Rough Sets Approach to Symbolic Value Partition

Fan Min*,1, Qihe Liu2, and Chunlan Fang

School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 610054, China

Abstract

Symbolic value partition refers to the problem of dividing each attribute domain of a data table into a family of disjoint subsets. It could help obtaining a new data table with less attributes and smaller attribute domains, thus facilitating further machine learning processing. The optimal symbolic value partition (OSVP) problem of supervised data, where the optimal metric is defined by the cardinality sum of new attribute domains, is devoted to this issue. In this paper, we propose the concept of partition reducts for this problem, and develop an algorithm whose all possible outputs form the set of all partition reducts. The main idea is to convert the symbolic value partition problem into a series of attribute reduction problems. The optimal substructure property of the OSVP-problem and some properties of our algorithm indicate that through choosing optimal or suboptimal reducts in each turn, we can obtain a suboptimal partition reduct, which is also a suboptimal solution of the OSVP-problem. Empirical studies on some datasets of the UCI library showed that our algorithm could help computing smaller rule sets with better coverage compared with that of the reduction approach.

Key words: Rough sets, attribute reduction, continuous attribute discretization, symbolic value partition, scaling, reduct, discretization reduct and partition reduct.

* Corresponding author

Email addresses: minfan@uestc.edu.cn (Fan Min), qiheliu@uestc.edu.cn (Qihe Liu), fangcl@uestc.edu.cn (Chunlan Fang).

1 Fan Min is supported by an information distribution project under grant No. 9140A0600106DZ223 and the Youth Foundation of UESTC.
2 Qihe Liu is supported by a key problem tackling project of Sichuan province under grant No. 05GG006-004.
1 Introduction

Many problems in machine learning, pattern recognition, and signal process involve high dimensional descriptions of data. In some applications, such as text process and Web content classification, data sets are often represented in the form of information tables or decision tables with both huge numbers of attributes (in the order of tens of thousands) [1] and large cardinalities of attribute domains (in the order of hundreds). Most generalization techniques such as rule induction [2] and decision tree construction [3] on such data are rather hard, or even infeasible.

It is, therefore, desirable to develop some preprocessing techniques to reduce both the number of attributes and the cardinalities of attributes domains. Another important motivation of these techniques is that lowering the degree of precision in the data makes the data pattern more visible [4].

Attribute reduction, continuous attribute discretization and symbolic value partition are three such techniques. In the Rough Sets society, attribute reduction methods are based on the observation that attributes are not independent, thus some of them are superfluous [5]. People proposed a number definitions (see, e.g., [5][6][7][8][9]) of reducts in order to provide different levels of information reservation, along with many heuristic algorithms (see, e.g., [10][11][12]), which have been successfully applied in numerous applications. However, this technique can only reduce the number of attributes, thus unsuitable for continuous attributes or attributes with large domains [13]. Continuous attribute discretization has been investigated by a wider range of research groups (see, e.g., [14][15][16]) than that of attribute reduction, whereas there is still more works to do.

Symbolic value partition is more general, thus more complex than attribute reduction and continuous attribute discretization. In fact, both the attribute reduction problem and the continuous attribute discretization problem are special cases of the partition problem [15]. Moreover, given an information system with \( N \) attributes, each having \( M \) attribute values, the search spaces of the attribute reduction problem, the continuous attribute discretization problem and the symbolic value partition problem (or the partition problem for briefness) are \( 2^N \), \( 2^{N(M-1)} \), and \( (M!2^{M-1})^N \), respectively.

Often we are interested in optimal consistent partition schemes, where the optimal metric is defined by the cardinality sum of new attribute domains, and consistency means preserving the discernibility relation between objects from different decision classes [17]. Nguyen H.S. [15] and Nguyen S.H. [18] proposed the concept of optimal symbolic value partition (OSVP) problem and two approaches to this issue: A Decision Tree approach and a Rough
Sets approach. The first approach called MDG-method addresses the Binary Optimal Partition (BOP) problem, i.e., partition each attribute value set into two disjoint subsets, in a top-down manner until some terminating condition holds. The second approach is interesting since it converts this problem into the graph-coloring problem. But it has a few drawbacks, as will be analyzed in Section 5. In fact, the partition problem has not been thoroughly investigated, and it deserves more in-depth research.

In this paper, we propose the concept of partition reducts since finding optimal consistent partition schemes is computationally infeasible. The concepts of optimal consistent partition scheme and optimal partition reduct are equivalent. We develop an algorithm whose all possible outputs form the set of all partition reducts. The main idea is to convert the partition problem into a series of attribute reduction problems. We point out that through choosing optimal or suboptimal reducts in each turn, a suboptimal partition reduct can be gradually obtained. Moreover, since the generalization ability of symbolic value partition is stronger than that of attribute reduction, our algorithm can help obtaining smaller decision tables, smaller decision rule sets with better predication ability compared with that of some existing reduction algorithms.

The rest of the paper is organized as follows: Section 2 enumerates some basic concepts that will be used throughout the paper. Section 3 proposes an algorithm called the Reduction Based Symbolic Value Partition (RBSVP) algorithm for the partition problem. Section 4 validates our analysis through experiments. The approach of Nguyen H.S. [15] and Nguyen S.H. [18] is very interesting, but since our approach is totally different from it, we analyzed it and give a counterexample in Section 5. Finally, Section 6 concludes and points out further works.

2 Preliminaries

In this section we enumerate relative concepts including decision tables [19], partition schemes [15][18], M-reducts [20] and scaling [21]. Some well known concepts such as indiscernibility relation, positive region and relative reducts [5][17] will be employed without further explanation.

2.1 Decision Tables

Data are often presented as a table, columns of which are labelled by attributes, rows by objects of interest and entries of the table are attribute values. This paper only concerns decision tables with only one decision attribute.
Formally, a decision table is a triple $S = (U, C, \{d\})$ where $d \notin C$ is the decision attribute and elements of $C$ are called conditional attributes or conditions for briefness. Table 1 lists a decision table where $U = \{x_1, x_2, \ldots, x_9\}$, $C = \{\text{Occupation, Temperature, Cough}\}$ and $d = \text{SARS}$. It will be employed throughout the paper.

### 2.2 $M$-Relative Reduct

The number of relative reducts of a decision table may be large [22][20], and there may exist many minimal reducts of a decision table. Bazan presented the concept of dynamic reduct [6] to obtain stable reducts.

In order to obtain more preferred reducts, we presented the concept of $M$-relative reduct [20] which ensures that an attribute set specified by the user or the algorithm always be included. Since this concept will be used in Section 3, we list it as follows:

**Definition 1** Given a decision table $S = (U, C, \{d\})$ and a set of specified attributes $M \subseteq C$, any $B \subseteq C$ is called an $M$-relative reduct of $S$ iff:

1. $M \subseteq B$;
2. $\text{POS}_B(\{d\}) = \text{POS}_C(\{d\})$;
3. $\forall a \in (B - M), \text{POS}_{B - \{a\}}(\{d\}) \subseteq \text{POS}_C(\{d\})$.

The set of all $M$-relative reducts of $S$ is denoted by $\text{Red}(S, M)$.

### Table 1

An exemplary decision table $S$

<table>
<thead>
<tr>
<th>$U$</th>
<th>Occupation</th>
<th>Temperature</th>
<th>Cough</th>
<th>SARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>student</td>
<td>low</td>
<td>yes</td>
<td>suspicious</td>
</tr>
<tr>
<td>$x_2$</td>
<td>doctor</td>
<td>high</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$x_3$</td>
<td>nurse</td>
<td>high</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$x_4$</td>
<td>nurse</td>
<td>normal</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>teacher</td>
<td>normal</td>
<td>no</td>
<td>suspicious</td>
</tr>
<tr>
<td>$x_6$</td>
<td>teacher</td>
<td>normal</td>
<td>yes</td>
<td>suspicious</td>
</tr>
<tr>
<td>$x_7$</td>
<td>lawyer</td>
<td>normal</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$x_8$</td>
<td>student</td>
<td>normal</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$x_9$</td>
<td>student</td>
<td>high</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
2.3 Partitions

In order to facilitate our discussion, we use definitions with more general forms than that of Nguyen S.H. [18]. Note that most definitions are essentially equivalent to exiting ones.

Let \( S = (U, C, \{d\}) \) be a decision table where \( C = \{a_i : U \rightarrow V_{a_i}\} \) for \( i \in \{1, \ldots, |C|\} \). Any function \( P_i = P_{a_i} : V_{a_i} \rightarrow W_{a_i} \cup V_{a_i} \) where \( W_{a_i} \cap V_{a_i} = \emptyset \), \( P_i(v) \in W_{a_i} \) or \( P_i(v) = v \) is called a partition of \( V_{a_i} \). The function \( P_i \) defines a new partition attribute \( a_{P_i} = P_i \circ a_i \), i.e., \( a_{P_i}(u) = P_i(a_i(u)) \) for any object \( u \in U \). The range of \( a_{P_i} \) is \( V_{P_i} = \bigcup_{u \in U} \{a_{P_i}(u)\} \). Often \( P_i \) is also expressed by a set of value pairs, i.e., \( P_i(v_1) = v_2 \iff (v_1, v_2) \in P_i \).

For example, \( P_1 = \{(\text{student}, 1), (\text{doctor}, 2), (\text{nurse}, 2), (\text{teacher}, \text{teacher}), (\text{lawyer}, 1)\} \) is a partition of \( V_{a_1} \) where \( W_{a_1} = \{1, 2\} \) and \( a_1 \) is an attribute of \( S \) listed in Table 1. \( P_1 \) divides \( V_{a_1} \) into three subsets: \{\text{student, lawyer}\}, \{\text{doctor, nurse}\} and \{\text{teacher}\}. \( V_{P_1} = \{1, 2, \text{teacher}\} \).

Any array of partition \( P = [P_1, \ldots, P_{|C|}] \) is called a partition scheme of \( S \). \( P \) defines from \( S \) a new decision table \( S^P = (U, C^P, \{d\}) \) where \( C^P = \{a_{P_1}^1, \ldots, a_{P_{|C|}}^{|C|}\} \). The rank of \( S \) is the value \( \sum_{i=1}^{|C|} |V_{a_i}| \). The rank of \( P_{a_i} \) is the value \( \text{rank}(P_{a_i}) = |V_{a_i}| \).

Similar to the definition of a reduct [5], we propose two concepts as follows. A partition scheme \( P \) is consistent iff \( POS_{C^P}(\{d\}) = POS_C(\{d\}) \). A decision table \( S \) is unpartitionable iff there does not exist a partition scheme \( P \) such that \( P \) is consistent and \( \text{rank}(S^P) < \text{rank}(S) \).

**Definition 2** \( P \) is called a partition reduct of \( S \) iff \( P \) is consistent and \( S^P \) is unpartitionable.

The set of all partition reduct of \( S \) is denoted by \( PR(S) \). A partition reduct \( P \) is optimal iff \( \text{rank}(S^P) \) is minimal. We consider the following problem which was proven NP-hard [18]:

**Problem 3** **Optimal Symbolic Value Partition (OSVP)**

**Input:** A decision table \( S = (U, C, \{d\}) \) where all attributes are symbolic.

**Output:** An optimal partition reduct \( P \) of \( S \).
Table 2

$S_B$, the scaled decision table of $S$

<table>
<thead>
<tr>
<th>(O, s) (O, d) (O, n) (O, t) (O, l) (T, l) (T, h) (T, n) (C, y) (C, n)</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1 0 0 0 0 1 0 0 0 1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0 1 0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0 0 1 0 0 0 1 0 1 0</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0 0 1 0 0 0 0 1 1 0</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0 0 0 1 0 0 0 1 0 1</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0 0 0 1 0 0 0 1 1 0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0 0 0 0 1 0 0 1 1 0</td>
</tr>
<tr>
<td>$x_8$</td>
<td>1 0 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>$x_9$</td>
<td>1 0 0 0 0 0 0 1 0 1</td>
</tr>
</tbody>
</table>

2.4 Scaling

In some theories, especially Formal Context Analysis [21], there is a need to transform a many-valued attribute into a number of binary valued attributes. This process is called scaling [21]. Here we require that the decision attribute not to be changed in the scaling process.

**Definition 4** Given a decision table $S = (U, C, \{d\})$, the set of scaled attributes of $Q \subseteq C$ is

$$Q_B = \{(a, v) | a \in Q, v \in V_a\},$$

(1)

where $(a, v) : U \rightarrow \{0, 1\}$ and

$$ (a, v)(u) = \begin{cases} 1 & \text{if } a(u) = v; \\ 0 & \text{otherwise}. \end{cases}$$

(2)

Table 2 lists the scaled decision table $S_B = (U, C_B, \{d\})$ of $S$ listed in Table 1, where $(O, s), (O, d), \ldots, (C, n)$ stand for (Occupation, student), (Occupation, doctor), \ldots, (Cough, no), respectively.

The scaling process has no influence on the indiscernibility relations or positive regions of attribute sets, this is given by the following lemma:

**Lemma 5** Given a decision table $S = (U, C, \{d\})$ for any $i \in \{1, \ldots, |C|\}$,

$$\text{Ind}(\{a_i\}) = \text{Ind}(\{a_i\}_B),$$

(3)
\[ \text{Ind}(C) = \text{Ind}(C_B), \]  
(4)

\[ \text{POS}_{C}(\{d\}) = \text{POS}_{C_B}(\{d\}). \]  
(5)

**PROOF.** We only prove Equation (4). According to the construction of \( C_B \),

\[ \forall a_i \in C \text{ and } u_j, u_k \in U, \]

\[ a_i(u_j) = a_i(u_k) \iff \forall (a_i, v) \in C_B, (a_i, v)(u_j) = (a_i, v)(u_k). \]

Hence \( \text{Ind}(a_i) = \bigcap_{(a_i, v) \in C_B} \text{Ind}((a_i, v)) \),

\[ \text{Ind}(C) = \bigcap_{1 \leq i \leq |C|} \text{Ind}(a_i) = \bigcap_{1 \leq i \leq |C|} (\bigcap_{(a_i, v) \in C_B} \text{Ind}((a_i, v))) \]

\[ = \bigcap_{(a, v) \in C_B} \text{Ind}((a, v)) = \text{Ind}(C_B). \]

3 The Reduction Based Symbolic Value Partition Algorithm

In this section, first we explain the main idea of the algorithm through an example, then we propose the OSGP-problem as the basis of our algorithm and list the algorithm, finally we analyze some properties of the algorithm.

3.1 An Example

In order to make our algorithm easier to comprehend, we analyze the example prior to any theoretic analysis.

One can apply attribute reduction to the scaled decision table. For example, \( R^1 = \{(O, d), (O, n), (O, t), (T, 1)\} \) is a minimal reduct of \( S_B \) listed in Table 2 and a reduced decision table \((U, R^1, \{d\})\) can be obtained. However, it is more interesting to convert \((U, R^1, \{d\})\) back to a “normal” decision table as listed in Table 3, where duplicated objects are removed.

Because \((O, s) \notin R^1\) and \((O, 1) \notin R^1\), we do not distinguish student from lawyer. From semantic point of view, student and lawyer can be replace by others, for the extending purpose 1 is used instead.

In fact, a partition scheme \( P^1 = \{\{(s, 1), (d, d), (n, n), (t, t), (1, 1)\}, \{(1, 1), (h, 1), (n, 1), \{(y, 1), (n, 1)\}\} \) could be constructed such that the decision table listed in Table 3 is just \( S^{P^1} \).

Another key idea is that this scaling, reduction and converting back process (where new values such as 2, 3, …instead of 1 should be used) could be
Table 3

<table>
<thead>
<tr>
<th></th>
<th>$O^{P_1}$</th>
<th>$T^{P_1}$</th>
<th>$C^{P_1}$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>low</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>$x_2$</td>
<td>doctor</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>$x_3$</td>
<td>nurse</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>teacher</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>$x_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

repeated until the cardinality of any attribute cannot be reduced further.

$S^{P_1}_B$ is obtained as listed in Table 4. $R^2 = \{ (O^{P_1}, 1), (O^{P_1}, t), (T, 1) \}$ is a reduct of $S^{P_1}_B$ and $S^{P_2}_B$ is obtained as listed in Table 5 where $P^2 = \{ (s, 1), (d, 2), (n, 2), (t, t), (1, 1) \}, \{ (1, 2), (h, 1), (n, 1) \}, \{ (y, 1), (n, 1) \}$. $S^{P_2}_B$ is listed in Table 6. $R^3 = \{ (O^{P_2}, 1), (O^{P_2}, 2), (T^{P_2}, 1) \}$ is a reduct of $S^{P_2}_B$ and $S^{P_3}_B$ is obtained as listed in Table 7 where $P^3 = \{ (s, 1), (d, 2), (n, 2), (t, 3), (1, 1) \}, \{ (1, 2), (h, 1), (n, 1) \}, \{ (y, 1), (n, 1) \}$. Since $S^{P_3}_B$ is unpartitionable, the whole process terminates and $P = P^3$ is a partition reduct. A more comprehensive version of $S^{P_3}_B$ is listed in Table 8.

3.2 The Optimal Single Group Partition Problem

The example indicated that the whole process is essentially recursive, hence we shall focus on the first round of the process, i.e., the computation of $P^1$.

As listed in Table 3, any conditional attribute has exactly one new value, which corresponds to one or more initial values. For example, for Occupation, 1 corresponds to both student and lawyer; while for Cough, 1 corresponds to both yes and no. That is, attribute values for each attribute form exactly one (single) new group. Hence we introduce a special form of partition scheme.

**Definition 6** A partition scheme $P = [P_1, \ldots, P_{|C|}]$ of $S$ is called a single group partition scheme (SGPS) if for any $i \in \{1, \ldots, |C|\}$, $|W_{a_i}| = 1$.

Given an SGPS $P = [P_1, \ldots, P_{|C|}]$, for any $i \in \{1, \ldots, |C|\}$, $P_i$ essentially divides $V_{a_i}$ into two disjoint subsets $V_{a_i}^F$ and $V_{a_i}^G$, and

$$P_i(v) = \begin{cases} 
 v & \text{if } v \in V_{a_i}^F; \\
 k & \text{if } v \in V_{a_i}^G,
\end{cases}$$

where $k \in W_{a_i}$. According to Definition 6, $V_{a_i}^G \neq \emptyset$. Hence any SGPS $P$ can
Table 4

<table>
<thead>
<tr>
<th></th>
<th>(O^{P^1,1})</th>
<th>(O^{P^1,1})</th>
<th>(O^{P^1,n})</th>
<th>(O^{P^1,t})</th>
<th>(T^{P^1,1})</th>
<th>(T^{P^1,1})</th>
<th>(C^{P^1,1})</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th></th>
<th>(O^{P^2})</th>
<th>(T^{P^2})</th>
<th>(C^{P^2})</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>teacher</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th></th>
<th>(O^{P^2,1})</th>
<th>(O^{P^2,2})</th>
<th>(O^{P^2,t})</th>
<th>(T^{P^2,1})</th>
<th>(T^{P^2,2})</th>
<th>(C^{P^2,1})</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 7

<table>
<thead>
<tr>
<th></th>
<th>(O^{P^3})</th>
<th>(T^{P^3})</th>
<th>(C^{P^3})</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>suspicious</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

also be represented by a set of attribute-value pairs, i.e., 
\[
P = \{(a_i, v)|i \in \{1, \ldots, |C|\} \text{ and } v \in V_{a_i}^{\text{F}}\}.
\]

For example, partition scheme \(P^1\) is also an SGPS and it can be represented by 
\[
P^1 = R^1 = \{(O, d), (O, n), (O, t), (T, 1)\}.
\]

With this form of SGPS we can define single group partition reducts as follows:
Definition 7 Any SGPS $P$ is called a single group partition reduct (SGPR) of $S$ iff $P$ is consistent and any $P' \subset P$ is not consistent.

The set of SGPR of $S$ is denoted by $SGR(S)$. A SGPR $P$ is optimal iff $rank(S^P)$ is minimal. We consider the following problem:

**Problem 8 Optimal Single Group Partition (OSGP)**

**Input:** A decision table $S = (U, C, \{d\})$ where all attributes are symbolic.

**Output:** An optimal SGPR $P$ of $S$.

In fact, any SGPS $P \subseteq C_B$ is as an attribute subset of $S_B$, and $C^P$ is the set of conditional attributes of $S^P$. We have the following lemma:

**Lemma 9** For any SGPS $P \subseteq C_B$,

$$ Ind(P) = Ind(C^P). $$

**PROOF.** For any $x_i, x_j \in U$,

$$(x_i, x_j) \in Ind(P) \iff \forall (a, v) \in P, (a, v)(x_i) = (a, v)(x_j) \iff \forall a \in C, \{a\}^P(x_i) = \{a\}^P(x_j) \iff \forall a \in C, (x_i, x_j) \in Ind(\{a\}^P) \iff (x_i, x_j) \in \bigcap_{a \in C} Ind(\{a\}^P) = Ind(C^P).$$

According to Definitions 7 and Lemma 9, we have the following theorem:

**Theorem 10** $SGR(S) = Red(S_B).$ (8)

Therefore the OSGP problem of $S$ (constructing an SGPR $P$) is converted into the reduction problem of $S_B$ (selecting an attribute subset $P$ from $C_B$).

Moreover, Equation (6) indicates

$$ rank(S^P) = \sum_{1 \leq i \leq |C|} (V_{a_i}^F + 1) = |P| + |C|. $$

(9)
Finally, according to the definition of optimal metrics of reducts and SGPR, we have the following corollary:

**Corollary 11** The OSGP-problem of $S$ is equivalent with the OR-problem of $S_B$.

For example, since $R^1$ is an optimal reduct of $S_B$, according to Theorem 10, $P^1 = R^1$ is an optimal SGPS of $S$.

Also note that $R^1 \cap \{a_3\}_B = \emptyset$, hence $a_3^{P^1}(u) \equiv 1$, indicating $a_3^{P^1}$ is reduced.

Since the reduct problem is NP-complete, the follows corollary holds:

**Corollary 12** The OSGP-problem is NP-complete.

It is also easily seen that the single group partition problem is more general than the reduct problem, while more specific than the partition problem. The following theorem is straightforward:

**Theorem 13** If $V_{a_i} = 2$ for any $i \in \{1, \ldots, |C|\}$, the OSGP-problem coincides with the OR-problem.

### 3.3 The Algorithm

#### 3.3.1 Algorithm Structure

As indicated in Subsection 3.2, an (optimal) SGPR, or the set of all SGPR, of a decision table can be obtained through employing three steps, namely, scaling, reduction and converting back. But in most cases an (optimal) SGPR is not a partition reduct.

The main structure of the algorithm is, therefore, repeating these three steps until the new decision table is unpartitionable. In other words, an SGPR of the new decision table should be computed recursively and new values such as 2, 3, \ldots should be assigned to $k$ in Equation (6). By doing so a partition reduct, or even the set of all partition reducts can be obtained.

#### 3.3.2 The Usefulness of $M$-relative reduct

In the reduction step, one should not just randomly choose a reduct for the partition purpose. In the example, $R^1$ is a reduct of $S_B$, also an SGPR of $S$. $S^{R^1} = S^{P^1}$ and $\text{Rank}(S^{R^1}) < \text{Rank}(S)$. However, $R^1$ is also a reduct of $S^{P^1}$, if it is used again as an SGPR of $S^{P^1}$, $(S^{P^1})^{R^1}$ would be equivalent with $S^{P^1}$. In fact, the only difference between $S^{P^1}$ and $(S^{P^1})^{R^1}$ lies in that all 1s in the
former decision table are replaced by 2s in the later. In the worst case, the whole process would enter a dead loop if \( R_1 \) is always chosen as the SGPR of the new decision table.

Hence the concept of \( M \)-relative reduct should be introduced to control the computation of SGPRs and ensure quick converge of the algorithm. \( M \) should be deliberately set such that new attribute values introduced would never be replaced by others. In the example, since 1 has replaced student of Occupation, it should not be replaced again by any other new values. For this purpose we let \( M^2 = \{(O, 1), (T, 1)\} \) for the computation of \( R^2 \) and \( M^3 = \{(O, 1), (T, 1), (O, 2)\} \) for the computation of \( R^3 \).

Another important influence of this approach is that once all attribute-value pairs have been processed, i.e., respective values replaced by new ones, the algorithm should terminate. A two-dimension vector \( H \) will be used to record unprocessed attribute-value pairs.

### 3.3.3 The Computation of Partition Schemes

According to our discussion in Subsection 3.2, \( R_1, R_2, R_3, \ldots \) could be taken as SGPRs of \( S, S^{P_1}, S^{P_2}, \ldots \), respectively. That is, \( S^{P_1} = S^{R_1}, S^{P_2} = (S^{R_1})^{R_2}, S^{P_3} = ((S^{R_1})^{R_2})^{R_3}, \ldots \)

The computation of \( P_1, P_2, P_3, \ldots \) will be given in the algorithm.

### 3.3.4 Algorithm Description

The algorithm is listed in Algorithm. 1. It should be noted that we can compute \( S_B^{P_i} \) without computing \( S^{P_i} \), but for completeness we still list the pseudo code.

**Algorithm 1 The Reduction Based Symbolic Value Partition Algorithm**

```plaintext
ReductionBasedSymbolicValuePartition (S = (U, C, \{d\}))
{input: A decision table S.}
{output: A partition reduct P.}
//Initialize. \( M^i \) is used for \( M \)-relative reduct.
Step 1. \( M^1 = \emptyset \),
//The initial partition scheme \( P^0 \). In fact \( S^{P_0} = S \).
Step 2. \( P^0 = [P^0_1, \ldots, P^0_{|C|}] \) where \( P^0_i(v_i) = v_i \) for any \( i \in \{1, \ldots, |C|\} \) and \( v_i \in V_{a_i} \);
//Initialize unprocessed attribute-values pairs for each attribute.
//Now all attribute-values pairs are unprocessed.
Step 3. for \( (i = 1; i \leq |C|; i++) \) \( H^0_i = \{a_i\}_B \);
//Attack the OSVP-problem through attacking the OSGP-problem recursively.
```

12
Step 4. for (i = 1; ; i++) begin
//**scaling.**
Step 4.1 compute $S_B^{p_i-1}$;
//**Reduction.**
Step 4.2 $R^i = \text{an optimal } M\text{-relative reduct of } S_B^{p_i-1} \text{ where } M = M^i$;
Step 4.3 $M^{i+1} = M^i$; //Initialize $M^{i+1}$.
Step 4.4 for (j = 1; j ≤ |C|; j++) begin
//Compute $P^i$.
Step 4.4.1 $\forall (a_j, v) \notin H^{i-1}_j - R^i$, $P^i_j(v) = P^{i-1}_j(v)$;
Step 4.4.2 $\forall (a_j, v) \in H^{i-1}_j - R^i$, $P^i_j(v) = i$;
Step 4.4.3 $H^i_j = H^{i-1}_j \cap R^i$ //Remove processed attribute-values pairs
//Compute $M^{i+1}$
Step 4.4.4 if ($H^i_j \not= \emptyset$) $M^{i+1} = M^{i+1} \cup \{(a_j, i)\}$;
end //of for j;
//**Converting back to a “normal” decision table.**
Step 4.5 compute $S^p_i$ where $P^i = [P^i_1, \ldots, P^i_{|C|}]$;
//See if all attribute-values pairs have been processed
Step 4.6 if $H^i = \bigcup_{j=1}^{|C|} H^i_j = \emptyset$ break; end; //of for i
Step 5. $P = P^i$, return $P$;

3.4 Algorithm Analysis

In this subsection we analyze the set of all possible outputs of the algorithm, explain why optimal $M$-reduct should be computed in Step 4.2, and address some implementation issues.

3.4.1 Algorithm Output

In Step 4.2, we required that “$R^i = \text{an optimal } M\text{-relative reduct of } S_B^{p_i-1} \text{ where } M = M^i$”. Now we loose this requirement through removing optimal from the assessment and investigate the output of the algorithm.

Given a partition reduct $P$, it is straightforward to traceback the algorithm to construct respective $P^i$, $M^i$ and $R^i$. Hence we have the following property:

**Property 14** Any $P \in PR(S)$ can be the output of the algorithm.

On the other hand, the output of the algorithm has the following property:

**Property 15** If $R^i \in Red(S_B^{p_i-1}, M^i)$ for any $1 \leq i \leq K$, then the algorithm output satisfies $P \in PR(S)$.

Integrating these two properties we obtain:
Corollary 16

\[ \bigcup_{i \in \{1, \ldots, K \}, R^i \in \text{Red}(S^{P_{i-1}}, M^i)} \{P^K\} = PR(S). \]  

(10)

Corollary 16 indicates that if we run the algorithm enough times and choose different \( M \)-relative reducts during the process we can generally obtain \( PR(S) \). In other words, the set of all possible outputs of the algorithm is just the set of all partition reducts.

3.4.2 Optimal Substructure

Since the goal of the OSVP-problem is to construct \( P = P^K \) such that \( \text{rank}(S^{P^K}) \) is minimal, it is natural to choose locally the optimal solution, i.e., the solution of the OSGP-problem. So we required that the \( M \)-relative reduct to be optimal in Step 4.2. Therefore, the algorithm is a greedy algorithm, and we need to explain why locally optimal solution can result in globally sub-optimal solution. The following theorem gives partial reason.

Theorem 17 The OSVP-problem has the optimal-substructure property.

PROOF. Let \( P = [P_1, P_2, \ldots, P_{[C]}] \) be an optimal partition reduct of \( S \), \( P' = [P'_1, P'_2, \ldots, P'_{[C]}] \) where \( P'_i = (P_i - \{(v, 1) | (v, 1) \in P_i\}) \cup \{(1, 1) \) for \( i \in \{1, \ldots, [C]\} \), we can see that \( (S^{P_1})^{P'} = S^P \). We need to prove that \( P' \) is an optimal partition reduct of \( S^{P_1} \).

Suppose that there is another partition reduct \( P'' = [P''_1, P''_2, \ldots, P''_{[C]}] \) of \( S^{P_1} \) such that \( \text{rank}((S^{P_1})^{P''}) < \text{rank}((S^{P_1})^{P'}) \). We can then construct another partition scheme \( P^x = [P^x_1, P^x_2, \ldots, P^x_{[C]}] \) where \( P^x_i = (P^x_i - \{(v, v) | (v, v) \in P^x_i\}) \cup (P^x_i - \{(1, 1)\}) \), and \( S^{P^x} = (S^{P_1})^{P''} \). This in turn gives that

\[ \text{rank}(S^{P^x}) = \text{rank}((S^{P_1})^{P''}) < \text{rank}((S^{P_1})^{P'}) = \text{rank}(S^{P}), \]  

(11)

which means that \( P \) is not an optimal partition reduct and contradicts with the assumption.

Hence an optimal solution \( P \) of the OSVP-problem of \( S \) contains the optimal solution \( P' \) of the same problem of \( S^{P_1} \), and the proof is completed.

In Step 4.1 an optimal \( M \)-relative reduct instead of an optimal reduct is computed for two reasons. The first reason has been given in Subsection 3.3.2, the second reason is that optimal \( M \)-relative reducts computed in the algorithms are also optimal reducts. This is given by the following property:
Property 18 \( R^i \) is an optimal reduct of \( S_B^{P^i-1} \).

Unfortunately, the OSVP-problem does not have the greedy-choice property. In what follows we will discuss a property which indicate that the algorithm can compute at least suboptimal partition reducts.

In the algorithm, \( H_{i}^{j-1} \) where \( 1 \leq i \leq K \) is the set of all unprocessed attribute-value pairs before the \( i \)th round of Step 4, and \( (H_{j}^{i-1} - H_{j}^{i}) \) is the set of all attribute-value pairs processed in the \( i \)th round of Step 4. Since \( R^i \) is optimal, the following property holds.

Property 19 Given \( 1 \leq i < k \leq K \), for any \( 1 \leq j \leq |C| \),

\[
|H_{j}^{i-1} - H_{j}^{i}| \geq |H_{j}^{k-1} - H_{j}^{k}|. \tag{12}
\]

In the example, since in the first round two values of Occupation, namely, student and lawyer are partitioned into one group, we have \(|H_{1}^{0} - H_{1}^{1}| = 2\). In the two rounds followed, we have \(|H_{1}^{1} - H_{1}^{2}| = 2 \leq 2\) and \(|H_{1}^{2} - H_{1}^{3}| = 1 \leq 2\).

This property also indicates that if equation (12) does not hold, then \( R^k \) is not a optimal \( M \)-relative reduct of \( S_{P^j-1} \) and we should trace back to the \( k \)th round for a better solution of \( R^k \).

3.4.3 Complexity Analysis

In most applications, \( K \) is small. Table 10 lists some representative experimental results. Also, \( S_B^{P^i-1} \) has much less attributes and objects (duplicated objects are removed) than that of \( S_B^i \). Hence both the time and space complexities of the algorithm are determined by the the first round, i.e., the reduct computation of \( S_B \).

Any reduction algorithm could be employed. If we use the entropy method [11], the time complexity would be

\[
O(|U|^2 \text{rank}(S) + |U|^3). \tag{13}
\]

One very interesting approach is that we can borrow the idea of MD-heuristic for the reduct computation. This is because \( S_B \) contains only binary conditional attributes, thus the optimal discretization (OD) problem is equivalent with the OR-problem. If we use this approach, according to the complexity analysis in given in [15], the space complexity of our algorithm is

\[
O(|U| \text{rank}(S)), \tag{14}
\]
and the time complexity is
\[ O(rank(S)|U|(|R| + \log |U|)), \]
(15)

where \( R \) is the reduct of \( S_B \).

In most applications \(|R| < |U| \) and \( \text{Rank}(S) < |U||C| \), hence this approach is applicable.

4 Experiments with Data

In this section, we investigate the performance of the algorithm according to two aspects: 1) generalization ability, which is evaluated by \( \text{Rank}(S^P) \), and 2) predication ability, which is evaluated by both the coverage and the \( F \)-measure of respective rule sets.

Two other approaches were compared with the partition approach: 1) the basic approach, i.e., no process other than continuous attributes discretization is employed, and 2) the attribute reduction approach.

4.1 The Experimental Setup

4.1.1 Datasets Information

The 8 datasets obtained from the UC Irvine ML repository [23] to test our algorithm are:

1. Monks dataset, including three training datasets and respective testing datasets (Monk1, Monk2 and Monk3),
2. Mushroom Database;
3. Nursery Database;
4. Letter Image Recognition dataset (LetterRec), where the first 16,000 items was used for training.
5. Statlog Project Heart Disease dataset (Heart),
6. Iris Plants dataset (Iris).

Basic information of these datasets is given in Table 9, where “Cont.” denotes the number of continuous attributes in the dataset.
4.1.2 Experiment Process

Some datasets contain continuous attributes, and the discretization process was required to convert them into discrete ones. MD-heuristic implemented in RSES 2.2 was employed for this purpose. For simplicity, the decision table after discretization, instead of the initial decision table, was denoted by $S$.

The exhaustive algorithm implemented in RSES 2.2 was employed to obtain all reducts. Since different optimal reducts may produce new decision tables with different ranks, the optimal reduct $R$ with least $\text{Rank}(S^R)$, where $S^R = (U, R, \{d\})$, was chosen.

The RBSVP algorithm was employed on $S$ to produce $S^P$.

In order to keep the influence of the rule generation algorithm to a minimal, the exhaustive algorithm was employed to produce decision rules for all datasets but Mushroom, where LEM2 was employed with the cover parameter set to 1. Conflicts were resolved by standard voting.

For datasets with specified training data, the numbers of decision rules were computed based on the training data, while for other datasets they are computed based on whole datasets.

Also, for datasets without specified training sets, CV-10 were employed to obtain respective rule coverage and $F$-measure ($F$).

The evaluation metrics include rule coverage (cov.) and $F$-measure ($F$) since high predication accuracy cannot prove the usefulness of the rule set while the coverage is low.
4.2 Analysis of The Results

4.2.1 Analysis of Ranks

The experimental results about the comparison of different generalization methods are listed in Table 10.

Table 10
Comparison of the ranks of 8 decision tables and respective \( S^R \) and \( S^P \)

<table>
<thead>
<tr>
<th>dataset</th>
<th>( Rank(S) )</th>
<th>( Rank(S^R) )</th>
<th>( Rank(S^P) )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monk1(train)</td>
<td>17</td>
<td>13</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Monk2(train)</td>
<td>17</td>
<td>17</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Monk3(train)</td>
<td>17</td>
<td>15</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>Mushroom</td>
<td>116</td>
<td>35</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>Nursery</td>
<td>27</td>
<td>27</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>LetterRec (train)</td>
<td>256</td>
<td>160</td>
<td>56</td>
<td>5</td>
</tr>
<tr>
<td>Heart</td>
<td>25</td>
<td>25</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>Iris</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

It is easily seen that

\[
Rank(S) \geq Rank(S^R) \geq Rank(S^P).
\]

The cardinality of conditional attribute sets are also listed since in some cases \( Rank(S^P) - |C| \) instead of \( Rank(S^P) \) may represent the partition quality better. For example, in the Mushroom dataset, \( S^P \) has only 7 “valid” conditional attributes, all of which are binary.

For all datasets \( K \) is small, indicating that only a few rounds are needed to obtain the partition reduct. Note that in LetterRec the cardinalities of attributes are all 16, while only 5 rounds (the last round essentially does nothing) are needed.

4.2.2 Analysis of the Classification Results Using the Generalized Data Sets

Respective results of rules generated in these datasets are listed in Table 11.

For the Monks datasets, the RBSVP algorithm can help obtaining smaller rule sets than that of the attribute reduction approach, which in turn has a significant enhancement over the direct approach. Also, in 2 out of 3 Monks datasets the RBSVP algorithm performed the best in terms of \( F \)-measure.
Table 11
Comparison of rules (Bolded values indicate the best results)

<table>
<thead>
<tr>
<th>dataset</th>
<th>$S$ rules</th>
<th>$S^R$ rules</th>
<th>$S^P$ rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S$ cov.</td>
<td>$F$</td>
<td>$S^R$ cov.</td>
</tr>
<tr>
<td>Monk1</td>
<td>161</td>
<td>1</td>
<td>0.866</td>
</tr>
<tr>
<td>Monk2</td>
<td>247</td>
<td>1</td>
<td>0.736</td>
</tr>
<tr>
<td>Monk3</td>
<td>135</td>
<td>1</td>
<td><strong>0.944</strong></td>
</tr>
<tr>
<td>Mushroom</td>
<td><strong>14</strong></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nursery</td>
<td>638</td>
<td>1</td>
<td>0.983</td>
</tr>
<tr>
<td>LetterRec</td>
<td>3725</td>
<td><strong>0.661</strong></td>
<td><strong>0.725</strong></td>
</tr>
<tr>
<td>Heart</td>
<td>918</td>
<td>1</td>
<td>0.796</td>
</tr>
<tr>
<td>Iris</td>
<td>18</td>
<td>1</td>
<td>0.967</td>
</tr>
</tbody>
</table>

Even for the last Monks dataset, the $F$-measure of the new approach is only slightly worse than the direct approach.

For the Mushroom dataset, more rules were obtained while attribute reduction was introduced. But the RBSVP algorithm does not incur such problem here. In fact, the Rule learning process after symbolic value partition is much faster.

For three datasets, i.e., Nursery, Heart and Iris, attributes in $S$ cannot be reduced further, hence the results correspond with $S$ and $S^R$ are the same. But the RBSVP algorithm can still take effect here and helped producing better results.

For the LetterRec dataset, worse results were obtained while attribute reduction was introduced, and even worse results was obtained while the RBSVP algorithm was employed. There are at least two reasons: first, attributes in this dataset is essentially ordered rather than symbolic; and second, different attributes represent grey values of different points, which are not dependent.

5 An Approach of Nguyen

Nguyen H.S. [15] and Nguyen S.H. [18] proposed an interesting partition approach as follows. Suppose that $x_1, x_2 \in U$, $a \in C$, $a(x_1) = v_1$ and $a(x_2) = v_2$. If $v_1 \neq v_2$ then $x_1$ and $x_2$ are discerned by $a$, or more precisely, by the triple $a_{v_1}^{v_2} = (a, v_1, v_2)$. Hence we can construct a new type of discernibility matrix, elements of which are triple sets instead of attribute sets.

An exemplary decision table and the corresponding new type discernibility
matrix are listed in Table 12.

Table 12

<table>
<thead>
<tr>
<th>U</th>
<th>a</th>
<th>b</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>white wood</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x₂</td>
<td>black iron</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x₃</td>
<td>white plastic</td>
<td>0</td>
<td>→</td>
</tr>
<tr>
<td>x₄</td>
<td>green iron</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x₅</td>
<td>black plastic</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x₆</td>
<td>white glass</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Through using discernibility function [22], two prime implicants can then be found:

\[ R_1 = a_w \land a_b \land b_1^{li} \land b_1^{lp} \land b_p^{lg}, \]
\[ R_2 = a_w \land a_b \land b_1^{lg} \land b_1^{lp} \land b_p^{lg}, \]
both are the shortest.

Suppose that \( R_1 \) is chosen, we obtain a graph as depicted in Fig. 1.

Now the partition problem is converted into the graph coloring problem, namely, finding coloring schemes with least colors such that adjacent nodes have different colors. For the graph depicted in Fig. 1, two colors are enough, indicating a partition scheme \( P_1 = [P_a, P_b] \) where

\[ P_a(\text{white}) = P_a(\text{green}) = 1, P_a(\text{black}) = 2; \]
\[ P_b(\text{wood}) = P_b(\text{plastic}) = 1, P_b(\text{iron}) = P_b(\text{glass}) = 2; \]
and \( P_1 \) is a partition reduct of the decision table.

Unfortunately, this approach has the following drawbacks: first, the space complexity is very high, as indicated by Nguyen S.H. [18]: “[t]he constructed Boolean formula has \( O(knll^2) \) variables and \( O(n^2) \) clauses, where \( l \) is the maximal value of \(|V_a| \) for \( a \in C \)” ; second, the partition scheme obtained may not
be a partition reduct. Suppose that $R_2$ is employed to generate the graph as depicted in Fig. 2, we obtain a partition scheme $P_2 = [P_a, P_b]$ where

$$P_a(\text{white}) = 1, P_a(\text{black}) = 2, P_a(\text{green}) = 3;$$

$$P_b(\text{wood}) = P_b(\text{plastic}) = 1, P_b(\text{iron}) = P_b(\text{glass}) = 2;$$

which is not a partition reduct. In fact, it can be proven that any partition scheme $P$ produced by this approach is consistent, while $S^P$ is not necessarily unpartitionable.

Fig. 2. The graph for coloring when $R_2$ is chosen

6 Conclusions And Further Works

In this paper we proposed the concept of partition reducts for the OSVP problem. Since Nguyen’s approach might not result in a partition reduct (see Section 5 for the counterexample), we developed an algorithm for this purpose. The new algorithm, called RBSVP, is both efficient (see equations (14) and (15) for the space and time complexities) and tends to give suboptimal results (see Theorem 17, Properties 15 through 19). Experimental results also show that it could help obtaining small rule sets with good performance for most datasets tested.

In further works we will extend this algorithm to suit mixed-mode data and/or decision tables with missing value, and apply it to applications such as intelligent information distribution. Generating rules in the partition process is also a very interesting issue.

References


