: (a) maximal time, (b) average time.
5.4. Comparison of three approaches

Here we compare the performance of the three approaches mentioned in Section 4. All three are based on Algorithm 2. The first approach, called the non-weighting approach, is implemented by setting $k = 0$. The second approach, called the best $k$ approach, chooses the best $k$ value as depicted in Figs. 7–9. The third approach is the competition approach discussed in Section 4.3. $L = \{0, -0.25, -0.5, \ldots, -2\}$.

Table 11 lists results for all three approaches. We observe the following:

---

Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031
1. The non-weighting approach never finds the optimal reduct. It is unacceptable from all three metrics. This is because without considering test costs, the algorithm tends to choose tests with narrower error ranges, which are more expensive.

2. The competition approach significantly improves the quality of results especially in datasets where the optimal reduct is hard to find. For example, on the Glass dataset, the finding optimal factor of the competition approach is 0.6510, 0.224 better than the best $k$ setting.

In general, the competition approach is a simple and effective method to improve the performance of the algorithm. From the application point of view, it is also much easier to specify $L$ than to guess the best $k$. We can specify as many $k$ values as we want, as long as run-times are acceptable.

### 6. Conclusions and further works

In this paper, we addressed the TARER problem covering data model, computational model, problem and algorithm. Experimental results indicate the efficiency of the backtrack algorithm, the effectiveness of the $\lambda$-weighting heuristic algorithm, and the significance of the competition approach in terms of performance improvement. In the future, much work needs to be undertaken at four levels:

1. At the data model level, new data models addressing common-test-cost [38], different error ranges, and other types of uncertainty [1,7] can be built. These models are generally more complex than that presented in this work. These could have some interesting areas of application.

2. At the computational model level, error-range-based covering rough sets deserve more investigation. This paper only considered concepts closely related to attribute reduction. Other aspects of the theory should be studied.

3. At the problem level, other problems such as rule synthesis should be defined. Moreover, with different data models, attribute reduction problems should also be reconsidered.

4. At the algorithm level, other algorithms should be developed. One could design discernibility-matrix-based exhaustive algorithms [52], entropy-based heuristic algorithms [34,54,57], genetic algorithms [31,61], or set-covering-based algorithms [8].

In summary, this study suggests new research trends concerning covering rough set theory, the attribute reduction problem, and cost-sensitive learning applications.

### Acknowledgements

We are grateful to the anonymous reviewers for their valuable comments and suggestions. This work is in part supported by National Science Foundation of China under Grant No. 61170128, the Natural Science Foundation of Fujian Province, China under Grant No. 2011J01374 and the Education Department of Fujian Province under Grant No. JA11176.

### References


Please cite this article in press as: F. Min, W. Zhu, Attribute reduction of data with error ranges and test costs, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.04.031


