A Novel Variable Precision Reduction Approach to Comprehensive Knowledge Systems

Chao Yang, Hongbo Liu, Sean McLoone, Senior Member, IEEE, C. L. Philip Chen, Fellow, IEEE, Xindong Wu, Fellow, IEEE

Abstract—A comprehensive knowledge system reveals the intangible insights hidden in an information system by integrating information from multiple data sources in a synthetical manner. In this paper, we present a variable precision reduction theory, underpinned by two new concepts: distribution tables and genealogical binary trees. Sufficient and necessary conditions to extract comprehensive knowledge from a given information system are also presented and proven. A complete variable precision reduction (CVPR) algorithm is proposed, in which we introduce four important strategies, namely, distribution table abstracting, attribute rank dynamic updating, hierarchical binary classifying, and genealogical pruning. The completeness of our algorithm is proven theoretically and its superiority to existing methods for obtaining complete reducts is demonstrated experimentally. Finally, having obtaining the complete reduct set, we demonstrate how the relationships between the complete reduct set and comprehensive knowledge system can be visualized in a double-layer lattice structure using Hasse diagrams.

Index Terms—Variable precision reduction, information system, knowledge system, knowledge system structure, data science.

I. INTRODUCTION

DATA science employs theories and techniques drawn from many fields for knowledge extraction from data and information systems [1]. They are usually presented as decision tables with rows and columns, i.e. objects and attributes. Knowledge extraction [2] or rule generation [3] is achieved by reducing the number of attributes in the decision tables in such a way that there is no loss of the information hidden in the information systems. In other words, if there is a metric λ for information, the value of λ is not changed by the reduction process. It is also one of the most common problems in data mining and knowledge discovery and has been widely studied. Considerable progress has been made on topics such as feature selection [4], [5], dimensionality reduction [6], feature evaluation [7], dynamic updating approximation [8], noisy processing [9], multi-granulation analysis [10], [11] and inconsistent data filtering [12], etc.

When performing knowledge extraction using rough set theory [13], the goal is to obtain a minimum reduct [14], [15] at an affordable computational cost and algorithmic complexity. The corresponding knowledge derived from a single reduct is called a single-knowledge [16]. However, in practice multiple reducts usually exist in information systems, with each reduct having a different attribute combination but the same classification capabilities. These can be thought of as providing insights from different perspectives [17]. Multiple reducts can constitute a multi-knowledge system [18], but this does not guarantee that they capture all available knowledge. For a given information system, only a complete reduct set contains all possible reducts without redundancy, and therefore can be used to deduce a comprehensive knowledge system. Unfortunately, obtaining a complete reduct set is an NP-hard problem [19], and as such represents a major challenge computationally.

This paper focuses on how to obtain the complete reducts set, formulate the corresponding knowledge structure and generate comprehensive knowledge systems. The paper has the following contributions:

- We present a variable precision reduction theory for comprehensive knowledge systems. Some underpinning concepts relating to the completeness of reduct set and knowledge are defined. In particular, we introduce two new concepts, namely, distribution tables and genealogical binary trees, and establish the necessary and sufficient conditions to achieve complete multi-knowledge extraction without redundancies.

- An efficient and effective complete variable precision reduction algorithm is then proposed, the key elements of which are distribution table abstracting, attribute rank dynamic updating, hierarchical binary classifying and genealogical tree pruning. We prove theoretically the completeness of our algorithm and compare its efficacy to existing methods with the aid of experimental studies.

- We also explore the relationships between a complete reduct set and its knowledge system, introducing a double-layer lattice structure to enable them to be illustrated visually in Hasse diagrams.
The remainder of the paper is organized as follows. Related work is reviewed in Section II. In Sections III we present our variable precision reduction theory to include the completeness of reduct set and knowledge extraction. In Section IV, we propose a complete variable precision reduction algorithm. In particular, the dynamic attribute ranking to a genealogical binary tree is presented in Subsection IV-A, while the hierarchical binary classifying and pruning strategies are introduced in Subsection IV-B. Then, our complete reduction and knowledge extraction are presented in Subsection IV-C. The experimental results and discussions are provided in Section V. Finally, conclusions and areas for future work are given in Section VI.

II. RELATED WORK

A. Knowledge Comprehensiveness

Pancerz and his colleagues discussed information and dynamic information systems, which can be used in prediction problems [20], [21]. Compared to single-knowledge, multi-knowledge is usually more comprehensive, and as such is likely to correctly classify new objects. Abu-Donia [22] presented rough set approximations using multi-knowledge base to raise the efficiency of decision support system. Gams [23] demonstrated the creation of a multi-knowledge system by integrating ten single-knowledge systems using 10-fold partition learning. They illustrated that a multi-knowledge system can improve the accuracy of classification significantly. However, although they put the multi-knowledge system into practice, issues of redundancy and comprehensiveness of the knowledge extracted were not considered. Ma [24] introduced a completeness condition to determine whether an attribute combination is a reduct or not, but again does not consider the completeness of multi-reducts. A complete reduct set is stricter than multi-reducts, since in a complete reduct set, all reducts are included without redundancy. Knowledge comprehensiveness also depends on the corresponding complete reduct set, because different reduct provide knowledge from different perspectives.

B. Knowledge Visualisation

Qian et al. presented a space distance [25] which is used to characterize the similarity between reducts. After attribute reduction, a knowledge system can be constituted by rule extraction tools, such as rough set [26], apriori algorithm [27], formal concept analysis [28] and so on. Formal concept analysis not only gives implications “$X \rightarrow Y$”, but also builds a hierarchical structure of knowledge concepts through concept lattices [29] which can be used to describe the relationships between objects and attributes. Through constructing complete concept lattices, the lattice structure can be introduced to describe the similarity between reducts or between knowledge, and the relationship between a reduct and its corresponding knowledge can also be depicted. Hasse diagrams [30] are a suitable way to visualise the lattices of the comprehensive knowledge system structure as developed in this paper.

C. Knowledge Extraction Approaches

Different heuristic algorithms have been proposed to reduce attributes and to extract knowledge from information systems in the classical rough set model [31]. However, these algorithms are subject to a rigid premise in the classical rough set model that the error classification rate [32] must be zero.

To relax this rigid premise of the classical rough set model, variable precision rough sets (VPRS) [33] are presented to introduce an error classification rate $\beta$ [34]. The positive and boundary regions in rough set theory are designed to distinguish certain and uncertain objects, respectively [35], [36]. The predictable (certain) objects will increase by reducing $\beta$.

Ziarko presents a $\beta$ − reduct method [33] for attribute reduction while Park and Choi propose a novel VPRS method using information entropy [37]. However, a side effect of their approaches is the so called “jumping” phenomenon [14], [38], where the output alternates between reduct and non-reduct.

Wang et al. [38] show that the “jumping” phenomenon is due to the fact that the changes in classification quality and positive region are non-monotonous. They further illustrate that the distribution of the lower approximation bound about decision classes varies monotonically. The reduction approach in this case follows a decision monotonicity criterion [39], [40]. It implies that rough set reduction is subject to the distribution of the objects with $\beta$-dominant decision value in each equivalence class to $\beta$ extent within the positive region, while the distribution of other objects have little influence because of the error classification rate $\beta$.

The attribute reduction methods based on the discernibility matrix and the ones based on the positive region can both be used to obtain multiple reducts from a given information system. However, the former is usually much more computationally intensive and is limited to much lower dimension problems than the later, even when using the approach introduced by Yang [41] to minimize the number of elements in the discernibility matrix to decrease the computational load. In the case of multi-reducts and multi-knowledge extraction methods based on the positive region, the related work mainly falls into two categories:

- non-core attribute combination algorithms, such as the worst-attribute-drop-first (WADF) algorithm [16]. In these algorithms, all attributes are divided into core attributes and non-core attributes. A reduct is obtained and then used as a seed reduct to generate other reducts in the multi-reducts set through a non-core attribute replacement process, whereby one non-core attribute is excluded in the identified reduct. In [42] a multi-knowledge system is constituted to help robots identify their environment. However, in the algorithms employed, the number of multi-reducts obtained is strongly dependent on the initial seed reducts.

- non-deterministic random or quasi-random search algorithm, such as swarm intelligent algorithms [43], [44]. These algorithms utilize a swarm of intelligent individuals to search for the reducts in an information system and can usually obtain multi-reducts quickly. They have been employed to obtain a multi-knowledge system using
rough sets and to complete multi-factor analysis of violent crime [45]. However, although random search algorithms can converge theoretically with a probability of 1, they cannot guarantee the completeness of multi-reducts within a finite run time in practice.

III. REDUCTION AND KNOWLEDGE THEORY

The basic concepts of attribute reduction and its philosophy are presented in the related literature [38], [46] and multiple reducts and multi-knowledge systems are described in [17]. Here, we will recall the reduction concepts that are relevant to the present work and then extend the relevant theories and concepts to provide a theoretical description of comprehensive knowledge systems.

A. Distribution Table

An information system can be defined as a decision table by $T = \{(C, D, V, f)\}$, where $C$ is the set of condition attributes and $D$ is the set of decision attributes, respectively. $V$ is the value set of all attributes. $f : U \times (C \cup D) \rightarrow V$ is the information function such that $f(x, a) \in V$ for every $a \in C \cup D$, $x \in U$. For $D \subseteq C$, an equivalence relation $IND(E)$ is defined as follows:

$$IND(E) = \{(x, y) \in U \times U \mid \forall a \in E, f(x, a) = f(y, a)\}$$

(1)

$IND(E)$ partitions $U$ into disjoint subsets. Let $U/E$ denote the family of all equivalence classes of the relation $IND(E)$, i.e. $U/E = \{E_1, E_2, \ldots, E_i, \ldots\}$, where $E_i$ is an equivalence class of $E$, which is denoted $[x]_{E}$. Note that equivalence classes are defined with respect to their own attribute set. Equivalence classes $U/C$ and $U/D$ will be called condition and decision classes, respectively.

Definition 1: [Positive region] Given a decision table $T = (U, C, D, V, f)$. Let $B \subseteq C$. The $B$-positive region of $D$ is the set of all objects from $U$ which satisfy Eqn. (2).

$$POS_B^D(D) = \bigcup_{|\beta| \geq \beta} [x]_{B}$$

(2)

where $\beta \in (0, 1)$. In particular, the positive region with $\beta$ will degenerate into the classical rough set model if $\beta = 1$.

Definition 2: [$\beta$-decision table] Given a decision table $T = (U, C, D, V, f)$, determine its condition classes $[x]_C$. The decision value of objects in each condition class to $\beta$ extent within $POS_B^D(D)$ is normalized to the $\beta$-dominant decision value in the respective condition class. Namely, the decision value of $[x]_C$ in the positive region is normalized such that:

$$\frac{|\beta| \cap [x]_{D}}{|[x]_C|} = 1$$

(3)

and then, the information function is updated as $f_\beta : U \times (C \cup D) \rightarrow V$. In this way a $\beta$-decision table is formulated, which will be denoted as $T_\beta = (U, C, D, V_\beta, f_\beta)$. Accordingly, attribute $b \in B \subseteq C$ is $D$-dispensable in $B$, if $POS_B^D(D) = POS_B^{D \setminus \{D\}}(D)$; otherwise attribute $b$ is $D$-indispensable in $B$. If $\forall b \in B$ are $D$-indispensable in $B$, then $B$ will be called $D$-independent. A subset of attributes $B \subseteq C$ is a $D$-reduct of $C$, iff $POS_B^D(D) = POS_B^C(D)$ and $B$ is $D$-independent.

Theorem 1: Given $T_\beta$, set $NEG_B^C(D) = U \setminus POS_B^C(D)$. $B \subseteq C$ is a reduct, therefore $\forall [x]_{B}$ either $[x]_{B} \subseteq NEG_B^C(D)$ or $[x]_{B} \subseteq POS_B^C(D)$ and $\beta > 1$ is satisfied.

Proof 1: $B$ is a reduct, so $POS_B^C(D) = POS_B^D(D)$ and $NEG_B^C(D) = NEG_B^D(D)$. If $[x]_{B} \subseteq NEG_B^D(D)$, $[x]_{B} \subseteq NEG_B^C(D)$. Similarly, $[x]_{B} \subseteq POS_B^C(D)$. According to Definition 2, $\frac{|[x]_{B} \cap [x]_{D}}{|[x]_C|} = 1 \Leftrightarrow \frac{|[x]_{B}/D|}{1} = 1$.

Definition 3: [Distribution table] Given a decision table $T = (U, C, D, V, f)$, transform it into $\beta$-decision table $T_\beta = (U, C, D, V_\beta, f_\beta)$, in which the decision values of $x_i$ according to a decision monotonicity criterion is adjusted by Eqn. (4).

$$f_\beta(x_i, D) = \begin{cases} \arg \max \{\frac{|\beta| \cap [x]_{D}}{|[x]_C|}\} & \text{if } \beta \leq \rho(x_i) \leq 1, \\ f(x_i, D) & \text{otherwise.} \end{cases}$$

(4)

where $\rho(x_i) = \frac{|[x]_C \cap [x]_{D}}{|[x]_C|}$ $U/C = \{[x_1, \ldots, x_i, \ldots, x_m]\}$ is the equivalence class set. Here a distribution table $T_D = (U_D(\omega), C, D, V_\beta, f_\beta)$ can be formulated with the same attributes while $U_D = \{[x_1, \omega(x_1)), \ldots, (x_i, \omega(x_i)), \ldots, (x_m, \omega(x_m)\}$, where the distribution region label $\omega(x_i)$ of object $x_i$ is determined by Eqn. (5).

$$\omega(x_i) = \begin{cases} 1 & \text{if } \beta \leq \rho(x_i) \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

(5)

These objects are in the positive region if $\omega(x_i) = 1$ and in the negative region if $\omega(x_i) = 0$. In other words, all objects have their deterministic distributions during variable precision reduction, which helps us avoid the “jumping” phenomenon. Furthermore, the distribution table collects only one object from each equivalence class in a given information system. It is an essential abstract from the original universe of discourse, and reduces significantly the number of considered objects, especially for large datasets. These benefits motivate our use of the distribution table instead of working directly with the decision table. We will prove the equivalence between the distribution table and the original decision table in Theorem 2 for attribute reduction.

Theorem 2: Given $T$, compute $D_T$, where $U_D = \{[x_1, \omega(x_1)), \ldots, (x_i, \omega(x_i)), \ldots, (x_m, \omega(x_m))\}$. $\forall B \subseteq C$, if an arbitrary equivalence class $[x]_{B}$ in $D_T$ satisfies $\omega(x_i) = 0$ or $\omega(x_i) = 1 \Leftrightarrow \frac{|[x]_B \cap [x]_{D}}{|[x]_C|} = 1 \Leftrightarrow \forall x_i \in [x]_B$, then $POS_B^D(D) = POS_B^C(D)$.

Proof 2: $[x]_{B}$ respects an arbitrary equivalence class about $x_i$ in $T$. By Definition 3, $\forall x_i \in [x]_B$, $\omega(x_i) = 0 \Leftrightarrow [x]_B \subseteq NEG_B^C(D) \Leftrightarrow [x]_B \subseteq NEG_B^C(D)$.

By Theorem 1 $(\forall x_i \in [x]_B, \omega(x_i) = 1 \Leftrightarrow \frac{|[x]_B \cap [x]_{D}}{|[x]_C|} = 1 \Leftrightarrow POS_B^D(D) = POS_B^C(D)$ and $\beta > 1$ $\Rightarrow [x]_B \subseteq POS_B^C(D)$, Thus, $POS_B^D(D) = POS_B^C(D)$. 

IEEE TRANSACTIONS ON CYBERNETICS, VOL. **, NO. **, AUGUST 2016 3
B. Reduct Completeness

Definition 4: [Complete reduct set] Consider all possible attribute subsets \( \{b_1, \ldots, \{b_i, b_{i+1}, \ldots, b_{i'}\}\} \), where \( b_i \in B \subseteq C, 1 \leq i \leq |C| \). Let RED(\( \beta \)) represent the set of complete reducts, i.e.,

\[
RED(\beta) = \{ B | POS_B^\beta(D) = POS_C^\beta(D), \forall b \in B, POS_B^\beta(B - \{b\}) < POS_B^\beta(D) \}.
\]

(6)

Definition 5: [Reduct completeness ratio] The reduct completeness ratio is defined as follows.

\[
\vartheta = \frac{|RED(\beta)|}{|RED(\beta)|}
\]

(7)

where RED(\( \beta \)) represents the set of reducts in practice.

Theorem 3: Given \( \beta \)-decision table \( T_\beta = (U, \bar{C}, \bar{D}, \bar{B}, f_\beta) \), \( B \subseteq C \), then \( x \in POS_B^\beta(\bar{D}_\beta) \) iff \( x \) satisfies the predicate formulas \( (\forall y \in U) (D_\beta(x) \neq D_\beta(y)) \Rightarrow B(x) \neq B(y) \) is true where \( D_\beta(x) \) represents the decision value of \( x \) in \( T_\beta \).

Proof 3:

\[
x \in POS_B^\beta(\bar{D}_\beta) \Leftrightarrow [x]B \subseteq [x]D_\beta
\]

\[
\Leftrightarrow (\forall y \in U)(B(x) = B(y) \Rightarrow D_\beta(x) = D_\beta(y))
\]

\[
\Leftrightarrow (\forall y \in U)(B(x) \neq B(y) \lor D_\beta(x) = D_\beta(y))
\]

\[
\Leftrightarrow (\forall y \in U)(D_\beta(x) = D_\beta(y) \land B(x) = B(y))
\]

\[
\Leftrightarrow (\forall y \in U)(D_\beta(x) \neq D_\beta(y) \Rightarrow B(x) \neq B(y)).
\]

Theorem 4: \( POS_B^\beta(\bar{D}_\beta) \subseteq POS_C^\beta(D_\beta) \) if \( B \subseteq C \).

Proof 4: Set \( x \in POS_B^\beta(\bar{D}_\beta) \), we have \([x]B \subseteq [x]D_\beta\). Since \( B \subseteq C \), \([x]C \subseteq [x]B \). Therefore \([x]C \subseteq [x]D_\beta\). We have \( x \in POS_C^\beta(D_\beta) \).

Theorem 5: Set \( B \subseteq C \), \( POS_B^\beta(\bar{D}_\beta) = POS_C^\beta(D_\beta) \) iff \( B \) satisfies the following predicate formulas:

\[
(\forall x \in U)((\exists y \in U)(D_\beta(x) \neq D_\beta(y) \land C(x) = C(y)) \lor (\forall z \in U)((D_\beta(x) \neq D_\beta(z) \Rightarrow B(x) \neq B(z))).
\]

Proof 5: Since \( B \subseteq C \), we have \( POS_B^\beta(\bar{D}_\beta) \subseteq POS_C^\beta(D_\beta) \) from Theorem 4. So \( POS_B^\beta(\bar{D}_\beta) = POS_C^\beta(D_\beta) \) iff \( POS_B^\beta(D_\beta) = POS_C^\beta(D_\beta) \), that is \( POS_B^\beta(D_\beta) = POS_C^\beta(D_\beta) \) iff \( (\forall x \in U)(x \in POS_B^\beta(D_\beta) \Rightarrow x \in POS_C^\beta(D_\beta)) \).

From Theorem 3, we have

\[
POS_B^\beta(D_\beta) = POS_C^\beta(D_\beta) \Leftrightarrow
\]

\[
((\forall x \in U)((\forall y \in U)(D_\beta(x) \neq D_\beta(y) \Rightarrow C(x) \neq C(y)) \Rightarrow ((\forall x \in U)(D_\beta(x) \neq D_\beta(z) \Rightarrow B(x) \neq B(z)))) \Leftrightarrow
\]

\[
((\forall x \in U)((\forall y \in U)(D_\beta(x) \neq D_\beta(y) \Rightarrow C(x) \neq C(y)) \lor (\forall z \in U)((D_\beta(x) \neq D_\beta(z) \Rightarrow B(x) \neq B(z)))) \Leftrightarrow
\]

\[
((\forall x \in U)((\exists y \in U)(D_\beta(x) \neq D_\beta(z) \land C(x) = C(y)) \lor (\forall x \in U)(D_\beta(x) \neq D_\beta(z) \Rightarrow B(x) \neq B(z))).
\]

When \( \beta = 1 \), Theorem 5 is consistent with the results in [24).

Theorem 6: \( \forall B \subseteq C \), if \( [x]_B \cap [C] \neq \emptyset \land [x]_B \cap [D] = \emptyset \),

Proof 6: Because \( B \subseteq C \), \([x]_C \subseteq [x]_B \). And \([x]_C \cap [D] \neq \emptyset \), it must be \([x]_B \cap [D] \neq \emptyset \).

C. Genealogical Binary Tree

In a left-child-right-sibling (LCRS) binary tree \( M = (N, L, R) \), \( N \) denotes a node, its parent node \( N_f \) and sibling node \( N_s, L \) and \( R \) is the left and right child nodes, respectively. There is no child node in a leaf node.

Definition 6: [Brotherhood] In the right path of the LCRS binary tree, the previous node of the starting node is null or \( N_f \) and the terminator node is a leaf node, then all of nodes in this path is a brotherhood in the binary tree.

In particular, the brotherhood is also called the ancestor when the previous node of the starting node is null.

Definition 7: [Genealogical binary tree] Given \( T = (U, C, D, V, f) \), a LCRS binary tree maps \( C \) to generate a genealogical binary tree, denoted as \( G(C) \), where any attribute \( c \in C \) is mapped into a node \( c \). In the right path of \( G(C) \), the previous nodes of node \( c \) are its elder siblings, while those next nodes of node \( c \) are its younger siblings. In the left path of \( G(C) \), the path from a node to its ancestor is called as a genealogical generation path, denoted as \( \sim \). The nodes in a genealogical generation path only have parent-child relationships without brotherhood. By traversing all genealogical generation paths the complete attribute combinations set, denoted \( PG(G(C)) \), can be obtained.

The genealogical binary tree of \( T \) will provide a deterministic roadmap to perform an orderly and complete search of all candidate reducts in \( T \).

As shown in Figure 1, a genealogical binary tree is mapped from \( C = \{c_1, c_2, c_3, c_4\} \) in a given \( T = (U, C, D, V, f) \). The number at the upper right of each node is its sequence number. Here, the order of the attributes in \( T \) is used as the order of the attributes in the nodes. In practice, the attributes need to be ranked before attribute reduction. We will discuss the ranking process later (see Equ. (15)).

Nodes 1, 2, 3 and 4 are the brotherhood (ancestor). Nodes 1 and 2 are the elder siblings of node 3 and its younger sibling is node 4. In the left paths of \( G(C) \), all attribute combinations can be obtained from \( PG(G(C)) \), i.e. \( PG(G(C)) = \{\{c_1\}, \{c_2\}, \{c_3\}, \{c_4\}, \{c_1, c_2\}, \{c_1, c_3\}, \{c_1, c_4\}, \{c_2, c_3\}, \{c_2, c_4\}, \{c_1, c_2, c_3\}, \{c_1, c_2, c_4\}, \{c_2, c_3, c_4\}\} \). The corresponding genealogical generation paths are \( 1 \sim 2 \sim 3 \sim 4 \sim 5 \sim 6 \sim 7 \sim 2 \sim 8 \sim 9 \sim 10 \sim 11 \sim 5 \sim 1 \sim 12 \sim 13 \sim 14 \sim 15 \sim 1 \sim 6 \sim \).

![Fig. 1: A genealogical binary tree](image-url)
D. Knowledge Comprehensiveness

Theorem 7: Given a decision table \( T = (U, C, D, V, f) \), denote \( RED(\beta) \) as its reduct set with a given \( \beta \). The set of attribute combinations obtained by traversing \( G(C) \) is denoted as \( RED^* \). \( (POS_B^\beta(D) = POS_C^\beta(D)) \land ((\forall a \in B)(POS_B^{(B-a)}(D) < POS_B^\beta(D))) \) for \( \forall B \in RED^* \) and \( (POS_B^\beta(D) = POS_C^\beta(D)) \lor ((\exists b \in B)(POS_B^{(B-b)}(D) = POS_B^\beta(D) \land POS_C^\beta(D))) \) for \( \forall B \notin RED^* \), which is the necessary and sufficient condition for \( RED(\beta) = RED^* \).

Proof 7: The number of nodes is \( 2^n - 1 \) for a complete graph \( G \) of depth \( n \). This implies that there are \( 2^n - 1 \) attribute combinations in \( P^G(C) \). Then \( RED(\beta) \subseteq P^G(C) \) in \( G(C) \).

By Definition 4, if \( RED(\beta) = RED^* \), it follows that \( RED(\beta) = RED^* \)

\[
(\forall B \in RED^*, (POS_B^\beta(D) = POS_C^\beta(D)) \land ((\forall a \in B)(POS_B^{(B-a)}(D) < POS_B^\beta(D))))
\]

For \( \forall B \in RED^* \), \( (POS_B^\beta(D) = POS_C^\beta(D)) \land ((\forall a \in B)(POS_B^{(B-a)}(D) < POS_B^\beta(D))) \), \( B \in RED(\beta) \). For \( \forall B \notin RED^* \), \( (POS_B^\beta(D) \neq POS_C^\beta(D)) \lor ((\exists b \in B)(POS_B^{(B-b)}(D) = POS_B^\beta(D) = POS_C^\beta(D))) \), \( B \notin RED(\beta) \). So \( RED(\beta) = RED^* \).

Hence, the theorem follows.

Let \( RED(\beta) \) represent the set of reducts, and \( \tau \) is a mapping from the condition attributes set to the decision attributes set. Then the knowledge from the reduct set can be defined as follow.

\[
\Psi(\beta) = \{\tau_B | B \in RED(\beta)\}
\]

(8)

where each of the elements in \( \Psi(\beta) \) is a rule, all of which form a comprehensive knowledge system \( K = (U_D', A, I) \), where \( U_D' \) is \( U_D \) without the \( \omega \) component, and \( I \) represents the relation between \( U_D' \) and \( A \). Defining \( x \in U_D', a \in A \), the notation \( \tau a \) is introduced to signify that the object \( x \) has the attribute \( a \) [47].

In an information system, the condition attribute \( \forall c \in C \), the condition vector \( V_C \) and its size \( |V_C| \) [48] are as defined in Eqn. (9).

\[
V_C = V_{c_1} \times V_{c_2} \times \cdots \times V_{c_{|C|}}
\]

\[
|V_C| = \prod_{i=1}^{C} |V_{c_i}|
\]

(9)

where \( |V_C| \) is the number of objects with different values that can theoretically exist in the condition vector, while \( |C(U)| \) is the actual number of objects with different values that exists for a given information system. Since the attribute values can be formulated as vectors of Cartesian products of attribute values in the information system, they can be regarded as an extension of the information system [20], [21].

Definition 8: [Information system comprehensiveness] Information system comprehensiveness is the ratio of the actual and theoretical number of objects with different values, as defined in Eqn. (10).

\[
\xi = \frac{|C(U)|}{|V_C|}
\]

(10)

if \( \xi = 1 \), the system is called a comprehensive information system.

Definition 9: [Knowledge system comprehensiveness] In a knowledge system, the number of objects with different value is \( |\Psi(\beta)| \) in the condition vector, and hence knowledge system comprehensiveness, \( \xi \), can be expressed as:

\[
\hat{\xi} = \frac{|\Psi(\beta)|}{|C(U)|}
\]

(11)

if \( \hat{\xi} = 1 \), the system is called a comprehensive knowledge system.

Definition 10: [Basic rule ratio] Some of the rules in \( |\Psi(\beta)| \) already exist in \( |C(U)| \). These rules are called basic rules and explicit. The ratio of knowledge extraction, referred to as the basic rule ratio, is defined as:

\[
\hat{\xi} = \frac{|C(U) \cap \Psi(\beta)|}{|C(U)|}
\]

(12)

if \( \hat{\xi} = 1 \) in a comprehensive knowledge system, then the system contains all basic rules from the given information system.

E. Knowledge Structure

We define a structure for a comprehensive knowledge system to illustrate a complete reduct set, the corresponding knowledge and their relationships.

Definition 11: [Reduce chain] All possible attribute subsets for \( 2^{|C|} \) form a complete lattice. The subset \( \{c_1, \cdots, c_{|C|}\} \) is the maximum node of the complete lattice, while the empty set \( \Phi \) is the minimum node. \( \forall B \subseteq C \) and \( B \) is a reduct, the chain which passes through \( B \) is called a reduct chain from the minimum to the maximum node in the lattice.

The reducts in \( RED \) form a complete reduct set lattice. A Hasse diagram of the attributes for \( \{c_1, c_2, c_3, c_4\} \) is shown in Figure 2(a) for a given \( \beta \). These are twenty-four attribute chains, for example \( \Phi - c_1 - c_1 - c_3 - c_1 - c_1 - c_1 - c_2, c_3, c_4 \). If \( \{c_1, c_3, c_4\} \) and \( \{c_2, c_4\} \) are considered as reducts, the two chains highlighted in red are the reduct chains. A Hasse diagram of a complete reduct set is shown in Figure 2(b).

In a comprehensive knowledge system, a formal context with complete reducts can be used to reveal the internal relationships between knowledge from multiple perspectives. The knowledge lattice is obtained from the formal context [47]. Its minimum node is an empty set, and the maximum node is a complete attribute set with corresponding attribute values. For \( RED(\beta) = \{B_1, \cdots, B_i\} \), the lower the value of \( | \bigcup_{i=1}^{B_i} B_i | - | \bigcap_{i=1}^{B_i} B_i | \), the less the dispersion of complete reducts.

Definition 12: [Double-layer Complete Structure] A double-layer complete structure consists of two layers, in which one is a complete reduct set lattice and the other is a comprehensive knowledge lattice. Set reduct \( B \in RED \) and \( B^d = B \cup \{d\} \).
In a comprehensive knowledge lattice, each node has a lattice connotation, in which \( c_i^j \) denotes that the value of attribute \( c_i \) is \( j \). Similarly, \( d^j \) denotes that the value of the decision attribute \( d \) is \( j \). Set lattice connotation \( B' \) is from an upper-neighbor \( t_{B'} \) of the maximum node in a comprehensive knowledge lattice, where \( B' = B^d \) with corresponding attribute values and denote their relationship as \( B' \in B^d \). The set of nodes on the chains passing node \( t_{B'} \), excluding the minimum and maximum nodes, forms a set \( T(t_{B'}) \) in a comprehensive knowledge lattice. The relationship between the complete reduct set and the comprehensive knowledge lattices is denoted by “\( \rightarrow * \)”. If \( B' \in B^d \), \( B \rightarrow * T(t_{B'}) \).

Property 1: Some double-layer structure properties are summarized as follows.
1) \( B'_1 \cap B'_2 \neq \emptyset \Rightarrow B_1 \cap B_2 \neq \emptyset \);
2) \( B'_1 \cap B'_2 \neq \emptyset \Leftrightarrow B_1 \cap B_2 \neq \emptyset \);
3) \( T(t_{B'_1}) \cap T(t_{B'_2}) \neq \emptyset \Rightarrow B'_1 \cap B'_2 \neq \emptyset \Rightarrow B_1 \cap B_2 \neq \emptyset \).

where \( B_1, B_2 \in RED \) and \( B'_1 \in B^d \), \( B'_2 \in B^d \).

IV. KNOWLEDGE EXTRACTION METHODOLOGY

In this section, our complete variable precision reduction (CVPR) algorithm is presented in detail. The computational time complexity and completeness of the methodology are also analyzed and proven.

A. Distribution Table Abstracting and Attribute Rank Dynamic Updating

Given a decision table \( T \), we transform it into \( \beta \)-decision table \( T_{\beta} \), in which the decision value of \( x_i \) according to a decision monotonicity criterion is adjusted by Eq. (4). Then only object \( x_i \) in \( \forall [x_i]_c \) is collected and \( \omega(x_i) \) is computed by Eq. (5) in \( T_{\beta} \). We get the distribution table \( T_D = (U_D(\omega), C, D, V_D, f_D) \), where \( U_D = \{ (x_1, \omega(x_1)), \ldots, (x_i, \omega(x_i)), \ldots, (x_m, \omega(x_m)) \} \). It is our distribution table abstracting strategy, which helps us to reduce the considered universe (objects) significantly compared to the original decision table through the equivalence class. From Theorem 2 it follows that this strategy can keep the equivalence between obtaining reducts from \( T_D \) and \( T \). An abstracting proportion denotes the percentage reduction of objects from \( T \) to \( T_D \) as follows.

\[
\varsigma = \frac{|U| - |U_D(\omega)|}{|U|}
\]

where \( |U| \) and \( |U_D(\omega)| \) are the numbers of objects in \( T \) and \( T_D \), respectively.

Consider here an attribute set \( H \), which is \( \emptyset \) initially. We combine incrementally an attribute \( c \) into \( H \) from the attribute set \( C - H \) \((i.e., c \in (C - H))\) in distribution table \( T_D = (U_D(\omega), C, D, V_D, f_D) \) to obtain a reduct. The combination priority of any attribute \( c \in (C - H) \) will be mainly dependent on its attribute rank. If \( \exists [x_i]_H \) with \( [x_i]_H/\omega \neq 1 \), we denote the objects with \( \omega(x_i) = 1 \) in \( [x_i]_H \) with \( [x_i]_H/\omega \neq 1 \) as \( S_{obj} \). In the next loop, if \( [x_i]_H/\omega = 1 \), but \( \exists [x_i]_H \) with \( \omega(x_i) = 1 \land [x_i]_H/D \neq 1 \), we denote the objects with the same decision values in all \( [x_i]_H \) with \( \omega(x_i) = 1 \land [x_i]_H/D \neq 1 \) as \( S_{obj} \). If there are multiple different decision values in these equivalence classes, other objects with other different decision values would be denoted as \( S_{obj} \) one by one in the next loops in our algorithm. At worst, it will loop \(|D| + 1 \) times, while at best, it will only loop once, as will be discussed in Subsection IV-B. If \( S_{obj} \) has been denoted, our binary classifying label \( \varphi \) is determined as follows.

\[
f(x, \varphi) = \begin{cases} 
1 & \text{if } x \in S_{obj}, \\
0 & \text{otherwise.} 
\end{cases}
\]

We use \( c \) to partition \( [x_i]_H \) with \( [x_i]_H/\omega \neq 1 \), and then obtain \( [x_i]_c \). Then the attribute rank \( r(c) \) can be calculated as

\[
r(c) = -\frac{1}{|\{x_i\}_H/\varphi \neq 1 \land [x_i]_c/\varphi \neq 1 \land f(x, \varphi) = 1|}
\]

It means that \( r(c) \) is the negative of the number of \( \varphi \)-inconsistency equivalence classes with respect to attribute \( c \) against \( [x_i]_H \). The larger \( r(c) \) is, the higher the combination priority of attribute \( c \). When \( r(c) \) is the same for two attributes priority is determined according to the from-left-to-right order in \( T_D \). If \( r(c) = 0 \), \( H \cup c \) is a candidate reduct, since all objects have been partitioned clearly. If all the equivalence classes with respect to \( H \cup c \) satisfy Theorem 2, then this candidate reduct is a reduct according to Definition 2.

Let \( U_D, |n'| \) and \( |n'| \) represent the universe of discourse, its number of objects and the number of condition attributes in the distribution table, respectively. The pseudo-code of our attribute rank updating (ARU) algorithm for genealogical brotherhood is illustrated in Algorithm 1.

We analyze the time complexity of ARU algorithm as follows. Because the loop from Steps 1 to 9 occurs at most \(|C| \) times, the time complexity is \( O(|U||C| + \sum_{i=1}^{|C|} (M_i - m_i + 1)) \).
we transform the considered
Secondly, attribute rank is calculated dynamically when
our pruning strategy is also triggered at Step 6 if rank
Theorem 2 in our complete attribute reduction algorithm, as
validity of each of the candidate reducts is checked using
Lemma 1 provides us with a pruning opportunity. At Step
Using the genealogical binary tree of attributes to partiti on
attribute combinations are partitioned into \( |x_y|/\varphi = 1 \) and \( |x_y|/\varphi = 2 \), respectively,
and the average case time complexity through Steps 3 to 21 is \( O(2^{|C|}/|U|) \), it can be concluded that the average case time complexity of Algorithm 2 is \( O(2^{|C|}/|U|) \).

C. Knowledge Extraction

In order to obtain a complete reduct set, we have to perform
four tasks as follows. Firstly, we transform the considered
decision table of the information system into its distribution
table. Secondly, attribute rank is calculated dynamically when
any attribute combination set is obtained for the reducts.
Thirdly, a genealogical binary tree is generated dependent on
the attribute rank. Fourthly, the objects in the positive region
of the information system will be binary classified hierarchically.
During binary classifying, the attributes not included in the
by calling Algorithm 1; Generate the left child node according to its
attribute ranks; goto the youngest sibling in the
brotherhood; Collect the attributes in this genealogical
generation path as a candidate reduct and add it into \( RED' \);
end end end while all ancestor nodes are visited;

B. Hierarchical Binary Classifying and Genealogical Tree Pruning

The distribution state of objects is determined once the
information system is given. Attribute reduction is essential to
find some subsets of attributes (i.e., reducts) to partition the
objects distinctly through updating the equivalence classes
with respect to different tentative attribute combinations. According to Theorem 2, the partition can be implemented by multi-binary classifying of the equivalence classes with
\( |x_H|/D \neq 1 \land \omega = 1 \) hierarchically.

Using the genealogical binary tree of attributes to partition
the objects distinctly, the hierarchical binary classifying (HBC)
algorithm illustrated in Algorithm 2 is proposed. Here, \( \Omega \)
represents the condition, i.e., the attribute combination \( Z \)
containing \( c \) and its younger siblings satisfies \( |x_z|/\varphi \neq 1 \)
for \( \forall x_z \in Z \), and \( Y \) represents the corresponding attributes, i.e.,
the attributes in the path from the \( N_f \) node of the current node
to its ancestor.

From Theorem 6, Lemma 1 immediately follows.

\textbf{Lemma 1}: \( \forall Z' \subseteq Z, \) if \( |x_z|/\varphi \neq 1, |x_z|/\varphi \neq 1 \).

Lemma 1 provides us with a pruning opportunity. At Step
7 in Algorithm 2, our pruning strategy is triggered in \( G \) if
\( \exists Z \Rightarrow |x_z|/\varphi = 2 \), since the attributes at these pruned
nodes cannot be included in any reduct. In the same way,
our pruning strategy is also triggered at Step 6 if \( r = 0 \).
This is very helpful in terms of reducing computational complexity, and enables multiple reducts to be determined with
significantly reduced computational effort.

Note that the output of the HBC algorithm is the candidate
multiple reduct set \( RED' \) for the binary classifying. The
validity of each of the candidate reducts is checked using
Theorem 2 in our complete attribute reduction algorithm, as
will be discussed in detail in Subsection IV-C. Here we further analyze the time complexity of Algorithm 2 as follows. To execute Algorithm 1, Steps 1 and 11 take no longer than \( |U||C| \) time. The useful attributes and objects obtained using Steps 8 and 10 is dominated by \( O(|U||C|) \). Since the best,
worst and average case time complexity for pre-rank of the binary tree is \( O(1), O(2^{|C|}) \) and \( O(2^{|C|}/|C|) \), respectively,
and the average case time complexity through Steps 3 to 21 is \( O(2^{|C|}/|U|) \), it can be concluded that the average case time complexity of Algorithm 2 is \( O(2^{|C|}/|U|) \).

\begin{algorithm}[H]
\caption{Attribute Rank Updating (ARU) algorithm}
\begin{algorithmic}
\State \textbf{input :} \( [x_i]_H \) with \( \varphi \)
\State \textbf{output:} attribute rank of a genealogical brotherhood
\For {\( i = 1 \) to \( |n'| \)}
\State Obtain the maximum and minimum values of \( c_i \)
\For {\( j = 1 \) to \( |n'| \)}
\State \( \text{Assign } f(x_j, c_i) = m_i \text{ into the corresponding queue; } \)
\EndFor
\State Set \( r(c_i) = 1 \)
\State Calculate \( r(c_i) \) by Equ. (15);
\EndFor
\State Sort the attributes by \( r(c_i) \) and output.
\end{algorithmic}
\end{algorithm}

\begin{algorithm}[H]
\caption{Hierarchical Binary Classifying (HBC) algorithm}
\begin{algorithmic}
\State \textbf{input :} \( H \) and \( [x_i]_H \) with \( \varphi \)
\State \textbf{output:} Candidate multiple reducts set \( RED' \)
\State \textbf{Call} Algorithm 1 to calculate rank \( r \) of every attribute in \( C - H \);
\State \textbf{Generate} the ancestor branches according to their attribute ranks;
\For {\( r \neq 0 \)}
\State \textbf{Visit} an ancestor following the ancestor branch one-by-one;
\State Obtain the attribute combination set \( Z \);
\If {\( \Omega \) is satisfied}
\State \textbf{Current} attributes are collected into set \( HA \);
\State \textbf{Obtain} the attribute combination set \( Y \);
\State \textbf{Current} objects \( U_O \) are partitioned into \( [x_y]_Y \) \( \forall x \)
\State \textbf{Calculate} rank \( r \) of every attribute in \( HA \)
\With \( U_O \) by calling Algorithm 1;
\State \textbf{Generate} the left child node according to its attribute ranks;
\Else
\State \textbf{Goto} the youngest sibling in the brotherhood;
\EndIf
\State \textbf{Collect} the attributes in this genealogical generation path as a candidate reduct and add it into \( RED' \);
\EndFor
\end{algorithmic}
\end{algorithm}
current considered attribute combination will be combined into the current considered attribute combination following any branch of the genealogical tree from the root to the attribute node with rank \( r = 0 \). After completing the four tasks, all candidate reducts are checked and redundancies removed, and the algorithm outputs the complete reduct sets. Our variable precision reduction (CVPR) algorithm is summarized in Algorithm 3.

**Algorithm 3:** Complete Variable Precision Reduction (CVPR) Algorithm

<table>
<thead>
<tr>
<th>Input</th>
<th>( T ) and ( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>Complete reduct set ( \text{RED} )</td>
</tr>
</tbody>
</table>

1. Formulate the distribution table \( T_D \);
2. Candidate complete reduct set \( \text{RED}' = \emptyset \) and complete reduct set \( \text{RED} = \emptyset \);
3. \textbf{do}
   4. Choose the first candidate reduct as \( H \) in \( \text{RED}' \);
   5. Current objects are partitioned into all of \([x]_H \) in \([x]_H \);
   6. Assign \( \varphi \) values of the objects using Eq. (14);
   7. Call Algorithm 2 to obtain the candidate reduct set \( \text{RED}'_H \) set;
   8. if \( \text{RED}'_H = \text{null} \) then
      9. Set \( \text{RED}'_H = \{R_1, \ldots, R_j, \ldots, R_n\} \);
      10. for \( j = 1 \rightarrow n \) do
          11. if all of \([x]_R_j \) satisfy Theorem 2 then
              12. Union \( R_j \cup H \) into \( \text{RED}' \);
          else
              13. Union \( R_j \cup H \) into the end of \( \text{RED}' \);
          end
      end
   end
   19. Remove the first candidate reduct from \( \text{RED}' \);
20. while \( \text{RED}' = \text{null} \);
21. Remove the redundant reduct(s) in \( \text{RED} \);

In Algorithm 3, Step 1 takes \( O(|U||C|) \) time to formulate \( T_D \). The algorithm chooses the attribute sets at least once, on average \( 2^{|C|}/|C| \) times and at most \( 2^{|C|} \) times. Therefore the operation has \( O(2^{|C|}/|U||C|) \) average case time complexity from Steps 3 to 20. Step 21 involves removing reducts with redundant attributes which can be done in \( O(4^{|C|}) \) time. Therefore the best, worst and average case time complexity of Algorithm 3 is \( O(|U||C|), O(4^{|C|}/|U||C|) \) and \( O(4^{|C|}/|U||C|) \), respectively.

In general, \(|C| < |U| \), however, the rate of exponential growth with respect to \( C \) is far faster than the quadratic growth with \( U \), hence in Algorithm 3 the number of attributes has more influence than the number of objects in terms of computational complexity. The pruning strategy is a very important mechanism for reducing the number of objects and consequently decreasing algorithm time complexity.

**Theorem 8:** \( \text{RED}(\beta) \) extracted by the CVPR algorithm is the complete reduct set.

**Proof:** From Theorem 7, the complete reduct set is a subset of \( P^{G(C)} \) in the genealogical binary tree \( G(C) \). According to Lemma 1, the genealogical paths of these attributes in \( G(C) \) are pruned during hierarchical binary classifying from Steps 7 to 16 in Algorithm 2 once there is no candidate reduct in the paths. The redundant attribute combinations are removed at Step 21 in Algorithm 3. The theorem follows.

The reduct lattice and its comprehensive knowledge system can be built once the complete reduct set \( \text{RED}(\beta) \) is obtained by the CVPR algorithm. The comprehensive knowledge system \( K = \{U_D, A, I\} \) is considered as a formal context. Then \( c'_i \) and \( d' \) in every knowledge lattice are determined through formal concept analysis [47]. The double-layer structure of the comprehensive knowledge system is established as in Definition 12.

V. EXPERIMENTS, RESULTS AND DISCUSSION.

In this section, our complete attribute reduction algorithm is evaluated based on an implementation in the C language and a computational environment consisting of an Intel® Core(TM) i5-3230M CPU @2.60GHz processor with 4G memory. We begin by providing a relatively simple example to illustrate comprehensive knowledge extraction and then demonstrate the performance of our algorithm on 30 benchmark datasets with different instances and attributes.

A. Benchmark Problems

To illustrate the effectiveness and performance of the proposed CVPR algorithm, we evaluate its performance on 30 well-known benchmark datasets from the UC Irvine machine learning repository\(^1\). Some of these datasets (e.g. heart, iris, glass) are frequently used to test classification methods. We also consider some of the newer datasets, such as Connectionist Bench, Fertility, and MAGIC Gamma Telescope. The average number of attributes in the datasets is 12 and the maximum number is 60. The average number of objects is 1,980, with ten datasets having more than 1,000 objects and the maximum number is 19,020. The results obtained with different \( \beta \) values are presented and discussed in Subsection V-B. In this section we consider the datasets listed in Table I with \( \beta = 1 \) to enable a fair comparison with the alternative approaches considered.

In the case of the heart dataset, the ancestor of its attribute genealogical tree is \( c_5 - c_8 - c_{10} - c_4 - c_1 - c_2 - c_3 - c_6 - c_7 - c_9 - c_{11} - c_{12} - c_{13} \). The time performance curve during extraction of complete reducts without pruning is illustrated in Figure 3(a). Searching the left subtree of \( c_5 \), we obtain the reducts from the first to the 53rd reduct. Accordingly, \( c_8 \) corresponds to the 54th to the 83rd reduct, \( c_{10} \) corresponds to the 84th to the 95th reduct, \( c_4 \) corresponds to the 96th to the 106th reduct, and \( c_1 \) corresponds to the 107th to the 109th reduct. With regard to the child generation, for example, \( c_{10} - c_4 \) corresponds to the 84th to the 91st reduct, and \( c_{10} - c_4 \) corresponds to the 92nd to the 95th reduct.

Figure 3(b) shows the time performance curve comparison during the extraction of complete reducts from the heart

\(^1\)http://archive.ics.uci.edu/ml/
dataset with and without pruning. The pruning strategy saves significantly on computational time without omitting any reduct. When traversing the ancestors $c_5 - c_8 - c_{10} - c_4 - c_1$, the child nodes from $c_2$ to $c_{13}$ do not satisfy the $\Omega$ condition, hence all of them are pruned. Even traversing the first 5 attribute nodes (i.e., $c_5 - c_8 - c_{10} - c_4 - c_1$), the complete traverse procedure takes much longer on five slopes, denoted $\begin{array}{l} 1, 2, 3, 4 \text{ and } 5 \end{array}$, respectively. For example, to obtain the 95th reduct after traversing $c_{10}$, takes nearly 3000 ms to get the next reduct (i.e. the 96th reduct) without pruning (the child branches of $c_{10}$). Since $c_{10}$’s child node $c_4$ has eight younger siblings, there are $2^8 - 1$ nodes to be traversed and calculated. Even when sharing one father node, child branch pruning also makes a difference. For example, under the ancestor $c_{10}$, it takes nearly 1000 ms from traversing its child $c_1$ to traversing its child $c_4$ without pruning, as shown in Figures 3(a) and 3(b). In general, there is a positive correlation between the time cost and the number of nodes. The idea of the pruning strategy is that if one attribute node cannot be included in a reduct, then none of its child nodes can be included in the reduct either.

There are three kinds of attribute node in Figure 4, the black ones which are traversed normally, the red one which are skipped, and the blue ones which are pruned. Since the red nodes are rank $r(c) = 0$, it is impossible for the candidate reduct to include more attributes. The blue ones do not satisfy the $\Omega$ condition, and therefore are not valid to include in any reduct. In other words, the red and blue nodes will not generate child nodes, i.e. our pruning strategy is triggered. For example, traversing the path $c_5 \rightarrow c_4 \rightarrow c_9$, we get $c_9$ with rank $r(c) = 0$, the candidate reduct $\{c_5, c_4, c_9\}$ is obtained, and the left child nodes of $c_9$ are not traversed. Instead $c_{10}$ is traversed. Note that the attributes of child nodes of one node are the same as its younger siblings, only with different ranks. Since the child node attribute combination set $c_2, c_3, c_{12}, c_{13}$ of $c_6$ in the attribute set $\{c_5, c_4, c_6\}$ cannot partition its objects, it is not possible for its subset to partition the objects inerrably, hence the branch $c_2 - c_3 - c_{12} - c_{13}$ is pruned.

In Figure 4, there are two paths $c_7 - c_{11} - c_2 - c_3 - c_{12} - c_{13}$ and $c_6 - c_{13} - c_2 - c_3 - c_7 - c_{11} - c_{12}$. Although the attributes are the same with the exception of $c_6$, they have different attribute ranks. They provide dynamic opportunities to join the candidate reduct(s), which is obviously different from the importance of attributes in the model [49]. This is a static quantity and only equivalent to our attribute ranks in the ancestor branch in Figure 4. In the heart dataset there is a reduct $\{c_5, c_8, c_7, c_{10}\}$. According to the classical importance of attributes approach to obtaining the reduct, after selecting $c_5$ and $c_8$, $c_{10}, c_4, c_1, c_2, c_3$, and $c_6$ would be considered successively. While we select the attributes following the genealogical tree generated through these attribute ranks, after selecting $c_5$ and $c_8$, $c_7$ should be considered, since our attribute ranks are dynamic. After $c_5$ and $c_8$ are selected the objects which need to be considered are reduced with the remaining attributes in a new distribution table. The attribute ranks are updated accordingly, which is very helpful for obtaining the reduct(s) successfully.

In Figure 4, there are two different genealogical generation paths $c_5 \sim c_4 \sim c_4 \sim c_7 \sim c_{12}$ and $c_5 \sim c_4 \sim c_7 \sim c_{12}$, with the former contained in the latter. The former is redundant with respect to the latter. The reason is that when combining the attribute set with different sequences to obtain the candidate reduct, they both partition all the objects equally. Therefore it is necessary to remove the redundancies in the collected reduct set.

In Figure 5, the left subfigure shows the comparison be-
between the number of normally traversed nodes and the number of pruned nodes about the child nodes of the five first attributes in the ancestor branch for the heart dataset. There are 12 nodes in the next generation of \( c_5 \), as shown in the left bar of the left subfigure, no node is skipped, five pruned, and seven (i.e. \( c_1, c_8, c_4, c_9, c_{12}, c_3, c_{11} \)) normally traversed. For the next generation of \( c_5 \), the right subfigure of Figure 5 shows further the comparison among the three kinds of node in terms of their visit states. It is clear that the number of normally traversed nodes is a small portion of the nodes in the whole genealogical binary tree in our algorithm, with most nodes either skipped or pruned. Our pruning strategy reduces computational costs significantly. Figure 6 shows more performance comparison curves during reduct extraction for the seismic-bump, Credit Approvals and ILPD datasets.

**TABLE I: Considered datasets and the reduct number**

<table>
<thead>
<tr>
<th>Data set</th>
<th>No</th>
<th>Na</th>
<th>Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Approval</td>
<td>690</td>
<td>15</td>
<td>60</td>
</tr>
<tr>
<td>heart</td>
<td>270</td>
<td>13</td>
<td>109</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>5759</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Lenses</td>
<td>24</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>345</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Statlog (German Credit Data)</td>
<td>1000</td>
<td>20</td>
<td>846</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>Abalone</td>
<td>4177</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>Auto MPG</td>
<td>398</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>banknote</td>
<td>1372</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Blood Transfusion</td>
<td>748</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Breast Cancer Wisconsin</td>
<td>699</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>Breast Tissue</td>
<td>106</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Connectionist Bench</td>
<td>208</td>
<td>60</td>
<td>1314</td>
</tr>
<tr>
<td>Contraceptive Method Choice</td>
<td>1473</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>EEG Eye</td>
<td>14980</td>
<td>14</td>
<td>597</td>
</tr>
<tr>
<td>Fertility</td>
<td>100</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>ILPD</td>
<td>583</td>
<td>10</td>
<td>56</td>
</tr>
<tr>
<td>Knowledge Modeling Data</td>
<td>403</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>MAGIC Gamma Telescope</td>
<td>19020</td>
<td>10</td>
<td>32</td>
</tr>
<tr>
<td>Qualitative Bankruptcy</td>
<td>250</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>seeds</td>
<td>210</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>seismic-bumps</td>
<td>2584</td>
<td>18</td>
<td>34</td>
</tr>
<tr>
<td>SPECT Heart</td>
<td>80</td>
<td>22</td>
<td>725</td>
</tr>
<tr>
<td>Thoracic Surgery Data</td>
<td>470</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>Wine Quality (red)</td>
<td>1599</td>
<td>11</td>
<td>227</td>
</tr>
<tr>
<td>Wine Quality (white)</td>
<td>4898</td>
<td>11</td>
<td>127</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

\( ^0 \) Number of Objects.
\( ^A \) Number of Condition Attributes.
\( ^R \) Number of Reducts.

In the seeds dataset, the three decision values are the three different varieties of wheat: Kama, Rosa and Canadian. The objects with a decision value of 1 are selected and their classifying value, \( \phi \) set to 1 at Step 6 in Algorithm 3. After the first calling of Algorithm 2 at Step 7, all 21 reducts are obtained, because \( RED' = \emptyset \) at Step 20, which leads directly to Step 21 and then to the end of the whole algorithm. When using the classic classification quality \( \gamma \) [38] to evaluate the attribute’s importance (indicating the attribute’s impact on all decision classes), there is little difference between all attributes. However our attribute ranks are calculated by considering only two classes during any binary classifying with binary classifying label \( \phi \). The significant differences between attributes are very helpful in determining straightforwardly
whether the attribute and its attribute combinations should be included in a candidate reduct or not. For the Contraceptive Method Choice dataset, the search stops once the first binary classifying is completed, since it is impossible to include any attribute to obtain a reduct. In other words, there is no reduct in the dataset. The binary classifying strategy speeds up distinguishing whether there is a candidate reduct or not. Of course, the binary classifying strategy also makes it possible to generate the genealogical binary tree with the attribute ranks. The fourth column in Table I shows the number of complete reducts in each dataset identified by the CVPR algorithm.

In Table I, the highest abstracting proportion defined in Equ. (13) reaches 58.80% for the Qualitative_Bankruptcy dataset. The proportion is more than 30%, for the Blood Transfusion and Breast Cancer Wisconsin datasets, while the proportions for the SPECT Heart, Wine Quality (red) and Wine Quality (white) datasets are 18.75%, 15.01% and 19.13%, respectively.

Table II provides a comparison of the results obtained using the CVPR algorithm and the WADF (worst-attribute-drop-first) algorithm [16]. For the relatively small scale datasets, i.e. credit approval, heart, iris and liver disorders, CVPR and WADF algorithms both determined all of the reducts, while only the CVPR algorithm identified the complete reduct set for the relatively large scale datasets, i.e. Ionosphere and Statlog.

In WADF, the maximum number of multi-reducts that can be identified is the number of non-core attributes in the seed reduct, and the reducts which the algorithm can identify is strongly dependent on the seed reducts. For example, WADF is able to extract all reducts from the Credit Approval and heart datasets, but not from the Ionosphere and Statlog datasets.

### Table II: Comparisons about the number of obtained multi-reducts

<table>
<thead>
<tr>
<th>dataset</th>
<th>WADF [16]</th>
<th>$\vartheta$</th>
<th>CVPR</th>
<th>$\vartheta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Approval</td>
<td>60</td>
<td>100%</td>
<td>60</td>
<td>100%</td>
</tr>
<tr>
<td>heart</td>
<td>109</td>
<td>100%</td>
<td>109</td>
<td>100%</td>
</tr>
<tr>
<td>iris</td>
<td>4</td>
<td>100%</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>Lenses</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>9</td>
<td>100%</td>
<td>9</td>
<td>100%</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>32</td>
<td>5.56%</td>
<td>5759</td>
<td>100%</td>
</tr>
<tr>
<td>Statlog</td>
<td>16</td>
<td>18.9%</td>
<td>846</td>
<td>100%</td>
</tr>
</tbody>
</table>

To compare the performance of the CVPR and FSRS (fuzzy swarm rough set) algorithms [45], we firstly test a classical algorithm, the BinTree algorithm [32], in which only the CVPR algorithm identified the complete reduct set for the Glass dataset. Here all 18 reducts are obtained within 593 ms reflecting the fact that CVPR algorithm is a deterministic algorithm that considers distribution information and employs effective pruning strategies to reduce the computational overhead.

**Fig. 7:** Multiple reducts processing using FSRS

**Fig. 8:** Reduction performance curve using CVPR from the Glass dataset

The CVPR algorithm is designed based on the theory of comprehensive knowledge extraction introduced in the paper. It is a deterministic search procedure in which dynamic attribute rank is used as heuristical information to traverse its genealogical tree, such that the completeness of the reduct set is ensured. This contrasts with which is different obviously from WADF and FSRS which do not guarantee completeness. Furthermore, CVPR algorithm is optimised with respect to computational overhead through the use of distribution table abstracting, hierarchical binary classifying and genealogical tree pruning.

The numbers of complete reducts for each of the 30 benchmark datasets is listed in Table I, with all reducts satisfying Definition 2. The number of objects in the datasets has little influence on the algorithm search time and completeness of reducts.
system, and then only the objects in the subsequent equivalence classes with \( |x_i|_H/D \neq 1 \wedge \omega = 1 \) are considered during binary classification. The number of attributes in the datasets has a relatively larger influence on CVPR algorithm performance. The pruning strategy is very helpful in obtaining sound and complete reducts. The average case time complexity of CVPR is \( O(4^{|C|}/|C|) \), which also verifies further the theoretical analysis of the time complexity of algorithms and theoretical proofs of completeness in Section IV.

B. From Information Systems to Double-layer Lattice Structures

The previous section illustrated the effectiveness and performance of the proposed CVPR algorithm. Here we provide an illustrative example of how a given information system can be represented as double-layer lattice structures.

Table III shows an instance of a decision table in which the condition attributes are \( c_1, c_2, c_3, c_4 \) and \( d \), and \( \beta \) is the decision attribute. Its condition equivalence classes are \( \{x_1\}, \{x_2, x_5, x_{10}\}, \{x_3\}, \{x_4, x_5, x_{11}\}, \{x_7\} \) and \( \{x_9\} \). The universe of discourse is \( \{x_1, x_2, x_3, x_4, x_5, x_7\} \) in \( T_D \).

The \( POS^{C\beta}(D) \) and complete reduct sets corresponding to different values of \( \beta \) are shown in Table IV. When \( \beta = 0.70 \), \( x_6 \) is added to \( POS^{C\beta}(D) \). If \( \beta \) is changed to 0.60, \( x_9 \) is also within \( POS^{C\beta}(D) \). Because \( POS^{C\beta}(D) \) does not change between \( \beta = 1 \) and \( \beta = 0.80 \), the complete reduct set is also unchanged. For the complete reduct set with \( \beta = 0.7 \), the comprehensive knowledge system is illustrated in Table V.

<table>
<thead>
<tr>
<th>TABLE III: Decision table</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>( x_1 )</td>
</tr>
<tr>
<td>( x_2 )</td>
</tr>
<tr>
<td>( x_3 )</td>
</tr>
<tr>
<td>( x_4 )</td>
</tr>
<tr>
<td>( x_5 )</td>
</tr>
<tr>
<td>( x_7 )</td>
</tr>
<tr>
<td>( x_9 )</td>
</tr>
<tr>
<td>( x_{10} )</td>
</tr>
<tr>
<td>( x_{11} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE IV: Complete reduct sets for Table III</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta )</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>1.00</td>
</tr>
<tr>
<td>0.80</td>
</tr>
<tr>
<td>0.70</td>
</tr>
<tr>
<td>0.60</td>
</tr>
</tbody>
</table>

When \( \beta \) is set to three different values, three comprehensive knowledge systems are extracted accordingly. Their completeness metric results with different \( \beta \) are illustrated in Table VI. The multi-knowledge system \( \xi \) is the same, because it is determined by the original information system, while the knowledge system comprehensiveness \( \xi \) is influenced by its precision parameter \( \beta \). In the classical rough set model (i.e. \( \beta = 1 \)), \( \xi \) is only 0.5, while \( \xi = 1 \) when \( \beta \) decreases to 0.6. Everywhere, the basic rule ratio \( \xi \) is 1, which indicates our algorithm can extract the comprehensive knowledge from the given information system.

The double-layer structures of the comprehensive knowledge system with \( \beta = 0.7 \) values are presented in Figure 9. The complete reduct set lattice is depicted in the upper left while the comprehensive knowledge lattice is shown in
the lower right of each plot in Figure 9. The relationship $\epsilon^*$ between the two types of lattice is highlighted using dashed red lines. 

$$RED = \{\{c_1, c_2, c_3\}, \{c_1, c_4, c_5\}, \{c_1, c_6, c_0\}\} \text{ when } \beta = 0.7 \text{ in Figure 9. Node } \mathcal{O} \text{ is an upper-neighbor of the maximum node 28 and its connotation is } c_1, c_2, c_3, c_0 \rightarrow Y(\mathcal{O}) = \{2, 8, 9, 10, 11, 12, 13\}. \text{ And } \{c_1, c_2, c_3\} \rightarrow Y(\mathcal{O}) = \{2, 8, 9, 10, 11, 14, 15, 16\}. \text{ The connotation of node } \mathcal{O} \text{ is } c_1, c_2, c_3, d_1, \{c_1, c_2, c_3, d_1\} \text{ is } c_1, c_4, c_5, d, \text{ so } \{c_1, c_4, c_5\} \rightarrow Y(\mathcal{O}) = \{2, 8, 9, 10, 11, 12, 13\}. \text{ And } \{c_1, c_2, c_3\} \rightarrow Y(\mathcal{O}) = \{2, 8, 9, 10, 11\}.$$ 

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a theory of complete variable precision knowledge extraction has been developed. The concept of complete reduct set and knowledge have been proposed. An efficient complete variable precision reduction (CVPR) algorithm which is able to extract comprehensive knowledge has also been developed. This is achieved by transforming the information system into a distribution table, which only collects one object from each equivalence class in a given information system. It is an essential abstract from the original universe of the object discourse. Then the objects in the distribution table are binary classified hierarchically. A genealogical tree is generated according to the rank of attributes, which facilitates dynamically combining attributes and triggering of a pruning strategy to obtain the complete reduct set with significantly low computational cost. The theoretical proof and experimental results presented illustrate that the CVPR algorithm is complete. The comprehensive knowledge system structure is presented through concept lattices and is visualized using Hasse diagrams.

Our CVPR algorithm is suitable for the variable precision reduction and can also be degenerated to a classic rough set model. The underlying approach can also degenerate to give a single minimum reduce, since the first reduct obtained tends to be the minimum reduce. The earlier a reduct is obtained, the fewer attributes are involved, due to the hierarchical binary classifying strategy employed.

The genealogical tree employed is a left-child-right-sibling binary tree. Binary trees and forests can be converted to each other [50]. Therefore in a genealogical binary tree, the offspring of the nodes in a brotherhood will not be mutually interfering when there are similar ways to search for reducts. It is feasible to design a parallel complete reduction algorithm with mapreduce, which will be considered in future work. It is also of interest to explore the application of CVPR algorithm to the analysis of multiple pathways in brain neural networks and big data networks.

ACKNOWLEDGMENT

The authors sincerely thank the editors and the anonymous reviewers for the very helpful and kind comments with regard to improving the presentation of our paper.

REFERENCES


VII. SUPPLEMENTAL MATERIALS I

Table VII is a decision table where $U/C = \{\{x_1, x_3\}, \{x_2\}, \{x_4, x_5, x_7\}, \{x_6, x_8\}\}$. The distribution tables with $\beta = 1$ and $\beta = 0.6$ are shown in Table VIII according to Definition 3. Note that $x_4$ is in the positive region with $\beta = 0.6$ but it is not with $\beta = 1$. The decision values of $x_4$ are adjusted by Equ. (4) as shown in Table VIII and the values of $\omega$ are obtained according to Equ. (5).

**Table VII: A decision table**

<table>
<thead>
<tr>
<th>Object</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_2$</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$x_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_4$</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_5$</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$x_6$</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$x_7$</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$x_8$</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table VIII: The distribution tables with different $\beta$**

<table>
<thead>
<tr>
<th>Object</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$\beta = 1$</th>
<th>$\beta = 0.6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d$</td>
<td>$\omega$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table IX is a decision table on playing tennis. The condition attributes set is $\{\text{Outlook, Temp, Humidity, Wind}\}$ and $\{\text{Play}\}$ is the decision attribute. $RED$ of playing tennis decision table is $\{\{\text{Outlook, Humidity, Wind}\}, \{\text{Outlook, Temp, Wind}\}\}$.

**Table IX: A decision table about playing tennis**

<table>
<thead>
<tr>
<th>Object</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Playing</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>$x_3$</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_6$</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>$x_7$</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_8$</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>$x_9$</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

In Table IX, there are 14 actual objects, i.e. $C(U)=14$. $|\nu_{\text{Outlook}}| = 3$ because $\nu_{\text{Outlook}} = \{\text{Sunny, Overcast, Rain}\}$. $V_C = 3 \times 3 \times 2 \times 2 = 36$ by Equ. (9), so the theoretical number of objects is 36. The first and eighth objects have the same condition attribute values according to $\{\text{Outlook, Humid, Wind}\}$, and the fifth and tenth objects are also the same. Because $|\nu_{\text{Temp}}| = 3$, the number of the knowledge from $\{\text{Outlook, Humid, Wind}\}$ is $(14 - 2) \times 3 = 36$. Likewise, the cardinality of the knowledge from $\{\text{Outlook, Temp, Wind}\}$ is 24. So $|\Psi(\beta)| = 36$ without redundancy. $\xi = \frac{14}{36} = \frac{7}{18}$, $\hat{\xi} = 1$ and $\xi = 1$ according to Equs. (10) to (12).

The double-layer structures of the comprehensive knowledge system with $\beta = 1$ and $\beta = 0.6$ from Table III are presented in Figure 10.
Fig. 10: Double-layer structure of the comprehensive knowledge system with different $\beta$
VIII. SUPPLEMENTAL MATERIALS II

An instance procedure is illustrated in Figure 11. There are two decision values, i.e., 0 and 1, which are partitioned by the red curves. The central white region is $POS^d_C(D)$ and the peripheral gray region is $NEG^d_C(D)$, as shown in Figure 11. Based on the attribute rank obtained using Algorithm 1, its attribute genealogical binary tree $G$ can be generated to combine attribute sets sequentially. Suppose we get an attribute combination set $B_1$, which some equivalence classes satisfy $\omega(x_i) = 1$, i.e., $[x_1]_{B_1}, [x_2]_{B_1}, [x_3]_{B_1}$ while all other satisfy $\omega(x_i) = 0$, i.e., $[x_4]_{B_1}, [x_5]_{B_1}, [x_6]_{B_1}$ in Figure 11(b). Note that $[x_1]_{B_1}$ and $[x_2]_{B_1}$ do not satisfy $|[x_i]_{B_1}/D| = 1$. Now we only focus on these two equivalence classes in the next steps. An object set can be collected from the objects in $[x_1]_{B_1}$ and $[x_2]_{B_1}$ in Figure 11(c) with the attributes $C - B_1$. Accordingly for this object set, all attribute ranks in $C - B_1$ are re-calculated and the corresponding subtree of the attribute genealogical binary tree $G$ also continues to generate further to the current attribute nodes. From this attribute subtree, suppose we get an attribute combination set $B_2$ to partition $[x_1]_{B_1}$ in Figure 11(c) into $[x_{11}]_{B_2}$ and $[x_{12}]_{B_2}$, and to partition $[x_2]_{B_1}$ into $[x_{21}]_{B_2}, [x_{22}]_{B_2}$ and $[x_{23}]_{B_2}$, as shown in Figure 11(d). Thus far, all equivalence classes of $B_1 \cup B_2$ satisfy Theorem 2. It means that $\forall x_i \in [x_i]_{(B_1 \cup B_2)}$, $\omega(x_i) = 0$ or $\omega(x_i) = 1 \land |[x_i]_{(B_1 \cup B_2)}/D| = 1$. Hence, a reduct is obtained as $B_1 \cup B_2$. 

![Fig. 11: Multi-binary classification for a reduct](image-url)
IX. SUPPLEMENTAL MATERIALS III

Considering a simple example to demonstrate our algorithms, Table X and $\beta = 1$ are used as the input of Algorithm 3. $U_D = \{(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0), (x_6, 1), (x_8, 1)\}$ in $T_D$ can be obtained using Equs. (4) and (5) in Step 1. The $\varphi$ values of $x_1$ and $x_4$ are assigned 1 by Equ. (14) in Step 6. When calling Algorithm 2 in Step 7, $r(c_4) = 0, r(c_5) = -1, r(c_3) = -1, r(c_2) = -2, r(c_1) = -2$ according to Algorithm 1 using Equ. (15) and $RED'_H = \{\{x_4\}, \{x_5, x_3\}, \{x_5, x_2\}, \{x_3, x_2, x_1\}\}$. Every candidate reduct in $RED'_H$ is checked to see if it is a reduct or not in Step 11. Then $RED = \{\{x_4\}, \{x_5, x_3\}, \{x_5, x_2\}, \{x_3, x_2, x_1\}\}$ and $RED' = \emptyset$ from Steps 12 to 14. The algorithms end and the complete reduct set $RED$ is returned.

TABLE X: A decision table for the algorithm demonstration

<table>
<thead>
<tr>
<th>Object</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_4$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$x_8$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
X. SUPPLEMENTAL MATERIALS IV

The flowchart of our attribute reduction algorithm is presented in Figure 12.

Fig. 12: Flowchart of the complete attribute reduction algorithm