A Reduction Algorithm Meeting Users' Requirements

ZHAO Kai (赵 凯) and WANG Jue (王 珏)

Institute of Automation, The Chinese Academy of Sciences, Beijing 100080, P.R. China

E-mail: kai.zhao@mail.ia.ac.cn

Received February 1, 2002; revised June 12, 2002.

Abstract Generally a database encompasses various kinds of knowledge and is shared by many users. Different users may prefer different kinds of knowledge. So it is important for a data mining algorithm to output specific knowledge according to users' current requirements (preference). We call this kind of data mining requirement-oriented knowledge discovery (ROKD). When the rough set theory is used in data mining, the ROKD problem is how to find a reduct and corresponding rules interesting for the user. Since reducts and rules are generated in the same way, this paper only concerns with how to find a particular reduct. The user's requirement is described by an order of attributes, called attribute order, which implies the importance of attributes for the user. In the order, more important attributes are located before less important ones. Then the problem becomes how to find a reduct including those attributes anterior in the attribute order. An approach to dealing with such a problem is proposed. And its completeness for reduct is proved. After that, three kinds of attribute order are developed to describe various user requirements.

Keywords requirement-oriented knowledge discovery, attribute order, discernibility matrix, reduction algorithm

1 Introduction

As stated in [1], "$\cdots$*data mining tasks can be classified into two categories: descriptive and predictive. Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions.* $\cdots$" Since prediction has the same motivation with machine learning, description is more distinct than prediction in data mining. In detail, description aims at finding a concise explanation (reduction) of the given data set, or what we call data extraction.

There are three issues in data extraction. First, a database is usually shared by many users who have their own requirements (preferences) for the knowledge discovered. As the database does contain all the knowledge required, it is important for the data mining algorithm to provide the knowledge a user prefers. We call this problem *requirement-oriented knowledge discovery* (ROKD). Second, efficient algorithms are needed to deal with huge data sets. Third, the result should be presented in a natural language or visual style. While a great effort has been made on the last two issues[2−8], we concentrate on the first one, that is how to discover knowledge according to the users' requirements.

Generally speaking, reduction on words has a close relation with inductive machine learning, with which a data set is typically consistent with several hypotheses (knowledge). Since it is generally NP-complete to find all the hypotheses and the hypotheses have little meaning for a particular user except making him bored, some strategies are embedded in an algorithm to determine just one hypothesis as the output. Those strategies are called *biases* of the algorithm[9−15]. Abstractly, inductive machine learning may be represented as a function, $y = f(x)$, where x denotes the data set, and y denotes the knowledge discovered. Taken biases into consideration, a coefficient α is added in and the function is varied to $y = \alpha f(x)$. In classical learning algorithms, biases are usually implicitly fixed in the algorithms and some modifications, such as shift bias, are introduced later[9,10,16]. The criteria of

This work is supported by the National Key Project for Prime Research on Image, Speech, Natural Language Understanding and Knowledge Mining (NKBRSF, Grant No.G1998030508).

designing a bias consist of predictive accuracy, the computational complexity of the algorithm, the syntactic complexity of the induced rules, and the stability of the algorithm[17]. We suggest another kind of criteria, the requirements of users.

Using the fixed or shift bias, traditional algorithms actually assume that all the users have the same requirements for the knowledge. Therefore the same knowledge are output for all users. But in many applications, that is problematic. Most huge data sets are shared by many users, who usually have their own requirements. That means different users may require different knowledge.

Consider a simple example about the production of soybean.

Table 1. Example About Production of Soybean

Temperature	Humidity	Wind	Fertilization	Pesticide	Production
high	middle	middle	fer1	pes1	high
middle	middle	middle	fer2	pes1	high
middle	high	weak	fer1	pes2	high
middle	high	middle	fer1	pes1	low
low	middle	strong	fer1	pes1	low
high	middle	middle	fer3	pes2	low
high	high	middle	fer1	pes2	low

There are seven records in Table 1. Three of them are about high production of soybean, while the other four about low production. Among the six attributes, *production* is a decision attribute and the remaining five are conditions. In the *fertilization* and *pesticide* attributes, *fer*1, *fer*2 and *fer*3 represent three kinds of fertilizers, and *pes*1 and *pes*2 represent two brands of pesticides.

Suppose there are two users, A and B. A wonders how *temperature*, *humidity* and *wind* affect the soybean production, while B cares about the effect of pesticide and fertilization. If the data set does contain those two kinds of knowledge, how does an algorithm output customized knowledge for the user? That is, how to provide A with the knowledge about *temperature*, *humidity* and *wind*, and B with the knowledge about *pesticide* and *fertilization*, respectively?

To accomplish such a task, we have to answer two questions. First, how does a user describe his requirement to the computer? Second, after a user has given the description of his requirement, how does the algorithm support the requirement, i.e., discover the particular knowledge according to the requirement description. In the following, we try to answer these two questions.

For the first question, we suggest a method based on the order of attributes, by which a user describes his requirement by ordering attributes in terms of their importance for him, e.g., more important attributes are located before less important ones. In this way, user A may order the attributes as *temperature*>*humidity*>*wind*>*fertilization*>*pesticide*, which means *temperature* is the most important attribute for him, while *pesticide* is the least important attribute. Here, $a_1 > a_2$ means attribute a_1 is more important than a_2 for the user. Then, user B may order the attributes as *pesticide*>*fertilization*>*wind*>*humidity*>*temperature*, which means *pesticide* is the most important attribute for him, while *temperature* is the least important one.

Note that usually to mark the importance of attributes is to associate every attribute with a specific argument, i.e., a weight. And the higher the weight, the more important the attribute. But the essentiality of such a method is to rank the attributes in an order. Therefore we directly use the ordering method.

What has to be pointed out here is that the idea of *attribute order* is not firstly presented in this paper. It has been used in [18]. But in that paper, it serves as a necessary condition that ensures the completeness of the algorithm for reduct. However, in this paper, it serves as the specification of users' requirements.

For the second question, we propose a new algorithm that is based on the rough set theory.

The rough set theory, developed by Pawlak and his co-workers in the early 1980s, has become an important approach of knowledge discovery and data mining[19,20]. It constitutes a sound basis for data mining applications. In the rough set theory, reduct is a significant concept and attribute reduction has been one of the main thrusts in current applications[21]. For a consistent data set with decision attributes, a reduct (or relative reduct) is a minimal subset of attributes that maintains the

same capability of classification on objects as the whole set of attributes. Usually more than one reduct can be obtained from a data set. In the rough set theory, rules (knowledge) are induced from the data set after a reduct is achieved, concerning only with attributes within the reduct. So, for the ROKD problem, it is natural to find out firstly a reduct containing attributes interesting for the user and then induce rules from the reduct. For the above example about soybean, we may firstly find a reduct like {*temperature, humidity, wind*} for user A, and a reduct like {*pesticide, fertilization*} for user B, and then generate their corresponding rules. So the rules induced for user A only concern with *temperature, humidity,* and/or *wind*, and are actually the knowledge about how these factors affect the production of soybean. And for user B, the rules are about how *pesticide* and/or *fertilization* effect the production of soybean. In this way, we reach the goal of this example.

But such attribute sets as {*temperature, humidity, wind*} and {*pesticide, fertilization*} may not be reducts of data set, so the problem becomes how to find a reduct containing attributes as anterior in the attribute order as possible. This problem is not as simple as it seems to be.

For example, it is easy for an algorithm to identify reduct attributes one by one from the head of the order until all the objects are correctly classified. We check the performance of the simple algorithm with Table 1.

To view the data clearly, Table 1 is transformed to Table 2.

Table 2. Example Transformed from Table 1

a	b	c	d	e	Production
1	1	1	1	1	1
2	1	1	2	1	1
2	2	3	1	2	1
2	2	1	1	1	2
3	1	2	1	1	2
1	1	1	3	2	2
1	2	1	1	2	2

In Table 2, a, b, c, d and e correspond to *temperature, humidity, wind, fertilization and pesticide* respectively.

The discernibility matrix is {*ab, ac, de, bd, acd, ade, ce, abce, abcd, be, abde*}. Assume that user A sets the attribute order as $a > b > c > d > e$. Then attribute a, the most important for the user, is firstly selected as a reduct attribute. According to the principle of discernibility matrix, all the entries containing a can be dropped from the discernibility matrix. So the discernibility matrix becomes {*de, bd, ce, be*}. Secondly, attribute b is selected, and discernibility matrix becomes {*de, ce*}. Then, c is selected, and discernibility matrix becomes {*de*}. Finally, d is selected, and discernibility matrix becomes empty. So the current result is $\{a, b, c, d\}$.

But the result is not a reduct of the data set. It still has some redundancy. In fact, attribute a can be eliminated from the result and $\{b, c, d\}$ becomes a reduct. Moreover, none of $\{a, b, c\}$, $\{a, c, d\}$ and $\{a, b, d\}$ is a reduct. Thus, to cut down the redundancy, we have to eliminate attribute a from the result, and the final result is $\{b, c, d\}$. Then the rules induced from the data set in terms of the reduct only concern with b, c, and/or d. Here the drawback is clear, for a, the most important attribute for the user, has been missed. However, the data set actually contains a reduct $\{a, b, e\}$, which is more preferred by the user in the sense that it includes the first two important attributes of the attribute order.

This example illustrates why such a simple algorithm does not fit for ROKD problem. In fact, the algorithm fails because it cannot ensure that all the attributes kept so far construct part of a reduct. For example, in a step the algorithm kept $\{a, b, c\}$, but there is no reduct containing a, b, and c simultaneously. Therefore, to ensure the final result to be a reduct, some important attributes that are originally kept have to be moved out.

Since attribute order firstly appears in [18], where a reduction algorithm *RA-Order* is proposed, it is natural to think about using *RA-Order* for the ROKD problem. Unfortunately, *RA-Order* also meets some difficulty, because it begins to remain attribute from middle part of the attribute order, and from end to head. Therefore, some most important attributes for the user may be missed. That

is the same as what happens in the simple algorithm. For the above example, *RA-Order* also outputs $\{b, c, d\}$ as the final reduct that excludes the most important attribute a.

To sum up, both the simple algorithm and *RA-Order* do not work well in supporting attribute order as the specification of user's requirement, because they cannot ensure keeping in the result some most important attributes for the user.

In fact, the main challenge in supporting attribute order as the specification of user's requirement lies in two points. First, the algorithm should identify reduct attributes one by one from the head of the attribute order, that means the more important an attribute is, the more prior it will be included in the final result. Second, after an attribute is selected, the algorithm should ensure that the attribute will not be moved out from the final result, that implies that the current selected attributes must be part of a reduct. In this paper, we propose the reduction algorithm based on free attributes, which just has these two important features.

This algorithm identifies reduct attributes one by one from the head of the attribute order. Additionally, to ensure all the attributes currently identified being part of a reduct, the algorithm determines some non-reduct-attributes, called free attributes, at the same time. For example, given a discernibility matrix and an attribute order $S = a > b > c > d > e > f$, the algorithm first identifies a in the reduct, but at the same time, it determines f not in the reduct. Then it identifies b in the reduct, and c not in the reduct. At last it includes d in the reduct, and excludes e. So the final result is $\{a, b, d\}$.

The paper is organized as follows. In the next section, we introduce some related work including the study on preference and reduct. Sections 3, 4, and 5 are related to the theoretical foundation of the new algorithm. In Section 6 we introduce the algorithm. In Section 7 we prove the completeness of the algorithm for reduct. Section 8 is devoted to three kinds of attribute order. Section 9 shows some experimental results. Section 10 summarizes the paper with some conclusive remarks.

2 Related Work

2.1 Preference

Preference has been studied by some researchers[22–24]. In those papers, a special kind of problem, *sorting problem*, is taken into consideration. In a sorting problem, a database includes two kinds of attributes, *criteria*, (i.e., attributes with preference ordered domains (scales), like return on investment or market share), and *regular attributes*, whose domains are not preference ordered, like color or texture (If those terms are used, all the attributes mentioned in this paper are actually *regular attributes*). To deal with sorting problems, a new relation, dominance relation, substitutes the classical discernibility relation of the rough set theory. The goal is to infer the preference model hidden in the database. Those studies do not aim at ROKD problems. They obtain the same knowledge for all the users.

Another research on user preference can be found in [25]. It indicates that an algorithm usually generates a large number of rules from a database, but most of them are not interesting for the user. So it is necessary to identify those interesting rules from the total rules for the user. The user's preference for the rules is described by a kind of specification language, called *general impressions*. Then the system analyzes the discovered rules by matching them against the general impressions, and then ranks them in terms of the matching result. Finally, high-rank rules are shown to the user. In fact, the mining course is separated into two steps. In the first step, all the rules are discovered, while in the second step, interesting ones are identified for the user. [25] only concerns with the second step. As to the first step, it leaves it to the current data mining algorithms. But that separation may have a trouble when dealing with ROKD problems, as some interesting rules for a user may have been missed after the first step so that they cannot be identified in the second step. If all the interesting rules for all users have to be included after the first step, the data mining algorithm must cover the total solution space, which is usually NP-complete in time complexity so that it loses efficiency when facing large databases. Different from it, the algorithm we present directly induces interesting rules

for the current user, and, the specification of user's preference are different.

2.2 Reduction Algorithms of Rough Set Theory

Like data mining, we can classify researches on attribute reduction into two kinds, prediction and description. The former obtains reducts from a data set to make predictions on new cases, such as approximate reduct and dynamic reduct[26,27]. In such methods, the quality of classification on training data may be decreased in order to receive higher quality of classification on new objects[26]. The latter obtains reducts to make a short but accurate description of the data set. Here, the accuracy of the data, rather than new ones, plays an important role. The methods include the *RA-Order* algorithm[18], approximate algorithms based on statistics[28], and optimization algorithms for minimal reduct[29,30].

Since reduction algorithms generally find one reduct from the reduct set, there are some biases embedded in the algorithms. The biases may be accuracy of prediction[26], stability of the result[27], completeness for reduct[18], or minimal length of reduct[29,30]. Here we suggest another criterion, requirement of the user. By such a criterion, a long reduct preferred by the user is better than a short but unsatisfactory one.

3 Discernibility Function

Before the new algorithm is described, and its completeness for reduct is proved, it is necessary to study the nature of discernibility function. Because only the propositional logic is involved, the formulas below are all propositional ones.

Let $\langle U, C \cup D\rangle$ be an information system (data set), where C is the condition attribute set and D is the decision attribute set, and $U = \{x_1, x_2, \ldots, x_n\}$. Its discernibility matrix[31], denoted by Φ, is an $n \times n$ matrix defined as

$$m_{ij} = \{a \in C : a(x_i) \neq a(x_j) \wedge (d \in D, d(x_i) \neq d(x_j))\}, \quad \text{for } i, j = 1, 2, \ldots, n.$$

Write Φ as a list $\{p_1, \ldots, p_t\}$. In Φ, each p_i is called a *discernibility entry*, and is usually written as $p_i = a_{i1} \ldots a_{im}$, where each a_{ik} corresponds to a condition attribute of the information system, $k = 1, \ldots, m$, $i = 1, \ldots, t$. Furthermore, the discernibility matrix can be represented by the discernibility function f, a *conjunctive normal form* (CNF), i.e., $f = p_1 \wedge \cdots \wedge p_t$, where each $p_i = a_{i1} \vee \cdots \vee a_{im}$ is called a *clause*, and each a_{ik} is called an *atom*. Note that the discernibility function contains only atoms, but not negations of atoms.

Although the discernibility matrix and discernibility function have different styles of expression, they are actually the same in nature. So the discussion on one subject is easily extended to the other one. In this section, we discuss the discernibility function.

A clause p is said to *subsume* a clause p' if p' contains all the atoms in p. A clause p is called a *prime clause* of f if no clause of f subsumes p. A CNF is called *prime* if it only consists of prime clauses.

Applied the absorption law, a CNF is transformed to a prime CNF, which is written as

$$f = \bigwedge_{p \in PC(f)} p$$

where $PC(f)$ denotes the set of prime clauses of f.

Example 1. $f = (a \vee b \vee d) \wedge (a \vee b \vee c) \wedge (b \vee d) \wedge (c \vee e)$. Applied the absorption law, $(a \vee b \vee d)$ is absorbed by $(b \vee d)$, and f is transformed to the prime CNF $f' = (a \vee b \vee c) \wedge (b \vee d) \wedge (c \vee e)$. Each clause of f' is a prime clause.

Proposition 1. *Let f be the discernibility function of an information system in the form of CNF. Applied the absorption law, f can be transformed to a unique prime CNF.*

Since the prime CNF transformed from a CNF is unique, the above proposition is immediately proved. The proposition shows that, each information system may be expressed uniquely by a prime CNF.

In the following, we discuss a property of prime CNF. Firstly, we define a set operator "$\sim$".

Definition 1 (Set Operator "$\sim$"). *If f is a prime CNF, $\alpha \in f$ is a clause, and B is an empty set, an atom, or a conjunction of atoms, then $\alpha \sim B = \alpha - (\alpha \cap B)$.*

Proposition 2. *If f is a prime CNF, $\alpha \in f$ is a clause and $\alpha = a \vee B$, then for any $\beta \in f$, $\beta \sim B \neq \emptyset$.*

Proof. Suppose there is a clause $\gamma \in f$ such that $\gamma \sim B = \emptyset$, then $\gamma \subseteq B \subseteq \alpha$, which means γ subsumes α so that α is not prime. Then f is not a prime CNF, which contradicts the known condition. Therefore, there is no such clause as γ in f. □

Example 2. $f = (a \vee b \vee c) \wedge (b \vee d) \wedge (c \vee e)$ is a prime CNF. Consider $\alpha = a \vee b \vee c = a \vee B$, where $B = b \vee c$. Dropping B from f, we have $f' = a \wedge d \wedge e$, while $a \neq \emptyset$, $d \neq \emptyset$, and $e \neq \emptyset$.

Remark. This proposition implies that if a is identified as a reduct attribute, all the atoms in B may be dropped from f while keeping all the clauses nonempty. This is the basis of the reduction algorithm presented in this paper.

4 Prime CNF and DNF

By applying the multiplication (distributive) law, the discernibility function represented by a CNF is transformed to a *disjunctive normal form* (DNF), i.e., $f = q_1 \vee \cdots \vee q_s$, where each q_i, $i = 1, \ldots, s$, is a conjunction of atoms and called an *implicant* of f.

Then the concept of *subsume* is extended to DNF. An implicant q is said to *subsume* an implicant q' if q' contains all the atoms in q. An implicant q is called a *prime* implicant of f if no implicant of f subsumes q. A DNF is called *prime* if it only consists of prime implicants. Moreover, the set operator $\sim$ and Proposition 2 are also extendable from CNF to DNF.

By applying the absorption law, a DNF is transformed to a prime DNF, $f = q_1 \vee \cdots \vee q_k$. In fact, each q_i, $i = 1, \ldots, k$, is a reduct of the information system, and $\{q_1, \ldots, q_k\}$ is the reduct set of the given information system[31].

Denote the prime CNF of f by F, and the prime DNF of f by G. P and Q are the atom sets of F and G, respectively. A question arises, whether $P = Q$ or not? If the answer is yes, then for every atom in F, there must exist a prime implicant (reduct) in G including the atom. That means, to compute a reduct, we may start from any clause in F, and select one of its atoms as a reduct attribute. Furthermore, we may select the first one in the attribute order as a reduct attribute. This is why we base our discussion on prime CNF of a discernibility matrix, but not on the discernibility matrix itself. Before answering the above question, we first introduce *atom order* (*attribute order*).

Given a CNF f with atom set C, the *atom order*, denoted by S, is an order over C. Given an atom order S, each clause of f may be rearranged according to S so that the atoms in the clause appear in the same order with S. For example, $f = (a \vee c \vee b) \wedge (d \vee b) \wedge (c \vee e)$, $C = \{a, b, c, d, e\}$, and $S = a > b > c > d > e$. Rearrange f according to S, $f = (a \vee b \vee c) \wedge (b \vee d) \wedge (c \vee e)$. After the rearrangement, each clause is said to *satisfy* S.

When each clause satisfies S, we define an equivalence relation $L(S)$ over f:

$$\{\alpha : \alpha \in f, \text{ where } \alpha \text{ takes } a_k \text{ as the first atom, i.e., } \alpha = a_k B\}$$

where $a_k \in C$ and $B \subset C$. By $L(S)$, f is partitioned into equivalence classes, i.e., $f/L(S) = \{[a_1], \ldots, [a_n]\} = \{[1], \ldots, [n]\}$, where a_i locates before a_{i+1} in S, $i = 1, \ldots, n-1$. Consider $\emptyset \neq [k] \in f/L(S)$. Select $\alpha \in [k]$ and $\alpha = a_k \vee B$, then for every $a_p \in B$, a_p is located behind a_k in S. a_k is called the *labeled atom* of $[k]$. Sometimes we write $\alpha = a_k \vee B$ as $\alpha = a_k B$.

The following proposition is about clauses in the equivalence class $[k]$.

Proposition 3. *Given a CNF f and an atom order S, $[k] \in f/L(S)$ with labeled atom a_k, and $[k] = \{\alpha_1, \ldots, \alpha_n\}$. Transform $\alpha_1 \wedge \cdots \wedge \alpha_n$ to DNF, denoted the result by Ω_k. There must exist an implicant $\{a_k\}$ in Ω_k.*

Remark. This proposition suggests that, $\{a_k\}$ is an implicant of Ω_k. Then, if the absorption law is applied to Ω_k, all the implicants in Ω_k containing a_k will be absorbed. That means, in the prime

DNF of Ω_k, there must exist a prime implicant $\{a_k\}$, and all the other prime implicants exclude the atom a_k. Especially, if there is just one clause in $[k]$, for example, $a \vee b \vee c$, then all the atoms in the clause, a, b, and c, are implicants in Ω_k. In addition, the proposition must still hold for a DNF.

Example 3. $f = (a \vee d) \wedge (a \vee e) \wedge (b \vee c)$ is a CNF and $S = a > b > c > d > e$. $[a] = [1] = \{(a \vee d), (a \vee e)\}$. Transforming $[a]$ to DNF, we get $\Omega_1 = a \vee (a \wedge d) \vee (a \wedge e) \vee (d \wedge e)$. Then $\{a\}$ is an implicant of Ω_1. Applying the absorption law to Ω_1, $\Omega_1 = a \vee (d \wedge e)$. $\{a\}$ is a prime implicant of Ω_1 and $(d \wedge e)$ excludes the atom a.

In fact, what we are looking for is an algorithm that can obtain a reduct including those attributes the user preferred. According to Proposition 3, for each $[k]$, if its labeled atom, a_k, is selected as an reduct attribute, all the clauses in $[k]$ can be dropped from the CNF of the discernibility function.

Let F be the prime CNF of discernibility function f, and G be the prime DNF of f. The atom sets of F and G are both subsets of C, and are arranged according to the atom order S.

Consider $\alpha \in [k]$, $\alpha = a_k B$. Let $F' = F - [k]$, $G' = \{\theta_1 \vee \cdots \vee \theta_q\}$ be the prime DNF of F'. For the convenience of discussion, move away the subscript k and write $\alpha = a_k B$ as $\alpha = aB$. Applying the multiplication law to $\alpha \wedge G'$, we have the following formula.

$$\alpha \wedge G' = \{(a_1 \wedge \theta_1) \vee \cdots \vee (a \wedge \theta_q)\} \vee \{(B \wedge \theta_1) \vee \cdots \vee (B \wedge \theta_q)\}$$

Obviously, all the $(a \wedge \theta_i)$, $i = 1, \ldots, q$, are implicants of $\alpha \wedge G'$. Then two questions arise: (1) does every $(a \wedge \theta_i)$ definitely keep prime in $\alpha \wedge G'$? Clearly, the answer is no. (2) is there any $(a \wedge \theta_i)$ keeping prime in $\alpha \wedge G'$? This is an important question that has to be answered before the algorithm is designed. If the answer is no, it is not possible for a to work as a reduct attribute, so that any algorithm should not identify a as a reduct attribute. The following proposition answers this question.

Proposition 4. *$(a \wedge \theta_m)$ is a prime implicant of $\alpha \wedge G'$ if and only if for all $b \in B \subset \alpha$, $b \notin \theta_m$ holds, where $1 \leq m \leq q$.*

Proof. By the definition of G', $a \notin \theta_m$ holds.

($\Leftarrow$) Assume that $(a \wedge \theta_m)$ is a prime implicant of $\alpha \wedge G'$. Suppose there is $b \in B \subset \alpha$ and $b \in \theta_m$. Since $b \in \theta_m$ and $a \notin \theta_m$, we have $b \wedge \theta_m = \theta_m$ and $(b \wedge \theta_m) \subset (a \wedge \theta_m)$, that is, $(a \wedge \theta_m)$ can be absorbed by $(b \wedge \theta_m)$ (or $(B \wedge \theta_m)$). Thus, $(a \wedge \theta_m)$ is not a prime implicant of $\alpha \wedge G'$, which contradicts the assumption.

($\Rightarrow$) Assume that for all $b \in B \subset \alpha$, $b \notin \theta_m$ holds. Since $b \notin \theta_m$ and $a \notin \theta_m$, and $a \neq b$, we have $(a \wedge \theta_m) \not\subset (b \wedge \theta_m)$ and $(b \wedge \theta_m) \not\subset (a \wedge \theta_m)$. Thus, $(a \wedge \theta_m)$ cannot be absorbed by any $(b \wedge \theta_m)$ (or $(B \wedge \theta_m)$). And obviously, $(a \wedge \theta_m)$ cannot be absorbed by any other $(B \wedge \theta_t)$ either, $1 \leq t \leq q$, $t \neq m$. Therefore, $(a \wedge \theta_m)$ is prime in $\alpha \wedge G'$. □

Remark. Since only multiplication and absorption laws are employed in the above proof, the proposition must hold for DNF, that is, we can prove in the same way that the proposition holds when transforming a DNF to CNF.

Proposition 4 implies an algorithm that is complete for reduct.

Example 4. $F = (a \vee b \vee c) \wedge (b \vee d) \wedge (c \vee e)$ is a prime CNF and $S = a > b > c > d > e$. $[a] = \{(a \vee b \vee c)\}$. $F' = F - [a] = (b \vee d) \wedge (c \vee e)$. $\alpha = a \vee b \vee c = a \vee B$ and $B = b \vee c$. Applying multiplication and absorption laws to $(b \vee d) \wedge (c \vee e)$, we have its prime DNF $G' = (b \wedge c) \vee (b \wedge e) \vee (d \wedge c) \vee (d \wedge e)$. Then applying the multiplication law to $(a \vee b \vee c) \wedge G'$, we have

$$a \wedge G' = (a \wedge b \wedge c) \vee (a \wedge b \wedge e) \vee (a \wedge d \wedge c) \vee (a \wedge d \wedge e),$$
$$b \wedge G' = (b \wedge b \wedge c) \vee (b \wedge b \wedge e) \vee (b \wedge d \wedge c) \vee (b \wedge d \wedge e),$$
$$c \wedge G' = (c \wedge b \wedge c) \vee (c \wedge b \wedge e) \vee (c \wedge d \wedge c) \vee (c \wedge d \wedge e).$$

The prime DNF of $(a \vee b \vee c) \wedge G'$ is $(a \wedge d \wedge e) \vee (b \wedge c) \vee (b \wedge e) \vee (d \wedge c)$.

Note that in G', only $(d \wedge e)$ does not contain b or c. Therefore, $(a \wedge d \wedge e)$ is a prime implicant of $\alpha \wedge G'$, and none of $(a \wedge b \wedge c)$, $(a \wedge b \wedge e)$, $(a \wedge d \wedge c)$ is. The result validates Proposition 4.

Now, we can answer the question, whether $P = Q$ or not, put forward at the beginning of this section.

Let F be a prime CNF and S be an atom order. Let $[k] \in F/L(S)$, $\alpha \in [k]$, and $\alpha = aB$. Applied multiplication and absorption laws, F is transformed to prime DNF G. P and Q are atom sets of F and G, respectively.

Proposition 5. $P = Q$.

Proof. First, we prove that $P \subseteq Q$ holds.

Since $\alpha = aB \in F$, $a \in P$. For any $\gamma \in F$, by Proposition 2, $\gamma \sim B \neq \emptyset$ holds. Applied multiplication and absorption laws to F, there must exist an implicant $\beta = a \wedge A$ in G, such that $A \cap B = \emptyset$. Then according to Proposition 4, $\beta = a \wedge A$ is a prime implicant of G, i.e., $a \in Q$. Because of the liberty of α, for any $a \in P$, we have $a \in Q$. Then $P \subseteq Q$ holds.

Because Propositions 2, 3, 4 hold for CNF and DNF, in the same way, we can prove $Q \subseteq P$.

$P = Q$ holds. □

This proposition implies that any atom in a prime CNF must be included in an implicant of a prime DNF. Since the discernibility function of an information system can be expressed in the form of prime CNF, and the prime DNF is just the reduct set, for any atom of the prime CNF, there must exist a reduct containing that atom. This is the necessary condition of the algorithm presented in this paper.

5 Reduct

Reduct is one of the most important contributions of Pawlak to machine learning theory. It can be regarded as a goal between minimal reducts and trivial reducts, with the computational complexity of $O(n^2)$. This concept is originally defined on the positive region of the information system (low approximation) ①.

Definition 2 (Reduct, Pawlak). *Given an information system $\langle U, C \cup D\rangle$, and $R \subseteq C$. If $POS_R(D) = POS_C(D)$ and for any $r \in R$, $POS_R(D) \neq POS_{R-\{r\}}(D)$, then R is a reduct of $\langle U, C \cup D\rangle$.*

Roughly speaking, for a subset of condition attributes, the positive region is the maximal subset of the universe U, in which all the objects are consistent with each other. To make it clear, we give a direct explanation following Pawlak's definition.

Given an information system $\langle U, C \cup D\rangle$, $R \subseteq C$ and $E \subseteq U$. For R, $U - E$ is the positive region of the information system, if and only if for all x, $y \in U - E$, if $R(x) = R(y)$, then $D(x) = D(y)$, and for any $z \in E$, there exists a $z' \in E$, such that $R(z) = R(z')$ but $D(z) \neq D(z')$. And, $\{y : R(z) = R(y)$ but $D(z) \neq D(y)$, $z \in E$ and $y \in U - E\} = \emptyset$.

Remark. In fact, the original definition of positive region given by Pawlak has considered the inconsistent case of an information system. Although it is significant for prediction tasks, it loses essentiality in description ones, because we only need to consider the case $\mathrm{POS}_C(D) = U$ for reduction. If the given information system is inconsistent, according to Pawlak's definition, the reduction is accomplished on the positive region for decision attribute D. That is, the reduct on $\langle U, C \cup D\rangle$ is replaced by the reduct on $\langle U - E, C \cup D\rangle$.

By this remark, reduct may be defined in another way.

Definition 3 (Reduct). *Given an information system $\langle U, C \cup D\rangle$, $R \subseteq C$ is a reduct, if and only if* (1) *for every x, $y \in U$ with $D(x) \neq D(y)$, $R(x) \neq R(y)$ holds and* (2) *for any $r \in R$, $P = R - \{r\}$, there exist x, $y \in U$, such that $D(x) \neq D(y)$, $R(x) \neq R(y)$, but $P(x) = P(y)$.*

Remark. Definition 3 is closely related to the discernibility matrix.

In the principle of discernibility matrix, reduct is related to core attributes.

Proposition 6. *R is a reduct of $\langle U, C \cup D\rangle$ if and only if R is the core attribute set of $\langle U, R \cup D\rangle$.*

① In Pawlak's papers, reduct is defined with and without decision attributes, respectively. Since we can transform the latter case to the former case, it is only necessary to consider reducts in an information system with decision attributes, which is based on the positive region (low approximation).

Remark. By this proposition, the information system $\langle U, \{R-\{r\}\} \cup D\rangle$ is constructed by moving out a core attribute r from $\langle U, R \cup D\rangle$, which must cause a change of the positive region.

In this paper, we use another definition of reduct that is based on the principle of discernibility matrix. The new definition is equivalent to Definition 3.

Let Φ be the discernibility matrix of the information system.

Definition 4. *Given an information system $\langle U, C \cup D\rangle$, and $R \subseteq C$. For $r \in R$, if there is an $\alpha \in \Phi$, such that $R \cap \alpha = r$, then r is said to be independent in R. R is a reduct of the information system, if and only if* (1) *$R \cap \alpha \neq \emptyset$ for any $\alpha \in \Phi$ and* (2) *any $r \in R$ is independent in R.*

Proposition 7. *Given an information system $\langle U, C \cup D\rangle$ and $POS_C(D) = U$, Definition 4 is equivalent to Definition 3.*

Proof. Let $R \subseteq C$, $r \in R$ and $P = R - \{r\}$.

By Definition 3, (1) for every x, $y \in U$ with $D(x) \neq D(y)$, $R(x) \neq R(y)$ holds. Then α, the discernibility element of x and y, satisfies $R \cap \alpha \neq \emptyset$. (2) For any $r \in R$, $P = R - \{r\}$, there exist x, $y \in U$ such that $D(x) \neq D(y)$, $R(x) \neq R(y)$ but $P(x) = P(y)$. Hence, by the principle of discernibility matrix, α satisfies $r \in \alpha$ but $P \notin \alpha$. Then $R \cap \alpha = r$. Thus, the two conditions of Definition 3 correspond to those of Definition 4 respectively.

By Definition 4, (1) $R \cap \alpha \neq \emptyset$ for any $\alpha \in \Phi$, i.e., for every x, $y \in U$ with $D(x) \neq D(y)$, there is some difference between them on a subset of R, i.e., $R(x) \neq R(y)$. (2) For any $r \in R$, there is an $\alpha \in \Phi$, such that $R \cap \alpha = r$. That is, for any $r \in R$, there is an $\alpha \in \Phi$, such that $R \cap \alpha \neq \emptyset$ and $P \cap \alpha = \emptyset$. By the principle of discernibility matrix, there exist x, $y \in U$ satisfying $D(x) \neq D(y)$ and $R(x) \neq R(y)$, but $P(x) = P(y)$. That is just Definition 3. □

We close this section by defining free attributes. In fact, the algorithm we will present tries to drop non-reduct-attributes from the discernibility matrix. That is different from traditional algorithms, where it is the reduct attributes that would be dropped. The non-reduct-attributes are called free attributes.

There are two sorts of free attributes. First, free attributes for the information system, which are free for all the reducts. Second, free attributes for a reduct, which are free just for that reduct.

Given an information system $\langle U, C \cup D\rangle$, by Proposition 5, the attribute set of reduct set (prime DNF) is the atom set of prime CNF of its discernibility function, denoted by P.

Definition 5 (Free Attribute). *For any $a \in C$, if $a \notin P$, it is called free attribute of the information system. Let $R \subseteq C$ be a reduct. Any $b \in C - R$ is called free attribute of the reduct R.*

6 The Algorithm Based on Free Attributes

Some reduction algorithms based on the principle of discernibility matrix take the following reduction rule.

Proposition 8 (Reduction Rule Based on Discernibility Matrix). *Let Φ be the discernibility matrix of an information system. If $\emptyset \neq R \subseteq C$ is a reduct, then $\alpha \cap R \neq \emptyset$ holds for any $\alpha \in \Phi$.*

The reduction method implied by Proposition 8 is based on dropping entries from the discernibility matrix. Although various strategies may be embedded in the above rule, it has been proved that not all the strategies are complete for reduct[18]. Also in [18], a strategy based on attribute order is proposed, and the corresponding algorithm is proved to be complete for reduct. However, since the strategy replies on the nonempty equivalence class with maximum subscript in F/L(S), it meets some difficulty in supporting user requirement.

The algorithm to be presented in this paper adds another reduction rule to Proposition 8, which in fact identifies the free attributes.

Proposition 9 (Reduction Rule Based on Free Attributes). *Let Φ be the discernibility matrix. For a reduct $\emptyset \neq R \subseteq C$, there must exist $\Phi' \subseteq \Phi$, such that for any $\alpha \in \Phi'$, where $\alpha = aB$, $\cup_{a \in \alpha} a = R$ and $B \cap R = \emptyset$ must hold.*

Proof. Let $\emptyset \neq R \subseteq C$ be a reduct. By Definition 4, for any $a \in R$, there is an $\alpha \in \Phi$ satisfying $R \cap \alpha = a$. Written α as $\alpha = aB$, $B \cap R = \emptyset$ holds. Construct Φ' by including all these α. Then

$\cup_{a\in\alpha} a = R$ holds. □

Proposition 9 extends the reduction rule in Proposition 8. The key point lies in the selection of B. Since the attributes in B are free for R, they can be dropped from discernibility matrix before the reduction rule in Proposition 8 is applied.

Example 5. $\Phi = \{abc, ab, ad, bd, be, ce, cf\}$ and $R = \{a, b, c\}$.

$\Phi' = \{ad, bd, ce\}$. Note that $\{d, e\} \cap R = \emptyset$.

Let F be the CNF of the discernibility function, S be the given attribute order and $R = \emptyset$.

Algorithm 1. (Reduction Algorithm Based on Free Attributes)

(1) If $F = \emptyset$, stop.
(2) Apply the absorption law to F and transform it to prime CNF.
(3) $F/L(S) = \{[a_1], \ldots, [a_n]\}$. $R = R \cup \{a_1\}$. Select an $\alpha \in [a_1]$, $\alpha = a_1 B$.
(4) $F = F - [a_1]$.
(5) $\forall \alpha \in F$, $\alpha = \alpha \sim B$.
(6) Goto (1).

When the algorithm stops, R is a reduct.

Example 6. $F = (a \vee b \vee f) \wedge (a \vee c \vee f) \wedge (a \vee c) \wedge (a \vee d) \wedge (b \vee d) \wedge (b \vee e) \wedge (c \vee e) \wedge (c \vee f)$. $S = a > b > c > d > e > f$.

To have a clear view, write F as $\Phi = \{abf, acf, ac, ad, bd, be, ce, cf\}$, and replace using the algorithm on F with using it on Φ. Both the process and result are the same.

(1) Since $\Phi \neq \emptyset$, go to (2).
(2) Apply the absorption law to Φ. acf is absorbed by ac. So $\Phi = \{abf, ac, ad, bd, be, ce, cf\}$.
(3) $\Phi/L(S) = \{[a], [b], [c]\}$. $[a] = \{abf, ac, ad\}$, $[b] = \{bd, be\}$, $[c] = \{ce, cf\}$. $R = \{a\}$. Select $\alpha = abf$, $\alpha = aB$, where $B = \{b, f\}$.
(4) $\Phi = \Phi - [a] = \{bd, be, ce, cf\}$.
(5) Drop $B = \{b, f\}$ from Φ. $\Phi = \{d, e, ce, c\}$.
(6) Goto (1).
(7) Since $\Phi \neq \emptyset$, go to (2).
(8) Apply the absorption law to Φ. ce is absorbed by c. So $\Phi = \{d, e, c\}$.
(9) $\Phi/L(S) = \{[c], [d], [e]\}$.
...

When the algorithm stops, $R = \{a, c, d, e\}$ is a reduct.

Note that in Step (3), since $[a_1]$ usually contains more than one entry, how to select $\alpha \in [a_1]$ is a strategic problem that must be considered carefully. Different strategies usually lead to different result. For the above example, $[a] = \{abf, ac, ad\}$. If $\alpha = abf$, the result is $\{a, c, d, e\}$; if $\alpha = ac$, the result is $\{a, b, e, f\}$; if $\alpha = ad$, the result is $\{a, b, c\}$. Obviously $\{a, b, c\}$ is the best reduct for the user because it contains the first three attributes in S. The details of strategies will be discussed in Section 8.

The algorithm is based on forming the discernibility matrix of an information system that has a time complexity of $O(n^2)$, where n is the number of objects in the information system. As to the algorithm itself, since the absorption law is used, the time complexity becomes $O(Card(\Phi)^2)$, where $Card(\Phi)$ is the entry number of discernibility matrix.

7 The Completeness of Algorithm 1 for Reduct

To prove that the algorithm based on free attributes is complete for reduct, we need the following proposition that is proposed by Skowron[31].

Proposition 10 (Reduct Set). *The prime DNF of the information system is its reduct set. Each implicant of the prime DNF is a reduct.*

By this proposition, to prove the algorithm's completeness for reduct, it only needs to prove that the output of the algorithm is one of the implicants of the prime DNF of the information system.

Proposition 10 has a useful corollary, which may be regarded as another definition of reduct.

Corollary 1 (Reduct). *For an information system, F is the prime CNF of the discernibility function, and G is the DNF of F. For $\beta \in G$, β is a reduct, if and only if for any $\gamma \in G$, $\gamma \not\subset \beta$ holds.*

Since the algorithm based on free attributes identifies one reduct from the reduct set, it will be complete for reduct if its output is actually one β in Corollary 1 for every information system.

By Proposition 1, F is unique for the given information system. Let S be an attribute order, and $[k] \in F/L(S)$, whose labeled attribute is 'a'. $\alpha = aB \in F$, $\beta \in G$. We have the following statements prepared for the proof of the completeness of the algorithm..

(1) By Proposition 5, $a \in \beta$. There must be an implicant in G that contains attribute a (Step 3).

(2) By Proposition 3, if a is selected as a reduct attribute, all the clauses in $[k]$ may be dropped (Step 4).

(3) By Proposition 2, select a clause $\alpha = aB$, for any $\gamma \in F$, $\gamma \sim B \neq \emptyset$. We get the new CNF H (Step 5).

(4) Assume $\delta, \phi \in H$, $\delta \subseteq \phi$, that is, $\phi = \delta \vee K$. Then $\alpha \wedge \delta \wedge \phi = \alpha \wedge \delta \wedge (\delta \vee K) = \alpha \wedge \delta$ (Step 2).

Let R be the output of the algorithm. It must be an implicant of the DNF of F, but may not be prime. To prove the completeness of the algorithm for reduct, we must prove that R is a prime implicant.

Proposition 11. *The algorithm based on free attributes is complete for reduct.*

Proof. Let R be the output of the algorithm.

Let $\alpha \in F$, $\alpha = aB$. Drop B from $F-[a]$, and then transform it to prime DNF G'. $G' = \theta_1 \vee \cdots \vee \theta_k$. Applying the multiplication law to $\alpha \wedge G'$, we have

$$\alpha \wedge G' = (a \wedge \theta_1) \vee \cdots \vee (a \wedge \theta_k) \vee (B \wedge \theta_1) \vee \cdots \vee (B \wedge \theta_k).$$

Consider $\theta_m \in G'$. By Proposition 4, $(a \wedge \theta_m)$ is a prime implicant of $\alpha \wedge G'$ if and only if $b \notin \theta_m$ for all $b \in B$. Since B has been dropped before G' is formed, that condition is satisfied. By Proposition 2, for any $\gamma \in F$, we have $\gamma \sim B \neq \emptyset$. Therefore, $G' \neq \emptyset$ holds, so that there must exist an θ_m. Thus, the output of the algorithm, $R = a \wedge \theta_m$ is a prime implicant of $\alpha \wedge G'$. According to Proposition 10, R is a reduct. □

8 Three Kinds of Attribute Orders

In the above sections, the attribute order is supposed to have a unique form, $S = a_1 > a_2 > \cdots > a_n$, which implies that the user can distinguish each couple of attributes clearly in terms of their importance for him. But that is a rather strict condition, which may make a user feel difficult when, for example, some attributes are equally important for him.

An alternative is to develop different forms of attribute orders for different situations. In this section, three kinds of attribute orders are developed, i.e., total order, group order and balance order. In addition, a problem still remains in Algorithm 1, that is, how to identify the particular α from $[a_1]$ at Step (3). Since that strategy closely concerns with user requirement, it will be discussed within the three kinds of attribute orders, respectively. In the following, the α is called *free entry*, for all its attributes will become free attributes except the label one.

In this section, the discussion is made on the discernibility matrix.

8.1 Total Order

The total order over the attribute set C is denoted by $S = a_1 > a_2 > \cdots > a_n$. A user sets a total order when he can distinguish each couple of attributes by their importance for him. Below, we discuss the strategy of identifying the free entry. The idea is to make the free attributes contained in the free entry as posterior in S as possible.

Let c be the current reduct attribute. $\beta \in [c]$. Let $r(\beta, c)$ be the position of c in β, $|\beta|$ be the number of attributes of β. Let $Apos(\beta, i)$ be the position of the i-th attribute of β in S. Then $\min_{i=1,\ldots,|\beta|, i \neq r(\beta,c)}(Apos(\beta, i))$ represents the position of the most anterior attribute of β except c

in S, which is called *secondary attribute* of β. Obviously, the more posterior the position is, the more unimportant the free attributes implied by β are. Therefore, among all $\beta \in [c]$, the one with $\max_{\beta\in[c]}(\min_{i=1,\ldots,|\beta|,i\neq r(\beta,c)}(Apos(\beta,i)))$ should be selected as the free entry. The attribute taking position $\max_{\beta\in[c]}(\min_{i=1,\ldots,|\beta|,i\neq r(\beta,c)}(Apos(\beta,i)))$ in S is called the *maximal secondary attribute* of $[c]$.

Example 7. $\Phi = \{ab, ac, adf, be, cf, dg\}$. $S = a > b > c > d > e > f > g$.

The current reduct attribute is a. $[a] = \{ab, ac, adf\}$.

For $\beta = ab$, $\min(Apos(ab,i)) = \min(Apos(ab,2)) = 2$. (The position of b in S is 2)

For $\beta = ac$, $\min(Apos(ac,i)) = \min(Apos(ac,2)) = 3$.

For $\beta = adf$, $\min(Apos(adf,i)) = \min(Apos(adf,2), Apos(adf,3)) = 4$.

Thus, $\max(\min(Apos(\beta,i))) = \max(2,3,4) = 4$. Therefore, $\beta = adf$ is selected as the free entry, and d, f become free attributes. The final reduct is $\{a, b, c, g\}$.

Consider the example about soybean production mentioned in Section 1. $\Phi = \{ab, ac, de, bd, acd, ade, ce, abce, abcd, be, abde\}$. For user A, $S = a > b > c > d > e$. The final reduct is $\{a, b, e\}$. For user B, $S = e > d > c > b > a$. The final reduct is $\{e, d, a\}$. Both of them include the first two attributes in their corresponding S.

Sometimes more than one $\beta \in [c]$ takes the maximal secondary attribute of $[c]$ as its own secondary attribute. For instance, $[c] = \{cde, cdf\}$ and $S = c > d > e > f$. Here, d is the maximal secondary attribute of $[c]$, and both *cde* and *cdf* take d as the secondary attribute. In such a case, the Max-Min strategy should be refined to identify the free entry of $[c]$. A simple supplementation may compare their third attribute, and then the forth attribute, ..., until the end of one entry. Since the absorption law has been used, not any two entries of $[c]$ are exactly the same. So there must be a time when the difference between the two entries appears. For example, in the above case, *cdf* will be selected as the free entry since its third attribute, f, has a more posterior position in S than e, the third attribute of *cde*.

8.2 Group Order

Sometimes it is difficult for a user to distinguish clearly among all the attributes, for some of them are equally important in his mind. In such cases, the user may use a group order instead of a total order.

For a group order, all the attributes are divided into some groups, $A_1, \ldots, A_k$, where $A_1 \cup \cdots \cup A_k = C$, $A_i \cap A_j = \emptyset$, $i, j = 1, \ldots, k$, $i \neq j$. The total order relation is satisfied among the groups, i.e., $A_1 > \cdots > A_k$, which means all the attributes in A_i are more important for the user than those in A_{i+1}, $i = 1, \ldots, k-1$. While in each group, all the attributes are equally important.

The strategy of identifying the free entry under a group order shares the same principle as in total order cases. It also tries to choose free attributes as posterior in S as possible, so that the Max-Min principle still works in the group order. However, one of the differences between the two kinds of orders is that no matter how to refine the strategy, sometimes the group order cannot identify a single free entry of $[c]$. See a simple example, $\Phi = \{ab, ac, bc\}$, $S = \{a\} > \{b, c\}$. Since b and c stay in the same group, they have the identical position in S. Therefore, in $[a]$, ab and ac are indistinguishable, and the free entry cannot be determined directly. To solve it, a simple solution constructs artificially a quasi-total-order inside a group, for instance, following the alphabetic order, which implies that b is more important than c and ac should be selected as the free entry in this example.

8.3 Balance Order

A balance order is developed when the following real case is considered.

Human development database (HDD) is a large database including hundreds kinds of social, economic and natural data. Many researchers study HDD to get knowledge about the sustainability of a country. Because *the sustainability of a country is connected with not only population quality and economic level, but also social and economic structure, resource and environmental state, and*

economic and ecological function, it requires keeping balance in all the social, economic and environmental aspects[32]. This raises a special kind of user requirement. To meet the requirement, all the attributes are divided into some groups, and the algorithm is invoked to find a reduct containing attributes distributed evenly among all the groups.

Here we define the balance order. All the attributes are divided into some groups, $A_1, \ldots, A_k$, where $A_1 \cup \cdots \cup A_k = C$, $A_i \cap A_j = \emptyset$, $i, j = 1, \ldots, k$, $i \neq j$. All the groups are at the same level of importance, while inside each group, the total order relation is satisfied, i.e., all the attributes inside a group are totally ordered.

Under the balance order, a strategy is designed following the rules:

1) Select reduct attributes in the following way. At first, all the groups form a candidate set. Then the minimal group contributes a reduct attribute. Since all the attributes in each group are totally ordered, the first attribute in the minimal group is identified as the current reduct attribute. Then that group quits the candidate set. In this way, every group contributes a reduct attribute and quits the candidate set in turn. This process continues until the candidate set becomes empty. If the discernibility matrix is not empty, all the groups form the candidate set again except the empty ones, and the above steps repeat.

2) Select free attributes in the free entry under the following principles. Let c be the current reduct attribute and denote its group by A_c.

a) If two attributes a and b, $a \in A_c$, $b \notin A_c$, select a as the free attribute.

b) If two attributes a and b, $a, b \in A$ and $a > b$ in the order, select b as the free attribute.

c) If two attributes belong to different groups and have the same position in their own order, select the attribute within the larger group.

The above principles consider some simple cases of identifying free entry. In more complex situations, some weights of attributes have to be introduced.

Let c be the current reduct attribute. Denote the number of attributes in A_i by m_i. $m = \sum_{i=1}^{k} m_i$. $m_{\max} = \max\limits_{i=1,\ldots,k}(m_i)$. For an attribute b having position q in the order of the l-th group,

$$w_1 = \begin{cases} 1, & \text{if } b \in A_c \\ 0.5, & \text{otherwise} \end{cases}$$

$$w_2 = q/m_{\max}$$

$$w_3 = m_l/m_{\max}$$

$$w_b = w_1 * w_2 * w_3$$

For $\beta \in [c]$, $\beta = cB$, the weight of β, $w_\beta = \sum_{b \in B} w_b$. Then the β with $\max\limits_{\beta \in [c]}(w_\beta)$ is identified as the free entry.

Example 8. $\Phi = \{ab, af, acg, aeg, agh, bef, beg, cefg, efgh\}$. $S = \{a > c\}\{b > g\}\{e > f > h\}$. In terms of S, there are three groups, $A_1 = \{a > c\}$, $A_2 = \{b > g\}$, $A_3 = \{e > f > h\}$. $m_1 = 2$. $m_2 = 2$. $m_3 = 3$. $m_{\max} = \max(m_1, m_2, m_3) = 3$.

A_1 and A_2 are the minimal groups. Select attribute a in A_1 as reduct attribute. Then select the free entry of $[a]$.

For ab, $w_{ab} = w_b = w_1 * w_2 * w_3 = 0.5 * (1/3) * (2/3) = 2/18$

For af, $w_{af} = w_f = 0.5 * (2/3) * (3/3) = 6/18$

For acg, $w_{acg} = w_c + w_g = 1 * (2/3) * (2/3) + 0.5 * (2/3) * (2/3) = 12/18$

For aeg, $w_{aeg} = w_e + w_g = 0.5 * (1/3) * (3/3) + 0.5 * (2/3) * (2/3) = 7/18$

For agh, $w_{agh} = w_g + w_h = 0.5 * (2/3) * (2/3) + 0.5 * (3/3) * (3/3) = 13/18$

Because agh has the biggest weight, it is identified as the free entry, while g and h become free attributes. The final reduct is $\{a, b, f\}$. It selects one attribute from each group, and among the attributes, a and b are both the first one in their own group, while f is the second one in its group. In fact, that is the best result under the balance order, for $\{a, b, e\}$, consisting of the first attribute of each group, is not a reduct of Φ.

Note that the above weight computing method is an approximate strategy to keep balance among the attribute groups. We cannot ensure to remain in the result even one attribute from each group. First, such a reduct may not exist. Second, even if there is such a reduct, it may be missed because the strategy is actually heuristic rather than deterministic.

9 Experimental Results

In this section, we present three experiments to test the performance of Algorithm 1. The experiments are taken on a computer with AMD 1GHz processor and 256M memory.

Example 9. The experiment is taken on DATA1, which is a data set made artificially and has 10,000 records, 20 condition attributes, and one decision attribute. The attributes are labeled by their index, i.e., 1,2, ..., 21. Among them, attribute 1 is the decision attribute, and 2, ..., 21 are condition ones. The discernibility matrix contains 131,048 different entries. No core exists in DATA1. The following table shows the results of nine experiments taken on DATA1 under different attribute orders. Each experiment takes about 76 seconds (The time is that of Algorithm 1 but does not include that of generating the discernibility matrix).

Table 3. Experiments Taken on the DATA1

Case	Attribute order	Results
1	$2,3,\ldots,21$	$2,3,4,5,6,7,8,9$
2	$21,20,\ldots,2$	$21,20,19,18,17,16,15,14$
3	$3,5,\ldots,21,2,4,\ldots,20$	$3,5,7,9,13,15,17,4$
4	$2,4,\ldots,20,1,3,\ldots,19$	$2,4,6,8,10,12,14,16$
5	$3,8,7,12,15,2,18,14,6,10,11,21,5,17,4,16,9,13,19,20$	$3,8,7,12,15,18,14$
6	$13,3,18,16,7,5,14,9,19,17,11,8,21,4,15,6,20,2,12,10$	$13,3,18,16,7,5,14,9$
7	$14,10,17,2,16,3,7,8,18,4,19,6,21,13,12,15,5,9,20,11$	$14,10,17,2,16,3,7,9$
8	$10,3,8,7,13,17,19,12,4,20,15,16,11,18,6,21,14,5,2,9$	$10,3,8,7,13,17,20$
9	$8,7,10,13,14,4,15,21,11,16,12,3,19,17,18,20,6,9,2,5$	$8,7,10,13,14,4,21,17$

In the attribute order, the symbol ">" is omitted and $a_1,\ldots,a_n$ means $a_1 > \cdots > a_n$. In the above nine experiments, the reducts found by the algorithm are rather satisfactory. For example, in case 1, the reduct $\{2,3,4,5,6,7,8,9\}$ remains the first eight attributes of the given order. And that also happens in case 2 and case 4. In fact, the algorithm has a good property, that is, if the first k attributes in the order happen to make up a reduct, the algorithm surely finds that reduct. In case 5, the result contains the first eight attributes in the order except the sixth attribute 2. And in other cases, most attributes in the final result take anterior positions in the attribute order.

Example 10. The experiment is taken on DATA2, which is also made artificially and has the same number of attributes with DATA1 but ten times of the records in DATA1, i.e., 100,000 records. The discernibility matrix contains 411,943 different entries, which is three times more than that of DATA1. The experiments take about 216.96 seconds. Since in this experiment we only care about how the running time goes up when the data set expands, no details of experiment results are shown.

Table 4. Experiments Taken on Data Set of Cars

Case	Attribute order	Results
1	$5,3,2,8,7,6,4,9,1$	$5,3,2,8,6,4,9$
2	$2,4,9,3,1,5,8,6,7$	$2,4,9,3,1,8$
3	$4,8,5,9,2,1,3,6,7$	$4,8,5,9,2,3,6$
4	$7,5,9,6,3,4,1,8,2$	$7,9,6,4,2$
5	$4,1,5,7,3,2,8,6,9$	$4,1,5,9$
6	$1,7,6,2,4,9,3,5,8$	$1,7,6,4,9$
7	$1,3,2,9,4,7,8,6,5$	$1,3,2,9,4,8$
8	$4,9,7,5,6,2,3,8,1$	$4,9,7,6,2$
9	$8,7,1,9,2,6,5,3,4$	$8,1,9,2,3,4$
10	$3,4,1,9,5,2,8,7,6$	$3,4,1,9,2,8$

Example 11. The experiment is taken on the data set of cars[33], which includes nine condition attributes, one decision attribute, and 21 records. The attributes are labeled by their index, i.e., 1,2,

..., 10. Attribute 10 is the decision attribute, and $1, \ldots, 9$ are condition ones, among which attributes 4 and 9 are cores. The discernibility matrix contains 104 different entries. The following table shows ten experimental results under different attribute orders.

Since the discernibility matrix contains only a few entries, the algorithm takes a short running time (less than 0.1 second). As in Example 9, most attributes in the final results take anterior positions in the attribute order (Table 4).

10 Conclusions

In data mining, there are two tasks, description and prediction. Prediction has little difference from machine learning in nature. Both of them try to find out model upon a data set, while keeping their capability to other objects in the same domain, i.e., generalization. However, description has a rather different objective from them. It regards the data set as a closed world. Under a process, the data set is transformed to a more concise form, while keeping the possibility of explanation, i.e., reduction. Thus, compared with prediction, description is more distinct in data mining. In this paper, we focus on description tasks.

Generally speaking, a data set contains several kinds of knowledge. In addition, a data set is usually shared by several users. And different users may require different knowledge. So it is important for a data mining algorithm to provide users the knowledge they wanted, which we call *requirement-oriented knowledge discovery* (ROKD). In this paper, we present a new method to find customized reducts to meet the users' requirements, which is useful in a lot of applications because one reduct may be better in one sense than another. Thus, how to describe users' requirements, and then how to compute a reduct according to the requirements become the key point. This paper just regards attribute ordering as a special kind of method. It surely needs much more research to establish a general language for describing users' requirements. Some basic strategies are designed to describe attribute order. Then an algorithm supporting such description is proposed.

We emphasize the completeness of the algorithm for reduct because reduct is an objective between minimal reduct and trivial reduct, with the computational complexity of $O(n^2)$. In the future, we will try to promote the efficiency of the algorithm based on free attributes.

References

[1] Han J, Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

[2] Catlett J. Megainduction: Machine learning on very large databases [Dissertation]. Dept. of Computer Science, University of Sydney, Australia, 1991.

[3] Musick R, Catlett J, Russell S. Decision theoretic subsampling for induction on large databases. In *Proceedings of the Tenth International Conference on Machine Learning*, Utgoff P E (ed.), San Francisco, CA: Morgan Kaufmann, 1992, pp.212–219.

[4] Chan P K, Stolfo S J. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA: AAAI Press, 1995, pp.39–44.

[5] Shafer J, Agrawal R, Mehta M. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the Twenty-Second VLDB Conference*, San Francisco, CA: Morgan Kaufmann, 1996, pp.544–555.

[6] Mehta M, Agrawal R, Rissanen J. SLIQ: A fast scalable classifier for data mining. In *5th Int. Conf. on Extending Database Technology*, New York: Springer, 1996, pp.18–32.

[7] Provost F, Kolluri V. Scaling up inductive algorithms: An overview. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (KDD-97), 1997, pp.239–242.

[8] Ronen F, Willi K, Amir Z. Visualization techniques to explore data mining results for document collections. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (KDD-97), AAAI Press, 1997, pp.16–23.

[9] Utgoff P, Mitchell T. Acquisition of appropriate bias for inductive concept learning. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI-82, Pittsburgh, 1982, pp.414–417.

[10] Utgoff P. Shift of bias for inductive concept learning. In *Machine Learning: An Artificial Intelligence Approach*, Michalski R S, Carbonell J G, Mitchell T M (eds.), Volume II, California: Morgan Kaufmann, 1986, pp.107–148.

[11] Rendell L. A general framework for induction and a study of selective induction. *Machine Learning*, 1986, 1(2): 177–226.
[12] Haussler D. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 1988, 36(2): 177–221.
[13] *Machine Learning*, Vol.20, Issue 1/2, *Special Issue of ML on Bias Selection*, July, 1995.
[14] Dietterich T G, Kong E B. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Tech. Rep., Department of Computer Science, Oregon State University, Corvallis, Oregon, 1995.
[15] Wilson D R, Tony R M. Bias and the Probability of Generalization. In *Proc. the Int. Conf. Intelligent Information Systems* (IIS'97), 1997, pp.108–114.
[16] Turney P D. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal for AI Research*, 1995, 2: 369–409.
[17] Turney P D. Technical note: Bias and the quantification of stability. *Machine Learning*, 1995, 20(1–2): 23–33.
[18] Wang Jue, Wang Ju. Reduction algorithms based on discernibility matrix: The ordered attributes method. *J. Computer Science and Technology*, 2001, 16(6): 489–504.
[19] Pawlak Z. Rough sets. *Int. J. Comput. Inform. Sci.*, 1982, 11(5): 341–356.
[20] Polkowski L, Skowron A (eds.). Rough sets in knowledge discovery. Heidelberg: Physica-Verlag, 1998.
[21] Duntsch I, Gediga G. Rough set data analysis. *Encyclopedia of Computer Science and Technology*, 2000, 43(Supplement 28): 281–301.
[22] Greco S, Matarazzo B, Slowinski R. Rough approximation of a preference relation by dominance relations. *European Journal of Operational Research*, 1999, 117(1): 63–83.
[23] Greco S, Matarazzo B, Slowinski R. The use of rough sets and fuzzy sets in MCDM. Gal T, Stewart T, Hanne T (eds.), Chapter 14, *Advances in Multiple Criteria Decision Making*, Kluwer Academic Publishers, Dordrecht, Boston, 1999, pp.14.1–14.59.
[24] Greco S, Matarazzo B, Slowinski R. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 2001, 129(1): 1–47.
[25] Liu B, Hsu W, Chen S. Using general impressions to analyze discovered classification rules. *Knowledge Discovery and Data Mining*, 1997, pp.31–36.
[26] Bazan J, Skowron A, Synak P. Discovery of decision rules from experimental data. In *Proc. the Third International Workshop on Rough Sets and Soft Computing,* Lin T L (ed.), San Jose CA, November 10–12, 1994, pp.526–533.
[27] Bazan J, Skowron A, Synak P. Dynamic reducts as a tool for extracting laws from decision tables. In *Proc. the Symp. Methodologies for Intelligent Systems*, Charlotte, NC, Lecture Notes in Artificial Intelligence, Berlin: Springer-Verlag, 1994, pp.346–355.
[28] Wang J, Cui J, Zhao K. Investigation on AQ11, ID3 and the principle of discernibility matrix. *J. Computer Science and Technology*, 2001, 16(1): 1–12.
[29] Wroblewski J. Finding minimal reducts using genetic algorithms. In *Proceedings of the International Workshop on Rough Sets Soft Computing at Second Annual Joint Conference on Information Sciences* (JCIS'95), Wang P P (ed.), Wrightsville Beach, North Carolina, USA, September 28 – October 1, 1995, pp.186–189.
[30] Wroblewski J. Genetic algorithms in decomposition and classification problems. In *Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems,* Polkowski L, Skowron A (eds.), Physica-Verlag, Heidelberg, 1998, pp.472–492.
[31] Skowron A, Rauszer C. The discernibility matrices and functions in information systems. Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory, Slowinski R (eds.), 1991, pp.331–362.
[32] Wang X F, Wang R S, Wang J. Sustainability knowledge mining from human development database. In *Third Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD99), Zhong N, Zhou L Z (eds.), 1999, pp.279–283.
[33] Ziarko W. The discovery, analysis, and representation of data dependencies in databases. In *IJCAI Workshop on Knowledge Discovery in Databases Proceedings,* Piatetsky-Shapiro G, Frawley W J (eds.), AAAI/MIT Press, 1991, pp.195–209.

ZHAO Kai received his B.S. degree from Beijing Institute of Technology in 1993, and Ph.D. degree from the Institute of Automation, the Chinese Academy of Sciences. His research interests are adaptation systems, genetic programming and data mining.

WANG Jue is a professor of computer science and artificial intelligence at the Institute of Automation, the Chinese Academy of Sciences. His research interests include artificial neural network, machine learning and knowledge discovery in database.