Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5850410, 16 pages
https://doi.org/10.1155/2019/5850410

Research Article

Rule Extraction Model Based on Decision Dependency Degree

Xinying Chen,[1,2] **Guanyu Li**,[1] **and Yunhao Sun**[1]

[1]*College of Information Science and Technology, Dalian Maritime University, Dalian, Liaoning 116026, China*
[2]*School of Software Technology, Dalian Jiao Tong University, Dalian, Liaoning 116028, China*

Correspondence should be addressed to Guanyu Li; rabitlee@163.com

Received 1 April 2019; Revised 16 August 2019; Accepted 6 November 2019; Published 29 November 2019

Academic Editor: Ali Ramazani

Rule extraction is the core in rough set. Two procedures are contained in rule extraction: one is attribute reduction and another is attribute value reduction. It was proved through computational complexity perspective that obtaining all the reduction, minimum attribute reduction, and minimum attribute value reduction is an NP problem. So, generally, a heuristic reduction method is used to solve attribute reduction and attribute value reduction. However, for most heuristic methods, it is hard to put into practice and has high cost on computational complexity. Moreover, part of the methods extracted redundant rules. To approach a quick and effective model for rule extraction in decision systems, against the concept of distinguishable relation, relevant concepts and basic theorems of rule extraction are proposed. In order to get concise and accurate rules quickly, algorithms for finding conflict object set, finding duplicate object set, and finding redundant rules are given. After that, using decision dependency degree as attribute importance to determine the importance of each attribute in rule object, a new rule extraction model based on decision dependency degree is proposed in this paper. Compared with the previous models, this model does not generate matrix; instead, it finds conflict object set and duplicate object set by equivalence class, and consequently, improves the time performance to $\max\{O(|C||U|), O(|C|^2|U/C|)$, and $O(|\mathrm{RED}||(U/C)/\mathrm{RED}|^2)\}$. The theoretical analysis and experimental research show that the new model more accurately and effectively reduces the redundant data and extracts more concise decision rules from dataset.

1. Introduction

Today, in this ubiquitous and intelligent interconnection environment, after integration, cleansing, and transformation, the data information in the dataset (denoted by information system in this paper) from all kinds of perceived data and context data is not equally important [1]. Some are even redundant, which will interfere with policymakers and will seriously affect the efficiency on the followup operations. Hereby, it is necessary to perform attribute reduction and attribute value reduction or similar simplifying process on the dataset to reduce redundancy and extract decision rules before executing the followup operations.

Rough set is a data induction tool, which can effectively process massive data [2, 3]. One of the main advantages of Rough Set is that it does not need any preliminary or additional information about data, such as dataset probability distribution or grade membership. Without any information other than the dataset itself, Rough Set can keep original classification information unchanged and start with the description set of a problem and then get the essential features and inherent law of the problem by deleting redundant information.

Rule extraction is the core in Rough Set. By rule extraction, clarity of the knowledge system can be improved and decision knowledge underneath dataset can be found out. Two procedures are contained in rule extraction: one is attribute reduction [4–13] and another is attribute value reduction [14–28].

By conducting the attribute reduction on the decision system, redundant information can be removed from the system. In the premise of ensuring decision system has the same distinguishability, attribute reduction can simplify and improve clarity of the original system to a certain extent. However, after attribute reduction on the whole system, the system is not the most simplified system. In the system, each rule object may contain redundant information; that is, all condition attribute values of each rule object are not equally

important and are not necessary to impact decision rules extraction execution. Therefore, to improve the quality of data reduction, it is mandatory to evaluate all condition attribute values for each rule object and remove redundant condition attribute values which have no impact on performing rule extraction. This process is named as attribute value reduction in decision systems. Using attribute reduction and attribute value reduction, small and concise decision rules can be extracted from the original system. After that, data from the system can be simply presented as the valuable and straightforward information, which can be easily used.

It was proved through computational complexity perspective that to get all the reduction, minimum attribute reduction, and minimum attribute value reduction is an NP problem by Wong and Ziarko [3]. Because of the computational complexity of getting core attributes and value-core attributes during the process of attribute reduction and attribute value reduction, moreover, with big volume of candidates for reduction, the NP problem indicates that it is easy to get combination explosion. So, generally, a heuristic reduction method is used to solve attribute reduction and attribute value reduction.

Today, value reduction of decision systems is one of important topics in research and application of Rough Set. Researchers have done a lot of research work and gradually put forward some methods, primarily including value reduction algorithms based on discernibility matrix [14–22] and heuristic value reduction algorithms [23–28]. However, when using discernibility matrix to get value reduction on decision systems with massive data, a huge matrix is required, which leads to high cost on time-space complexity during constructing and traversing the matrix. Therefore, it needs furthermore research on discernibility matrix methods. For most heuristic methods, it is hard to put into practice and has high cost on computational complexity. Moreover, part of the methods extracted redundant rules, for example, methods in [23].

To resolve the above problems and reduce the time-space complexity, with learning from tradition reduction algorithms, we propose a new heuristic value reduction and rule extraction model. The model will use an improved chain base sorting method to perform sorting and equivalence class partitioning on the system and then will compare rule objects. Moreover, it will use conditional equivalence class to determine conflict objects and duplicate objects. So, the time cost generated by comparison of rule objects is significantly decreased. At the same time, the model uses theory of decisive distinguishable matrix (but never create the interim result) to determine the importance of attributes, which effectively control the complexity. This model will perform attribute reduction before performing value reduction. After the reduction, the count of condition attributes in reduction will not be very big. Compared with the current known methods, even with massive data, this model can significantly reduce time-space complexity and improve efficiency.

Term distinguishable object set is named under distinguishable relationship. Against this concept, theorems on distinguishable object set and consistency in decision system are given. With the theorems, formalized concept on attribute value reduction, theorem on value-core attribute of object, and formalized concepts on conflict object set and duplicate object set during value reduction are defined. After that, algorithms for conflict object set, duplicate object set, and redundant rules are proposed, respectively. With dependency based on distinguishable relation, an attribute value reduction and rule extraction model are given for decision systems. At last, the correctness and feasibility of the model are proved by specific examples and experiments. The model can provide effective solution on data value reduction and decision rule extraction for all kinds of intelligent networking systems and intelligent control systems.

2. Attribute Reduction of Decision Systems

Definition 1 (decision system). Let $S = (U, A)$ be an information system. $U = \{x_1, x_2, \ldots, x_n\}$ is a nonempty finite object set, called universe. $A = \{a_1, a_2, \ldots, a_m\}$ is a nonempty finite attribute set, called attribute set. If A consists of a condition attribute set C and a decision attribute set D and two sets satisfy $C \cup D = A$ and $C \cap D = \varnothing$, then S is a decision system which is denoted by $(U, C \cup D)$. When there is only one decision attribute in D, the decision system is denoted by $(U, C \cup \{d\})$ commonly.

In general, the basic structure of decision system $S = (U, C \cup \{d\})$ is shown in Table 1.

Definition 2 (equivalence class). Let $S = (U, C \cup D)$ be a decision system. For any $P \subseteq (C \cup D)$, $P \neq \varnothing$, define the indiscernibility relation as follows:

$$\operatorname{ind}(P) = \left\{\left(x_i, x_j\right) \mid \left(x_i, x_j\right) \in U \times U, \forall a_k \in P, f_k(x_i) = f_k\left(x_j\right)\right\}. \tag{1}$$

And the equivalence class of object $x_i\,(i \leq n)$ according to attribute set P is defined as follows:

$$[x_i]_{\operatorname{ind}(P)} = \left\{x_j \mid \left(x_i, x_j\right) \in \operatorname{ind}(P)\right\}. \tag{2}$$

The indiscernibility relation $\operatorname{ind}(P)$ corresponds to a partition of U, and the partition is denoted by $U/\operatorname{ind}(P)$. In general, U/P is used as the abbreviation.

If $(x_i, x_j) \in \operatorname{ind}(P)$, then x_i and x_j are indiscernible by P. $[x_i]_{\operatorname{ind}(P)}$ contains all the objects that are indiscernible with x_i by attribute set P. That is, an equivalence class is a set of objects with the same condition attributes and the same decision attributes in universe.

Definition 3 (attribute reduction of decision system). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. If $R_P \subseteq R_C$ (that is, $\operatorname{ind}_P \subseteq \operatorname{ind}_C$), then P is a consistent set of S [28]. Furthermore, if P is a consistent set and any $Q \subset P$ is not a consistent set of S, then P is called as an attribute reduction of S [28].

The attribute reduction of a decision system is a reduction for its condition attributes. The attribute reduction is the smallest subset of the condition attribute set found on the premise of maintaining the original system classification ability.

Table 1: Basic structure of decision system.

U	Condition attributes							Decision attribute
	a_1	a_2	a_3	...	a_j	...	a_m	d
x_1	0	1	1	...	1	...	3	1
x_2	1	1	0	...	1	...	2	1
...	...	...	...	...	...	...	...	...
x_i	3	1	0	...	1	...	4	1
...	...	...	...	...	...	...	...	...
x_n	0	1	3	...	1	...	1	0

3. Distinguishable Relation and Consistent Reduction with Distinguishability

In order to facilitate subsequent discussion and research, related concepts and theorems are given as follows.

Among them, Definition 4 "Distinguishable Relation of Attribute Set" is discussed from the perspective of distinguishability between objects, while Definition 2 "Indiscernibility Relation and Equivalence Class" in the previous section is discussed from the perspective of indiscernibility between objects.

And, definitions "Consistent Reduction with Distinguishability" and "Decisive Distinguishable Matrix" are based on Definition 4, while Definition 3 "Attribute Reduction of Decision System" in the previous section is based on Definition 2.

Based on the distinguishable relation of attribute set, the traditional logical knowledge granulation process related to Definition 3 can be transformed into matrix computing and the rapid reduction of redundant attributes can be realized.

Definition 4 (distinguishable relation of attribute set). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. Define the distinguishable relation of P as follows:

$$\mathrm{DIS}_P = \left\{\left(x_i, x_j\right) \mid \left(x_i, x_j\right) \in U \times U, \exists a_k \in P, f_k(x_i) \neq f_k\left(x_j\right)\right\}. \tag{3}$$

DIS_P is used as the distinguishable relation of P.

If $(x_i, x_j) \in \mathrm{DIS}_P$, then x_i and x_j are distinguishable by P. That is, there is a condition attribute, and the attribute values of the two objects are different.

Definition 5 (distinguishable unit set of attribute set). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. If $(x_i, x_j) \in \mathrm{DIS}_P$, then (x_i, x_j) is called a distinguishable unit. The distinguishable unit set of P is denoted by $\mathrm{DIS}_P(U)$.

$\mathrm{DIS}_P(U)$ contains all the units that can be distinguished by P.

Definition 6 (distinguishable unit set of attribute). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. For any attribute $a_k \in C$, let

$$\mathrm{DIS}_{\{a_k\}} = \left\{\left(x_i, x_j\right) \mid \left(x_i, x_j\right) \in U \times U, f_k(x_i) \neq f_k\left(x_j\right)\right\}. \tag{4}$$

$\mathrm{DIS}_{\{a_k\}}$ is used as the distinguishable relation of a_k. The distinguishable unit set of a_k is denoted by $\mathrm{DIS}_{a_k}(U)$, which contains all the units that can be distinguished by a_k.

Based on Definition 4, Definition 5, and Definition 6, the definition of 7 "Consistent Reduction with Distinguishability" is given as follows.

Definition 7 (consistent reduction with distinguishability). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. If $\mathrm{DIS}_P \cap \mathrm{DIS}_D = \mathrm{DIS}_C \cap \mathrm{DIS}_D$, then P is a consistent attribute set with distinguishability of S [28]. For any $Q \subset P$ and $Q \neq \varnothing$, if $\mathrm{DIS}_P \cap \mathrm{DIS}_D = \mathrm{DIS}_C \cap \mathrm{DIS}_D$ and $\mathrm{DIS}_Q \cap \mathrm{DIS}_D \neq \mathrm{DIS}_C \cap \mathrm{DIS}_D$, then P is a consistent reduction with distinguishability of S [28].

Under the premise of keeping the system classification ability unchanged, a consistent reduction with distinguishability can be achieved after deleting the irrelevant or unimportant knowledge.

Definition 8 (decisive distinguishable matrix). Let $S = (U, C \cup D)$, $|C| = m$, and $|U| = n$. For any attribute $a_k \in C$, let

$$\overline{m_k}^* = \left\{\left(x_i, x_j\right) \mid \left(x_i, x_j\right) \in \mathrm{DIS}_D, \left(x_i, x_j\right) \in \mathrm{DIS}_{\{a_k\}}\right\}. \tag{5}$$

$\overline{m_k}^*$ is used as the kth column vector of the decisive distinguishable matrix $\overline{M_k}^*$, which contains n^2 rows and m columns.

$\overline{m_k}^*$ is the intersection of DIS_D and $\mathrm{DIS}_{\{a_k\}}$. Given a decision system $S = (U, P \cup D)$, $P \subseteq C$ and $P \neq \varnothing$, $\overline{M_p}^*$ is used as the decisive distinguishable matrix of S.

Definition 9 (core with distinguishability). For a decision system $S = (U, C \cup D)$, which has r reductions, $P_k (k = 1, 2, \ldots, r)$, a core set with distinguishability of S is defined as follows:

$$\mathrm{CORE}^* = \bigcap_{1 \leq k \leq r} P_k. \tag{6}$$

In the process of attribute reduction, the core cannot be deleted from the attribute set. Because the distinguishability of the core cannot be replaced by other attributes, the core is indispensable in expressing the distinguishability of the system. The distinguishability of other noncore attributes can be replaced by other attributes.

Theorem 1. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent:*

(1) S is a consistent decision system

(2) $DIS_C \supseteq DIS_D$

(3) $\mathrm{ind}_C \subseteq \mathrm{ind}_D$

Proof. (2) $\Longleftrightarrow$ (3). Evidently, $\mathrm{DIS}_C \supseteq \mathrm{DIS}_D$ is equivalent to $U \times U - \mathrm{DIS}_P \supseteq U \times U - \mathrm{DIS}_C$. That is, for any $(x_i, x_j) \in U \times U$, if $(x_i, x_j) \in U \times U - \mathrm{DIS}_C$, then $(x_i, x_j) \in U \times U - \mathrm{DIS}_D$.

According to Definition 5 and Definition 6, if $(x_i, x_j) \in U \times U - \mathrm{DIS}_C$, then x_i and x_j are indistinguishable by C. That is, $[x_i]_{\mathrm{ind}(C)} = [x_j]_{\mathrm{ind}(C)}$. Similarly, if $(x_i, x_j) \in U \times$

$U - \mathrm{DIS}_D$, then x_i and x_j are indistinguishable by D. That is, $[x_i]_{\mathrm{ind}(D)} = [x_j]_{\mathrm{ind}(D)}$.

Thus, $U \times U - \mathrm{DIS}_P \supseteq U \times U - \mathrm{DIS}_C$ is equivalent to any $(x_i, x_j) \in U \times U$, if $[x_i]_{\mathrm{ind}(C)} = [x_j]_{\mathrm{ind}(C)}$, then $[x_i]_{\mathrm{ind}(D)} = [x_j]_{\mathrm{ind}(D)}$. That is, $\mathrm{ind}_C \subseteq \mathrm{ind}_D$. Therefore, (2) $\Longleftrightarrow$ (3).

Similarly, it can be proved that (1) $\Longleftrightarrow$ (2). □

Theorem 2. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. DIS_P has the following properties.*

(1) If $P_1 \subseteq P_2 \subseteq C$, then $\mathrm{DIS}_{P_1} \subseteq \mathrm{DIS}_{P_2} D \subseteq \mathrm{DIS}_C$

(2) $\mathrm{DIS}_P = \cup_{a_k \in P} \mathrm{DIS}_{a_k}$

Proof. The proof of Theorem 2 is similar to that of Theorem 2 in [27]. □

Theorem 3. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$, then $|M_P^*| = |DIS_P \cap DIS_D|$.*

Proof. According to Definition 8, for any $a_k \in P$, $\overline{m_k^*} = \mathrm{DIS}_{a_k} \cap \mathrm{DIS}_D$, and $|M_P^*| = |\cup \overline{m_k}^*|$. Thus, according to property 2 in Theorem 2, $|M_P^*| = |\mathrm{DIS}_P \cap \mathrm{DIS}_D|$. □

Theorem 4. *Let $S = (U, C \cup D)$ be a decision system; then, consistent reduction with distinguishability must exist.*

Theorem 5. *Let $S = (U, C \cup D)$ be a decision system. In decision system S, consistent set and consistent attribute set with distinguishability are equivalent.*

Theorem 6. *Let $S = (U, C \cup D)$ be a decision system. In decision system S, attribute reduction and consistent reduction with distinguishability are equivalent.*

Theorem 7. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent.*

(1) P is a consistent attribute set with distinguishability

(2) $\mathrm{DIS}_P \cap \mathrm{DIS}_D \supseteq \mathrm{DIS}_C \cap \mathrm{DIS}_D$

(3) $P \equiv C$

Theorem 8. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent.*

(1) P is a consistent attribute set with distinguishability

(2) $\mathrm{DIS}_P \cap \mathrm{DIS}_D \supseteq \mathrm{DIS}_C \cap \mathrm{DIS}_D$, and $\forall Q \subset P, \mathrm{DIS}_Q \cap \mathrm{DIS}_D \neq \mathrm{DIS}_C \cap \mathrm{DIS}_D$

(3) $P \equiv C$, and $\forall Q \subset P$, $Q \equiv C$ is not true

The proof of above theorems is similar to the proof of Theorems 11 to 15 in [29] (not elaborated here).

4. Decision Dependency Degree Based on Distinguishable Relation

Based on the distinguishable relation, Definition 10 and Definition 11 are proposed as follows. Definition 10 "Dependency Based on Distinguishable Relation" is used to determine whether there is a dependency relationship between the attribute sets P_1 and P_2. And Definition 11 "Dependency Degree based on Distinguishable Relation" is used to assess the extent of the dependency or replacement relationship that exists between P_1 and P_2.

Definition 10 (dependency based on distinguishable relation). Let $S = (U, C \cup D)$ be a decision system, $P_1, P_2 \subseteq C$ and $P_1, P_2 \neq \varnothing$. Then, the definition "dependency based on distinguishable relation" is defined as follows:

(1) Attribute set P_2 depends on attribute set P_1 based on distinguishable relation iff $\mathrm{DIS}_{P_2} \cap \mathrm{DIS}_D \subseteq \mathrm{DIS}_{P_1} \cap \mathrm{DIS}_D$. This decision dependency can be denoted as $P_1 \Longrightarrow P_2$.

(2) Attribute set P_2 is equivalent to attribute set P_1 based on distinguishable relation iff $P_1 \Longrightarrow P_2$ and $P_2 \Longrightarrow P_1$. This equivalent relationship can be denoted as $P_1 \equiv P_2$. It is obvious that $P_1 \equiv P_2$, iff $\mathrm{DIS}_{P_2} \cap \mathrm{DIS}_D \subseteq \mathrm{DIS}_{P_1} \cap \mathrm{DIS}_D$.

Definition 11 (distinguishable relation-based dependency degree). Let $S = (U, C \cup D)$ be a decision system, $P_1, P_2 \subseteq C$ and $P_1, P_2 \neq \varnothing$; if decision dependency degree $k\,(0 \leq k \leq 1)$ is denoted as

$$k_{p_2}(P_1) = \frac{1 - \left|\mathrm{DIS}_{P_2} \cap \mathrm{DIS}_D\right|}{\left|\mathrm{DIS}_{P_1} \cap \mathrm{DIS}_D\right|}, \tag{7}$$

then the extent to which P_2 depends on attribute set P_1 based on distinguishable relation is $k\,(0 \leq k \leq 1)$, called as P_2. k depends on P_1 (P_2 depends on P_1 in k decision dependency degree), or P_1 can be substituted by P_2 in $1 - k$ decision substitution degree, which is denoted as $P_1{}^k \Longrightarrow P_2$.

(1) If $k = 0$, named P_2 does not depend on P_1 based on distinguishable relation, or P_1 can be completely replaced (substituted) by P_2

(2) If $0 < k < 1$, named P_2 partly depends on P_1 based on distinguishable relation, or P_1 can be partly replaced (substituted) by P_2

(3) If $k = 1$, named P_2 completely depends on P_1 based on distinguishable relation, or there does not exist decision substitution relationship between P_1 and P_2

Theorem 9. *Let $S = (U, C \cup D)$ be a decision system, $P_1, P_2 \subseteq C$, $P_1, P_2 \neq \varnothing$, and $P_1 \Longrightarrow P_2$; then, the following propositions are true:*

(1) Decision dependency degree $k_{P_2}(P_1)$ is 0 if $|M_{P_1}^*| = |M_{P_2}^*|$

(2) Decision dependency degree $k_{P_2}(P_1)$ is 1 if $|M_{P_2}^*| = 0$ or $\mathrm{DIS}_{P_2} \cap \mathrm{DIS}_D = \varnothing$

Proof

(1) According to Theorem 3, it is obvious that $k_{P_2}(P_1) = 1 - |M^*_{P_2}|/|M^*_{P_1}|$. Thus, $k_{P_2}(P_1)$ is 0 if $|M^*_{P_1}| = |M^*_{P_2}|$.

(2) According to Definition 11 and Theorem 3, it is obvious that $k_{P_2}(P_1)$ is 1 if $\mathrm{DIS}_{P_2} \cap \mathrm{DIS}_D = \varnothing$ or $|M^*_{P_2}| = 0$. □

Theorem 10. *Let $S = (U, C \cup D)$ be a decision system, $P_1, P_2 \subseteq C$, $P_1, P_2 \neq \varnothing$, and $P_1 \Longrightarrow P_2$; $|M^*_{P_2}|$ is inversely proportional to decision dependency degree $k_{P_2}(P_1)$ and is proportional to decision substitution degree.*

Proof. According to Definition 11 and Theorem 3, it is obvious that $|M^*_{P_2}|$ is inversely proportional to decision dependency degree $k_{P_2}(P_1)$ and is proportional to decision substitution degree.

With the above analysis, $k_{P_2}(P_1) = 1 - |M^*_{P_2}|/|M^*_{P_1}|$ can well reflect the decision dependency and decision substitution relationship between P_1 and P_2. Due to the denominator M^*_C being a fixed value and inversely proportional to $k_P(C)$, the decision dependence degree of P can be directly calculated by $|M^*_P|$. □

5. Value-Core Attribute Set of Object and Attribute Value Reduction Set

Performing reduction on the decision system, a simplified attribute set can be achieved without impacting the distinguishability of the original system. However, for each object instance in the decision system after reduction, there still exists an attribute redundancy problem. Here, obtaining minimum attribute reduction for each object instance is equivalent to value reduction.

Because the distinguishability of core attributes in the decision system cannot be substituted during attribute reduction, core attributes are objected to be removed. The others can be replaced. Similar to the existence of core attributes during attribute reduction, there are value-core attributes during attribute value reduction. The definition "value-core attribute" is defined in value reduction in Rough Set.

After performing attribute reduction and deleting duplicate object instances, there is not any duplicate object instance in the system. Each object instance corresponds to a rule. These object instances are named as "rule object" (or simply "rule" for short) in this paper. At this point, the rule object has less clustering power, which is not conducive to matching object instances. In order to improve the clustering ability, similar to attribute reduction, for each rule object, the distinguishability of each condition attribute should be evaluated. Rules can be simplified and summarized by removing redundant and non-value-core condition attributes. Thus, the ability of rules to match instances is improved.

Given a decision system $S' = (U', P \cup D)$, $U' \subseteq U$ and $P \subseteq C$. If S' is a consistent decision system, then in S', $\forall x_i \in U'$ and $\forall a_j \in P$, after removing a_j from x_i, the consistency of S' will be changed. And there exist three scenarios as follows:

(1) After deleting a_j from x_i, in the S', $\exists x_k \in U'$, those remaining condition attribute values for x_i and x_k are the same while decision attribute values are not. That is, decision conflict occurs. In such a case, a_j is a value-core attribute of x_i and should not be deleted. Hereby, a_j has to be recovered.

Similarly, if a_j is removed from x_k, in the S', $\exists x_i \in U'$, decision conflict is surely found between x_i and x_k. In such a case, a_j is also a value-core attribute of x_k and should not be deleted. Hereby, a_j has to be recovered.

This shows that in system S', $\forall x_i \in U'$, $\forall a_j \in P$, if a_j is removed, x_i has decision conflict with other objects. That is, consistency of system S' is changed after removing a_j, which makes S' to be an inconsistent system. In such a case, a_j is the value-core attribute of x_i and of other conflict objects with x_i. a_j can avoid decision conflict between x_i and other objects. So it should not be removed from those objects.

(2) After deleting a_j, in the S', $\exists x_k \in U'$, both condition attribute values and decision attribute values for x_i and x_k are the same. That is, no decision conflict is generated while duplicate objects arise. In such a case, a_j is not a value-core attribute of x_i and it will not impact the decision for new duplicate object. Hereby, a_j can be removed.

Similarly, if a_j is removed from x_k, in the S', $\exists x_i \in U'$, both condition attribute values and decision attribute values for x_i and x_k are the same; that is, x_k is the new duplicate object of x_i. In such a case, a_j is not a value-core attribute of x_k. Hereby, a_j can be removed.

This shows that in system S', $\forall x_i \in U'$, $\forall a_j \in P$, if a_j is removed, there are new objects which are duplicate objects for x_i; then, a_j is not a value-core attribute for all duplicate objects including x_i. So it can be removed.

(3) After deleting a_j, in the S', there is no decision conflict or duplicate objects. Hereby, according to existing information, it cannot be determined whether the deletion on a_j of x_i will impact decision consistency of the system or not. Additional information has to be considered on how to proceed on a_j.

Combined with above analysis, definitions "Distinguishable Set of Object", theorems of "Distinguishable Set of Object" and "System Consistency" are given. With the theorems, formalized definitions "Attribute Value Reduction," "Value-core Attribute of Object," "Conflict Object Set," and "Duplicate Object Set" during value reduction are proposed.

Definition 12 (distinguishable set of object). Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. For any $x_i \in U$, let

$$\mathrm{DIS}_P(x_i) = \left\{ \left\{ x_j \mid \left(x_i, x_j\right) \in U \times U, \left(x_i, x_j\right) \in \mathrm{DIS}_P(U) \right\}. \right. \tag{8}$$

$\mathrm{DIS}_P(x_i)$ is used as the distinguishable set of x_i. Object x_j belongs to distinguishable unit (x_i, x_j) which is generated by DIS_P in universe. Hereby, $\mathrm{DIS}_P(x_i)$ can also be defined as

$$\mathrm{DIS}_P(x_i) = \left\{ x_j \mid \left(x_i, x_j\right) \in U \times U, \exists a_k \in P, f_k(x_i) \neq f_k\left(x_j\right) \right\}. \tag{9}$$

Let $S = (U, C \cup D)$ be a decision system and $S' = (U', \mathrm{RED} \cup D)$ be a decision system after attribute reduction where RED is an attribute reduction of S. For $\forall x_i \in U'$, $\mathrm{DIS}_{\mathrm{RED}}(x_i) = |U'|\text{-}1$ is true.

It shows that after reduction, derived from attribute reduction RED, for $\forall x_i \in U'$, the distinguishable set of x_i is $\mathrm{DIS}_{\mathrm{RED}}(x_i) = U' - \{x_i\}$. That is, after reduction, for $\forall x_i \in U'$, x_i can be distinguishable with all the other objects in U' by RED.

Theorem 11. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent:*

(1) $S = (U, P \cup D)$ is a consistent decision system

(2) $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$

(3) For $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$

Proof. According to Theorem 1, it is obvious that (1) $\Longleftrightarrow$ (2). □

(2) $\Longleftrightarrow$ (3). According to Theorem 1, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\mathrm{ind}_P \subseteq \mathrm{ind}_D$, that is, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\forall x_i \in U$, $[x_i]_P \subseteq [x_i]_D$ (namely, $[x_i]_{\mathrm{ind}(P)} \subseteq [x_i]_{\mathrm{ind}(D)}$). As $[x_i]_P = U - \mathrm{DIS}_P(x_i)$ and $[x_i]_D = U - \mathrm{DIS}_D(x_i)$, $[x_i]_P \subseteq [x_i]_D$ is equivalent to $U - \mathrm{DIS}_P(x_i) \subseteq U - \mathrm{DIS}_D(x_i)$, that is equivalent to $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$. Thus, for $\forall x_i \in U$, $[x_i]_P \subseteq [x_i]_D$ is equivalent to $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$. Therefore, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$.

From the above theorems, if objects that can be distinguished with x_i by condition attributes are more or at least equal to objects that can be distinguished with x_i by decision attributes, the system is a consistent decision system. That is, only when all the distinguishable objects deduced by condition attribute set P contain those deduced by decision attribute set D, the system is a consistent decision system. In another word, if all the objects in system meet the condition $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$, the system is a consistent decision system.

Theorem 12. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent:*

(1) For $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$

(2) For $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)$

(3) For $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P \cup D}(x_i)$

Proof. (1) $\Longleftrightarrow$ (2). □

According to Theorem 1, it is obvious that $\forall x_i \in U$ and $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$ is equivalent to $S = (U, P \cup D)$, which is a consistent decision system; that is, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is true.

According to Theorem 1, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\mathrm{ind}_P \subseteq \mathrm{ind}_D$, and $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\forall x_i \in U$, $[x_i]_P \subseteq [x_i]_D$ (that is, $[x_i]_{\mathrm{ind}(P)} \subseteq [x_i]_{\mathrm{ind}(D)}$). As $[x_i]_P \subseteq [x_i]_D$ is equivalent to $[x_i]_P \cap [x_i]_P \subseteq [x_i]_D \cap [x_i]_P$, $[x_i]_P \subseteq [x_i]_D$, is equivalent to $[x_i]_P \subseteq [x_i]_{P \cup D}$. Thus, $\mathrm{DIS}_P \supseteq \mathrm{DIS}_D$ is equivalent to $\forall x_i \in U$ and $[x_i]_P \subseteq [x_i]_{P \cup D}$.

Because $[x_i]_P = U - \mathrm{DIS}_P(x_i)$ and $[x_i]_{P \cup D} = U - \mathrm{DIS}_{P \cup D}(x_i)$, $[x_i]_P \subseteq [x_i]_{P \cup D}$ is equivalent to $U - \mathrm{DIS}_P(x_i) \subseteq U - \mathrm{DIS}_{P \cup D}(x_i)$. That is, $[x_i]_P \subseteq [x_i]_{P \cup D}$ is equivalent to $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)$. Thus, $\forall x_i \in U$, $[x_i]_P \subseteq [x_i]_{P \cup D}$ is equivalent to $\forall x_i \in U$, and $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)$.

Therefore, $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$ is equivalent to $\forall x_i \in U$, and $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P \cup D}(x_i)$.

(2) $\Longleftrightarrow$ (3).

It is obvious that $P \subseteq P \cup D$. According to Theorem 2, $\mathrm{DIS}_P(x_i) \subseteq \mathrm{DIS}_{P \cup D}(x_i)$ is true. Thus, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)$ is equivalent to $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P \cup D}(x_i)$.

Theorem 13. *Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$ and $P \neq \varnothing$. The following propositions are equivalent: (1) $S = (U, P \cup D)$ is a inconsistent decision system (2) $\exists x_i \in U$, $\mathrm{DIS}_P(x_i) \subset \mathrm{DIS}_{P \cup D}(x_i)$ (3) $\exists x_i \in U$, $\mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{P \cup D}(x_i)$.*

Proof. (1) $\Longleftrightarrow$ (2).

According to Theorem 11, it is obvious that $S = (U, P \cup D)$ is a consistent decision system, which is equivalent to $\forall x_i \in U$ and $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$. Then, $S = (U, P \cup D)$ is a inconsistent decision system, which is equivalent to $\neg(\forall x_i \in U, \mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i))$.

According to Theorem 12, it is obvious that $\forall x_i \in U$, $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)$ is equivalent to $\forall x_i \in U$, and $\mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)$:

$$\begin{aligned} &\text{so } \neg\left(\forall x_i \in U, \mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i)\right), \\ &\quad \Longleftrightarrow \neg\left(\forall x_i \in U, \mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P \cup D}(x_i)\right), \\ &\quad \Longleftrightarrow \neg\left(\forall x_i \in U, U - \mathrm{DIS}_P(x_i) \subseteq U - \mathrm{DIS}_{P \cup D}(x_i)\right). \end{aligned} \tag{10}$$

Because $[x_i]_P = U - \mathrm{DIS}_P(x_i)$ and $[x_i]_{P \cup D} = U - \mathrm{DIS}_{P \cup D}(x_i)$,

$$\begin{aligned} &\neg\left(\forall x_i \in U, U - \mathrm{DIS}_P(x_i) \subseteq U - \mathrm{DIS}_{P \cup D}(x_i)\right), \\ &\quad \Longleftrightarrow \neg\left(\forall x_i \in U, [x_i]_P \subseteq [x_i]_{P \cup D}\right). \end{aligned} \tag{11}$$

Because $[x_i]_{P \cup D}$ is equivalent to $[x_i]_D \cap [x_i]_P$, according to the definition and properties of equivalence class, $[x_i]_P \supseteq [x_i]_{P \cup D}$ is true, so the reasoning process is as follows:

$$\begin{aligned} &\neg\left(\forall x_i \in U, [x_i]_P \subseteq [x_i]_{P \cup D}\right), \\ &\quad \Longleftrightarrow \exists x_i \in U, [x_i]_P \supseteq [x_i]_{P \cup D}, \\ &\quad \Longleftrightarrow \exists x_i \in U, U - \mathrm{DIS}_P(x_i) \supseteq U - \mathrm{DIS}_{P \cup D}(x_i), \\ &\quad \Longleftrightarrow \exists x_i \in U, \mathrm{DIS}_p(x_i) \subset \mathrm{DIS}_{P \cup D}(x_i). \end{aligned} \tag{12}$$

$S = (U, P \cup D)$ is a inconsistent decision system $\Longleftrightarrow \exists x_i \in U$ and $\mathrm{DIS}_p(x_i) \subset \mathrm{DIS}_{P\cup D}(x_i)$.

Thus, Proposition (1) is equivalent to Proposition (2).

(1) $\Longleftrightarrow$ (3).

Because $[x_i]_{P\cup D}$ is equivalent to $[x_i]_D \cap [x_i]_P$, according to the definition and properties of equivalence class, $[x_i]_P \supseteq [x_i]_{P\cup D}$ is true, so the reasoning process is as follows:

$$\begin{aligned}
&\neg(\forall x_i \in U, [x_i]_P \subseteq [x_i]_{P\cup D}),\\
&\Longleftrightarrow \neg(\forall x_i \in U, [x_i]_P = [x_i]_{P\cup D}),\\
&\Longleftrightarrow \exists x_i \in U, [x_i]_P \neq [x_i]_{P\cup D},\\
&\Longleftrightarrow \exists x_i \in U, U - \mathrm{DIS}_P(x_i) \neq U - \mathrm{DIS}_{P\cup D}(x_i),\\
&\Longleftrightarrow \exists x_i \in U, \mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{P\cup D}(x_i).
\end{aligned} \tag{13}$$

Alternatively, according to Theorem 1, Theorem 2, and Theorem 12, the reasoning process is as follows:

$S = (U, P \cup D)$ is a consistent decision system:

$$\begin{aligned}
&\Longleftrightarrow \forall x_i \in U, \mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_D(x_i),\\
&\Longleftrightarrow \forall x_i \in U, \mathrm{DIS}_P(x_i) \supseteq \mathrm{DIS}_{P\cup D}(x_i),\\
&\Longleftrightarrow \forall x_i \in U, \mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i).
\end{aligned} \tag{14}$$

Thus, $S = (U, P \cup D)$ is an inconsistent decision system:

$$\begin{aligned}
&\Longleftrightarrow \neg(\forall x_i \in U, \mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)),\\
&\Longleftrightarrow \exists x_i \in U, \mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{P\cup D}(x_i).
\end{aligned} \tag{15}$$

□

Definition 13 (value-core attribute set of object). Let $S = (U, P \cup D)$ be a consistent decision system, if $\forall x_i \in U$, $\exists a_j \in P \subseteq C$, and $P \neq \varnothing$ satisfy $\mathrm{DIS}_{P-a_j\cup D}(x_i) \neq \mathrm{DIS}_{P-a_j}(x_i)$, then attribute set $\mathrm{CORE}(x_i) = \{a_j \mid a_j \in P \subseteq C, \mathrm{DIS}_{P-a_j\cup D}(x_i) \neq \mathrm{DIS}_{P-a_j}(x_i)\}$ is called value-core attribute set of x_i and $P - \mathrm{CORE}(x_i)$ is called non-value-core attribute set of x_i.

The distinguishability of the value-core cannot be replaced by other condition attributes, while the distinguishability of non-value-core attributes can be replaced by other attributes. The rule can be simplified and generalized by evaluating and deleting redundant non-value-core attributes.

Definition 14 (attribute value reduction set). Let $S = (U, P \cup D)$ be a consistent decision system; if $\forall x_i \in U$, $\exists P \subseteq C$, and $P \neq \varnothing$ satisfy $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)$ and satisfy $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)$ for $\forall Q \subset P$, then P is called an attribute value reduction set of x_i.

Based on the above definitions and theorems, it can be proved that the distinguishability of x_i and the original consistency of S will not change based on the attribute value reduction set P only, and no objects can be found with the same condition attribute values and different decision attribute values with x_i. For any $x_i \in U$, if attribute set $P \subseteq C$ arises and meets the conditions $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)$, then the process is finished.

Theorem 14. *Let $S = (U, C \cup D)$ be a decision system, $x_i \in U$, if x_i has r value reduction sets, $P_k (k = 1, 2, \ldots, r)$, then value-core attribute sets and attribute value reduction sets of x_i satisfy (comply with) the following formula:*

$$\mathrm{CORE}(x_i) = \bigcap_{1\le k\le r} P_k. \tag{16}$$

The value-core attribute set is the intersection of all value reductions, and all attributes in it are essential and indispensable for every value reduction.

6. Algorithm for Conflict Object Set and Duplicate Object Set

Based on Theorem 11, Theorem 12, and Theorem 13, the definitions of "Conflict Object Set" and "Duplicate Object Set" are given as follows.

Definition 15 (conflict object set). Let $S = (U, C \cup D)$ be a consistent decision system, RED be an attribute reduction of S, and $S' = (U', \mathrm{RED} \cup D)$ be a decision system after reduction. If $\forall x_i \in U'$, $\exists a_j \in \mathrm{RED}$, and $P = \mathrm{RED} - a_j$ satisfy $\mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{P\cup D}(x_i)$ (or $|\mathrm{DIS}_P(x_i)| < |\mathrm{DIS}_{P\cup D}(x_i)|$), then object set $U - \mathrm{DIS}_P(x_i)$ ($[x_i]_P$) is called as conflict object set of P.

Condition $\mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{P\cup D}(x_i)$ shows that $\mathrm{DIS}_P(x_i)$ is not equal to $U' - \{x_i\}$. Due to the lack of a_j in P, $\mathrm{DIS}_P(x_i) \cup \{x_i\}$ becomes a conflict object set. That is, $\exists x_k \in U'$ satisfies $(x_i, x_k) \notin \mathrm{DIS}_P$ ($x_k \notin \mathrm{DIS}_P(x_i)$) and $(x_i, x_k) \in \mathrm{DIS}_{P\cup D}$ ($x_k \in \mathrm{DIS}_{P\cup D}(x_i)$). In another word, x_i and x_k are indistinguishable by P only, but x_i and x_k are distinguishable by $P \cup D$. In one word, $S'' = (U', P \cup D)$ is an inconsistent decision system. In the system, conflict objects exist which have the same condition attribute values and different decision attribute values. Moreover, the conflict objects lead to the inconsistency of S''.

Definition 16 (duplicate object set). Let $S = (U, P \cup D)$ be a consistent decision system, RED be an attribute reduction of S, and $S' = (U', \mathrm{RED} \cup D)$ be a decision system after reduction. If $\forall x_i \in U'$, $\exists a_j \in \mathrm{RED}$, and $P = \mathrm{RED} - a_j$ satisfy $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)$ (or $|\mathrm{DIS}_P(x_i)| = |\mathrm{DIS}_{P\cup D}(x_i)|$) and $\mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{\mathrm{RED}}(x_i)$ (or $\mathrm{DIS}_P(x_i) \neq |U'| - 1$ or $\mathrm{DIS}_P(x_i) < |U'| - 1$), then object set $U - \mathrm{DIS}_P(x_i)$ ($[x_i]_P$) is called duplicate object set of S' after reduction.

Conditions $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i)$ and $\mathrm{DIS}_P(x_i) \neq \mathrm{DIS}_{\mathrm{RED}}(x_i)$ show that $\mathrm{DIS}_P(x_i) = \mathrm{DIS}_{P\cup D}(x_i) \neq U' - \{x_i\}$ is true by P or by $P \cup D$. Due to the lack of a_j in P, $\mathrm{DIS}_P(x_i) \cup \{x_i\}$ becomes an indistinguishable object set without any conflict. That is, $\exists x_k \in U'$, which satisfies $(x_i, x_k) \notin \mathrm{DIS}_P$ ($x_k \notin \mathrm{DIS}_P(x_i)$) and $(x_i, x_k) \notin \mathrm{DIS}_{P\cup D}$ ($x_k \notin \mathrm{DIS}_{P\cup D}(x_i)$) and $(x_i, x_k) \in \mathrm{DIS}_{\mathrm{RED}}$ ($x_k \in \mathrm{DIS}_{\mathrm{RED}}(x_i)$). In another word, x_i and x_k are indistinguishable by P or $P \cup D$, but x_i and x_k are distinguishable by RED. In one word, $S'' = (U', P \cup D)$ is a consistent decision system, where duplicate objects exist with the same condition attribute values and same decision attribute values, and the consistency of S'' is not changed while the distinguishability of S'' is changed by the conflict objects.

Input: a sorted decision system $S = (U'', P \cup D)$
Output: a duplicate object set U^* and a conflict object set U^+
(1) Initialize arrays C and D to 1
(2) Let pointer s point to x_1
(3) **for each** $x_j, x_{j+1} \in U$ **do**
(4) Flag = 0
(5) **for each** $a_i \in P$ **do**
(6) **if** $f_i(x_j) \neq f_i(x_{j+1})$, **then**
(7) **if** $C[x_j]! = 1 \&\& D[x_j] == 1$, **then**
(8) $[x_j] \longrightarrow U^*$
(9) **end**
(10) **if** $C[x_j]! = 1 \&\& D[x_j]! = 1$, **then**
(11) $[x_j] \longrightarrow U^*$
(12) **end**
(13) $j = j + 1$
(14) Flag = 1
(15) Let pointer s point to x_j
(16) break
(17) **else**
(18) $i = i + 1$
(19) **end**
(20) **end**
(21) **if** Flag $== 0 \&\& \forall d_i \in D(f_i(x_j) == f_i(x_{j+1}))$, **then**
(22) $C[x_j] + +$
(23) $j = j + 1$
(24) **end**
(25) **if** Flag $== 0 \&\& \exists d_i \in D(f_i(x_j) \neq f_i(x_{j+1}))$, **then**
(26) $C[x_j] + +$
(27) $D[x_j] + +$
(28) $j = j + 1$
(29) **end**
(30) **end**
(31) Output U^* and U^+

Algorithm 1: *ConDup* algorithm.

Based on the above formal analysis and definitions (especially, Definitions 15 and 16), an algorithm for conflict object set and duplicate object set (*ConDup* algorithm) is designed in Algorithm 1.

In Algorithm *ConDup*, $[x_j]$ denotes the current condition equivalence class, C is used to count the base of $[x_j]$, and D is used to count the number of decision values in $[x_j]$.

Steps 3–30 are the main part of the algorithm. Steps 5–20 are used to calculate the conflict object set and duplicate object set of x_j. The condition "$C[x_j]! = 1 \&\& D[x_j] == 1$" shows the base of $[x_j]$ is greater than 1, and all the objects in $[x_j]$ have the same decision attribute values, namely, $[x_j]$ is not a conflict equivalence class but a duplicated equivalence class. Then, all objects in $[x_j]$ should be added into U^*. If the condition "$C[x_j]! = 1 \&\& D[x_j] == 1$" is not true while "$C[x_j]! = 1 \&\& D[x_j]! = 1$" is true, it shows that the base of $[x_j]$ is greater than 1 and all the objects in $[x_j]$ have different decision attribute values, namely, $[x_j]$ is not a duplicated equivalence class but a conflict equivalence class. Then, all objects in $[x_j]$ should be added into U^+. In addition, steps 21–24 are used to judge whether x_j and x_{j+1} are duplicate objects or not. Steps 25–29 are used to judge whether x_j and x_{j+1} are conflict objects or not.

The time complexity of *ConDup* is mainly decided by steps 3–30. Generally, $|D|$ is 1 or far less than $|P|$. So, the time complexity is $O(|P||U|) + O(|D||U|) = \max\{O(|P||U|), O(|D||U|)\} = O(|P||U|)$.

7. Algorithm for Redundant Rule Object Set

After performing value reduction, some rule objects may contain the same decision attributes. Moreover, the attribute value set of one object may be the subset of another object's. Namely, inclusion relation may exist among these rule objects. Actually, the longer rule objects are redundant, which should be removed from the current system.

With the removal of the redundant rules, the rule generalization can be promoted in the system and a more simplified system can be achieved.

Based on the above analysis, for $\forall x_t \in U^*$, the condition that rule object x_t is a redundant rule is that at least one rule object $x_s \in U^*$ exists, which complies with the following conditions:

(1) x_s and x_t belong to the same decision equivalent class. (2) x_s and x_t have the same values among their condition attribute intersection. (3) The rule length of x_s (that is, the

number of x_s's condition attributes with value) is less than x_t's.

If there exist x_s and x_t satisfying condition (3), it means there exists proper relationship between the attribute values set of x_s and x_t's. If there exist x_s and x_t satisfying all the three conditions, it means that x_t is a redundant rule and should be added into redundant rule object set U^-.

In [29], algorithm *RedSet* $(U, P \cup D)$ sorts objects based on $D + P$ firstly and then adds the first objects of consistent condition equivalent classes into U' to get U/P. According to *RedSet* $(U, P \cup D)$ and the characteristic of redundant rules, an algorithm for redundant rule object set (*RedRul* algorithm) is designed in Algorithm 2.

8. Rule Extraction Model against Decision Dependency Degree

8.1. Rule Extraction Model. As the core problem in Rough Set, many scholars have studied the value reduction and rule extraction algorithms and achieved more results [14–28].

After analyzing most of the existing algorithms, it is found that some algorithms are unreasonable for the selection and deletion of condition attributes, and some algorithms have more complicated calculation processes. These algorithms have a higher time and space complexity due to the construction of a discernibility matrix or discernibility function. Moreover, most algorithms are based on algorithms in [23]. The basic principle of those algorithms in [23] is as follows: when performing value reduction on the decision system, for each row of the decision system, where attribute reduction has been finished, it is treated as one decision rule object. For each rule object, its attributes are deleted one by one; at the same time, decision conflict is determined to verify if the deletion is valid or not. With the above principle repeated, value reduction will be achieved finally.

Compared with the previous models, the rule extraction model designed in this paper does not generate matrix and its basic idea is when seeking the minimum reduction, the decision dependency degree is achieved and the value-core attributes that have the greatest impact on decision-making are achieved. After achieving the value-core attributes of the object, the decision dependency degree is used as the attribute importance information to determine the importance of other attributes. Under the premise of without causing a decision conflict or maintaining system consistency, subsequent attribute values are evaluated by the decision dependency degree. With the above principle repeated, a more refined decision rules set (abbreviated as DRS) is achieved and the rule extraction process is completed.

Based on the above basic idea, a rule extraction model against decision dependency degree (*DecDep_Rul* Algorithm) is designed as follows.

8.2. Related Algorithms. Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$. Several algorithms from [29] will be used in the rule extraction model proposed in this paper. Here is a brief introduction.

Generally, the sorting algorithm mainly implements the sorting process by comparing and moving keywords. The average time complexity of the sorting process is preferably $O(|U||\log U|)$. For example, Liu Shao-Hui et al. [30] use the quick sort method against condition attribute set C to perform equivalence class partitioning on universe, which results in time complexity $O(|C||U||\log U|)$ for the reduction model. However, it is not necessary to compare key in pair for base sort. One universe sorting method *RadSort* $(U, P \cup D)$ is given in [29] against static chain base sorting, which has time complexity $O(|P||U|)$ (normally $|D|$ equals to 1 or far less than $|P|$ in reality).

In [29], the algorithm *RedSet* $(U, P \cup D)$ extracts the first element of all equivalence classes of the universe. During the extraction from the whole universe, *RadSort* ensures the consistency of the system and has time complexity $O(|P||U|)$. Working with the sorting method *RadSort*, the algorithm RedSet can effectively reduce the size of universe from $|U|$ to $|U/P|$.

In [29], the algorithm *DecDep _Deg* $(U', P \cup D)$ is used for obtaining $|M_P^*|$ of P. When obtaining $|M_P^*|$, *DecDep _Deg* uses the idea of Definition 9 (corresponding to the Definition 8 in this paper), which does not create decisive distinguishable matrix as the interim result; hereby, the complexity is controlled to $O(|P||U'|)$ (Algorithm 3).

In [29], the attribute importance reduction model based on decision dependency degree is given with Theorems 11 to 15 (corresponding to Theorems 4 to 8 in this paper). The model *DecDep _Red* $(U, C \cup D)$ uses the bottom-up method, with core CORE as fundamentals and with decision dependency degree $\tau_{a_i}(C - \text{CORE})$ as heuristic information. The time complexity of the model is $\max\{O(|C||U|), O(|C|^2|U/C|)\}$.

8.3. Analyzing the Rule Extraction Model Based on Decision Dependency. Let $S = (U, C \cup D)$ be a decision system, $P \subseteq C$. The steps of model *DecDep _Rul* are analyzed in the order of execution as follows.

Firstly, an attribute reduction model *DecDep_Red* $(U, C \cup D)$ as defined in [29] is called in step 1 to complete the attribute reduction and to get reduced decision system $S' = (U', \text{RED} \cup D)$. After that, in step 2, *RadSort* $(U', \text{RED} \cup D)$ and *RedSet* $(U', \text{RED} \cup D)$ in [29] are called to complete extraction on global universe. With deletion of redundant object, new system $S'' = (U'', \text{RED} \cup D)$ is achieved.

Meanwhile, based on decision dependency degree, in steps 3–5, $|M_P^*|$ of attribute set $P = \text{RED} - a_i$ is achieved for any $a_i \in \text{RED}$, where $|M_P^*|$ is distinguishability of P. Since the distinguishability of RED is constant, so when $|M_P^*|$ is smaller, it means without a_i the distinguishable relationship of P becomes more weak. That is, the smaller the $|M_P^*|$ is, the stronger distinguishability of a_i is. In step 6, the count of $|M_P^*|$ is |RED|, and relevant attributes are added into Array according to each $|M_P^*|$ in the descending order.

Input: marked decision system $S^* = (U^*, \text{RED} \cup D)$
Output: redundant rule object set U^-

(1) Min_rule = x_1
(2) **for each** $x_j \in U$ **do**
(3) Flag = 0
(4) **for each** $d_i \in D$ **do**
(5) **if** f_i(Min_rule) $\neq f_i(x_j)$ **then**
(6) Min_rule = x_j
(7) Flag = 1
(8) Break
(9) **else**
(10) $i = i + 1$
(11) **end**
(12) **end**
(13) **if** *Flag* == 0&& *Min_rule and* x_j *have the same exact values among their condition attribute set* **then**
(14) **if** *the rule length of* x_j *is less than Min_rule's* **then**
(15) Min_rule $\longrightarrow U^-$
(16) Min_rule = x_j
(17) **else**
(18) $x_j \longrightarrow U^-$
(19) **end**
(20) **end**
(21) **if** *Flag* == 0&& *Min_rule and* x_j *do not have the same exact values among their condition attribute set* **then**
(22) Min_rule = x_j
(23) **end**
(24) $j = j + 1$
(25) **end**
(26) Output U^-

Algorithm 2: *RedRul* algorithm.

Steps 7–11 are to mark the system. For each object in the system, its useless attributes are marked as redundant attributes with "∗," its value-core attributes whose absence will lead to decision conflict are kept the original values, and the rest are marked with "!" for further assessment on whether to keep or remove. In step 8, $RadSort(U'', P \cup D)$ is called to sort all rule objects by $D + P$, which can make step 9 efficient. In step 9, $ConDup(U'', P \cup D)$ is called. It determines whether current equivalence class is conflict equivalence class or duplicate equivalence class and gets conflict object set U^+ and duplicate object set U^*. After that, the mark procedure is executed in step 10. The step replaces the value of a_i with "∗" for objects in U^*, keeps the original value of a_i for objects in U^+, and replaces the value of a_i with "!" for objects in $U'' - U^* - U^+$.

In steps 12–29, for each object, its non-value-core attributes are evaluated and marked with "!" to decide whether to keep or remove. The attribute set $P^!$ is used to store attributes marked as "!." And relevant attributes are added into $\text{Array}^!$ in descending order according to the $|M_P^*|$ values. $\text{Array}^![j]$ is the sequence number of the relevant attribute for each object. In step 22, $RadSort(U'', P \cup D)$ is called. As system S'' is sorted by $RadSort(U'', P \cup D)$, where attribute set P of x_i keeps original values, it is easy and fast to get conditional equivalence of x_i. In this condition equivalence class, objects which have the same condition attribute values with P of x_i and have different decision attribute values are counted to get decision conflict set $U^+(x_i)$ of x_i. In steps 23–27, the attribute of x_i with the minimum $|M_P^*|$ is restored to original value, where $\text{Array}^![j]$ is the sequence number of that attribute for x_i. After that, attribute set P is updated by adding attribute $a_{\text{Array}^![j]}$ and attribute set $P^!$ is updated by deleting $a_{\text{Array}^![j]}$. $RadSort(U'', P \cup D)$ is called to sort unmarked system S'' by $D + P$ and to update conflict set $U^+(x_i)$ of x_i. At the end of the loop processing, $U^+(xi)$ is empty. At that time, the decision can be determined by all of the attributes with original value from x_i. At last in steps 28-29, all "!" of x_i are set to "∗" to achieve a new marked system $S^* = (U'', \text{RED} \cup D)$.

Algorithm $RadSort(U'', P \cup D)$ is called in step 30 to sort universe U'' of the marked system $S^* = (U'', \text{RED} \cup D)$ by $D + \text{RED}$. $RadSet(U'', P \cup D)$ is called in step 31 to delete full rule and to delete potentially duplicate decision rules and then update universe U^* of the marked system $S^* = (U^*, \text{RED} \cup D)$.

RadSort uses two operations distribution and collection in [29] to sort object in universe by attribute set. In short, *RadSort* mainly adopts the base sort method, where "distribution" is actually to find equivalence class. Together with the characteristics of redundancy rules, if rule objects in universe can be aggregated according to decision attribute after the sorting, that is, rule objects with the same decision attribute can be adjacent; then, it is easy for searching and deleting redundancy rules. Thereafter, all the rule objects can be "distributed" and "collected" by condition attributes; after that, all the objects can be "distributed" and "collected" by

Input: decision system $S = (U, C \cup D)$
Output: the decision rule set DSR
(1) Call *DecDep_Red* $(U, C \cup D)$ reduction model in [29] to get $S' = (U', \text{RED} \cup D)$; initialize rule set DSR
(2) Call *RadSort* $(U', \text{RED} \cup D)$ and *RedSet* $(U', \text{RED} \cup D)$ in [29] to get $S'' = (U'', \text{RED} \cup D)$
(3) **for each** $a_i \in \text{RED}$ **do**
(4) Let $P = \text{RED} - a_i$; call *RadSort* $(U'', P \cup D)$ and *DecDep_Deg* $(U'', P \cup D)$ in [29] to get $|M_P^*|$ of P
(5) **end**
(6) Add relevant attributes into *Array* in descending order according to the achieved $|M_P^*|$ values
(7) **for each** $a_i \in \text{RED}$ **do**
(8) Let $P = \text{RED} - a_i$ and call *RadSort* $(U'', P \cup D)$
(9) Call *ConDup* $(U'', P \cup D)$ to get U^+ and U^*
(10) Execute a mark procedure to get $S^* = (U'', \text{RED} \cup D)$
(11) **end**
(12) **for each** $x_i \in S^*$ **do**
(13) **if** $P^! == \varnothing$, **then**
(14) For all attributes of x_{i++}, add attributes with original value to P, add attributes with "!" to $P^!$
(15) **end**
(16) **if** $P == \varnothing$, **then**
(17) Let $P = \{a_{\text{Array}}[0]\}$; Restore $f_{a_{\text{Array}^![f]}(x_i)}$ to its original value and update $P^!$
(18) **end**
(19) **for each** $a_j \in P^!$ **do**
(20) $\text{Array}[a_j] \longrightarrow \text{Array}^!$
(21) **end**
(22) Call *RadSort* $(U'', P \cup D)$ and get $U^+(x_i)$
(23) **while** $|U^+(x_i)! = 0|$
(24) Restore $f_{a_{\text{Array}^![j++]}(x_i)}$ to its original value
(25) Update P and $P^!$
(26) Call *RadSort* $(U'', P \cup D)$ and update $U^+(x_i)$
(27) **end**
(28) Replace each "!" with "$*$" for all attribute value of x_i; and update $S^* = (U'', \text{RED} \cup D)$
(29) **end**
(30) Call *RedSort* $(U'', \text{RED} \cup D)$
(31) Call *RedSet* $(U'', \text{RED} \cup D)$, delete duplicate rules, and delete rules only with "$*$" to update U^* of S^*
(32) Call *RedSort* $(U^*, \text{RED} \cup D)$
(33) Call *RedRul* $(U^*, \text{RED} \cup D)$ to get U^-
(34) For $S^* = (U^*, \text{RED} \cup D)$, update $U^* = U^* - U^-$
(35) Delete "$*$" in each rule of U^* to get DSR

Algorithm 3: *DecDep_Rul* algorithm.

decision attributes to sort the whole universe by decision equivalence class and sort decision equivalence class by conditional equivalence class.

To this end, *RadSort* $(U^*, \text{RED} \cup D)$ is called in step 32 to sort rule objects in current universe U^* by RED + D (which is not by $D + P$ as mentioned in [29]). That is, when sorting by RED + D, for any $a_i \in \text{RED} + D$, if columns 0 to $m-2$ from RED + D are condition attributes after reduction and column $m-1$ is a decision attribute, then i can take values from 0 to $m-1$ in ascending order. As *RadSort* will perform base sort by condition attributes firstly, followed by the decision attribute, rule objects with the same decision value will be gathered in groups; then, it is easy to get decision equivalence class. Therefore, it is easy to determine inclusion relationship among rule instances.

The importance of a_i will be evaluated by $|M_P^*|$ and attribute with smaller $|M_P^*|$ will give preference to change from "!" to original value. Similarly, when sorting rule objects by RED, it is reasonable to perform the sort later for those attributes with smaller $|M_P^*|$. As a result, after sorting universe, attributes of rule objects with original value will be found in the top column (top column attribute with stronger distinguishability). Then, it is convenient to compare, search, and locate multiple redundancy rule objects, among which exist inclusion relationships.

After sorting the whole universe by $\text{RED} \cup D$ with *RadSort* $(U^*, \text{RED} \cup D)$, rule objects will be aggregated by decision equivalence class and condition equivalence class. In step 33, *RedRul* $(U^*, \text{RED} \cup D)$ is called to get redundancy rules set U^- from U^*. At last, in step 34, the difference between rule objects set U^* and redundancy rules set U^- is nonredundancy rules set. After deleting "$*$" in this nonredundancy rules set, a rule set DSR is finally achieved after reduction.

8.4. Analysis of Model Time Complexity. Step 1 is the attribute reduction model in [29]. The time complexity of the model is $\max\{O(|C||U|), O(|C|^2|U/C|)\}$. The other steps, that is, steps 2–35, are composed of attribute value reduction model.

Thereinto, time complexity of step 2 is $O(|\text{RED}||U'|)$. Time complexity of steps 3–5 is $O(|\text{RED}|) \times (O(|\text{RED} - 1||U''|) + O(|\text{RED} - 1||U''|)) = O(|\text{RED}||\text{RED} - 1||U''|) = O(|\text{RED}|^2|U''|)$. The time complexity of *ConDup* is mainly decided by steps 3–30. Generally, $|D|$ is 1 or far less than $|P|$. So, the time complexity is $O(|P||U|) + O(|D||U|) = \max\{O(|P||U|), O(|D||U|)\} = O(|P||U|)$. Time complexity of steps 7–11 is mainly decided by steps 8 and 9, so time complexity of steps 7–11 is $O(|\text{RED}|) \times (O(|\text{RED} - 1||U''|) + O(|\text{RED} - 1||U''|)) = O(|\text{RED}||\text{RED} - 1||U''|) = O(|\text{RED}|^2 |U''|)$. Time complexity of steps 12–29 is mainly decided by steps 22–27, so time complexity of steps 12–29 is $O(|U''|) \times O(|P||U''|) + O(|U''|) \times O(|P||[x_i]_P|) + O(|U^+(x_i)|) \times (O(|P||U''|) + O(|P||[x_i]_P|)) = O(|P||U''|^2)$. Because P is the attribute set of keeping original value of object and $\max\{|P|\} = |\text{RED}|$, so time complexity of steps 12–29 is $O(|P||U''|^2) = O(|\text{RED}||U''|^2)$. Time complexity of step 30 is $O(|\text{RED}||U''|)$. Time complexity of step 31 is $O(|\text{RED}||U''|)$. And time complexity of steps 32–34 is $O(|\text{RED}||U^*|) + O(|\text{RED}||U^*|) = O(|\text{RED}||U^*|)$.

Given the above analysis, time complexity of steps 2–35 is $\max\{O(|\text{RED}||U'|), O(|\text{RED}|^2|U''|)$, and $O(|\text{RED}||U''|^2)\}$.

As U' is U/C and U'' is U'/RED, $|U'| = |U/C|$ and $|U''| = |U'/\text{RED}| = |(U/C)/\text{RED}|$. That is, time complexity of the rule extraction model is $\max\{O(|C||U|), O(|C|^2|U/C|), O(|\text{RED}||U'|), O(|\text{RED}|^2|U''|), O(|\text{RED}||U''|^2)\} = \max\{O(|C||U|), O(|C|^2|U/C|), O(|\text{RED}||U''|^2)\} = \max\{O(|C||U|), O(|C|^2|U/C|), O(|\text{RED}||(U/C)/\text{RED}|^2)\}$.

9. Simulation Example Analysis

Let decision system $S = (U, \text{RED} \cup D)$, which is the reduction result after performing steps 1-2, where $U = \{x_1, x_2, \ldots, x_{27}\}$, $\text{RED} = \{a, b, c, d\}$, and $D = \{E\}$. Illustrated below with demo in Figure 1(a) (rule objects in the table is processed by "ordering whole universe U by $D + RED$" and "the full universe extraction") is the detail on rule extraction procedure.

Firstly, $|M_P^*|$ for each $a_i \in \text{RED}$ is achieved by completing steps 3–5. The result is as follows: $|M_{\text{RED}-a}^*| = 151 + 26 + 21 * 2 = 219$, $|M_{\text{RED}-b}^*| = 152 + 26 + 21 * 2 = 220$, $|M_{\text{RED}-c}^*| = 149 + 24 + 21 + 19 = 213$, and $|M_{\text{RED}-d}^*| = 146 + 25 + 21 + 20 = 212$. Thereafter, in step 6, the order of attributes in Array (in descending order according to attribute importance) is $d \longrightarrow c \longrightarrow a \longrightarrow b$. Perform steps 7–11 for any $a_i \in \text{RED}$. In step 8, $RadSort(U'', \text{RED} - \{a\} \cup D)$ is called to sort universe by $D + P$. When $P = \text{RED} - a$, results are specified in Figure 1(b).

In step 9, $ConDup(U'', \text{RED} - \{a\} \cup D)$ is called to get conflict object set U^+ and duplicate object set U^*. Then, get $U^+ = \{x_1, x_{10}, x_{18}\} \cup \{x_3, x_{12}, x_{20}\} \cup \{x_4, x_{13}, x_{21}\} \cup \{x_5, x_{14}, x_{24}\} \cup \{x_6, x_{15}, x_{25}\} \cup \{x_7, x_{16}, x_{26}\}$ and $U^+ = \{x_2, x_{11}, x_{19}\} \cup \{x_8, x_{17}, x_{27}\}$. After marking the value column of a in step 10, a marked system is specified in Figure 1(c).

In a similar way, steps 7–11 are performed for other attributes in RED to get marked results, respectively, specified in Figure 1(d).

Steps 12–29 are called to evaluate rule objects. Let $P = \{a_{\text{Array}[0]}\} = \{d\}$ and $f_{a_{\text{Array}^![f]}} = f_d(x_9)$ be restored to original value "2," then $P^! = \{a, b, c\}$ and $\text{Array}^! = \{c, a, b\}$. In step 22, according to the value of x_9, $RadSort(U'', P \cup D)$ is called to sort unmarked system S'' by $D + P$. With reference to Figure 1(a), the conflict set of x_1 is $U^+(x_9) \neq \varnothing$. Then, steps 23–27 are performed to restore the value of attribute c for x_9 to 3 and to update $P = \{c, d\}$ and $P^! = \{a, b\}$. According to the value of x_9, $RadSort(U'', P \cup D)$ is called to sort the unmarked system S'' by $D + P$. The sorted result is $\{x_3, x_1, x_6, x_4, x_5, x_2\}$. With reference to Figure 1(a), conflict set $U^+(x_9) = \{x_{23}\}$ of x_9 is achieved. Repeat execution on steps 23–27, a of x_9 is restored to original value "2," $P = \{c, d, a\}$, $P^! = \{b\}$, and $\text{Array}^! = \{b\}$. At that time, according to the value of x_9, conflict set $U^+(x_9) = \varnothing$. At the completion of loop execution on steps 23–27, the remaining "!" of x_i is restored to "$*$."

Similarly, rule objects x_{22} and x_{23} are processed by sequence. The two objects can make decision in the original decision system according to current value, so the remaining "!" of x_{22} and x_{23} is restored to "$*$."

Get marked system $S^* = (U'', \text{RED} \cup D)$ specified in Figure 1(e). After steps 30-31, perform step 32 to order U^* by $\text{RED} + D$ and get marked system S^* specified in Figure 1(f). In Figure 1(f), rule objects with strikethrough are duplicate objects. Then, a rule set specified in Figure 1(g) is achieved after deleting redundant rules in steps 32–34.

Thereafter, for all rules, "$*$" in each rule are deleted and a reduced rule set $\text{DSR} = \{c1 \longrightarrow E3;\ d1 \longrightarrow E3;\ c1d2 \longrightarrow E2;\ c2d2 \longrightarrow E1;\ c3d1 \longrightarrow E0;\ c3d2 \longrightarrow E1;\ a2b2c2 \longrightarrow E3;\ a3b2c2 \longrightarrow E3\}$ is achieved.

10. Experiment Analysis

Dataset "Iris Plants Database" from UCI database [31] is used to compare rule extraction methods. Method A (proposed in this paper), method B (proposed in [22]), and method C (proposed in [23]) are compared on accuracy and effectiveness. There are 150 instances in Iris Plants Database, including 4 condition attributes "sepal length," "sepal width," "petal length," and "petal width" and one decision attribute "class." All the instances in the dataset are categorized into 3 decision classes. Because all the condition attribute data are continuous attribute data, the "discretization" method in ROSE [32] is used to discrete them as Rough Set has limited processing capacity on continuous data.

Experimental comparison results of three methods are given in Tables 2–4 and Figure 2. All of the instances are separated into two parts: one is for training set and another is for testing set randomly. One part is treated as the training set of random objects and it is used as learning samples. The other part is treated as the testing set which is the difference set of whole dataset and training set.

Testing of three groups is performed on the set, including

(1) Count of instances in training set/count of instances in testing set = 80/70

(2) Count of instances in training set/count of instances in testing set = 100/50

(3) Count of instances in training set/count of instances in testing set = 120/30

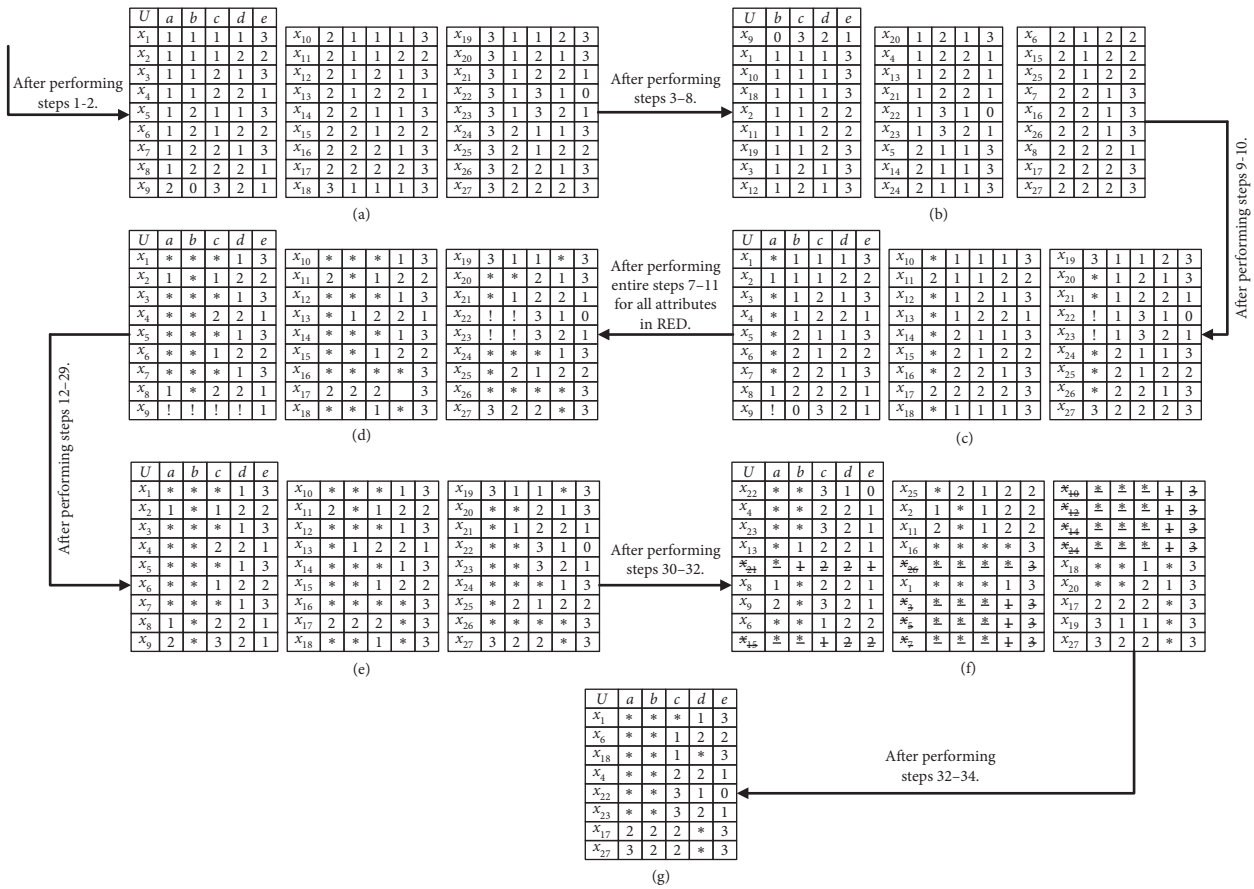

Figure 1: Simulation example analysis. (a) The reduction result after performing steps 1-2. (b) The result after performing steps 3–8. (c) The marked system after performing steps 9-10. (d) The marked system after performing steps 7–11 for all attributes in RED. (e) The marked system after performing steps 12–29. (f) The marked system after performing steps 30–32. (g) The rule set after deleting redundant rules.

Table 2: Groups.

Groups	Training set (count of instances)	Testing set (count of instances)
Group 1	80	70
Group 2	100	50
Group 3	120	30

Count of rules, average length of rules, and average accuracy of all the rules by each method are calculated in each testing, where average accuracy is $\sum_{i=1}^{i=|\text{rule set deduced}|}\text{accuracy of rule}_i/|\text{rule set deduced}|$. At the same time, to avoid randomness of a single experiment, it is executed 10 times for each group experiences and then the average value is obtained.

The comparison results are shown in Tables 2–4 and Figure 2: average accuracy of methods A and B in three-group testing is almost the same and average accuracy from two-group testing result is a bit higher by method A. Moreover, count of rules and average length of rules from method A are less than of methods B and C. With the practice instance increasing, count of rules getting from method A is not significantly increased.

Considering run time, "Geriatric Care Medical Dataset" and other 3 UCI datasets are used as testing datasets in Table 5. The "Geriatric Care Medical Dataset" is from the Canadian Study of Health and Aging. Detailed description of the dataset is available in [33]. The dataset has 8,547 instances, in which 5,089 are female and the rest are male. And, the dataset has 44 attributes, such as chest, diabetes, and dental. Its class or decision attribute is a binary value indicating whether an individual instance has died during follow-up.

LEM2 algorithm (Learning from Examples Module, version (2) [34]) is a rule extraction algorithm based on Rough Set. Since the LEM2 algorithm does not change the content and structure of the original system, and the extracted rules are not affected by the default values, and it has become one of the most commonly used rule extraction algorithms in recent years. Method D (LEM2 algorithm) and other three methods are compared on time.

Experimental comparison results of four methods are given in Table 5. The blind selection of attribute-value pairs and the multiple traversal of multiple cycles leads to the inefficiency of LEM2 algorithm. From the analysis in section "Analysis of model time complexity", time performance of method A is obviously better than other three comparison methods. And the experimental results also show that the proposed method A is better than other methods in terms of running time.

Table 3: Test results of method A.

Groups	Method A		
	Count of rules	Average length of rules	Average accuracy (%)
Group 1	6.5	2.50	94.5
Group 2	7.2	2.00	96.3
Group 3	8.4	1.88	98.5

Table 4: Test results of method B and method C.

Groups	Method B			Method C		
	Count of rules	Average length of rules	Average accuracy (%)	Count of rules	Average length of rules	Average accuracy (%)
Group 1	7.1	2.57	93.1	7.5	3.2	92.3
Group 2	8.0	2.12	96.5	8.2	2.85	93.6
Group 3	10.1	1.90	98.0	10.5	2.32	93.8

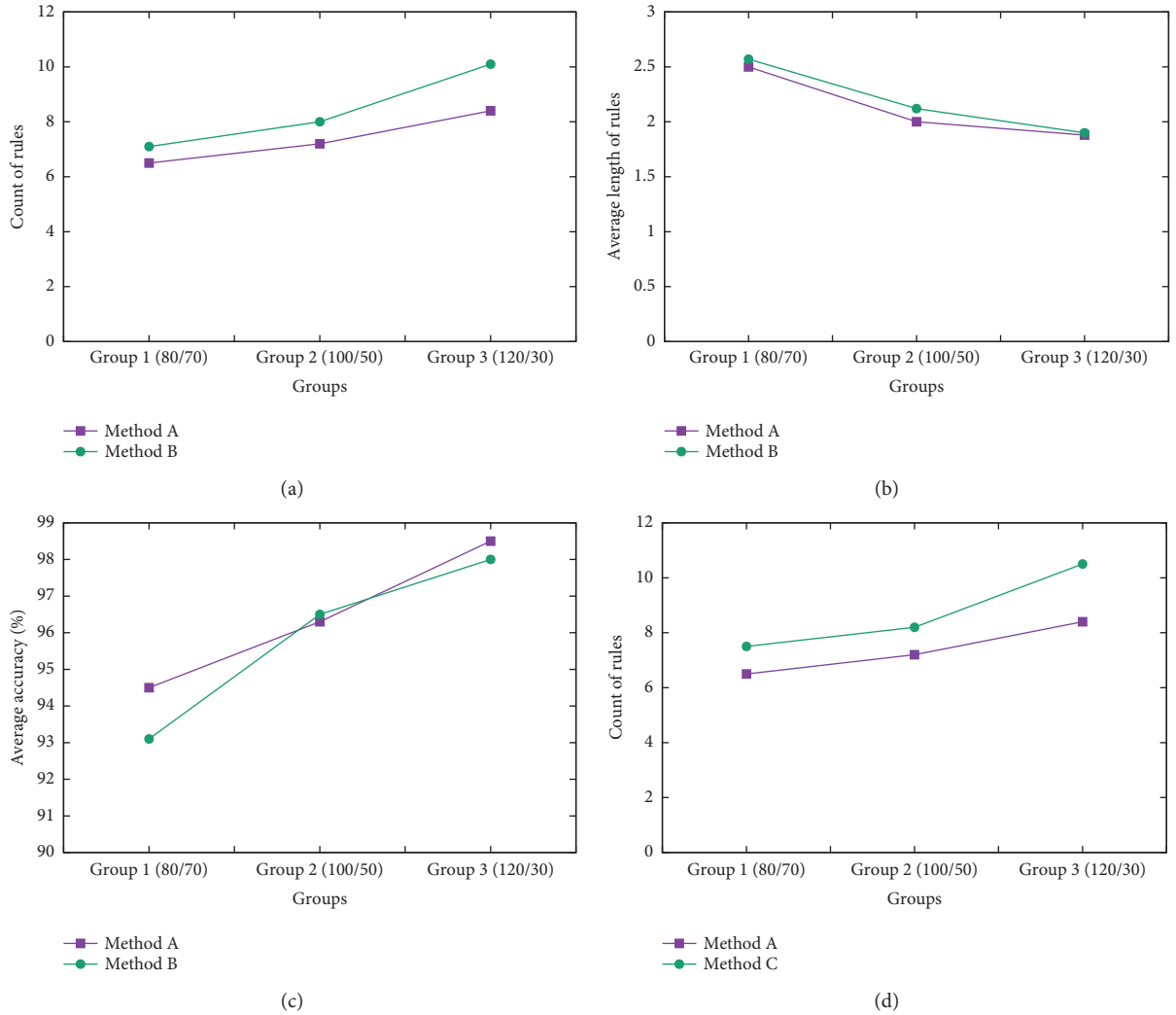

Figure 2: Continued.

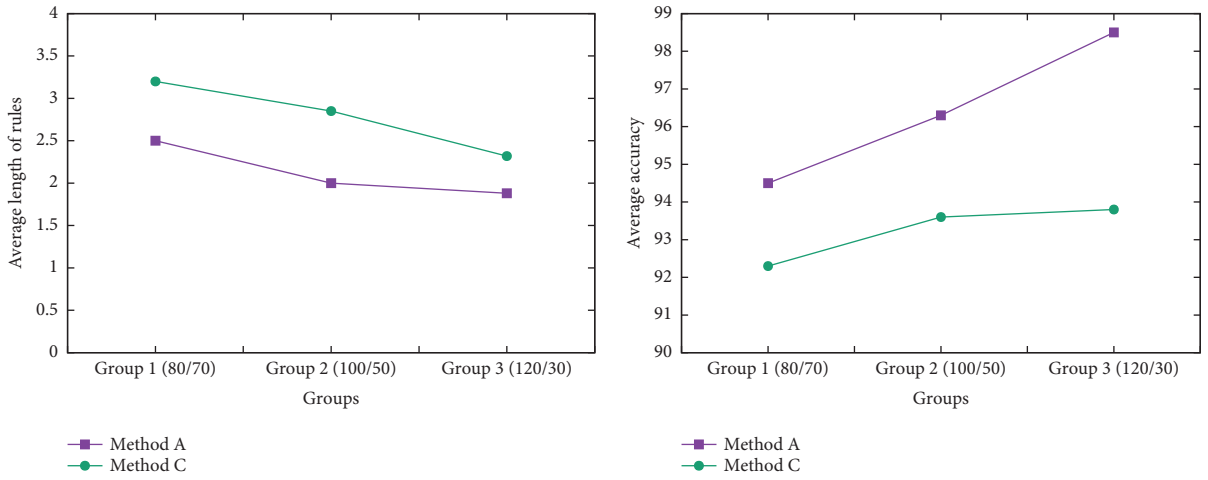

Figure 2: Experiment analysis.

Table 5: Comparison of run time.

	Testing dataset		Run time (S)			
Name	Count of instances	Count of attributes	Method A	Method B	Method C	Method D
Wine quality	4,898	11	0.805	0.947	1.135	0.933
Mushroom	8,124	22	1.246	1.562	1.921	1.601
Letter recognition	20,000	16	2.537	2.871	3.906	3.281
Geriatric care medical	8,547	44	3.201	4.251	5.163	4.592

11. Challenges

The model *DecDep_Rel* calls algorithm *RedSet*, which can extract the first element of all equivalence classes in the universe to complete extraction and get U/P. And before calling algorithms ConDup and RedRul, the model *DecDep_Rul* calls algorithm RedSort or algorithm *RedSet*. These two algorithms can effectively complete the extraction of the universe.

Therefore, the model *DecDep_Rul* is more suitable for processing datasets that contain more duplicate data and redundant data, while it is not well suitable for processing datasets that contain very few duplicate object instances or redundant data.

And, we also find the rule extraction model itself cannot work well in larger than or equal to PB dataset scales, because it takes too long to get rules. We believe that in combination with parallel processing, the model *DecDep_Rul* could better handle PB dataset scales. This is also what we will study further in the future.

12. Conclusion

In this paper, based on formalized definitions "Attribute Value Reduction" and "Value-Core Attribute," the algorithm ConDup for conflict object set and duplicate object set and the algorithm RedRul for redundant rule object set are proposed. Combined with algorithms in [29] and related theorems on attribute value reduction, the rule extraction model based on decision dependency degree *DecDep_Rul* is given. As the model does not generate matrix during reduction and uses object equivalent class to get conflict and duplicate object set, its time-space complexity is effectively controlled. The analysis results of simulation example and testing sets show that the new model can reduce redundant data more accurately and effectively while keeping the system classification ability unchanged. In conclusion, the new model can extract concise and key information and is effective and fast.

Data Availability

In the past, the data supporting the conclusions of the study were available at "Silicon Graphics International Corp. Sgi-Mlc++: Datasets from U-ci[EB/OL]. [2014-08-08]: "http://www.sgi.com/tech/mlc/db/". And now, the same data are available at "http://www.martinbroadhurst.com/stl/". These datasets are cited at relevant places within the text as references [31].

Conflicts of Interest

The author declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under (Grant nos. 61976032, 61371090,

61602076, and 61702072), the China Postdoctoral Science Foundation Funded Project (2017M621122 and 2017M611211), the Natural Science Foundation of LiaoNing Province (nos. 20170540144 and 20170540232), and the Fundamental Research Funds for the Central Universities (nos. 3132017118, 3132017121, and 3132017123).

References

[1] N. Zhong, J. H. Ma, R. H. Huang et al., "Research challenges and perspectives on wisdom web of Things (W2T)," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 862–882, 2013.

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

[3] S. K. M. Wong and W. Ziarko, "On optimal decision rules in decision tables," *Bulletin of the Polish Academy of Sciences Mathematics*, vol. 33, no. 11-12, pp. 693–696, 1985.

[4] Z.-L. Zhang and S.-Y. Liu, "Decision-theoretic rough set attribute reduction based on backtracking search algorithm," *Computer Engineering and Applications*, vol. 52, no. 10, pp. 71–74, 2016.

[5] J. Zhao, J.-J. Liang, and Z.-N. Dong, "Rough set attribute reduction algorithm using bit arithmetic and core attributes quick identification," *Journal of Chinese Computer System*, vol. 2, pp. 316–321, 2015.

[6] J. Xu and M. Guo, "Attribute reduction algorithm based on relative refinement capacity," *Computer Science*, vol. 42, no. s1, pp. 94–97, 2015.

[7] Y.-Y. Chen and Y.-M. Chen, "Attribute reduction algorithm based on information entropy and ant colony optimization," *Journal of Chinese Computer Systems*, vol. 36, no. 3, pp. 586–590, 2015.

[8] F. Jiang, S.-S. Wang, J.-W. Du et al., "Attribute reduction based on approximation decision entropy," *Control and Decision*, vol. 30, no. 1, pp. 65–70, 2015.

[9] Y.-Q. Wang and N.-B. Fan, "Improved algorithms for attribute reduction based on simple binary discernibility matrix," *Computer Science*, vol. 42, no. 6, pp. 210–215, 2015.

[10] C.-J. Yang, H. Ge, and L.-S. Li, "Attribute reduction of vertically partitioned binary discernibility matrix," *Control and Decision*, vol. 28, no. 4, pp. 563–568, 2013.

[11] T. Wang, Z.-Y. Xu, Y.-W. Chen et al., "Method of compressed discernibility matrix of the attribute reduction algorithm based on incompletion decision table," *Computer Science*, vol. 41, no. 6A, pp. 377–382, 2014.

[12] D. Ye, Z. Chen, and S. Ma, "A novel and better fitness evaluation for rough set based minimum attribute reduction problem," *Information Sciences*, vol. 222, pp. 413–423, 2013.

[13] W. Xu, Y. Li, and X. Liao, "Approaches to attribute reductions based on rough set and matrix computation in inconsistent ordered information systems," *Knowledge-Based Systems*, vol. 27, no. 3, pp. 78–91, 2012.

[14] F.-L. Yu, R.-J. Wang, and X.-H. Zhu, "Dynamic reduction and rule extraction based on discernibility matrix algorithm," *Automation and Instrumentation*, vol. 22, no. 6, pp. 1–4, 2007.

[15] J.-H. Gu, Y.-C. Zhou, and J. Song, "A false way of attributes value reduction," *Acta Scientiarum Naturalium Universitatis Nankaiensis*, vol. 36, no. 4, pp. 38–42, 2003.

[16] Y.-B. Liu, C.-Y. Hu, and D.-Y. Liu, "Value reduction algorithm DMBVR based on the discernible matrix of rough set," *Journal of Jilin University (Science Edition)*, vol. 42, no. 2, pp. 221–225, 2004.

[17] H. Rao, Y.-J. Xia, and M.-Z. Li, "Algorithm for rule extraction based on discernibility matrix and attribute significance," *Computer Engineering and Applications*, vol. 44, no. 23, pp. 163–165, 2008.

[18] J. Li and J.-Y. Wang, "Research on rule discernibility matrix based on rough set theory," *Computer Engineering and Applications*, vol. 42, no. 11, pp. 27–31, 2006.

[19] L.-P. Luo, E.-G. Liu, and G.-C. Wang, "Acquisition of briefest decision rule based on matrix," *Computer Engineering*, vol. 34, no. 19, pp. 41–43, 2008.

[20] X.-B. Lin and D.-Y. Ye, "A rule acquisition method based on extended discernibility matrix," *Computer Science*, vol. 35, no. 3, pp. 231–233, 2008.

[21] X. E., L.-S. Shao, and F. Yang, "Method of rule extraction based on discernible vector," *Journal of Liaoning Technical University (Natural Science)*, vol. 29, no. 5, pp. 787–790, 2010.

[22] X. E., L.-S. Shao, and Y.-Z. Zhang, "Method of rule extraction based on rough set theory," *Computer Science*, vol. 38, no. 1, pp. 232–235, 2011.

[23] L.-Y. Chang, G.-Y. Wang, and Y. Wu, "An approach for attribute reduction and rule generation based on rough set theory," *Journal of Software*, vol. 10, no. 11, pp. 1206–1211, 1999.

[24] H. Zhu, "A rough-set-based algorithm to determine the reduction of attributes and their values," *Natural Science Journal of Xiangtan University*, vol. 24, no. 3, pp. 36–39, 2002.

[25] J.-Y. Lin, H. Peng, and Q.-L. Zheng, "A new algorithm for value reduction based on rough set," *Computer Engineering*, vol. 29, no. 4, pp. 70-71, 2003.

[26] H.-B. Jin, "An improved heuristic value reduction algorithm based on rough set," *Journal of Taiyuan University of Science and Technology*, vol. 31, no. 3, pp. 181–184, 2010.

[27] L.-S. Zhao and J.-H. Shi, "Real value attribute reduction method based on rough set," *Journal of Inner Mongolia University*, vol. 1, pp. 97–101, 2010.

[28] W.-X. Zhang and G.-F. Qiu, *Uncertain Decision Making Based on Rough Sets*, Tsinghua University Press, Beijing, China, 2006.

[29] X.-Y. Chen, G.-Y. Li, and Y.-H. Liu, "Research on rough set reduction model based on decision dependency degree," *System Engineering ŀTheory & Practice*, vol. 36, no. 2, pp. 505–516, 2016.

[30] S.-H. Liu, Q.-J. Sheng, and B. Wu, "Research on efficient algorithms for rough set methods," *Chinese Journal of Computers*, vol. 26, no. 5, pp. 524–529, 2003.

[31] Silicon Graphics International Corp., Sgi-Mlc++: Datasets from Uci[EB/OL]. [2014-08-08], http://www.sgi.com/tech/mlc/db/.

[32] Z.-Z. Shi, *Knowledge Discovery*, Tsinghua University Press, Beijing, China, 2002.

[33] J. Li, *Rough set based rule evaluations and their applications*, Ph.D. thesis, University of Waterloo, Waterloo, Canada, 2007.

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