Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 864652, 24 pages
doi:10.1155/2012/864652

Research Article

Knowledge Reduction Based on Divide and Conquer Method in Rough Set Theory

Feng Hu and Guoyin Wang

Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Correspondence should be addressed to Guoyin Wang, Wanggy@cqupt.edu.cn

Received 27 June 2012; Revised 18 September 2012; Accepted 2 October 2012

Academic Editor: P. Liatsis

The divide and conquer method is a typical granular computing method using multiple levels of abstraction and granulations. So far, although some achievements based on divided and conquer method in the rough set theory have been acquired, the systematic methods for knowledge reduction based on divide and conquer method are still absent. In this paper, the knowledge reduction approaches based on divide and conquer method, under equivalence relation and under tolerance relation, are presented, respectively. After that, a systematic approach, named as the abstract process for knowledge reduction based on divide and conquer method in rough set theory, is proposed. Based on the presented approach, two algorithms for knowledge reduction, including an algorithm for attribute reduction and an algorithm for attribute value reduction, are presented. Some experimental evaluations are done to test the methods on uci data sets and KDDCUP99 data sets. The experimental results illustrate that the proposed approaches are efficient to process large data sets with good recognition rate, compared with KNN, SVM, C4.5, Naive Bayes, and CART.

1. Introduction

In the search for new paradigms of computing, there is a recent surge of interest, under the name of granular computing [1–3], in computations using multiple levels of abstraction and granulation. To a large extent, the majority of existing studies include rough sets, fuzzy sets, cluster analysis, and classical divide and conquer methods [4, 5] and aim at solving specific problems [6].

Rough set (RS) [7–10] is a valid mathematical theory for dealing with imprecise, uncertain, and vague information. It has been applied in such fields as machine learning, data mining, intelligent data analysis, and control algorithm acquiring, successfully since it was proposed by Pawlak and Skowron in 2007 [10]. Knowledge reduction is one of the most

important contributions of rough set theory to machine learning, pattern recognition, and data mining. Although the problem of finding a minimal reduction of a given information system was proven to be an NP-hard problem [8], many promising heuristics have been developed that are promising. A variety of methods for knowledge reduction and their applications can be found in [2, 7–43]. Among these existing methods, one group method focuses on the indiscernibility relation in a universe that captures the equivalence of objects, while the other group considers the discernibility relation that explores the differences of objects [42]. For indiscernibility relation, one can employ it to induce a partition of the universe and thereby to construct positive regions whose objects can be undoubtedly classified into a certain class with respect to the selected attributes. Thus, knowledge reduction algorithms based on positive regions have been proposed in [8, 10, 15, 16, 21, 28, 30]. For discernibility relation, we have knowledge reduction algorithms based on a discernibility matrix and information entropy. Reduction methods based on discernibility matrix [34] have high cost of storage with space complexity $O(m \times n^2)$ for a large decision table with n objects and m conditional attributes. Thus, storing and deleting the element cells in a discernibility matrix is a time-consuming process. Many researchers have studied discernibility matrix construction and contributed to a lot [10, 18, 28, 35, 38, 39, 43]. As well, knowledge reduction algorithms based on information entropy [20, 37, 42] have been developed. Although so many algorithms have been developed, it is valuable to study some new high efficient algorithms.

Divide and conquer method is a simple granular computing method. When the algorithms are designed by divide and conquer method, the decision table can be divided into many subdecision tables recursively in attribute space. That's to say, an original big data set can be divided into many small ones. If the small ones can be processed one by one, instead the original big one is processed on a whole, it will save a lot time. Thus, it may be an effective way to process large data set. The divide and conquer method consists of three vital stages.

Stage 1. Divide the big original problem into many independent subproblems with the same structure.

Stage 2. Conquer the sub-problems recursively.

Stage 3. Merge the solutions of sub-problems into the one of original problem.

So far, some good results for knowledge reduction based on divide and conquer method have been achieved, such as the computation of the attribute core and the computation of attribute reduction under given attribute order [12, 13]. Besides, decision tree-based methods [26, 27, 44] have been studied and are very popular. In fact, the construction of decision tree is a special case of divide and conquer method, because it can be generated from top to down recursively. In the methods based on decision tree, a tree should be constructed by using decomposition at first. It costs more in the first stage, which is convenient to the following stages, and costs less in the following two ones.

However, the systematic method for knowledge reduction based on divide and conquer method is still absent, especially "how to keep invariance between the solution of original problem and the ones of sub-problems." It results in the difficulty to design the high efficient algorithms for knowledge reduction based on divide and conquer method. Therefore, it is urgent to discuss the knowledge reduction method based on divide and conquer methods systematically and overall.

Contributions of this work are (1) some principles for "keeping invariance between the solution of original problem and the ones of sub-problems" are concluded. Then, the abstract process for knowledge reduction based on divide and conquer method in the rough set theory is presented, which is helpful to design high efficient algorithm based on divide and conquer method. (2) Fast approaches for knowledge reduction based on divide and conquer method, including an algorithm for attribute reduction and an algorithm for attribute value reduction, are proposed. Experimental evaluations show that the presented methods are efficient.

The remainder of this paper is organized as follows. The basic theory and methods dealing with the application of rough set theory in data mining are presented in Section 2. Section 3 introduces the abstract process for knowledge reduction based on divide and conquer method in the rough set theory. A quick algorithm based on divide and conquer method for attribute reduction is presented in Section 4. In Section 5, a fast algorithm for attribute value reduction using divide and conquer method is proposed. Some simulation experimental evaluations are discussed to show the performance of the developed methods in Section 6. The paper ends with conclusion in Section 7.

2. Preliminaries

Rough set theory was introduced by Pawlak as a tool for concept approximation relative to uncertainty. Basically, the idea is to approximate a concept by three description sets, namely, the lower approximation, upper approximation, and boundary region. The approximation process begins by partitioning a given set of objects into equivalence classes called blocks, where the objects in each block are indiscernible from each other relative to their attribute values. The approximation and boundary region sets are derived from the blocks of a partition of the available objects. The boundary region is constituted by the difference between the lower approximation and upper approximation and provides a basis for measuring the "roughness" of an approximation. Central to the philosophy of the rough set approach to concept approximation is the minimization of the boundary region [28].

For the convenience of description, some basic notions of decision tables are introduced here at first.

Definition 2.1 (decision table [36]). A decision table is defined as $S = \langle U, A, V, f \rangle$, where U is a non-empty finite set of objects, called the universe, A is a nonempty finite set of attributes, $A = C \cup D$, where C is the set of conditional attributes, and D is the set of decision attributes, $D \neq \phi$. For any attribute $a \in A$, V_a denotes the domain of attribute a. Each attribute determines a mapping function $f : U \times A \rightarrow V$.

Definition 2.2 (indiscernibility relation [36]). Given a decision table $S = \langle U, A, V, f \rangle$, each subset of attribute $B \subseteq A$ determines an indiscernibility relation $\mathrm{IND}(B)$ as follows: $\mathrm{IND}(B) = \{(x, y) \mid (x, y) \in U \times U,\ \forall b \in B(b(x) = b(y))\}$.

Definition 2.3 (lower approximation and upper approximation [36]). Given a decision table $S = \langle U, A, V, f \rangle$, for any subset $X \subseteq U$ and indiscernibility relation $\mathrm{IND}(B)$, the B lower approximation and upper approximation of X are defined as $B_-(X) = \bigcup_{Y_i \in U/\mathrm{IND}(B) \wedge Y_i \subseteq X} Y_i$, $B^-(X) = \bigcup_{Y_i \in U/\mathrm{IND}(B) \wedge Y_i \cap X \neq \phi} Y_i$.

Definition 2.4 (positive region [36]). Given a decision table $S = \langle U, A, V, f \rangle$, $P \subseteq A$, and $Q \subseteq A$, the P positive region of Q is defined as $\mathrm{Pos}_P(Q) = \cup_{x \in U/Q} P(X)$.

Definition 2.5 (relative core [36]). Given a decision table $S = \langle U, A, V, f \rangle$, $P \subseteq A$, $Q \subseteq A$, $r \in P$, r is unnecessary in P relative to Q if and only if $\mathrm{Pos}_P(Q) = \mathrm{Pos}_{P-\{r\}}(Q)$, otherwise r is unnecessary in P relative to Q. The core of P relative to Q is defined as $\mathrm{CORE}_Q(P) = \{r \mid r \in P; r \text{ is necessary in } P \text{ relative to } Q\}$.

Definition 2.6 (see [36]). Given a decision table $S = \langle U, A, V, f \rangle$, $P \subseteq A$ and $Q \subseteq A$, for all $r \in P$, if r is relatively necessary in P relative to Q, P is called independent relative to Q.

Definition 2.7 (relative reduction [36]). Given a decision table $S = \langle U, A, V, f \rangle$, $P \subseteq A$, $Q \subseteq A$, $R \in P$, if R is indispensable relative to Q and $\mathrm{Pos}_R(Q) = \mathrm{Pos}_P(Q)$, R is called a reduction of P relative to Q.

Definition 2.8 (see [18]). Given a decision table $S = \langle U, A, V, f \rangle$, the element of discernibility matrix M can be defined as $B^s_{xy} = \{a \in C \mid f(x,a) \neq f(y,a) \wedge w(x,y) = 1\}$, where, $x \in U$, $y \in U$, and

$$w(x,y) = \begin{cases} 1, & x \in \mathrm{Pos}_C(D),\ y \notin \mathrm{Pos}_C(D), \\ 1, & x \notin \mathrm{Pos}_C(D),\ y \in \mathrm{Pos}_C(D), \\ 1, & x, y \in \mathrm{Pos}_C(D),\ d(x) \neq d(y), \\ 0, & \text{others.} \end{cases} \tag{2.1}$$

3. The Abstract Process for Knowledge Reduction Based on Divide and Conquer Method in Rough Set Theory

3.1. *The Knowledge Reduction Based on Divide and Conquer Method under Equivalence Relation*

In the research of rough set theory, the divide and conquer method is an effective method to design high effective algorithm. It can be used to compute equivalence classes, positive region, and attribute core of decision table (see Propositions 3.1, 3.2, and 3.3) even execute some operators of discernibility matrix (see Propositions 3.4 and 3.5). In this section, the divide and conquer method under equivalence relation in rough set theory will be discussed.

Proposition 3.1 (see [13]). *Given a decision table $S = \langle U, A, V, f \rangle$, for all c $(c \in C)$, divide S into k $(k = |IND(U/\{c\})|)$ subdecision tables $S_1, S_2, \ldots, S_k$ by $U/\{c\}$, where $S_i = \langle U_i, (C \setminus \{c\}) \cup D, V_i, f_i \rangle$, which satisfies $\forall_{x \in U_i} \forall_{y \in U_i} c(x) = c(y)$ $(1 \leq i \leq k)$ and $\forall_{x \in U_i} \forall_{z \in U_j} c(x) \neq c(z)$ $(1 \leq i < j \leq k)$. Let $R = C - \{c\}$. Let us denote by $Pos^i_R(D)$ $(1 \leq i \leq k)$ the positive region of S_i and the one of S by $Pos_C(D)$, respectively. Then, $Pos_C(D) = \bigcup_{1 \leq i \leq k} Pos^i_R(D)$.*

Proposition 3.1 presents the approach of computing equivalence classes or positive region by using divide and conquer method. Compared with decision tree-based method (without pruning), the approach allows us to generate "clear" leaves (with the same decision) for objects in the positive region and "unclear" leaves where objects are with different decisions and correspond to the boundary region. It may be an effective way to prevent overfitting, because "conquer" can play a role of pruning of tree. Furthermore, the approach needs less space because the construction of tree is not necessary.

Proposition 3.2 (see [13]). *Given a decision table $S = \langle U, A, V, f\rangle$, for all c $(c \in C)$, divide S into k $(k = |IND(U/\{c\})|)$ sub-decision tables $S_1, S_2, \ldots, S_k$ by $U/\{c\}$, where $S_i = \langle U_i, (C \setminus \{c\}) \cup D, V_i, f_i\rangle$, which satisfies $\forall_{x\in U_i}\forall_{y\in U_i} c(x) = c(y)$ $(1 \le i \le k)$ and $\forall_{x\in U_i}\forall_{z\in U_j} c(x) \ne c(z)$ $(1 \le i < j \le k)$. Let us denote by $Core_i$ $(1 \le i \le k)$ the attribute core of S_i and the one of S by Core. Then, $\bigcup_{1\le i\le k} Core_i \subseteq Core \subseteq \{c\} \cup \bigcup_{1\le i\le k} Core_i$.*

Proposition 3.3 (see [13]). *Given a decision table S-$\langle U, A, V, f\rangle$, for all c $(c \in C)$, divide S into k $(k = |IND(U/\{c\})|)$ sub-decision tables $S_1, S_2, \ldots, S_k$ by $U/\{c\}$, where $S_i = \langle U_i, (C \setminus \{c\}) \cup D, V_i, f_i\rangle$, which satisfies $\forall_{x\in U_i}\forall_{y\in U_i} c(x) = c(y)$ $(1 \le i \le k)$ and $\forall_{x\in U_i}\forall_{z\in U_j} c(x) \ne c(z)$ $(1 \le i < j \le k)$. Let us denote by red_i $(1 \le i \le k)$ the attribute reduction of S_i and the one of S by R. Let $R = \bigcup_{1\le i\le k} red_i$, $R_1 = \{c\} \cup \bigcup_{1\le i\le k} red_i$. Then, $Pos_R(D) \subseteq Pos_C(D) = Pos_{R_1}(D)$.*

Propositions 3.2 and 3.3 present the approach of attribute core determining based on divide and conquer method. Compared with the method of computing attribute core on original decision table, it may be more efficient and can process bigger data set, since the big data set has been divided into many small data sets.

Obviously, Propositions 3.1, 3.2, and 3.3 hold.

Discernibility matrix by Skowron is a useful method to design some algorithms in rough set theory. However, due to its high complexity of algorithms based on explicit computing of the discernibility matrices, the efficiency of algorithms based on discernibility matrix needs to be improved. In the literature some useful methods have been proposed (see [26–31] by Nguyen et al., and decomposition methods implemented in RSES). Our methods differ from the existing ones as follows.

(1) "How to keep invariance between the solution of original problem and the ones of sub-problems" is a key problem. We conclude some principles for computing positive region, attribute core, attribute reduction, and value reduction (see Propositions 3.1, 3.2, 3.3, 3.4, 3.5, 3.11, and 3.12), which were not concluded before.

(2) Although the decision tree-based methods and our approaches both belong to divide and conquer method, our approaches cost more on "conquer" and "merge" while they cost less on "divide," compared with the decision tree-based methods. Furthermore, our approaches need not to construct a tree, which maybe save space.

(3) The existing heuristic ones in [26–28] can improve the efficiency by measuring the discernibility degree of different objects quickly. In our approaches, the element cells of discernibility matrix can be deleted by dividing decision table fast without storing the discernibility matrix. Thus, it may be a quick one for operating discernibility matrix with small space (see Propositions 3.4 and 3.5).

Given a decision table $S = \langle U, A, V, f\rangle$ and its discernibility matrix M (Definition 2.8); for all C_1 $(C_1 \subseteq C)$, let us denote by $\overline{M}^{C_1}$ a subset of element cells in M. $\overline{M}^{C_1}$ can be labeled as $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$ $(t = |\overline{M}^{C_1}|)$, where

(1) $\forall_{\alpha_i\in\overline{M}^{C_1}} \alpha_i \in M$ $(1 \le i \le |\overline{M}^{C_1}|)$;

(2) for all $c \in C_1$, if $\exists_{\alpha\in M}(c \in \alpha)$, then $\alpha \in \overline{M}^{C_1}$.

Proposition 3.4. *Given a decision table $S = \langle U, A, V, f\rangle$, for all c $(c \in C)$, divide the decision table S into k $(k = |IND(U/\{c\})|)$ sub-decision tables $S_1, S_2, \ldots, S_k$ by $U/\{c\}$, where $S_i = \langle U_i, (C \setminus$*

$\{c\}) \cup D, V_i, f_i\rangle$. *Let us denote by* M *the discernibility matrix of* S *and the discernibility matrix of* S_i $(1 \le i \le k)$ *by* M_i, *respectively. Then,* $\bigcup_{\alpha\in M} \alpha = \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta \cup \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$.

(Note: If $\forall_{x,y\in U_i}\forall_{c_1\in C} c_1(x) = c_1(y)$, then, $M_i = \phi$)

Proof. First, prove $\bigcup_{\alpha\in M} \alpha \subseteq \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta \cup \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$.

For all $\alpha \in M$, then, $\exists_{x,y\in U} \alpha = B^S_{xy}$ (Definition 2.8). After the partition of U/C_1, suppose x and y be divided into sub-decision tables S_i $(1 \le i \le |U/C_1|)$ and S_j $(1 \le j \le k)$, respectively.

If $i = j$, then, $\alpha \in M_i$, that is, $\alpha \in \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta$.

If $i \ne j$, then, $\alpha \in \overline{M}^{C_1}$, that is, $\alpha \in \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$.

Thus, $\alpha \in \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta \cup \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$. That is, $\bigcup_{\alpha\in M} \alpha \subseteq \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta \cup \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$.

Similarly, we can proof $\bigcup_{\alpha\in M} \alpha \supseteq \bigcup_{i=1}^{k} \bigcup_{\beta\in M_i} \beta \cup \bigcup_{\gamma\in\overline{M}^{C_1}} \gamma$.

Therefore, Proposition 3.4 holds. □

Proposition 3.5. *Given a decision table* $S = \langle U, A, V, f\rangle$, *for all* C_1 $(C_1 \subseteq C)$, *divide the decision table* S *into* k $(k = |IND(U/C_1)|)$ *sub-decision tables* $S_1, S_2, \ldots, S_k$ *by* U/C_1, *where* $S_i = \langle U_i, (C \setminus C_1)\cup D, V_i, f_i\rangle$. *Let us denote by* M *the discernibility matrix of* S. *Then, in the viewpoints of operating discernibility matrix, divide the decision table* S *by* U/C_1 *on attribute set* C_1 *if and only if one deletes all the element cells of* $\overline{M}^{C_1}$ *from* M *one by one.*

According to Proposition 3.4, it is easy to find that Proposition 3.5 holds.

Propositions 3.4 and 3.5 present the approach of deleting element cells of discernibility matrix. By using the approach, the element cells of discernibility matrix can be deleted quickly without constructing or storing the discernibility matrix. It may be an effective way to operate discernibility matrix quickly within small space. Thus, Propositions 3.4 and 3.5 can be used to design some efficient algorithms without explicit computing of discernibility matrix.

3.2. The Knowledge Reduction Based on Divide and Conquer Method under Tolerance Relation

In the course of attribute value reduction, tolerance relation is often adopted due to some attribute values on condition attribute being deleted. Thus, tolerance relation in attribute value reduction may be needed. A method is introduced by Kryszkiewicz and Rybinski [17] to process incomplete information system, where "$*$" is used to represent the missing values on condition attributes. Here, "$*$" can be also represent the deleted values on condition attributes. According to the tolerance relation by Kryszkiewicz, the divide and conquer method under the tolerance relation will be discussed.

Definition 3.6 (see [17]). Given an incomplete decision table $S = \langle U, A, V, f\rangle$, a tolerance relation T is defined as

$$\forall x, y_{\in U_1} \forall_{B\subseteq R} \left(T_B(x,y) \Longleftrightarrow_{\forall b\in B} \left(\left(b(x) = b(y)\right) \vee \left(b(x) = *\right) \vee \left(b(y) = *\right)\right)\right). \tag{3.1}$$

Definition 3.7 (see [17]). The tolerance class $T_B(x)$ of an object x relative to an attribute set B is defined as $T_B(x) = \{y \mid y \in U \wedge T_B(x,y)\}$. In the tolerance relation-based extension

of rough set theory, the lower approximation X_B^T and upper approximation X_T^B of an object set X relative to an attribute set B ($B \subseteq C$) are defined as $X_B^T = \{x \in U \mid T_B(x) \subseteq X\}$, $X_T^B = \{x \in U \mid T_B(x) \cap X \neq \emptyset\}$.

Definition 3.8 (the covering of the universe under tolerance relation). Given a decision table $S = \langle U, A, V, f \rangle$ and condition attribute set C_1 ($C_1 \subseteq C$), according to the tolerance relation by Kryszkiewicz, the covering of the universe of S can be defined as $U = U_1 \cup U_2 \cup \cdots \cup U_k$, where $\forall_{x,y \in U_p (1 \leq p \leq k)} \forall_{c \in C_1} (c(x) = c(y) \vee c(x) = * \vee c(y) = *)$, and $\forall_{x \in U_r} \forall_{y \in U_p \ (1 \leq r < p \leq k)} \exists_{c \in C_1} ((c(x) \neq * \wedge c(y) \neq *) \Rightarrow c(x) \neq c(y))$.

Definition 3.9 (certain decision rule). Given a decision table $S = \langle U, A, V, f \rangle$, for all $x_i \in \mathrm{Pos}_C(D)$, the object x_i can result in a certain decision rule $d_i : \mathrm{des}([x_i]_C) \Rightarrow \mathrm{des}([x_i]_D)$, $d_i(a) = a(x_i)$, $a \in C \cup D$. $d_i \mid C$ and $d_i \mid D$ are called the condition attribute set and decision attribute set of d_i, respectively.

Definition 3.10 (see [36]). Given a decision table $S = \langle U, A, V, f \rangle$, for arbitrary certain decision rule, there is $[x_i]_C \subset [x_i]_D$. For all $a \in C$, if $[x_i]_{C \setminus \{a\}} \subseteq [x_i]_D$ does not hold, then, a is necessary in d_i; otherwise, a is not necessary in d_i.

Proposition 3.11. *Given a decision table $S = \langle U, A, V, f \rangle$, and for all $C_1 \subseteq C$, given an decomposing order $c_1, c_2, \ldots c_p$ ($p = |C_1|$), according to the order and tolerance relation, S can be divided into k sub-decision tables $S_1, S_2, \ldots, S_k$, where $S_i = \langle U_i, A, V, f \rangle$. Assume S and $S_1, S_2, \ldots, S_k$ be processed with the same way. For each sub-decision table S_i ($1 \leq i \leq k$), for all x ($x \in S_i \wedge x \in Pos_C(D)$), let us denote by d_x a decision rule relative to object x in S_i and a decision rule relative to x in S by d_x^1, respectively. There is*

(1) *in S_i, if $\exists c$ ($c \in C$), c is necessary in d_x. Then, c is necessary in d_x^1;*

(2) *in S_i, if $\exists c$ ($c \in C$), c is not necessary in d_x. Then, c is not necessary in d_x^1;*

(3) $d_x = d_x^1$.

Proposition 3.12. *Given a decision table $S = \langle U, A, V, f \rangle$, for all $C_1 \subseteq C$, given an decomposing order $c_1, c_2, \ldots c_p$ ($p = |C_1|$), and according to the order, S can be divided into k sub-decision tables $S_1, S_2, \ldots, S_k$, where $S_i = \langle U_i, A, V, f \rangle$. Assume S and $S_1, S_2, \ldots, S_k$ be processed with the same. For each sub-decision table S_i ($1 \leq i \leq k$), let us denote by Rule the certain decision rule set of S and the one of S_i by $Rule_i$, respectively. Then, $Rule = \bigcup_{i=1}^{k} Rule_i$.*

Propositions 3.11 and 3.12 present the approach of value reduction based on divide and conquer method. It can keep invariance between the solution of original decision table and the ones of sub-decision tables. By using the approach, it allows us to generate decision rules from sub-decision tables, not from original decision table. It may be a feasible way to process big data set.

3.3. The Abstract Process for Knowledge Reduction Based on Divide and Conquer Method in Rough Set Theory

According to the divide and conquer method under equivalence relation and tolerance relation, the abstract process for knowledge reduction in rough set theory based on the divide and conquer method APFKRDAC(P, S) will be discussed in this section.

Algorithm 3.13 (APFKRDAC(P, S)).

Input: The problem P on S.

Output: The solution *Solu* of the problem P.

Step 1 (determine a similarity relation of different objects). Determine a similarity relation between different objects, such as equivalence relation or tolerance relation. Generally, reflexivity and symmetry of different objects may be necessary.

Step 2 (determine the decomposing order). Determine the order C_1 ($C_1 \subseteq C$) for decomposing the universe of decision table. Let $C_1 = \{c_1, c_2, \ldots, c_p\}$, $(p = |C_1|)$.

Step 3 (determine the decomposing strategy).

(3.1) Design a judgment criteria $CanBeDivide()$ for judging whether the universe can be decomposed.

(3.2) Design a decompose function $DecomposingFunc()$, which can be used to decompose the universe recursively.

(3.3) Design a boolean function $IsEnoughSmall()$, which can be used to judge if the size of problem is small enough to be processed easily.

(3.4) Design a computation function $ProcessSmallProblem()$, which can be used to process small problems directly.

(3.5) Design a computation function $MergingSolution()$, which can be used to merge the solutions of sub-problems.

Step 4 (process the problem based on divide and conquer method).

(4.1) IF $IsEnoughSmall(S)$ THEN $Solu = SolutionSubProblems(S)$,
goto Step 5.

(4.2) (Divide)
According to the decomposing order, divide S into k sub-decision tables $S_1, S_2, \ldots, S_k$ on C_1.

(4.3) (Conquer sub-problems recursively)
FOR $i = 1$ TO k DO
$Solu_i$=APFKRDAC(P_i, S_i).
END FOR.
Where, $S_i = \langle U_i, C_i \cup D, V, f \rangle$, $C_i \subseteq C$, P_i is the sub-problem of S_i.

(4.4) (Merge the solutions of sub-problems)
$Solu = MergingSolution(Solu_1, Solu_2, \ldots, Solu_k)$.

Step 5 (optimize the solution). If necessary, optimize the solution *Solu*.

Step 6 (return the solution). RETURN *Solu*.

Now, let us give an example for computing the positive region of decision table to explain Algorithm 3.13 (see Algorithm 3.14: the algorithm for computing positive region based on divide and conquer method).

Algorithm 3.14 (CPRDAC(P, S)).

Input: The problem P on S: compute positive region.

Output: The positive region $\mathrm{Pos}_C(D)$ of S.

Step 1 (Determine the similar relation). equivalence relation.

Step 2 (Determine the decomposing order). $c_1, c_2, \ldots, c_m$ $(m = |C|)$.

Step 3 (Determine the decomposing strategy).

(3.1) Design a judgment criteria $CanBeDivide()$:
On attribute c_i, for all U_i $(U_i \subseteq U)$,
IF $\exists_{x\in U_i}\exists_{y\in U_i}(c_i(x) \neq c_i(y)) \wedge \exists_{z\in U_i}\exists_{w\in U_i}(D(z) \neq D(w))$ THEN
$CanbeDivide(c_i)$=true;
ELSE $CanBeDivide(c_i)$=false;
END IF

(3.2) Design a decompose function $DecomposingFunc()$:
According to U/C, S can be divided into $k = |U/C|$ sub-decision tables $S_1, S_2, \ldots, S_k$ on attributes $c_1, c_2, \ldots, c_m$ recursively.

(3.3) Design a boolean function $IsEnoughSmall()$:
Let c_r be the attribute on which the universe is being decomposed.
IF $(|U_i| == 1)$ or $(r > m)$ or $\forall_{x\in U_i}\forall_{y\in U_i}(D(x) == D(y))$ THEN
$IsEnoughSmall(S_i) = true$.
END IF

(3.4) Design a computation function $ProcessSmallProblem()$:
For arbitrary sub-decision table S_i and its universe U_i,
IF $\forall_{x\in U_i}\forall_{y\in U_i}(D(x) == D(y))$ THEN $U_i \subseteq \mathrm{Pos}_C(D) \wedge Solu_i = U_i$;
ELSE $Solu_i = \phi$;
END IF

(3.5) Design a computation function $MergingSolution()$:
$Solu = Solu_1 \cup Solu_2 \cup \cdots \cup Solu_k$;

Step 4 (Process the problem based on the divide and conquer method).

(4.1) IF $IsEnoughSmall(S)$ THEN $Solu = ProcessSmallProblem(S)$; goto Step 5.

(4.2) (Divide)
According to the order $\{c_1, c_2, \ldots, c_m\}$, S can be divided into k sub-decision tables $S_1, S_2, \ldots, S_k$ on c_1 by using $U/\mathrm{IND}(\{c_1\})$.

(4.3) (Conquer sub-problems recursively)
FOR i = 1 TO k DO
$Solu_i$= CPRDAC(P_i, S_i).
END FOR
Where, $S_i = \langle U_i, (C - \{c_1\}) \cup D, V, f\rangle$, P_i denotes computing the positive region of S_i.

(4.4) (Merge the solutions of sub-problems)
$Solu = Solu_1 \cup Solu_2 \cup \cdots \cup Solu_k$.

Step 5 (Optimize the solution). *Solu* is an optimized result.

Step 6 (Return the solution). RETURN *Solu*.

Example 3.15. Given a decision table $S = \langle U, A, V, f\rangle$, $(U = \{x_1, x_2, x_3, x_4, x_5, x_6\}, C = \{c_1, c_2, c_3\})$, now compute the positive region of S according to Algorithm 3.14. The whole process can be found in Figure 1.

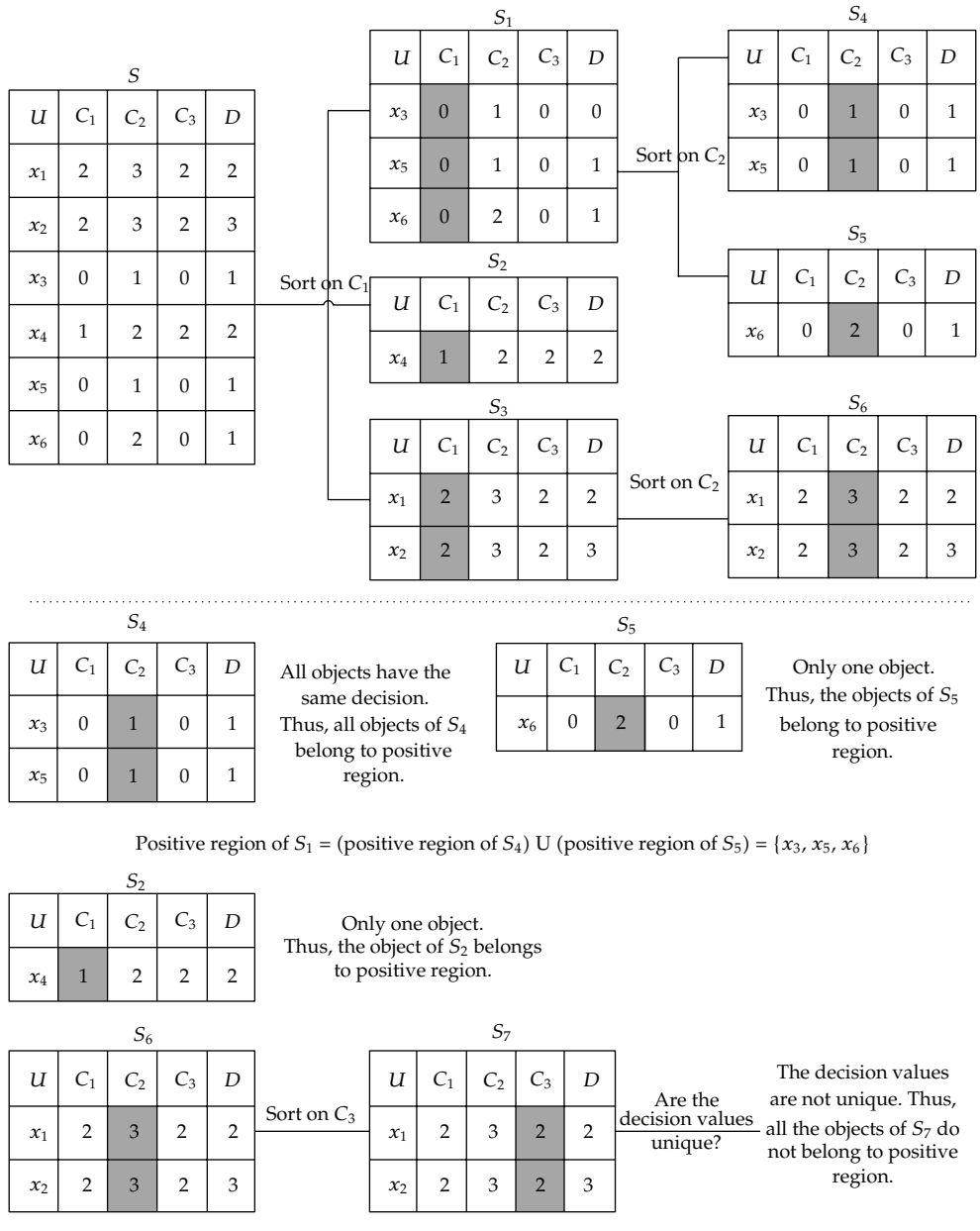

Figure 1: The course of computing positive region of S.

Let us denote by $\mathrm{Pos}_C(D), \mathrm{Pos}_C^1(D), \mathrm{Pos}_C^2(D), \ldots, \mathrm{Pos}_C^7(D)$ the positive region of $S, S_1, S_2, \ldots, S_7$, respectively.

Divide S into S_1, S_2, S_3 on c_1.

Conquer S_1.

Divide S_1 into S_4, S_5 on c_2.

Conquer S_4. $\mathrm{Pos}_C^4(D) = \{x_3, x_5\}$.
Conquer S_5. $\mathrm{Pos}_C^5(D) = \{x_6\}$.
Merge the solutions of S_4 and S_5. $\mathrm{Pos}_C^1(D) = \mathrm{Pos}_C^4(D) \cup \mathrm{Pos}_C^5(D) = \{x_3, x_5, x_6\}$.
Similarly, we can conquer S_2 and S_3.
Conquer S_2. $\mathrm{Pos}_C^2(D) = \{x_4\}$.
Conquer S_3. $\mathrm{Pos}_C^3(D) = \phi$.
Merge the solutions of S_1, S_2, and S_3.

$$\mathrm{Pos}_C(D) = \mathrm{Pos}_C^1(D) \cup \mathrm{Pos}_C^2(D) \cup \mathrm{Pos}_C^3(D) = \{x_3, x_4, x_5, x_6\}. \tag{3.2}$$

4. A Fast Algorithm for Attribute Reduction Based on Divide and Conquer Method

Knowledge reduction is the key problem in rough set theory. When the divide and conquer method is used to design the algorithm for knowledge reduction, some good results may be obtained. However, implementing the knowledge reduction based on the divide and conquer method is very complex, though it is only a simple granular computing method. Here, we will discuss the quick algorithm for knowledge reduction based on divide and conquer method.

In the course of attribute reduction, the divide and conquer method is used to compute the equivalence class, the positive region, and the non-empty label attribute set and delete the elements of discernibility matrix. Due to the complexity of attribute reduction, the following algorithm is not presented as Algorithm 3.14 in details.

According to Step 2 of Algorithm 3.13, the attribute set and the order must be determined, on which the universe of decision table will be partitioned in turns. Generally speaking, the decomposing order depends on the problem which needs to be solved. Furthermore, if the order is not given by field experts, it can be computed by the weights in [10, 15, 23, 25, 26, 28, 33, 36, 37, 41]. Of course, if the attribute order is given, it will be more suitable for Algorithm 3.13. Most techniques discussed below are based on a given attribute order and divide and conquer method. In this section, a quick attribute reduction algorithm based on a given attribute order and divide and conquer method is proposed.

4.1. Attribute Order

In 2001, an algorithm for attribute reduction based on the given attribute order is proposed by Jue Wang and Ju Wang [38]. For the convenience of illustration, some basic notions about attribute order are introduced here.

Given a decision table $S = \langle U, A, V, f \rangle$, an attribute order relation over C ($SO : c_1 \prec c_2 \prec \cdots \prec c_{|C|}$) can be defined. Let us denote by M the discernibility matrix of S. For any $\delta \in M$, the attributes of discernibility matrix δ inherit the order relation of SO from left to right, that is, $\delta = c_j B$, where $c_j \in C$ and $B \subset (C - \{a\})$, and c_j is the first element of δ by SO, called the *non-empty label attribute* of δ [33].

For c_j, a set $L(SO) = \{\delta \mid \delta = c_j B, \delta$ inherit the order relation of SO from left to right and $\delta \in M\}$ is defined. Hence, M can be divided into equivalence classes by label attributes defining a partition $\{[1], [2], \ldots, [|C|]\}$ of M denoted by $M/L(SO)$ [33]. Supposing $N = \max\{k \mid [k] \in M/L(SO) \wedge [k] \neq \phi\}$, its maximum non-empty label attribute is a_N.

4.2. Attribute Reduction Based on the Divide and Conquer Method

Lemma 4.1. $\forall_{B_{xy}^S \in M}(B_{xy}^S \neq \phi)$ *(Definition 2.8) if and only if* $(x \in Pos_C(D) \wedge y \in Pos_C(D) \wedge (d(x) \neq d(y)) \vee (x \in Pos_C(D) \wedge y \notin Pos_C(D)) \vee (x \notin Pos_C(D) \wedge y \in Pos_C(D))$.

According to Definition 2.8, obviously Lemma 4.1 holds.

Proposition 4.2. *Given a decision table* $S = \langle U, A, V, f \rangle$*, for all* $c \in C$*, let* M *be the discernibility matrix of* S*. Then,* $\exists_{\alpha \in M}(c \in \alpha)$ *if and only if the following conditions hold: (1)* $|U| > 1$*; (2)* $\exists_{x \in U}(x \in Pos_C(D))$*; (3)* $|V_D| > 1$*; (4)* $|V_{\{c\}}| > 1$*.*

Proof. (Necessity) according to Lemma 4.1, obviously Proposition 4.2 holds.

(Sufficiency):

for all x $(x \in \mathrm{Pos}_C(D) \wedge x \in U)$, according to $(|U| > 1) \wedge (|V_D| > 1)$, then,

$\exists_{y(y \neq x)}(y \in U \wedge D(x) \neq D(y))$.

If $c(x) \neq c(y)$, then $c \in B_{xy}^S$, that is, $\exists_{\alpha(\alpha \in M)} c \in \alpha$. The proposition holds.

If $c(x) = c(y)$, then, there are two cases.

Case 1 $(y \in \mathrm{Pos}_C(D))$. According to $|V_{\{c\}}| > 1$, $\exists_{z(z \neq x \wedge z \neq y)}(c(z) \neq c(x))$, thus $(c \in B_{xz}^S) \vee (c \in B_{yz}^S) = 1$.

That is, $\exists_{\alpha(\alpha \in M)} c \in \alpha$. The proposition holds.

Case 2 $(y \notin \mathrm{Pos}_C(D))$. According to $|V_{\{c\}}| > 1$, $\exists_{z(z \neq x \wedge z \neq y)}(c(z) \neq c(x))$. If $z \in \mathrm{Pos}_C(D)$, then $c \in B_{yz}^S$;

if $z \notin \mathrm{Pos}_C(D)$, then $c \in B_{yz}^S$. Thus, the proposition holds.

Therefore, Proposition 4.2 holds. □

According to the algorithm in [38], in order to compute the attribute reduction of a decision table, its non-empty label attribute set $L(SO)$ should be first calculated. Using the divide and conquer method, an efficient algorithm for computing the non-empty label attribute set $L(SO)$ is developed. A recursive function for computing the non-empty label attribute set is used in the algorithm.

Function 1

NonEmptyLabelAttr(S, r)

$//S$ is decision table. r is the number of attributes $(1 \leq r \leq |C|)$.

Step 1 (Ending Condition). According to Propositions 4.2 and 5.1,

IF $r = |C| + 1$ or $|U| == 1$ or $\mathrm{Pos}_C(D) == \phi$ or $|V_D| == 1$ THEN

return;

END IF

Step 2 (Compute non-empty labels based on divide and conquer method). Let NonEmptyLabel[] be an array used to store the solution.

Step 2.1. IF $|V_{\{c_r\}}| > 1$ THEN

Denote non-empty label attribute: NonEmptyLabel$[r] = 1$;

END IF

Step 2.2 (Divide). Divide S into $S_1, S_2, \ldots, S_k$ by $U/\mathrm{IND}(\{c_r\})$;

Step 2.3 (Conquer sub-problems). FOR $i = 1$ TO k DO
NonEmptyLabelAttr$(S_i, r+1)$;
END FOR.

Step 2.4 (Merge the solutions). Here, there is no operation because the solutions are stored in the array NonEmptyLabel[].
END Function 1.

Using the above recursive function, an algorithm for computing the non-empty label attribute set of a decision table is developed.

Algorithm 4.3. Computation of The Non-empty Label Attribute Set $L(SO)$
Input: A decision table S and an attribute order $SO : c_1 \prec c_2 \prec \cdots \prec c_{|C|}$
Output: The non-empty label attribute set R_1 of S.

Step 1. $R_1 = \phi$; $r = 1$;
FOR $j = 1$ TO $|C|$ DO
$NonEmptyLabel[j] = 0$;
END FOR

Step 2. NonEmptyLabelAttr$(S, 1)$;

Step 3. FOR $j = 1$ TO $|C|$ DO
IF $NonEmptyLabel[j] == 1$ THEN $R_1 = R_1 \cup \{c_j\}$;
END FOR

Step 4. RETURN R_1.

Suppose $n = |U|$, $m = |C|$. According to the conclusion of literature [45], the average time and space complexities of the Algorithm 4.3 are $T = O(n \times (m + \log n))$ and $S = O(m + n)$.

Obviously, Algorithm 4.3 is an instance of Algorithm 3.13. Given an attribute order of the conditional attributes in a decision table, using the Algorithm 4.3 and divide and conquer method, an efficient attribute reduction algorithm is developed.

Algorithm 4.4. Computation of Attribute Reduction Based on Divide and Conquer Method
Input: A decision table $S = \langle U, A, V, f \rangle$ and an attribute order $SO : c_1 \prec c_2 \prec \cdots \prec c_{|C|}$
Output: Attribute reduction R of S.

Step 1. $U_1^1 = U$, $R = \phi$.

Step 2. Compute the positive region $\mathrm{Pos}_C(D)$, according to Algorithm 3.14.

Step 3. Compute the non-empty label attribute set R_1 by Algorithm 4.3.

Step 4. //Suppose c_N be the maximum label attribute of R_1.
$R = R \cup \{c_{N'}\}$; $R_1 = R_1 - R$;
IF $R1 == \phi$ THEN RETURN R;
ELSE

Generate a new attribute order:
$c_1^1 \prec c_2^1 \prec \cdots \prec c_{|R|}^1 \prec c_1^2 \prec c_2^2 \prec \cdots \prec c_{|R_1|}^2$;
Compute new non-empty label attribute set R_1 of S by Algorithm 4.3;
GOTO Step 4.
END IF

Suppose $n = |U|$ and $m = |C|$, the average time complexity of the Algorithm 4.4 is $O(n \times m \times (m + \log n))$. Its space complexity is $O(m + n)$.

In Algorithm 4.4, Step 1 is the initialization. In Step 2 of Algorithm 4.4, divide and conquer method is used to compute equivalence classes and positive region, thus Step 2 is an instance of Algorithm 3.13. In Step 3, Algorithm 4.3 is used to compute non-empty label attribute set (Algorithm 4.3 is also an instance of Algorithm 3.13). Step 4 is responding to Step 5 of Algorithm 3.13. In Step 4, Algorithm 4.3 is called repeatedly to reduce the redundant attribute set. That is, Algorithm 4.4 is composed of instances of Algorithm 4.3, which illustrates that Algorithm 4.4 is implemented by divide and conquer method. Basically, divide and conquer method is used primarily to compute equivalence classes, positive region, and non-empty label attribute set and delete element cells in discernibility matrix in Algorithm 4.4.

5. A Fast Algorithm for Value Reduction Based on Divide and Conquer Method

Proposition 5.1. *Given a decision table $S = \langle U, A, V, f \rangle$, for all $x \in Pos_C(D)$, a certain rule can be induced by object x.*

Proposition 5.2. *Given a decision table $S = \langle U, A, V, f \rangle$, let us denote by DR a certain rule set of S. $\forall_{d_i \in DR(1 \le i \le |DR|)} d_i$, d_i must be induced by x $(x \in Pos_C(D))$.*

Proposition 5.3. *Given a decision table $S\langle U, A, V, f \rangle$, let us denote by DR certain rule set of S. For all $x \in Pos_C(D)$, let us denote by d_1 the certain rule by x. Let $M_1 = \{B_{xy}^s \mid y \in U\}$. Then, $\forall_{\alpha \in M_1}(M_1 \neq \phi) \Rightarrow \exists_{c \in (d_1|C)}(c \in \alpha)$.*

According to Algorithm 3.13, Propositions 5.1, 5.2, and 5.3, a recursive function and an algorithm for value reduction based on divide and conquer method are developed as follows.

Function 2

DRAVDAC(S, c_r)
//Denote by array CoreValueAttribute[] the result of value reduction of c_r.
//The values of array CoreValueAttribute[] are all 0 initially.

Step 1 (Ending Condition).
IF there is contradiction on S, THEN
return;
END IF

Step 2 (Value reduction on c_r based on divide and conquer method).
Step 2.1 (Divide).

Divide S into k sub-decision tables $S_1, S_2, \ldots, S_k$ on attribute c_r by using tolerance relation.

Step 2.2 (Conquer sub-problems recursively).

Denote by array $Solu_i[|U_i|]$ the solution of S_i.
Where, $S_i = \langle U_i, (C - \{c_r\}) \cup D, V, f \rangle$.
FOR $i = 1$ TO k DO
  $Solu_i$ = DRAVDAC(S_i);
END FOR

Step 2.3 (Merge the solutions).

FOR $j = 1$ TO $|U|$ DO
  FOR i =1 TO k DO
    IF $Solu_i[j] == 1$ THEN break; END IF
  END FOR
  IF $i == k + 1$ THEN CoreValueAttribute[j] =1 END IF
END FOR
END Function 2.

Using Function 2, we present an algorithm for value reduction based on divide and conquer method (see Algorithm 5.4).

Algorithm 5.4. An Algorithm for Value Reduction Based on Divide and Conquer Method

Input: A decision table $S = \langle U, A, V, f \rangle$
Output: The certain rule set DR of S.

Step 1 (Initiation). $ASet = \phi$, $DR = \phi$.

Step 2 (Compute the positive region). According to Algorithm 3.14, compute the positive region $\mathrm{Pos}_C(D)$ of S.

Step 3 (Compute the non-empty label attribute). Assume the order for dividing decision table be $c_1, c_2, \ldots, c_m$ $(m = |C|)$.

Compute the non-empty label attribute set $ASet$ by using Function 1.

Step 4 (Value reduction on attribute set $ASet$). Let $ASet = \{c'_1, c'_2, \ldots, c'_{|ASet|}\}$ and the divide order be $c'_1, c'_2, \ldots, c'_{|ASet|}$.

FOR i = $|ASet|$ TO 1 DO
  FOR $j = 1$ TO $|U|$ DO
    CoreValueAttribute[j] = 0.
  END FOR
  $C = ASet$.
  Invoke Function 2: DRAVDAC($S, \{c'_i\}$).
  Update S, according to the array CoreValueAttribute[].
END FOR.

Step 5 (Get the rule set).

FOR $i = 1$ TO $|U|$ DO
  IF $x_i \in \mathrm{Pos}_C(D)$ THEN
    Construct a rule d_i in terms with x_i; $DR = DR \cup \{d_i\}$;

ENE IF
END FOR

Step 6. RETURN DR.

Suppose $|C| = m, |U| = n$. The time complexity of Step 1 is $O(m+n)$. The average time complexities of Steps 2 and 3 are $O(n \times (m + \log n))$ [45]. The time complexities of Steps 5 and 6 are both $O(m \times n)$. Now, let us analyze the time complexity of Step 4.

In Step 4, let the number of non-empty label attribute set be u. Then, the time complexity of Step 4 is $O(u \times (2n + T(1,n)))$, where $T(1,n)$ is an instance of $T(r,n)$ which can be expressed by the following recursive equation:

$$T(r,n) = \begin{cases} 1. \quad n == 1 \\ 2n. \quad r \geq u \\ 2n + T(r+1,n). \quad \forall_{x,y \in U}(T_B(x,y)),\ (B = \{c_1, c_2, \ldots, c_r\}) \\ pn + T(r, n_1 + n') + T(r, n_2 + n') + \cdots + T(r, n_p + n'). \\ (n_1 + n_2 + \cdots + n_p + n') = n,\ p = |\{c_r(x) \mid c_r(x) \neq *, x \in U\}|. \end{cases} \tag{5.1}$$

$T(1,n)$ is between $O(u \times n)$ and $T(p_1 \times p_2 \times \cdots \times p_u \times n)$. So the time complexity of Step 4 is between $O(u^2 \times n)$ and $T(u \times p_1 \times p_2 \times \cdots \times p_u \times n)$. Thus, the time complexity of Algorithm 5.4 is between $\max(O(n \times (m + \log n)), O(u^2 \times n))$ and $O(u \times n \times p_1 \times p_2 \times \cdots \times p_u)$.

Suppose the data obey the uniform distribution. The time complexity of Algorithm 5.4 is $O(n^{\log_{((p+1)/2)}p} \times m) + O(n \times m) = O(n^{\log_{((p+1)/2)}p} \times m)$. When $p \geq 2$, the time complexity of Algorithm 5.4 is less than $O(n^2 \times m)$. When $p \geq 5$, the time complexity of Algorithm is less than $O(n^{1.5} \times m)$.

The space complexity of Algorithm 5.4 is $O(m \times n)$.

6. Experimental Evaluations

In order to test the efficiency of knowledge reduction based on divide and conquer method, some experiments have been performed on a personal computer. The experiments are shown as follows.

6.1. The Experimental Evaluations on UCI Data Sets

In this experiment, some experimental evaluations are done to present the efficiency and recognition results of Algorithms 4.4 and 5.4. In the mean time, some famous approaches for data mining are used to compare with our methods.

The test course is as follows. First, 11 uci data sets (Zoo, Iris, Wine, Machine, Glass, Voting, Wdbc, Balance-scale, Breast, Crx, and Tic-tac-toe) are used. Second, our methods: the algorithm for discretization [14] (it is an improved one based on the discretization method in [28]), the algorithm for attribute reduction (Algorithm 4.4), and the algorithm for attribute value reduction (Algorithm 5.4) are used to test the 11 uci data sets. Third, 5 methods (KNN, SVM, C4.5, Naive Bayes, and CART) are also used to test the data sets, which belong to the "top 10 algorithms in data mining" [44], and their source codes are afforded by Weka software. Weka (http://en.wikipedia.org/wiki/Weka_(machine_learning)) is used as the

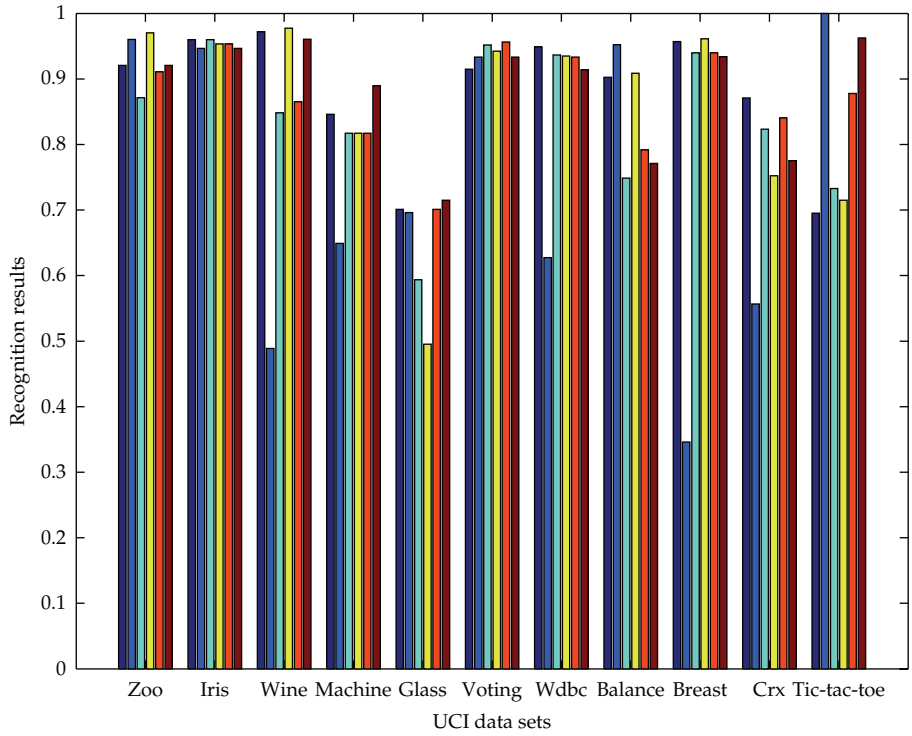

Figure 2: Recognition result on uci data sets.

Table 1: Specifications of 11 uci data sets.

Data sets	Number of records	Number of attributes	Number of decision values
Zoo	101	17	7
Iris	150	4	3
Wine	178	13	3
Machine	209	7	8
Glass	214	9	7
Voting	435	16	2
Wdbc	569	28	2
Balance scale	625	4	5
Breast	684	9	2
Crx	690	15	2
Tic-tac-toe	958	9	2

experimental plat and "Java Eclipse Weka" as the developing tool. The test method is LOOCV (Leave One Out Cross Validation). The specifications of the experimental computer are an Intel(R) Core(TM2) Quad CPU Q8200 @2.33 GHz CPU, 2 GB RAM, and Microsoft Windows 7. The specifications of 11 data sets and the experimental results are as follows.

From Table 1 and Figure 2, it can be found that the recognition results of our methods on the 11 uci data sets are closed to the ones of KNN and CART, which are better than Naive Bayes, C4.5, and SVM.

Table 2: The recognition of 6 algorithms on uci data sets.

Data sets	KNN	SVM	C4.5	Naive bayes	CART	Our methods
Zoo	0.9208	0.9603	0.8713	0.9702	0.9109	0.9208
Iris	0.9600	0.9467	0.9600	0.9533	0.9533	0.9467
Wine	0.9719	0.4887	0.8483	0.9775	0.8652	0.9607
Machine	0.8461	0.6490	0.8173	0.8173	0.8173	0.8894
Glass	0.7009	0.6963	0.5935	0.4953	0.7009	0.7149
Voting	0.9149	0.9333	0.9517	0.9425	0.9563	0.9333
Wdbc	0.9490	0.6274	0.9367	0.9349	0.9332	0.9139
Balance scale	0.9024	0.952	0.7488	0.9088	0.7920	0.7712
Breast	0.9570	0.3462	0.9399	0.9613	0.9399	0.9341
Crx	0.8710	0.5565	0.8232	0.7522	0.8406	0.7754
Tic-tac-toe	0.6952	1.0000	0.7328	0.7150	0.8779	0.9624
Average	0.8808	0.7415	0.8385	0.8571	0.8716	0.8839

Table 3: The recognition of 6 algorithms on KDDCUP99 data sets ($\leq 10^4$ records).

Number of records	KNN	SVM	C4.5	Naive bayes	CART	Our methods
1000	0.9950	0.9550	0.9800	0.9800	0.9870	0.9950
2000	0.9980	0.9700	0.9935	0.9925	0.9920	0.9935
3000	0.9977	0.9850	0.9947	0.9753	0.9950	0.9947
4000	0.9967	0.9750	0.9940	0.9483	0.9935	0.9940
5000	0.9984	0.9770	0.9956	0.9760	0.9958	0.9954
6000	0.9986	0.9791	0.9968	0.9553	0.9965	0.9975
7000	0.9980	0.9850	0.9958	0.9905	0.9951	0.9956
8000	0.9985	0.9831	0.9958	0.9555	0.9956	0.9965
9000	0.9983	0.9877	0.9970	0.9690	0.9955	0.9964
10000	0.9986	0.9880	0.9971	0.9449	0.9968	0.9971

6.2. The Experimental Results on KDDCUP99 Data Sets

In order to test the efficiency of our methods for processing large data sets, some experiments are done on KDDCUP99 data sets with 4898432 records, 41 condition attributes, and 23 decision classifications (http://kdd.ics.uci.edu/databases//kddcup99/kddcup99.html). Our methods consist of the discretization algorithm [14], Algorithms 4.4, and 5.4 still. Weka is used as the experimental plat and "Java Eclipse Weka" as the developing tool (Table 2). The test method is 10 cross-validation. The specifications of the experimentation computer are an Intel(R) Core(TM2) Quad CPU Q8200 @2.33 GHz CPU, 4 GB RAM, and Microsoft Windows Server 2008. The experimental results are as follows.

First, the experiments are done on 10 data sets ($\leq 10^4$ records) from the original KDDCUP99 data sets. The experimental evaluations are showed in Tables 3, 4, and 5, where the time unit is "ms" in Tables 4 and 5.

From Tables 3, 4, and 5, it can be found that it will cost much time to train by SVM, which shows that SVM is not a good way to process KDDCUP99 data sets with large records. Thus, in the following experiments, SVM will be not tested.

Second, the experiments are done on 10 data sets ($\leq 10^5$ records) from the original KDDCUP99 data sets. The experimental evaluations are showed in Tables 6 and 7, where "Tr" is the training time, "Te" is the test time, and the time unit is "ms" in Table 7.

Table 4: The training time of 6 algorithms on KDDCUP99 data sets ($\leq 10^4$ records).

Number of records	KNN	SVM	C4.5	Naive bayes	CART	Our methods
1000	3	87	50	9	120	143
2000	3	259	47	12	173	78
3000	2	631	18	16	280	130
4000	0	1071	125	22	384	192
5000	0	1731	173	31	543	226
6000	0	2467	223	41	672	311
7000	0	3617	281	129	861	541
8000	2	4591	329	70	1155	822
9000	2	6171	364	178	1262	1261
10000	3	7496	378	96.7	1629	1415

Table 5: The test time of 6 algorithms on KDDCUP99 data sets ($\leq 10^4$ records).

Number of records	KNN	SVM	C4.5	Naive bayes	CART	Our methods
1000	41	6	0	25	2	0
2000	122	9	0	34	0	0
3000	279	18	0	50	0	0
4000	495	40	0	67	0	0
5000	782	59	2	81	0	0
6000	1197	89	0	94	0	2
7000	1805	131	2	125	0	0
8000	2289	154	2	249	2	0
9000	2880	207	0	178	0	0
10000	3588	271	2	162	0	0

From Tables 6 and 7, it can be found that the recognition is lower than others by Naive Bayes and much test time is needed for KNN. Thus, Naive Bayes and KNN will be not tested in the following experiments.

Third, the experiments are done on 10 data sets ($\leq 10^6$ records) from the original KDDCUP99 data sets. The experimental evaluations are showed in Table 8, where "RRate" is the recognition rate, "Tr" is the training time, "Te" is the testing time, and the time unit is "ms" in Table 8.

Fourth, the experiments are done on 10 data sets ($< 5 \times 10^6$ records) from the original KDDCUP99 data sets. The experimental results are showed in Table 9, where "RRate" is the recognition rate, "Tr" is the training time, "Te" is the testing time, "-" is the overflow of memory, and the time unit is "second" in Table 9.

From Tables 8 and 9, it can be found that C4.5 is the best one and our method is the second best one among C4.5, CART, and our method. Due to the high complexity for discretization, our method can not complete the knowledge reduction of 4898432 records in this experiment.

Table 6: The recognition of 5 algorithms on KDDCUP99 data sets ($\leq 10^5$ records).

Number of records	KNN	C4.5	Naive bayes	CART	Our methods
10000	0.9985	0.9969	0.9804	0.9970	0.9973
20000	0.9987	0.9979	0.9490	0.9980	0.9980
30000	0.9990	0.9985	0.9560	0.9987	0.9987
40000	0.9989	0.9988	0.9365	0.9987	0.9989
50000	0.9991	0.9989	0.9613	0.9989	0.9989
60000	0.9992	0.9989	0.9627	0.9990	0.9989
70000	0.9992	0.9992	0.9438	0.9990	0.9990
80000	0.9992	0.9992	0.9249	0.9992	0.9992
90000	0.9992	0.9993	0.9097	0.9991	0.9992
100000	0.9992	0.9992	0.9118	0.9991	0.9992

Table 7: The running time of 5 algorithms on KDDCUP99 data sets ($\leq 10^5$ records).

Number of records	KNN		C4.5		Naive bayes		CART		Our methods	
	Tr	Te	Tr	Te	Tr	Te	Tr	Te	Tr	Te
10000	2	3622	412	0	100	164	1632	0	1496	3
20000	3	15045	1225	0	281	321	4984	0	4367	0
30000	6	34592	2225	2	474	557	8964	0	8165	0
40000	9	60845	3582	5	704	774	15833	2	12815	5
50000	13	95023	5527	2	930	1045	19140	2	20003	6
60000	17	139299	7920	6	1167	1148	28805	2	30450	9
70000	22	196827	10084	9	1422	1438	31774	3	34909	2
80000	28	248624	12069	5	1688	2073	40056	3	39407	9
90000	27	310826	14461	10	1959	1716	44023	3	47688	6
100000	33	386018	16673	13	2185	2044	54288	3	49394	8

6.3. The Conclusions of Experimental Evaluations

Now, we will give some conclusions for our approaches compared with KNN, SVM, C4.5, Naive Bayes, and CART, according to the LOOCV experimental results on the uci data sets and 10 cross-validation experimental results on KDDCUP99 data sets.

(1) Compared with KNN, SVM, and Naive Bayes, the LOOCV recognition results by our methods on uci data sets are better than KNN, SVM, and Bayes. Furthermore, our methods on KDDCUP99 data sets have higher efficiency than KNN, SVM, and Naive Bayes, while they also have good recognitions results.

(2) Compared with CART, the LOOCV recognition results by our methods on uci data sets are closed to CART. But our methods can process larger data sets than CART on KDDCUP99 data sets, while they both have good recognition results.

(3) Compared with C4.5, the LOOCV recognition results by our methods on uci data sets are better than C4.5. Furthermore, the test time by our methods on KDDCUP99 data sets is less than C4.5, while C4.5 can process larger data sets than our methods. After these two methods are analyzed, we find that our methods are more complex than C4.5, due to the complex discretization (C4.5 can process the decision table with continuous values directly, while the discretization should be necessary for

Table 8: The experimental results of 3 algorithms on KDDCUP99 data sets ($\leq 10^6$ records).

Number of records	C4.5			CART			Our methods		
	RRate	Tr	Te	RRate	Tr	Te	RRate	Tr	Te
100000	0.9992	18730	31	0.9991	56208	8	0.9992	51692	24
200000	0.9997	52971	32	0.9994	113297	8	0.9995	113496	16
300000	0.9997	78118	47	0.9997	182997	0	0.9997	178769	31
400000	0.9998	111026	86	0.9997	327640	23	0.9997	313085	40
500000	0.9997	163746	94	0.9998	391179	16	0.9997	410342	55
600000	0.9998	218152	110	0.9996	446004	16	0.9997	538169	55
700000	0.9998	226879	125	0.9997	610749	24	0.9998	719630	78
800000	0.9999	387911	148	0.9999	1015165	32	0.9999	1143149	109
900000	0.9998	304466	195	0.9999	1595899	382	0.9998	1512619	133
1000000	0.9999	303403	203	0.9997	1583910	367	0.9998	1590494	140

Table 9: The experimental results of 3 algorithms on KDDCUP99 data sets ($<5 \times 10^6$ records).

Number of records	C4.5			CART			Our methods		
	RRate	Tr	Te	RRate	Tr	Te	RRate	Tr	Te
489843	0.9998	158	0.124	0.9997	386	0.031	0.9998	499	0.063
979686	0.9999	330	0.187	0.9998	1517	0.312	0.9998	1472	0.141
1469529	0.9999	706	0.312	0.9999	5851	5.554	0.9999	4026	0.156
1959372	0.9999	883	0.499	—	—	—	0.9999	6650	0.421
2449216	0.9999	1143	0.624	—	—	—	0.9999	11446	0.476
2939059	0.9999	1176	0.655	—	—	—	0.9999	14023	0.578
3428902	0.9999	1849	0.827	—	—	—	0.9999	35361	0.606
3918745	0.9999	2043	0.923	—	—	—	—	—	—
4408588	0.9999	2567	1.106	—	—	—	—	—	—
4898432	0.9999	2916	1.217	—	—	—	—	—	—

our methods). As a coin has two sides, enough learning contributes to the better rule sets, thus less test time is needed by our methods than C4.5.

Therefore, the knowledge reduction approaches based on divide and conquer method are efficient to process large data set, although they need to be improved further in the future.

7. Conclusions

In this paper, the abstract process of knowledge reduction based on divide and conquer method is concluded, which is original from the approaches under the equivalence relation and the one under the tolerance relation. Furthermore, an example for computing positive region of the decision table is introduced. After that, two algorithms for knowledge reduction based on divide and conquer method, including an algorithm for attribute reduction and an algorithm for attribute value reduction, are presented, respectively. The proposed algorithms are efficient to process the knowledge reduction on uci data sets and KDDCUP99 data set, according to the experimental evaluations. Therefore, the divide and conquer method is an

efficient and, therefore, suitable method to be used to knowledge reduction algorithms in rough set theory. With this efficiency, widespread industrial application of rough set theory may become possible.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants no. 61073146, no. 61272060, no. 61203308, and no. 41201378, Scientific and Technological Cooperation Projects between China and Poland Government, under Grant no. 34-5, Natural Science Foundation Project of CQ CSTC under Grant no. cstc2012jjA1649, and Doctor Foundation of Chongqing University of Posts and Telecommunications under Grant no. A2012-08.

References

[1] A. Bargiela and W. Pedryc, *Human-Centric Information Processing Through Granular Modelling*, Springer, Berlin, Germany, 1997.

[2] W. Pedrycz, A. Skowron, and V. Kreinovich, *Handbook of Granular Computing*, Wiley Interscience, New York, NY, USA, 2007.

[3] J. T. Yao, *Novel Developments in Granular Computing, Applications for Advanced Human Reasoning and Soft Computation*, Information Science Reference, Herskey, Pa, USA, 2010.

[4] J. Yao, "A ten-year review of granular computing," in *Proceedings of the IEEE International Conference on Granular Computing (GRC '07)*, pp. 734–739, November 2007.

[5] Y. Y. Yao, "Granular computing: past, present and future," in *Proceedings of the IEEE International Conference on Granular Computing*, pp. 80–85, 2008.

[6] Y. Y. Yao and J. G. Luo., "Top-down progressive computing," in *Proceedings of the RSKT*, pp. 734–742, Springer, Regina, Canada, 2011.

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

[8] Z. Pawlak and A. Skowron, "Rudiments of rough sets," *Information Sciences*, vol. 177, no. 1, pp. 3–27, 2007.

[9] Z. Pawlak and A. Skowron, "Rough sets: some extensions," *Information Sciences*, vol. 177, no. 1, pp. 28–40, 2007.

[10] Z. Pawlak and A. Skowron, "Rough sets and boolean reasoning," *Information Sciences*, vol. 177, no. 1, pp. 41–73, 2007.

[11] J. W. Grzymala-Busse, "A new version of the rule induction system LERS," *Fundamenta Informaticae*, vol. 31, no. 1, pp. 27–39, 1997.

[12] F. Hu and G. Y. Wang, "A quick reduction algorithm based on attribute order," *Chinese Journal of Computers*, vol. 30, no. 8, pp. 1430–1435, 2007 (Chinese).

[13] F. Hu, G. Wang, and Y. Xia, "Attribute core computing based on divide and conquer method," in *Proceedings of the International Conference on Rough Sets and Intelligent Systems Paradigms (RSEISP '07)*, M. Kryszkiewicz et al., Ed., Lecture Notes in Artificial Intelligence 4585, pp. 310–319, springer, Warsaw, Poland, 2007.

[14] F. Hu, G. Wang, and J. Dai, "Quick discretization algorithm for rough set based on dynamic clustering," *Journal of Southwest Jiaotong University*, vol. 45, no. 6, pp. 977–983, 2010 (Chinese).

[15] K. Hu, Y. Lu, and C. Shi, "Feature ranking in rough sets," *AI Communications*, vol. 16, no. 1, pp. 41–50, 2003.

[16] X. Hu, N. Cercone, and N. Cercone, "Learning in relational databases: a rough set approach," *Computational Intelligence*, vol. 11, no. 2, pp. 323–338, 1995.

[17] M. Kryszkiewicz and H. Rybinski, "Computation of reducts of composed information systems," *Fundamenta Informaticae*, vol. 27, no. 2-3, pp. 183–195, 1996.

[18] D. F. Li, G. B. Li, and W. Zhang, "U/a partition based smallest reduction construction," *Journal of Wuhan University*, vol. 51, pp. 269–272, 2005 (Chinese).

[19] T. Y. Lin and N. Cercone, Eds., *Rough Sets and Data Mining-Analysis of Imperfect Data*, Kluwer Academic Publishers, Boston, Mass, USA, 1997.

[20] Q.-H. Liu, F. Li, F. Min, M. Ye, and G.-W. Yang, "Efficient knowledge reduction algorithm based on new conditional information entropy," *Control and Decision*, vol. 20, no. 8, pp. 878–882, 2005 (Chinese).

[21] S. W. Liu, Q.-J. Sheng, B. Wu, Z.-Z. Shi, F. Hu et al., "Research on efficient algorithms for rough set methods," *Chinese Journal of Computers*, vol. 40, pp. 637–642, 2003 (Chinese).

[22] D. Miao, C. Gao, N. Zhang, and Z. Zhang, "Diverse reduct subspaces based co-training for partially labeled data," *International Journal of Approximate Reasoning*, vol. 52, no. 8, pp. 1103–1117, 2011.

[23] J. M. Mikhail, P. Marcin, and Z. Beata, "On partial covers, reducts and decision rules with weights," in *Proceedings on Transactions on Rough Sets 6*, vol. 4374 of *Lecture Notes in Computer Sciences 4374*, pp. 211–246, Springer, Berlin, Germany, 2007.

[24] R. C. Michal, J. W. Grzymala-Busse, W. P. Neil, and T. Soe, "The rule induction system LERSa new version for personal computers," in *Proceeding of the International Workshop on Rough Sets and Knowledge Discovery (RSKD '93)*, Alberta, Canada, 1993.

[25] J. M. Moshkov, A. Skowron, and Z. Suraj, "On minimal rule sets for almost all binary information systems," *Fundamenta Informaticae*, vol. 80, no. 1–3, pp. 247–258, 2008.

[26] H. S. Nguyen, "From optimal hyperplanes to optimal decision trees," *Fundamenta Informaticae*, vol. 34, no. 1-2, pp. 145–174, 1998.

[27] H. S. Nguyen, "A soft decision tree," in *Proceedings of the Intelligent Information Systems (IIS '02)*, M. A. Klopotek, S. Wierzchon, and M. Michalewicz, Eds., Advanced in Soft Computing, pp. 57–66, Springer, Berlin, Germany, 2002.

[28] H. S. Nguyen, "Approximate Boolean reasoning: foundations and applications in data mining," in *Transactions on Rough Sets 5*, vol. 4100, pp. 334–506, Springer, Berlinm Germany, 2006.

[29] H. S. Nguyen, A. Skowron, and P. Synak, "Discovery of data patterns with applications to decomposition and classification problems," in *Rough sets in knowledge discovery 2*, L. Polkowski and A. Skowron, Eds., vol. 19, pp. 55–97, Physica, Berlin, Germany, 1998.

[30] S. H. Nguyen and H. S. Nguyen, "Some efficient algorithms for rough set methods," in *Proceedings of the Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '96)*, pp. 1451–1456, Granada, Spain, 1996.

[31] S. H. Nguyen and H. S. Nguyen, "Pattern extraction from data," *Fundamenta Informaticae*, vol. 34, no. 1-2, pp. 129–144, 1998.

[32] S. K. Pal, L. Polkowski, and A. Skowron, *Rough-Neural Computing: Techniques for Computing with Words, Cognitive Technologies*, Springer, Berlin, Germany, 2004.

[33] Y. Qian, J. Liang, W. Pedrycz, and C. Dang, "An efficient accelerator for attribute reduction from incomplete data in rough set framework," *Pattern Recognition*, vol. 44, no. 8, pp. 1658–1670, 2011.

[34] A. Skowron and C. Rauszer, "The discernibility functions matrics and functions in information systems," in *Intelligent Decision Support—Handbook of Applications and Advances of the Rough Sets Theory*, R. Slowinski, Ed., pp. 331–362, Kluwer Academic Publisher, Dordrecht, The Netherlands, 1992.

[35] A. Skowron, Z. Pawlak, J. Komorowski, and L. Polkowski, "A rough set perspective on data and knowledge," in *Handbook of KDD*, W. Kloesgen and J. Zytkow, Eds., pp. 134–149, Oxford University Press, Oxford, UK, 2002.

[36] G. Y. Wang, *Rough Set Theory and Knowledge Acquisition*, Xi'an Jiaotong University Press, 2001.

[37] G. Y. Wang, H. Yu, and D. C. Yang, "Decision table reduction based on conditional information entropy," *Chinese Journal of Computers*, vol. 25, no. 7, pp. 759–766, 2002 (Chinese).

[38] Jue Wang and Ju Wang, "Reduction algorithms based on discernibility matrix: the ordered attributes method," *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 489–504, 2001.

[39] Y. Yao and Y. Zhao, "Discernibility matrix simplification for constructing attribute reducts," *Information Sciences*, vol. 179, no. 7, pp. 867–882, 2009.

[40] H.-Z. Yang, L. Yee, and M.-W. Shao, "Rule acquisition and attribute reduction in real decision formal contexts," *Soft Computing*, vol. 15, no. 6, pp. 1115–1128, 2011.

[41] M. Zhao, *The data description based on reduct [Ph.D. dissertation]*, Institute of Automation, Chinese Academy of Sciences, Bejing, China, 2004.

[42] J. Zhou, D. Miao, W. Pedrycz, and H. Zhang, "Analysis of alternative objective functions for attribute reduction in complete decision tables," *Soft Computing*, vol. 15, no. 8, pp. 1601–1616, 2011.

[43] W. Ziarko, N. Cerone, and X. Hu, "Rule discovery from database with decision matrices," in *Proceedings of the 9th International Symposium on Foundation of Intelligent Systems (ISMIS '96)*, pp. 653–662, Zakopane, Poland, May 1996.

[44] X. Wu, V. P. Kumar, R. S. Quinlan et al. et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[45] F. Hu and G. Y. Wang, "Analysis of the complexity of quick sort for two-dimensional tables," *Chinese Journal of Computers*, vol. 30, no. 6, pp. 963–968, 2007 (Chinese).