Journal of Computer Science and Cybernetics, V.39, N.4 (2023), 313–321
DOI no 10.15625/1813-9663/18680

A NOVEL ALGORITHM FOR FINDING ALL REDUCTS IN THE INCOMPLETE DECISION TABLE

PHAM VIET ANH[1,2,*], VU DUC THI[3], NGUYEN NGOC CUONG[4]

[1]*Graduate University of Science and Technology, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet Street, Cau Giay District, Ha Noi, Viet Nam*
[2]*HaUI Institute of Technology, Hanoi University of Industry, 298 Cau Dien Street, Bac Tu Liem District, Ha Noi, Viet Nam*
[3]*Information Technology Institute, Vietnam National University, 144 Xuan Thuy Street, Cau Giay District, Ha Noi, Viet Nam*
[4]*General Department of Logistics and Engineering, Ministry of Public Security, 17 To Huu Street, Trung Van Ward, Nam Tu Liem District, Ha Noi, Viet Nam*

Abstract. Attribute reduction, or feature selection for decision tables is a fundamental problem of rough set theory. Currently, many scientists are interested in and developing these issues. Unfortunately, most studies focus mainly on the complete decision table. On incomplete decision tables, researchers have proposed tolerance relations and designed attribute reduction algorithms based on different measures. However, these algorithms only return a reduct and do not preserve information in the decision tables. This paper proposes an efficient method to determine entire reducts of incomplete decision tables according to the relational database approach. In the complex case, this algorithm has exponential computational complexity. However, this algorithm has polynomial computational complexity in the different cases of databases.

Keywords. The reduct, rough set theory, tolerance relation, incomplete decision table.

1. INTRODUCTION

Attribute reduction for decision information systems is the process of removing redundant attributes in the condition attribute set without affecting the classification of the objects. Based on the reduct set obtained, the rule generation and classification accuracy achieve the highest efficiency. Up to now, there have been many research works about attribute reduction algorithms according to rough set theory [1]. However, these algorithms only determine a reduct based on an evaluation criterion with polynomial computational complexity (algorithms following the heuristic approach) without solving the problem of finding all the reducts in the decision table. Compared with only finding a reduct, algorithms that find reducts in a decision table can provide results that are close to the performance of the best reduct set. Therefore, research on these algorithms brings lots of significance.

Nowadays, decision tables often miss information and are called incomplete decision

*Corresponding author.
E-mail addresses: anhpv@haui.edu.vn (P.V. Anh); vdthi@vnu.edu.vn (V.D. Thi); cuongnn.hvan@gmail.com (N.N. Cuong).

tables. The problem of finding reduct on the incomplete decision table is considered a general issue. Many algorithms eliminate object sets with missing data or replace missing data with other values to handle the issue of incomplete decision tables for knowledge mining problems. This dramatically affects the results of data mining algorithms, especially for attribute reduction problems. M. Kryszkiewics [2] introduced the concept of a tolerance relation and constructed a tolerance rough set model to study the problem of incomplete decision tables. Based on the tolerance relation and Boolean reasoning methodology, the authors in [3] have proposed a method to find the reducts in incomplete decision tables. According to the approach using the relational data model [4,5], authors in [6] proposed an algorithm with polynomial time for finding all the reducts in the consistent decision table. This algorithm can select columns (attributes) in the decision table. The authors in [7] have proposed a polynomial algorithm that reduces the rows (objects) in the decision table. Thus, these two efficient algorithms can remove redundant columns and rows on the decision table.

On the other hand, it is essential to find all reducts in the complete decision table. In [8], the authors have proposed a method to find all reducts in a consistent and complete decision table. The authors have demonstrated that the problem of finding all reducts on this decision table has an exponential computational complexity following the number of attributes. It means that we need to prove the existence of an algorithm with exponential computational complexity when finding all these reducts and prove that the time complexity of this problem is not less than the exponential function. We also know that finding a reduct on a complete and consistent decision table is performed by a polynomial algorithm. However, the problem of finding a reduct with the smallest size and ensuring the best classification efficiency on machine learning models is challenging. It indicates that no polynomial algorithm so far can solve this. Because the set of all reducts on the complete and consistent decision table is essential, we have proved this set of reducts is equivalent to the Sperner system (this system is a combinatorial system in which its elements do not contain each other) in [9]. It emphasizes that the study of reduct sets can lead to the study of Sperner systems.

2. PRELIMINARY

This section will summarize some basic concepts of rough set theory [7, 10–16]. The information system or decision table is known as a set of four elements $\mathcal{S} = (U, C \cup D, \mathcal{F}, \mathcal{P})$, where $U = \{u_1, u_2, \ldots, u_n\}$ is a non-empty set, comprising objects; $C = \{c_1, c_2, \ldots, c_m\}$ is a set of condition attribute; D is a set of decision attributes which satisfies $C \cap D = \varnothing$, and $\mathcal{F} = \bigcup_{b \in C \cup D} \mathcal{F}_{\{b\}}$ with $\mathcal{F}_{\{b\}}$ is the value set of the attribute b; $\mathcal{P} : U \times (C \cup D) \to \mathcal{F}$ is called the information function. For any $u \in U$, $b \in C \cup D$, function $\mathcal{P}$ has $\mathcal{P}(u, b) \in V_{\{b\}}$. Without losing the comprehensive features, suppose that D only contains one decision attribute d (in the case size of D higher than 1, it can transform into an attribute by using encryption [8]). Thus, we only need to consider $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, in which $\{d\} \notin C$.

An attribute subset $B \subseteq C \cup \{d\}$ determines an indiscernibility relation and is called an equivalence relation: $IR(B) = \left\{(u, v) \in U^2 \,|\forall b \in B, \mathcal{P}(u, b) = \mathcal{P}(v, b)\right\}$. $IR(B)$ generates a partition on U, denoted by $U/B = \{B_1, B_2, ..., B_m\}$. Each element $B_i \in U/B$ $(1 \leq i \leq m)$ is considered an equivalence class. For any $U' \subseteq U$ and $P \subseteq C$, we have P-upper approximation and P-lower approximation of U' are crisp sets and presented by $\overline{P}U' = \left\{u \in U \mid [u]_P \cap U' \neq \emptyset\right\}$ and $\underline{P}U' = \left\{u \in U \mid [u]_P \subseteq U'\right\}$, respectively. The P-

boundary region of $U^{'}$ is determined by the formula $\overline{P}U^{'} \backslash \underline{P}U^{'}$ and the P-positive region of $\{d\}$ is calculated by $POS_P(\{d\}) = \bigcup_{U' \in U/D} (\underline{P}U^{'})$. If $POS_C(\{d\}) = U$ or functional dependency (FD) $C \to \{d\}$ is true, then $\mathcal{S}$ is consistent, whereas $\mathcal{S}$ is inconsistent. If $\mathcal{S}$ is an inconsistent decision table, then $POS_C(\{d\})$ is the maximum subset of U that satisfies the FD $C \to \{d\}$.

Definition 2.1. Given a decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, a subset $A \subseteq C$ is called a reduct of C if satisfies:
(i) $POS_A(\{d\}) = POS_C(\{d\})$.
(ii) $\forall\ A' \subset\ B(POS_{A'}(\{d\}) \neq POS_C(\{d\}))$.

If $\mathcal{S}$ is a consistent decision table, then the Definition 2.1 indicates that A is a reduct set of C if $A \to \{d\}$ and $\forall\ A' \subset\ A, A' \nrightarrow \{d\}$.

Definition 2.2. Given a finite set of attributes $R = \{a_1, a_2, ..., a_n\}$ and the possible values set of attributes a_i is $\mathcal{D}(a_i)$ with $a_i \in R$, a relation $\mathcal{L}$ over R is the tuples set $\{l_1, ..., l_m\}$, in which $l_j : R \to \bigcup_{a_i \in R} \mathcal{D}(a_i),\ 1 \leq j \leq m$ is a function that satisfies $l_j(a_i) \in \mathcal{D}(a_i)$.

Consider a relation $\mathcal{L} = \{l_1, \ldots, l_m\}$ over R. Any pair of attribute sets $P, Q \subseteq R$ is considered the FD over R, and denoted by $P \to Q$, if and only if $(\forall\ l_i l_j \in \mathcal{L})\ ((\forall\ l \in P)\ (l_i(u) = l_j(u))) \Longrightarrow ((\forall\ v \in Q)\ (l_i(v) = l_j(v)))$. The set $FDr = \{(P, Q)\,|P, Q \subseteq R \wedge P \to Q\}$ is called the complete family of FDs in $\mathcal{L}$.

Definition 2.3. Let $\mathcal{L} = \{l_1, \ldots, l_m\}$ be a relation on $R = \{a_1, \ldots, a_n\}$. If $\forall a_i \in R$ has $\mathcal{D}_{a_i}$ and $* \in \mathcal{D}_{a_i}$ where $*$ is "missing value" $l_j : R \to \cup \mathcal{D}_{a_i}$ so $l_j(a_i) \in \mathcal{D}_{a_i}$.

Definition 2.4. Given a relation $\mathcal{L}$ on R and $B \subseteq R$, then we denote $l_i \sim l_j(B)$ if $b \in B$: $l_i(b) = l_j(b)$ or $l_i(b) = *$ or $l_j(b) = *$.

Definition 2.5. Let $\mathcal{L} = \{l_1, \ldots, l_m\}$ on $R = \{a_1, a_2, ..., a_n\}$. Then $A, B \subseteq R$ and A tolerance determines (TD) B presented by $A \xrightarrow{t} B$ if $(\forall\ l_i, l_j \in r)$ if $(l_i \sim l_j(A)$ then $l_i \sim l_j(B)$.

We set $TRr = \left\{(A, B) \,\middle|\, A, B \subseteq R \vee A \xrightarrow{t} B\right\}$. It can be easily seen that

(i) $(A, A) \in TR_r\ \forall A \subseteq R$.

(ii) $(A, B) \in TR_r$ then $A \subseteq \mathcal{X},\ \mathcal{Y} \subseteq B$ has $(\mathcal{X}, \mathcal{Y}) \in TR_r$.

(iii) $(A, \mathcal{X}) \in TR_r, (\mathcal{X}, B) \in TR_r \Longrightarrow (A, B) \in TR_r$.

Set $A^+ = \left\{a \in R : A \xrightarrow{t} \{a\}\right\}$.

Definition 2.6. Given an incomplete decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$ with $* \notin \mathcal{D}_d$ (it emphasizes that the value domain of d does not comprise $*$), if $C \xrightarrow{t} \{d\}$, then $\mathcal{S}$ is considered the incomplete decision table which is consistent.

It can be easily shown that if $\mathcal{S}$ is an inconsistent decision table, we can experiment by using a method that has the computational time according to the polynomial function on objects of U to remove the elements, making $\mathcal{S}$ consistently. After the removal process, we obtain the set $\mathbb{O}$ then $\mathcal{S} = (\mathbb{O}, C \cup \{d\}, \mathcal{F}, \mathcal{P})$ is consistent.

Definition 2.7. Given a consistent incomplete decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, an attribute subset G is called a reduct of $\mathcal{S}$ if $G \subseteq C : G \xrightarrow{t} \{d\}$ and $\forall\ G' \subsetneq G$ then $G' \overset{t}{\nrightarrow} \{d\}$ (it emphasizes that if G' is a crucial subset of G then G' does not tolerance determine d), set $PRD(C) = \{G\ :\ G$ is a reduct of $\mathcal{S}\}$.

Definition 2.8. Suppose that $R = \{a_1, a_2, ..., a_n\}$ and $\mathcal{K} = \{Z_1, Z_2, ..., Z_m\}$ is the Sperner system (SPS) on R if $\forall\ i, j,\ Z_i \nsubseteq Z_j$.

Definition 2.9. Given a Sperner system $\mathcal{K} = \{Z_1, Z_2, ..., Z_m\}$ on R, then $\mathcal{K}^{-1}$ is called the anti-key of $\mathcal{K}$ if $\mathcal{K}^{-1} = \{B \subsetneq R : (Z \in \mathcal{K} \Longrightarrow Z \nsubseteq B \text{ and } B \subsetneq C) \text{ then } \exists\ Z \in \mathcal{K} : Z \subseteq C\}$.

$\mathcal{K}^{-1}$ is one of the subsets of R, which does not include the elements of $\mathcal{K}$, essentially, they represent the largest non-key set. Evidently, $\mathcal{K}^{-1}$ is also considered an SPS. If a minimum key set exists, an anti-key set will also exist. From this basis, it can be easily seen that the role of the anti-key set is very crucial and leads to an algorithm to determine the minimum key set, creating a premise for building an algorithm to find all reducts in the decision table.

Assume that $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$ is a consistent incomplete decision table. Set $R = C \cup \{d\}$, $r = U = \{u_1, u_2, \ldots, u_n\}$ and $PRD(C)$ is called an SPS. Obviously, $PRD(C) = \mathcal{K}_d^t = \left\{Z \subseteq C : Z \xrightarrow{t} \{d\} \vee \nexists B : B \xrightarrow{t} \{d\} \vee B \subsetneq Z\right\}$.

From there, we have the following two important steps.

(i) Compute the equivalence sets $\epsilon_r = \{E_{\text{ij}} \,|1 \le i, j \le m \vee i \le j\}$ from r with $E_{ij} = \{b \in R : b(u_i) = b(u_j) \text{ or } b(u_i) = * \text{ or } b(u_j) = *\}$.

(ii) Set $N_d = \{Z \in \epsilon_r : Z \neq R,\ d \notin Z, \text{ and } \nexists B \in \varepsilon_r : d \notin B \text{ and } Z \subsetneq B\}$ from ϵ_r.

3. SOME ALGORITHMS BASED ON THE RELATIONAL DATABASE

3.1. Algorithm determining the anti-keys set

In this section, we will propose an algorithm for determining the set of anti-keys. The main steps of the algorithm are designed as below.

Algorithm 1 [17] Determining the anti-keys set

Input: Given a SPS $\mathcal{K} = \{Z_1, ..., Z_m\}$ on $R = \{\text{b}_1,...,\text{b}_n\}$.
Output: $\mathcal{K}^{-1}$.

1: **Step** 1: Set $\mathcal{K}_1 = \{R - \{b\} : b \in Z_1\}$. It can be seen that $\mathcal{K}_1 = \{Z_1\}^{-1}$.
2: **Step** p+1 **(**p<m**):** Suppose that $\mathcal{K}_p = W_p \cup \{Y_1, ..., Y_{tp}\}$, in which $Y_1, ..., Y_{tp}$ are elements of $\mathcal{K}_p$ comprising Z_{p+1} and $W_p = \{B \in \mathcal{K}_p : Z_{p+1} \nsubseteq B\}$.
3: For any i ($i = 1,\ldots,\text{t}_p$).
4: Calculate $\{Z_{p+1}\}^{-1}$ on Y_i as $\mathcal{K}_1$, which are the maximal subsets of Y_i not comprising Z_{p+1} , they are presented by $B_1^i, ..., B_{ri}^i$.
5: Set $\mathcal{K}_{p+1} = \text{W}_p \cup \left\{B_p^i \,|B \in \text{W}_p \Rightarrow B_p^i \textit{with}\, 1 \le p \le r_i \vee 1 \le i \le t_p\right\}$.
6: Finally, let $\mathcal{K}^{-1} = \mathcal{K}_m$.

We now examine the computational complexity of the proposed algorithm. Suppose that T_p with $1 \le p \le m-1$ is the number of elements in $\mathcal{K}_p$ from the above algorithm. Based on [14], the computational complexity of the algorithm is $O\left(|R|^2 \sum_{p=1}^{m-1} t_p u_p\right)$ with $u_p = T_p - t_p$ if $T_p > t_p$ and $u_p = 1$ if $T_p = t_p$. It is easy to see two following problems.

(i) $\mathcal{K}_p$ is the Sperner coefficient on R in each step of the method. According to [18], the cardinality of any SPSs on R does not exceed $C_n^{[n/2]} \approx 2^{n+1/2}/\left(\prod n^{1/2}\right)$. Therefore,

the worst computational complexity of the algorithm is an exponential function over n.

(ii) In case $T_p \leq T_m$ $(p = 1, \ldots, m-1)$, the computational complexity of the algorithm is not greater than $O\left(|\mathcal{K}|\,|R|^2\left|\mathcal{K}^{-1}\right|^2\right)$, then the algorithm complexity is a polynomial function according to $|R|$, $|\mathcal{K}|$, and $\left|\mathcal{K}^{-1}\right|$. If the number of elements of $\mathcal{K}$ is small, then the algorithm is very efficient when only requiring polynomial time following $|R|$.

3.2. Algorithm determining the minimum key set based on the anti-keys set

Based on the method proposed in Subsection 3.1, in this Section, we continue to design two algorithms as the basis for building an attribute reduction algorithm in the next section.

Algorithm 2 [18] Determine the minimum key set based on the anti-keys set

Input: Given an SPS $\mathcal{K}$ having the role of an anti-key set, $I = \{y_1, \ldots, y_n\} \subseteq R$ and $\mathcal{G}$ is an SPS in the role of key set $\left(\mathcal{G}^{-1} = \mathcal{K}\right)$ for $\exists Z \in \mathcal{K} : Z \subsetneq I$.
Output: $V \in \mathcal{G}$.

1: **Step** 1: Set $\mathfrak{c}(0) = I$.
2: **Step** $i+1$: Set $\mathfrak{c}(i+1) = \mathfrak{c}(i) - y_{i+1}$ if $\forall Z \in \mathcal{K}$ without $\mathfrak{c}(i+1) \subset Z$; On the contrary, $\mathfrak{c}(i+1) = \mathfrak{c}(i)$.
3: Finally, set $V = \mathfrak{c}(n)$.

It is noticeable that the computational complexity of the above method is polynomial according to n and $|\mathcal{K}|$.

Algorithm 3 [18] Determine the minimum key sets based on the anti-keys set

Input: Let $\mathcal{K} = \{Z_1, Z_2, \ldots, Z_k\}$ be the SPS on R.
Output: $\mathcal{G}$ with $\mathcal{G}^{-1} = \mathcal{K}$.

1: **Step** 1: Based on the Algorithm 2, calculate A_1 and set $\mathcal{K}_1 = A_1$.
2: **Step** $i+1$: If there is $Z \in \mathcal{K}_i^{-1}$ which satisfies $Z \not\subset Z_j$ $(\forall j : 1 \leq j \leq k)$, then calculate A_{i+1} $(A_{i+1} \in \mathcal{G}, A_{i+1} \subseteq Z)$ using the Algorithm 2 and set $\mathcal{K}_{i+1} = \mathcal{K}_i \cup A_{i+1}$. In the opposite case, set $\mathcal{G} = \mathcal{K}_i$.

We continue to evaluate the computational time of the Algorithm 3. Based on [18], the computational time of the Algorithm 3 is $O\left(|R|\left(\sum_{p=1}^{k-1}\left(|\mathcal{K}|\,T_p + |R|\,t_p u_p\right) + |\mathcal{K}|^2 + |R|\right)\right)$. The worst computational time of Algorithm 3 is an exponential function following $|R|$. In the case $T_p \leq |\mathcal{K}|$ $(p = 1, \ldots, k-1)$, the computational complexity of the Algorithm 3 is $O\left(|R|^2|\mathcal{K}|^2\,|\mathcal{G}|\right)$, this is the computational complexity of the polynomial function of $|R|$, $|\mathcal{K}|$, and $|\mathcal{G}|$. If $|\mathcal{G}|$ is a polynomial according to $|R|$ and $|\mathcal{K}|$, then the algorithm is efficient. If the number of elements in $\mathcal{G}$ is small, then the algorithm is very efficient.

4. ALGORITHM FOR FINDING THE ENTIRE REDUCT IN AN INCOMPLETE DECISION TABLE

Theorem 4.1. $N_d = (\mathcal{K}_d^t)^{-1}$.

Proof:

It can be easily seen that $X = X^+ \ \forall X \in N_d$ because if $X \subsetneq X^+$ then there is $x \in X^+$ and $x \notin X$. Since X is the equivalence maximum set, so $\exists i, j \ (1 \leq i < j \leq m)$ for $E_{ij} = X$, and based on the concept of the set X^+, we have $X \xrightarrow{t} \{x\}$. Besides, to be suitable for the definition of set E_{ij}, then $x \in E_{ij}$. Thus, if $X = X^+$ and $d \notin X$ then $d \notin X^+$. Therefore, $X \overset{t}{\nrightarrow} \{d\}$ (X does not TD d).

We consider P with $X \subsetneq P$, from the definition of set X if $d \notin P$ then $\forall i, j \ (1 \leq i < j \leq m)$ we have $l_i \sim l_j \, (P)$, which is false. Hence, from the concept of TD, we have $P \xrightarrow{t} R$. In the case $d \in P$, then we can see that $d \in P^+$. Thus, in two cases, we have $\forall P: \ X \subsetneq P \Rightarrow \ P^+ \xrightarrow{t} \{d\}$. Consequently, based on the definition of $\mathcal{K}_d^t$ then $I \in \mathcal{K}_d^t$, so $I \subseteq P$ and from the definition of set $(\mathcal{K}_d^t)^{-1}$, we have $X \in (\mathcal{K}_d^t)^{-1}$.

On the contrary, if $X \in (\mathcal{K}_d^t)^{-1}$ then $X^+ = X$. Since if $X \subsetneq X^+$ then from the concept of anti-key set, we have $I \in (\mathcal{K}_d^t)$ with $I \subseteq X^+$, means $X^+ \xrightarrow{t} \{d\}$, leading to $X \xrightarrow{t} \{d\}$. Based on the definition of $(\mathcal{K}_d^t)^{-1}$ then X is not TD $\{d\}$ ($X \overset{t}{\nrightarrow} \{d\}$). Hence, $X^+ = X$.

According to the definition of the sets N_d and $(\mathcal{K}_d^t)^{-1}$ (is the set of biggest sets do not TD d), this implies that $X \in N_d$. Thus, $N_d = (\mathcal{K}_d^t)^{-1}$. ∎

Algorithm 4 The algorithm for finding entire reducts in an incomplete decision table.

Input: Given a consistent incomplete decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, set $r = U = \{u_1, \ \ldots, \ u_m\}$, $R = C \cup \{d\}$.

Output: $PRD(C)$.

1: From r, calculate the equivalence sets: $\varepsilon_r = \{E_{ij} : 1 \leq \text{i} \leq \text{j} \leq m\}$ with $E_{ij} = \{a \in R$: $a(u_i) = a(u_j)$ or $a(u_i) = \ *$ or $a(u_j) = \ *\}$.

2: Based on the ε_R, set $N_d = \{X \in \varepsilon_r : X \neq R, \ d \notin X$ and $\nexists Z \in \varepsilon_r : d \notin Z$ and $X \subsetneq Z\}$.

3: Calculate the set $\mathcal{K}$ from $N_d \left(\mathcal{K}^{-1} = N_d\right)$ by the Algorithm 3.

4: Set $PRD(C)) = \mathcal{K} \backslash \{d\}$.

The computational complexity of the proposed algorithm in steps 1 and 2 is a polynomial function allowing the size of r. Therefore, the computational time of the Algorithm 4 is the same as Algorithm 2 when calculating the minimum key set from the anti-key set in step 3. Therefore, the complexity of the algorithm is

$$O\left(|R|\left(\sum_{p=1}^{m-1}\left(|N_d|\, T_p + |R|\, T_p u_p\right) + |N_d|^2 + |R|\right)\right),$$

where T_p, t_p, u_p are denoted as in Algorithm 1, and the computational complexity of this algorithm in the worst case is an exponential function over n, where n is the number of elements in R. In the case $T_p \leq |N_d| \, (p = 1, \ldots, m-1)$, the computational time of the algorithm is $O\left(|R|^2 |N_d|^2 \left|\mathcal{K}_d^t\right|\right)$, this computational time is a polynomial function according

to $|R|$, $|N_d|$, and $|\mathcal{K}|$. Evident in step 2, $|N_d|$ is a polynomial function of the size of r, so if $\left|\mathcal{K}_d^t\right|$ is a polynomial function according to $|R|$, then the computational complexity of the algorithm is a polynomial function based on the size of r. If the number of elements of $\left|\mathcal{K}_d^t\right|$ is small, the algorithm is very efficient.

We will illustrate a specific example for finding all reducts on an incomplete decision table to understand the algorithm more clearly.

Example 4.1. A decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$ where $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ and $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$.

Table 1: An example decision table

U	c_1	c_2	c_3	c_4	c_5	c_6	d
u_1	1	1	3	2	5	4	2
u_2	3	2	*	1	4	3	2
u_3	*	1	4	3	3	1	1
u_4	2	*	2	*	1	2	3
u_5	1	3	*	*	2	*	1
u_6	*	4	1	2	*	4	3

1) Calculate N_d:
$E_{12} = \{c_3, d\}$, $E_{13} = \{c_1, c_2\}$, $E_{14} = \{c_2, c_4\}$, $E_{15} = \{c_1, c_3, c_4, c_6\}$, $E_{16} = \{c_1, c_4, c_5, c_6\}$.
$E_{23} = \{c_1, c_3\}$, $E_{24} = \{c_2, c_3, c_4\}$, $E_{25} = \{c_3, c_4, c_6\}$, $E_{26} = \{c_1, c_3, c_5\}$.
$E_{34} = \{c_1, c_2, c_4\}$, $E_{35} = \{c_1, c_3, c_4, c_6, d\}$, $E_{36} = \{c_1, c_5\}$.
$E_{45} = \{c_2, c_3, c_4, c_6\}$, $E_{46} = \{c_1, c_2, c_4, c_5, d\}$.
$E_{56} = \{c_1, c_3, c_4, c_5, c_6\}$.
2) According to the condition of Algorithm 4, $A_1 = \{c_2, c_3, c_4, c_6\}$ and $A_2 = \{c_1, c_3, c_4, c_5, c_6\}$ satisfy the condition of N_d. Thus $N_d = \{\{c_2, c_3, c_4, c_6\}, \{c_1, c_3, c_4, c_5, c_6\}\}$.
3) It can be easily seen that based on Algorithm 3 from N_d, we have

$$PRD(C) = \{\{c_2, c_1\}, \{c_2, c_5\}\}.$$

Let $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$ be an incomplete decision table. Attribute $a \in C$ is called a core $\mathcal{S}$ if a participates in all reductions of $\mathcal{S}$. The notation $CORE(\mathcal{S})$ is the set of all core attributes of an $\mathcal{S}$. It can be seen that $CORE(\mathcal{S})$ is the intersection of the reducts of $\mathcal{S}$. From Algorithm 4, we have two following corollary.

Corollary 4.1. *Given an incomplete decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, then exists a method to find $CORE(\mathcal{S})$.*

As mentioned above, a core set of a decision table includes the attributes that belong to all reducts. Therefore, we can obtain a core set using the intersection of all reducts from Algorithm 4. From there, we have Corollary 1 presented as above. From Theorem 4.1 and Algorithm 2, we have Corollary 2 as below.

Corollary 4.2. *Given an incomplete decision table $\mathcal{S} = (U, C \cup \{d\}, \mathcal{F}, \mathcal{P})$, then exists a method to find a reduct of $\mathcal{S}$. This algorithm has polynomial computational complexity.*

We know that finding a reduct on the decision table is equivalent to determining a minimal key, as in Algorithm 2. On the other hand, before processing Algorithm 2, we need

to determine the anti-key set based on Algorithm 1. The computational complexity of both algorithms is a polynomial function. Therefore, determining a reduct on the decision table is also polynomial.

5. CONCLUSIONS

When dealing with big data, attribute reduction methods are crucial in knowledge mining and finding the necessary information. These methods typically follow rough set theory, and algorithms have proven to be highly effective in removing unnecessary attributes to enhance the quality of classification models. However, these methods may have limitations, mainly when dealing with tables with missing data. This paper identified specific properties of conditional attributes as well as developed a method to determine all reducts from the consistent incomplete decision tables in a polynomial time. This method serves as an effective tool in identifying essential attribute subsets while preserving information in decision tables. We plan to study more reduct properties to develop even more efficient attribute reduction models.

ACKNOWLEDGMENT

We thank the Simulation and High-Performance Computing Department (SHPC) of the HaUI Institute of Technology (HIT) for generously supporting our research in this paper.

REFERENCES

[1] Z. Pawlak, *Rough Sets - Theoretical Aspects of Reasoning about Data.* Dordrecht: Kluwer Academic Publishers, 1991.

[2] M. Kryszkiewicz, "Rough set approach to incomplete information systems," *Information Science*, vol. 112, pp. 39–40, 1998.

[3] J. Demetrovics, N. L. Giang, and V. D. Thi, "An efficient algorithm for determining the set of all reductive attributes in incomplete decision tables," *Journal of Cybernetics and Information Technologies, Bulgarian Academy of Sciences*, vol. 13, no. 4, pp. 118–126, 2013.

[4] V. D. Thi and N. L. Giang, "A method for extracting knowledge from decision tables in terms of functional dependencies," *Journal of Cybernetics and Information Technologies, Bulgarian Academy of Sciences*, vol. 13, no. 1, pp. 73–82, 2013.

[5] ——, "A method to construct decision table from relation scheme," *Journal of Cybernetics and Information Technologies, Bulgarian Academy of Sciences*, vol. 11, no. 3, pp. 32–41, 2011.

[6] N. L. Giang and V. D. Thi, "Some problems concerning condition attributes and reducts in decision tables," in *Proceedings of the 5th National Symposium Fundamental and Applied Information Technology Research (FAIR)*, 2011, pp. 142–152.

[7] J. Demetrovics, V. D. Thi, H. M. Quang, and N. V. Anh, "An efficient method to reduce the size of consistent decision tables," *Acta Cybernetica*, vol. 23, no. 4, pp. 167–180, 2018.

[8] J. Demetrovics, N. L. Giang, and V. D. Thi, "On finding all reducts of consistent decision tables," *Journal of Cybernetics and Information Technologies, Bulgarian Academy of Sciences*, vol. 14, no. 4, pp. 3–10, 2014.

[9] N. L. Giang, J. Demetrovics, V. D. Thi, and P. D. Khoa, "Some properties related to reduct of consistent decision systems," *Journal of Cybernetics and Information Technologies, Bulgarian Academy of Sciences*, vol. 21, no. 2, pp. 3–9, 2021.

[10] J. Demetrovics, "On the equivalence of candidate keys with sperner systems," *Acta Cybernetica*, vol. 4, no. 3, pp. 247–252, 1979. [Online]. Available: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3183

[11] J. Demetrovics and V. D. Thi, "Algorithms for generating armstrong relation and inferring functional dependencies in the relational data model," *Computer and Mathematics with Applications*, vol. 26, no. 4, pp. 43–55, 1993.

[12] D. János and V. Thi, "Keys, key, antikeys and prime attributes," *Ann. Univ. Scien. Budapest Sect. Comut*, vol. 8, pp. 37–54, 1987.

[13] D. János and V. D. Thi, "Relations and minimal keys," *Acta Cybernetica*, vol. 8, no. 3, pp. 297–285, 1998. [Online]. Available: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3342

[14] J. Demetrovics and V. D. Thi, "Some result about functional dependencies," *Acta Cybernetica*, vol. 8, no. 3, pp. 273–280, 1988. [Online]. Available: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3341

[15] N. L. Giang and V. D. Thi, "Algorithm for finding all attribute reduction of a decision," *Journal of Computer Science and Cybernetics*, vol. 27, no. 3, pp. 199–205, 2011.

[16] N. L. Giang, N. T. Tung, and V. D. Thi, "A new method for attribute reduction to incomplete decision table based on metric," *Journal of Computer Science and Cybernetics*, vol. 28, no. 2, pp. 129–140, 2012.

[17] V. D. Thi, "Minimal keys and antikeys," *Acta Cybernetica*, vol. 7, no. 4, pp. 361–371, 1986. [Online]. Available: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3305

[18] D. János and V. D. Thi, "Some remarks on generating, armstrong and inferring functional dependencies relation," *Acta Cybernetica*, vol. 12, no. 2, pp. 167–180, 1995. [Online]. Available: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3455

Received on August 21, 2023
Accepted on November 01, 2023