# An Algorithm for Computing Attribute Reducts Based on Graph Search Strategy

Minghui Shi, Changle Zhou, Fei Chao, Min Jiang
Department of Cognitive Science, Xiamen University, Fujian, China
Fujian Key Lab of the Brain-like Intelligent System, Fujian, China
Xiamen, China
smh@xmu.edu.cn; playtide@gmail.com; academic_mail@126.com

***Abstract*—Attribute reducts can discover previously unknown, non-trivial and useful abstractions from the data in large databases. However, many methods for finding attribute reducts from large data sets always meet a difficult problem of combination explosion. To overcome the problem and find some attribute reducts with high efficiency, the algorithm CARHS was proposed. The basic idea of CARHS is: 1) transform the problem into an equivalent one that searches paths, from which attribute reducts can be easily derived, from a graph; 2) employ high efficient heuristic rules during the course of depth-first search on the graph. By means of the heuristic rules, those paths that would not derive attribute reducts could be blocked as early as possible, furthermore, for those paths that would derive the same attribute reduct, only one of them could complete the course of search, and the others could be blocked as early as possible. Thus some attribute reducts could be found by CARHS with high efficiency even when dealing with huge data sets. The transformation of the problem, novel concepts, the heuristic search rules, and the algorithm CARHS were illustrated in detail by some examples. At last, The experiment on three classic UCI data sets showed the effect of the heuristic search rules and the efficiency of the algorithm CARHS.**

***Index Terms*—attribute reduct, data mining, reduct discernibility graph, machine learning**

## I. Introduction

In recent years, there are more and more huge data available to be processed by machine learning technologies. Knowledge acquisition from the data in large databases has become possible and has realistic significance. On the one hand, potential knowledge that human experts haven't known could be mined out automatically. On the other hand, the past experts' knowledge could be examined, verified, or even revised.

However, mining useful information from original data is often difficult or time-consuming, especially for huge data sets. Attribute reduct, which is one of the core subjects of the rough set theory [1], provides an an efficient way to discover potential, unknown, non-trivial and useful abstractions from the data in large databases. The goal of attribute reduct is to transform high dimensional data into low dimensional one by removing redundant attributes while keeping the useful information invariant. Although attribute reduct doesn't directly find knowledge, it conduces to delete the redundant data and reduce the computational complexity of the subsequent methods for knowledge discovery. Thus it has become one of focus topics in the fields such as data mining, knowledge discovery and machine learning.

So far, there are several methods for computing attribute reducts that are based on discernibility matrix [2], information entropy [3], positive region [4], genetic algorithm [5], etc. These methods can efficiently process some data sets and get a few attribute reducts. However, when dealing with huge data sets or computing more attribute reducts, they always meet a problem of combination explosion, which is called a NP-hard problem [6].

To find attribute reducts from huge data with high efficiency, in [7], we proposed an approach based on graph search strategy. Although [7] did not provide a complete algorithm, it indeed pointed out a meaningful direction: one can transform the problem of computing attribute reducts into the graph search problem.

Based on the past work [7], the algorithm CARHS was proposed in this paper. CARHS first constructs a reduct discernibility graph (RDG), then searchs those paths in RDG from which attribute reducts can be directly derived. CARHS employs four heuristic search rules: *Member Exclusive*, *Friend Dissuade*, *Barrier-layer Block*, *Stranger Join*. It has been proofed that during the depth-first search with these heuristic rules, those paths that are useless to derive attribute reducts can be blocked as early as possible, while for those paths that may derive the same attribute reduct, only one of them can complete the course of search, that is, the others can be blocked as early as possible. This ensures CARHS can compute attribute reducts with high efficiency even when dealing with huge data sets.

The following are organized as follows. In section 2, some novel and important concepts are systematically introduced. Then the principle to compute attribute reducts as well as its main difficulty when dealing with large data sets are discussed in section 3. To overcome the difficulty, the heuristic search rules are developed and the algorithm CARHS is presented in section 4. Section 5 provides an example to illustrate the algorithm CARHS and the heuristic search rules. To examine the algorithm CARHS, in section 6, CARHS is applied to deal with three classic UCI data sets [9]. Section 7 summarizes the whole paper.

## II. Concepts

In this section, we provide some examples to illustrate some basic concepts of rough set theory [1], and illustrate some

TABLE I
AN EXAMPLE OF INFORMATION SYSTEM (DECISION SYSTEM)

| | a | b | c | d |
|---|---|---|---|---|
| $u_1$ | 1 | 2 | 3 | 1 |
| $u_2$ | 2 | 3 | 3 | 2 |
| $u_3$ | 1 | 3 | 2 | 1 |
| $u_4$ | 2 | 1 | 1 | 2 |

TABLE II
AN EXAMPLE OF DISCERNIBILITY MATRIX

| | $u_1$ | $u_2$ | $u_3$ | $u_4$ |
|---|---|---|---|---|
| $u_1$ | $\phi$ | | | |
| $u_2$ | {a,b} | $\phi$ | | |
| $u_3$ | $\phi$ | {a,c} | $\phi$ | |
| $u_4$ | {a,b,c} | $\phi$ | {a,b,c} | $\phi$ |

novel concepts, most of which are proposed in our past work [7], that is important for understanding the following content.

***Information system***

In rough set theory, *Information system* can be presented by a table as Tab. I, and can be defined as: $IS = (U, A, V, f)$, where $U = \{u_1, u_2, \cdots, u_n\}$ is a set of objects; $A$ is a set of attributes; $V = \bigcup_{a_i \in A} V_{a_i}$, where $V_{a_i}$ is the domain of $a_i$; $f$ is an information function, $f : U \times A \to V$ .

For example, in Tab. I, $U = \{u_1, u_2, u_3, u_4\}$, $A = \{a, b, c, d\}$, $V_a = V_d = \{1, 2\}$, $V_b = V_c = \{1, 2, 3\}$, $V = \{1, 2, 3\}$.

***Decision system***

*Decision system* is also an information system, however, its attribute set $A$ may be divided into a condition attribute set $C$ and a decision attribute set $D$, which can determine the main characteristic (class) of objects. So a decision system may be denoted by $DS = (U, C \cup D, V, f)$.

In order to simplify the discussion and the implementation, in this paper we mainly discuss the decision system with discrete values and only one decision attribute, which is denoted by $DS = (U, C \cup d, V, f)$.

This simplification would not decrease the significance for general information systems, since 1) a general information system can always be treated as a decision system, if one attaches it an additional decision attribute according to the practical purpose; 2) although the range of $f$ is continuous in general, continuous values can be transformed into discrete values by some discretization methods; 3) if there are more than one decision attribute in the decision attribute set $D$, one can always transform them into one decision attribute [8].

In the following, a decision system $DS = (U, C \cup d, V, f)$ also can be understood as a classification system, in which the values of $d$ determine the classes that objects belong to, and Tab. I is treated as a decision system with only one decision attribute $d$.

***Discernibility attribute, discernibility attribute set***

Let $DS = (U, C \cup d, V, f)$, where $U = \{u_1, u_2, \cdots, u_n\}$, $C = \{a_1, a_2, \cdots, a_m\}$, and let $u_i \in U, u_j \in U, u_i \neq u_j$, a condition attribute $a_k \in C$ is a *discernibility attribute* with respect to $u_i$ and $u_j$ if $f(u_i, a_k) \neq f(u_j, a_k)$ and $f(u_i, d) \neq f(u_j, d)$.

For example, in Tab. I, attribute $c$ is a discernibility attribute with respect to $u_1$ and $u_4$ since $f(u_1, c) \neq f(u_4, c)$ and $f(u_1, d) = f(u_4, d)$, but $c$ is not a discernibility attribute with respect to $u_1$ and $u_2$ since $f(u_1, c) = f(u_2, c)$, and is not a discernibility attribute for $u_1$ and $u_3$ since $f(u_1, d) = f(u_3, d)$.

*Discernibility attribute set* $DAS(u_i, u_j)$ is defined as:

$$DAS(u_i, u_j) = \begin{cases} \phi & f(u_i, d) = f(u_j, d) \\ \{a_k : a_k \in C, f(u_i, a_k) \neq f(u_j, a_k)\} & f(u_i, d) \neq f(u_j, d) \end{cases}$$

For example, in Tab. I, $DAS(u_1, u_2) = \{a, b\}$, but $DAS(u_1, u_3) = \phi$ since $f(u_1, d) = f(u_3, d)$.

For ease of observation and understanding, one can list all discernibility attribute sets of $DS$ into a matrix $M(DS)$ such as Tab.II, which is referred to as *discernibility matrix* in some papers. Tab. II only displays its lower triangular part, since discernibility matrix is always symmetrical.

***Core attribute***

A condition attribute $a$ is a *core attribute* if there exists $DAS(u_i, u_j) = \{a\}$. That is, $DAS(u_i, u_j)$ has only one attribute $a$, which is necessary to discern $u_i$ and $u_j$, and is too important to be removed for keeping the classification information of the $DS$ invariant.

***Family of discernibility attribute set***

*Family of discernibility attribute set* of $DS$ is the set of discernibility attribute sets and defined as:

$$F(DS) = \{DAS(u_i, u_j) : 1 \leq i \leq n-1, i \leq j \leq n\}$$

According to Tab.II, one can easily find

$$F(DS) = \{\{a, b\}, \{a, c\}, \{a, b, c\}\} \tag{1}$$

***Absorbable and non-absorbable discernibility attribute set***

Let $F_i \in F(DS)$ , if there exists $F_j \in F(DS)$, and $F_j \subset F_i$, then $F_i$ is *absorbable*, else $F_i$ is *non absorbable*.

For example, in Eq.(1), let $F_1 = \{a, b\}$, $F_2 = \{a, c\}$, $F_3 = \{a, b, c\}$, then $F_3$ can be absorbed by $F_1$ and $F_2$, while $F_1$ and $F_2$ are non-absorbable.

***Reduct family of discernibility attribute set***

*Reduct family of discernibility attribute set* $RF(DS)$ is a set removing all absorbable discernibility attribute set from $F(DS)$.

For example, from Eq.(1), we get

$$RF(DS) = \{\{a, b\}, \{a, c\}\} \tag{2}$$

***Discernibility Subgraph***

Let $DS = (U, C \cup d, V, f)$, $F(DS) = \{F_1, F_2, \cdots, F_l\}$ $F_i \in F(DS), F_j \in F(DS)$, $1 \leq i < j \leq l$, $F_i = \{a_{i1}, a_{i2}, \cdots, a_{is}\}$, $F_j = \{a_{j1}, a_{j2}, \cdots, a_{jt}\}$, let us assume $F_i$ and $F_j$ also represent sets of vertices in a graph (that is, vertices are marked by the names of corresponding discernibility attributes in $F_i$ and $F_j$). If it does not cause confusion, we use marks (attributes' names) to represent vertices.

*Discernibility subgraph* with respect to $F_i, F_j$ is a directed complete bipartite graph as shown in Fig.1, and is denoted by $DSG(F_i, F_j)$.

For Eq.(1), $DSG(F_1, F_2)$, $DSG(F_2, F_3)$ and $DSG(F_1, F_3)$ are shown in Fig.2(a), Fig.2(b), Fig.2(c) respectively.

It should be noted that different vertices may be marked by the same attribute name. For example, in Fig.2(b), there are two vertices marked by the same attribute name $c$, but they are different vertices.

***Reduct discernibility subgraph***

Let $DS = (U, C \cup d, V, f)$, $RF(DS) = \{F_1, F_2, \cdots, F_l\}$ $F_i \in RF(DS)$, $F_j \in RF(DS)$, $i \neq j$, $F_i = \{a_{i1}, a_{i2}, \cdots, a_{is}\}$, $F_j = \{a_{j1}, a_{j2}, \cdots, a_{jt}\}$, *reduct Discernibility subgraph* with respect to $F_i, F_j$ is a directed complete bipartite graph similarly as disernibility subgraph, and is denoted by $RDSG(F_i, F_j)$.

For Eq.(2), $RDSG(F_1, F_2)$ is shown in Fig.2(a).

***Reduct discernibility graph***

Let $DS = (U, C \cup d, V, f)$, $RF(DS) = \{F_1, F_2, \cdots, F_h\}$, *reduct discernibility graph* (RDG) of $DS$ is defined as: $RDG(DS) = \bigcup RDG(F_i, F_j)$,where $F_i \in RF(DS), F_j \in RF(DS), 1 \leq i \leq h-1, j = i+1$; $RDG(DS)$ is also denoted by $RDG(DS) = (V(RDG), E(RDG))$, simply denoted as $RDG(DS) = (V, E)$, where $V$ is the set of vertices, $E$ is the set of edges.

For Eq.(2), $RDG(DS)$ is shown in Fig.2(a), where $RDG(DS)$ has only two layers since $RF(DS)$ has only two members. In general, $RF(DS)$ has more than two members, and $RDG(DS)$ has more than two layers. Fig.3 shows a bit more complicate RDG that has four layers.

***Complete path, Complete path set, Uncomplete path***

Let $G$ is a $RDG(DS)$, a *complete path* of $G$ is a path that passes through each layer once and only once. *Complete path set* of $G$ is a set containing all complete paths in $G$ and denoted by $CPS(G)$.

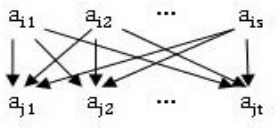


Fig. 1. Disernibility subgraph

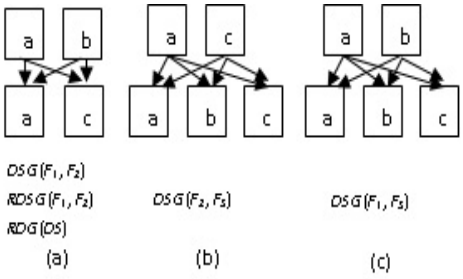


Fig. 2. Examples for DSG, RDSG and RDG.

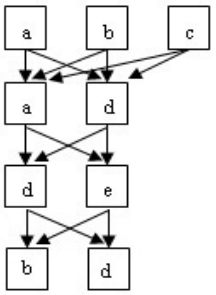


Fig. 3. An example of reduct discernibility graph

TABLE III
TOTAL PATHS WITH SEARCH PROCESS AND RESULTS

| No. | path | extend course | attribute reduct |
|---|---|---|---|
| 1 | (a,a,d,b) | $a \xrightarrow{I} a \xrightarrow{II} d \leftarrow$ Ⅰ | – |
| 2 | (a,a,d,d) | $a \xrightarrow{I} a \xrightarrow{II} d \xrightarrow{I} d$ | {a,d} |
| 3 | (a,a,e,b) | $a \xrightarrow{I} a \xrightarrow{II} e \leftarrow$ Ⅳ | – |
| 4 | (a,a,e,d) | $a \xrightarrow{I} a \xrightarrow{II} e \leftarrow$ Ⅳ | – |
| 5 | (a,d,d,b) | $a \leftarrow$ Ⅰ | – |
| 6 | (a,d,d,d) | $a \leftarrow$ Ⅰ | – |
| 7 | (a,d,e,b) | $a \leftarrow$ Ⅰ | – |
| 8 | (a,d,e,d) | $a \leftarrow$ Ⅰ | – |
| 9 | (b,a,d,b) | $b \leftarrow$ Ⅲ | – |
| 10 | (b,a,d,d) | $b \leftarrow$ Ⅲ | – |
| 11 | (b,a,e,b) | $b \leftarrow$ Ⅲ | – |
| 12 | (b,a,e,d) | $b \leftarrow$ Ⅲ | – |
| 13 | (b,d,d,b) | $b \xrightarrow{II} d \xrightarrow{I} d \xrightarrow{I} b$ | {b,d} |
| 14 | (b,d,d,d) | $b \xrightarrow{II} d \xrightarrow{I} d \leftarrow$ Ⅰ | – |
| 15 | (b,d,e,b) | $b \xrightarrow{II} d \leftarrow$ Ⅰ | – |
| 16 | (b,d,e,d) | $b \xrightarrow{II} d \leftarrow$ Ⅰ | – |
| 17 | (c,a,d,b) | $c \leftarrow$ Ⅲ | – |
| 18 | (c,a,d,d) | $c \leftarrow$ Ⅲ | – |
| 19 | (c,a,e,b) | $c \leftarrow$ Ⅲ | – |
| 20 | (c,a,e,d) | $c \leftarrow$ Ⅲ | – |
| 21 | (c,d,d,b) | $c \xrightarrow{II} d \xrightarrow{I} d \leftarrow$ Ⅰ | – |
| 22 | (c,d,d,d) | $c \xrightarrow{II} d \xrightarrow{I} d \xrightarrow{I} d$ | {c,d} |
| 23 | (c,d,e,b) | $c \xrightarrow{II} d \leftarrow$ Ⅰ | – |
| 24 | (c,d,e,d) | $c \xrightarrow{II} d \leftarrow$ Ⅰ | – |

Ⅰ: ME  Ⅱ: SJ  Ⅲ: FD  Ⅳ: BLB

*Uncomplete path* is an uncomplete (intermediate) state of an complete path during the depth-first search course on a RDG.

For example, the second column of Tab.III lists all compete paths in Fig.3. Let path $P = (a, a)$, then $P$ is an uncomplete path because it is an uncomplete state of a complete path such as $(a, a, d, b)$.

***Attribute set of a path***

Let $G$ is a $RDG(DS)$, Let $P$ is an uncomplete or a complete path of $G$, *attribute set* of $P$ is a set containing all attributes that mark its nodes, and denoted by $AttSet(P)$.

For example, let $P$ is the first path in Tab.III, i.e., $P = (a, a, d, b)$, then $AttSet(P) = \{a, b, d\}$.

***Absorbable complete path*, *non-absorbable complete path*, *Reduct complete path set***

Let $G$ is a $RDG(DS)$, a complete path $P_i \in CPS(DS)$ is an *absorbable complete path* if there exists another complete path $P_j \in CPS(DS)$ satisfying $AttSet(P_j) \subset AttSet(P_i)$, otherwise, $P_I$ is a *non-absorbable complete path*.

For example, let $P_1$ and $P_2$ are the first path and the second path in Tab.III respectively, i.e., $P_1 = (a, a, d, b)$, $P_2 = (a, a, d, d)$, then $P_1$ is an absorbable complete path since $AttSet(P_1) \subset AttSet(P_2)$, while $P_2$ is a non-absorbable complete path since there is no path $P$ satisfying $AttSet(P) \subset AttSet(P_2)$.

*Reduct complete path set* of $G$ is a set only containing non-absorbable complete paths $CPS(G)$, and denoted by $RCPS(G)$.

***Member set*, *member***

During the course of depth-first search on a RDG, let $P$ is an uncomplete path, its *member set* is a set of vertices marked by the names of attributes in $AttSet(P)$, and denoted by $MS(P)$. Any vertex in $MS(P)$ is called a *member* of $P$.

For example, in Fig.3, let $P = (a, a, d)$ is an uncomplete path, then $MS(P)$ includes all vertices denoted by $a$ or $d$ since $AttSet(P) = \{a, d\}$, that is, both of the vertices denoted by $d$ in the second layer and the last layer are the members of $P$.

***Friend set*, *friend***

During the course of depth-first search on a RDG, let $P = (v_1, v_2, \ldots, v_i, v_{i+1}, \ldots, v_n)$ is an uncomplete path or a complete path, *friend set* of $P$ denoted by $FS(P)$ is a set of vertices, which join $FS(P)$ according to the following recursive method:

1) if $P = (v_1)$, then $FS(P)$ includes vertices having the same mark (attribute name) as the brother nodes of vertex $v_1$;

2) let $P_i = (v_1, v_2, \ldots, v_i)$, and $P_{i+1} = (v_1, v_2, \ldots, v_i, v_{i+1})$, if $v_{i+1} \notin MS(P_i)$, then $FS(P_{i+1})$ is the union of $FS(P_i)$ with the sets of vertices that have the same mark as the brother nodes of vertex $v_{i+1}$, else $FS(P_{i+1}) = FS(P_i)$.

Any vertex in $FS(P)$ is called a *friend* of $P$.

For example, let $P_1 = (b)$, $P_2 = (b, a)$ are uncomplete paths in Fig. 3, then $FS(P_1)$ includes vertices marked by $a$ or $c$, $FS(P_2)$ includes vertices marked by $a$, $c$ or $d$.

***Stranger***

Let $P$ is an uncomplete path, a vertex is a *stranger* of $P$ if it is neither a member nor a friend of $P$.

For example, in Fig.3, let $P_1 = (c)$ is an uncomplete path, then $FS(P_1)$ includes vertices denoted by $a$ or $b$, and the vertex marked by attribute $d$ in the second layer is a stranger of $P_1$; let $P_2 = (c, d)$ is an uncomplete path, then $FS(P_2)$ is the union of $FS(P_1)$ with the set of vertices denoted by $a$, and the vertex marked by attribute $e$ in the third layer is a stranger of $P_2$. It should be noted that the vertex marked by $d$ in the third layer is not a stranger of $P_2$ since it is a member of $P_2$.

***Barrier-layer***

During the course of depth-first search on a RDG, let $P$ is an uncomplete path, *Barrier-layer* of $P$ is a layer in which every vertex is a friend of $P$.

For example, in Fig.3, let $P = (c, a)$ is an uncomplete path, then the last layer is its barrier-layer since each vertex in the last layer is belong to $FS(P)$.

## III. PRINCIPLE & PROBLEM

In this section, we illustrate the principle for computing *attribute reducts* from a decision system, and analyze its problem when dealing with huge decision systems. This problem will be overcome in section IV.

From the viewpoint of classification, condition attributes that are not in $DAS(u_i, u_j)$ is useless to discern $u_i$ and $u_j$ since they have the same values on $u_i$ and $u_j$. Furthermore, condition attributes in $DAS(u_i, u_j)$ are equivalent: one can remain any one of them and remove others while keeping the classification information about $u_i$ and $u_j$ invariant.

To formally represent this observation, we use the disjunction of condition attributes, which act as boolean variables, in $DAS(u_i, u_j)$. For example, let $DAS(u_i, u_j) = \{a_1, a_2, \ldots, a_k\}$, then the disjunction is $\vee a : a \in DAS(u_i, u_j)$ or $a_1 \vee a_2 \vee \ldots \vee a_k$.

Since $F(DS)$ includes all $DAS(u_i, u_j)$, where $u_i$ and $u_j$ are different objects in the decision system. From the viewpoint of classification, one can combine each one attribute from each member of $F(DS)$ so as to simplify the decision system while keeping its classification information invariant. Let $F(DS) = \{F_1, F_2, \ldots, F_k\}$, we use the following conjunction normal form to formally represent this observation.

$$\bigwedge_{i=1}^{k} \bigvee a : a \in F_i \tag{3}$$

For example, corresponding to $F(DS)$ in Eq.(1), Eq.(3) is

$$(a \vee b) \wedge (a \vee c) \wedge (a \vee b \vee c) \tag{4}$$

Obviously, by the absorption rule $p \wedge (p \vee q) = p$, Eq.(4) can be simplified to Eq.(5).

$$(a \vee b) \wedge (a \vee c) \tag{5}$$

Let $RF(DS) = \{RF_1, RF_2, \ldots, RF_l\}$, since $RF(DS)$ has removed all absorbable discernibility attribute set from $F(DS)$, Eq.6 is just the simplification form of Eq.(3).

$$\bigwedge_{i=1}^{l} \bigvee a : a \in RF_i \tag{6}$$

That is to say, one can directly obtain the simplification form of Eq.(3) from $RF(DS)$. Thus, corresponding to $RF(DS)$ in Eq.(2), Eq.(6) is just Eq.(5).

Eq.(6) can be transformed into a equivalent disjunction normal form. If the disjunction normal form has been simplified according to the absorption rule $p \vee (p \wedge q) = p$, then the set of attributes in each conjunction can maintain the classification information of $DS$ invariant without needing other attributes, that is to say, this set is just a *attribute reduct* of $DS$. In this way, all of attribute reducts can be derived from the disjunction normal form.

TABLE IV
REDUCT OF TAB.I

| | a | d |
|---|---|---|
| $u_1$ | 1 | 1 |
| $u_2$ | 2 | 2 |
| $u_3$ | 1 | 1 |
| $u_4$ | 2 | 2 |

For Eq.(5), its equivalent disjunction normal form is

$$a \vee (a \wedge c) \vee (a \wedge b) \vee (a \wedge c) \tag{7}$$

Eq.(7) can be simplified to $a$ according to the absorption rule. Thus $\{a\}$ is a *attribute reduct* of $DS$. That is, one can use only one condition attribute $a$ to keep the classification information of $DS$ invariant without needing other attributes.

So Tab.I can be reduct into Tab.IV while keeping the classification information invariant. That is, in Tab.I, any object $u_i \in U$ can be correctly classified just according to attribute $a$ without needing other attributes.

This also can be easily verified by observation since Tab.I and Tab.IV are very simple. From Tab.IV, one can easily find the classification rule: if $f(u_i, a) = 1$ then $f(u_i, d) = 1$; if $f(u_i, a) = 2$ then $f(u_i, d) = 2$. These rules are also true in Tab.I, although it is more difficult to find them from Tab.I.

For a huge data set, however, it would be very difficult to find and verify classification rules, and it would become very necessary to compute its attribute reducts, because 1) by remaining attributes in a attribute reduct and removing other ones, redundant data for classification could be removed from an decision system; 2) efficient information could be more easily mined out from the reduct decision system.

From the discussion above, one can summerize a method for computing attribute reduct that includes three basic steps:

Step_1: compute $RF(DS)$ and creating a conjunction normal form, such as Fig. (6);

Step_2: transform the conjunction normal form into a disjunction normal form, such as Fig. (7);

Step_3: simplify the disjunction normal form and attaining the attribute reducts from its conjunction.

However, step_2 may meet a problem of combination explosion when the $RF(DS)$ is big. For example, assume a $RF(DS)$ has 6 members, and each member has 9 attributes, after transformation in step_2, in the disjunction normal form, the number of conjunctions is $6^9$, which is more than ten million.

We try to solve this key problem by means of $RDG(DS)$. Let $G$ is a $RDG(DS)$. In fact, there exists a one-by-one mapping relation between complete paths in $CPS(G)$ and conjunctions in the disjunction normal form created in step_2. For example, let $P = (a_1, a_2, \ldots, a_l) \in CPS(G)$ is one of complete paths in $G$, then $a_1 \wedge a_2 \wedge \ldots \wedge a_l$ is one of conjunctions in the disjunction normal form created in step_2.

Furthermore, it is easy to find that there exists a one-by-one mapping relation between complete paths in $RCPS(G)$ and conjunctions in the disjunction normal form created in Step_3. Thus, according to step_3, attribute reducts may be attained by $RCPS(G)$ . For example, let $P = (a_1, a_2, \ldots, a_l) \in RCPS(G)$, then $a_1 \wedge a_2 \wedge \ldots \wedge a_l$ is one of conjunctions in the disjunction normal form created in step_3, and $AttSet(P)$ is a attribute reduct of $DS$.

From the discussion above, it can be concluded that one complete path in $RCPS(G)$ corresponds to one attribute reduct of $DS$, and all complete paths in $RCPS(G)$ corresponds to all attribute reducts of $DS$. Thus to get attribute reducts of $DS$, we only need to find out $RCPS(G)$.

To find out $RCPS(G)$, the basic method is: find out $CPS(G)$ at first, then get $RCPS(G)$ by removing all absorbable paths from $CPS(G)$.

Obviously, this method also meet the same problem of combination explosion since the size of $CPS(G)$ is very large when $G$ is big. For example, assume $G$ has 9 layers, and each layer has 6 attributes, the size of $CPS(G)$ is $6^9$, which is more than ten million.

To avoid the problem of combination explosion, the algorithm CARHS is proposed in the next section, which may get some attribute reducts by finding out some complete paths in $RCPS(G)$ while it may avoid the problem of combination explosion and even doesn't need know $CPS(G)$.

## IV. HEURISTC RULES & ALGORITHM

As discussed in section III, let $G$ is a RDG of a decision system $DS$, then one complete path in $RCPS(G)$ corresponds to one attribute reduct of $DS$. In this section, to find out some complete paths in $RCPS(G)$ while avoiding the problem of combination explosion, the algorithm CARHS is proposed based on depth-first search on $RDG(DS)$ with heuristic search rules.

These heuristic search rules have intuitive names: *Member Exclusive* (ME), *Friend Dissuade* (FD), *Barrier-layer Block* (BLB), and *Stranger join* (SJ).

ME: *Member Exclusive*

During the course of depth-first search, let $P$ is an *uncomplete path*, if there is a vertex $v$ in the next layer satisfying $v \in MS(P)$, then $P$ will extend through the vertex and don't extend through other vertices in the same layer. That is, vertex $v$ has the exclusive right over other vertices in the same layer. If there are several vertices in the next layer that are members of $P$, then the one at the most left has the exclusive right.

FD: *Friend Dissuade*

During the course of depth-first search, let $P$ is an *uncomplete path*, if vertex $v$ in the next layer satisfies $v \in FS(P)$, then $P$ will not extend through the vertex. That is to say, any vertex that is a *friend* of $P$ will dissuade $P$.

BLB: *Barrier-layer Block*

During the course of depth-first search, let $P$ is an *uncomplete path*, if there exists a *barrier-layer* of $P$, then $P$ will not continue to extend. That is to say, any *barrier-layer* of $P$ will block the search of $P$.

It seems that *BLB* is just a trivial result of *FD*. However, in some cases *BLB* can significantly improve the search, because one can prejudge whether there exists a barrier-layer of an

uncomplete path. If there exists a barrier-layer of a path, this path can be blocked as early as possible before it reaches the barrier layer so as to improve search efficiency.

SJ: *Stranger join*

During the course of depth-first search, let $P$ is an *uncomplete path*, if there is no barrier-layer in any following layer and no any member of $P$ in the next layer, then $P$ may pass through its stranger in the next layer.

When the depth-first search on $RDG(DS)$ employs these heuristic rules, it has been proofed that: 1) all absorbable complete paths in $CPS(G)$ must be blocked during the course of search; 2) the complete paths that don't be blocked must be non-absorbable complete paths in $CPS(G)$, i.e., must be in $RCPS(G)$; 3) the complete paths that don't be blocked must be different from each other.

Based on these heuristic search rules above and the principle in section III, the algorithm CARHS, presented in Fig.4, is proposed to compute attribute reducts of a decision system. It includes four sub processes:

1) compute $RF(DS)$ by the algorithm CRFDAS (see Fig.5);
2) construct $RDG(DS)$ based on $RF(DS)$;
3) search on $RDG(DS)$ with heuristic depth-first strategy;
4) output results.

The algorithm CRFDAS is proposed as part of the algorithm CARHS. CRFDAS can directly compute $RF(DS)$ without needing compute $F(DS)$ or discernibility matrix $M(DS)$, although the concept of $RF(DS)$ is based on the latter two. Thus CRFDAS can reduce the time complexity and the space complexity of CARHS.

Since the algorithm CARHS employs all of these heuristic rules (ME, FD, BLB, and SJ) (see Fig.4), it has high search efficiency, and all the members in its output set $ARS$ must be attribute reducts.

In the next section, by means of a simple example, the algorithm CARHS is illustrated in detail. In section VI, to examine its capability and efficiency, the algorithm CARHS is applied to three classic data sets from UCI database [9].

## V. EXAMPLE

In this section, by means of a simple example, we illustrate the basic procedure of the algorithm CARHS.

Let $RF(DS) = \{\{a,b,c\},\{a,d\},\{d,e\},\{b,d\}\}$, Fig. 3 shows the corresponding reduct discernibility graph $RDG(DS)$.

Tab. III clearly presents the total paths of $RDG(DS)$ (Fig. 3), the course of the depth-first search with heuristic rules, and the attribute reducts attained.

From Tab. III, one can conveniently observe and understand the detail of the search course and the result of each path. *The first column* shows the number of each complete path in Fig. 3. *The second column* shows the nodes through which each path passes. *The third column* shows the depth-first search course of each path with the heuristic rules. The character over "$\rightarrow$" denotes the heuristic rule supporting the search step, while the character on the right of "$\hookleftarrow$" denotes the heuristic rules blocking the search step. For instance, $a \xrightarrow{I} a$ indicates that the search can pass from the node $a$ in the current layer to the node $a$ in the next layer, according to the heuristic rule I, i.e., Member Exclusive, while $d \hookleftarrow ①$ indicates that the search is blocked to the next layer according to the heuristic rule I, that is, there exists another node in the next layer that is a *member* of the uncomplete search path $(d)$. *The fourth column* shows the attribute reducts attained from each path, and "-" means this path can not reach the last layer, that is, it is blocked

Algorithm: CARHS
// compute attribute reducts based on heuristic search.
Input: $DS = (U, C \cup d, V, f)$
Output: $ARS$ //ARS: attribute reduct set
Initial: $ARS = \phi$
Begin
1) Compute $RF(DS)$ by algorithm CRFDAS shown in Fig.5.
2) Construct $RDG(DS)$ based on $RF(DS)$.
3) Search on $RDG(DS)$ with heuristic depth-first strategy:
Let $P$ is an *uncomplete path*, the heuristic depth-first search strategy is:
Step 1 (BLB):
if there exists a *barrier-layer* of $P$, then
go to step 4; // $P$ is blocked by a barrier-layer
else
go to step 2;
Step 2 (ME):
examine the vertex $v$ in the next layer from left to right:
if $v \in MS(P)$, then
$P$ will not extend through other vertices in the layer;
$P = P + v$; // ME: $P$ extends through $v$
go to step 2;
else
go to step 3;
Step 3 (FD & SJ):
examine the current vertex $v$ in the next layer:
if $v \in FS(P)$, then
if $v$ is the last vertex in the next layer, then
go to step 4; // FD: $P$ is blocked by $v$
else
$v$=the direct right brother of $v$;
go to step 3;
else
$P = P + v$; // SJ: $P$ extends through $v$
if $P$ is a complete path, then
$ARS = ARS + AttSet(P)$;
go to step 4;
else
compute $FS(P)$;
go to step 1;
Step 4:
Finish this time of searching, and try other paths.
4) output $ARS$.
End.

Fig. 4. Algorithm CARHS

Algorithm: CRFDAS
// compute *reduct family of discernibility attribute set*
Input: $DS = (U, C \cup d, V, f)$, where $U = \{u_1, u_2, \cdots, u_n\}$
Output: $RF(DS)$
Initial: $RF(DS) = \phi$
Begin
For $i = 1$ to $n - 1$
For $j = i + 1$ to $n$
$absorbed := false$;
For each $R \in RF(DS)$
if ($R \subseteq DAS(u_i, u_j)$)
$absorbed := true$
break;
if ($DAS(u_i, u_j) \subset R$)
$RF(DS) = RF(DS) - R$
End
if not $absorbed$
$RF(DS) = RF(DS) + DAS(u_i, u_j)$
End
End
return $RF(DS)$
End

Fig. 5. Algorithm CRFDAS

in some earlier layer due to a heuristic rule, and no attribute reduct can be attained from this path.

In Tab. III, it is no coincidence that the attribute reducts attained in this example are different from each other. In fact, this can be ensured by the algorithm CARHS, and it is one of the main reasons why CARHS is high efficient.

However, there are some aspects that should be clarified:

1) In this example the attribute reducts attained are all minimal. However, it is not always true. That is, CARHS can ensure the results must be attribute reducts, but cannot ensure they are minimal attribute reducts;

2) CARHS cannot ensure attain all attribute reducts. For example, in fact, $\{a, b, e\}$ is a attribute reduct in this example, however, it cannot be attained by CARHS since path 3 and path 11 are blocked by the heuristic rules BLB and FD respectively;

3) From Tab. III, although the heuristic rule BLB appears to be not worthy of mention, in fact, sometimes it is very important and efficient in processing some large data sets that have RDGs with much more layers. This can be easily observed from the following experiments on the UCI data sets.

## VI. EXPERIMENT

To examine the capability and efficiency of the algorithm CARHS, from the famous UCI learning database [9] we select three classic data sets as the test data sets. The three data sets are: Zoo (data_1); Soybean_small (data_2); Poker_hand_training_true (data_3). All of them are numerical and no missing value data sets. For instance, Tab. V shows a part of data_1.

CARHS has been implemented with C language, and run on a T7100 1.8G CPU. Tab. VI shows the attribute reducts and the core attributes of the three data sets attained by CARHS. Although the core attributes set of data_2 is $\phi$, one can also find that attribute 23 and 35 are very important attributes since they are included in much of the attribute reducts sets attained.

Tab. VII shows the number of the objects, the number of the attributes, the number of the layers, the number of paths in the RDG, and the running time. It should be noted the running time includes the time of constructing the reduct discernibility graph and the time for the depth-first search.

From Tab. VII, one can find some significant facts:

1) For data_1 and data_2, the running time is less than 0.04 seconds, and for data_3, the running time is less than 100 seconds although it has more than twenty thousands of objects. So CARHS may be used to find attribute reducts for large data sets.

2) Although data_1 and data_2 have much less objects than data_3 does, they have much more paths in their RDGs. The main reason is that data_1 and data_2 have much more RDG layers than data_3 does. That is, the number of total path mainly depends on RDG rather than the number of objects. More objects does not mean more total paths in RDG.

3) Although data_1 and data_2 have much more paths in their RDGs than data_3 does, they need much less running time. The main reason is that the construction of RDG consumes most part of running time, while the search course on RDG is very quick and efficient even when the total path is greater than $1 \times 10^{96}$. This further verifies the efficiency of the algorithm CARHS.

To examine the effect of the different heuristic rules, Tab. VIII shows the contribution rates of each heuristic rule. The contribution rate of a heuristic rule is the rate of branches, which are cut out during the depth-first search according to the heuristic rule, on the total path of RDG. From Tab. VIII, one can further recognize the importance of the heuristic rules: 1) *Member Exclusive* always has more contribution rate than the other rules; 2) The effect of *Friend Dissuade* cannot be neglected in general; 3) Although *Barrier-layer Block* seems useless for some data sets such as data_2 and data_3, it is very important for some data sets such as data_1.

TABLE V
PART DATA OF ZOO.

| Values of sixteen attributes | | | | | | | Class |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | $\cdots$ | 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | $\cdots$ | 0 | 6 | 1 | 6 |
| 1 | 0 | 1 | $\cdots$ | 0 | 6 | 0 | 6 |
| 0 | 1 | 1 | $\cdots$ | 0 | 2 | 0 | 2 |
| 0 | 0 | 1 | $\cdots$ | 0 | 6 | 0 | 6 |
| 0 | 1 | 1 | $\cdots$ | 0 | 2 | 0 | 2 |
| 1 | 0 | 0 | $\cdots$ | 0 | 4 | 0 | 1 |

TABLE VI
ATTRIBUTE REDUCTS AND CORE ATTRIBUTES.

| Data set | attribute reduct | core attribute |
|---|---|---|
| data_1 | {3,6,8,9,11,13}; {3,6,8,9,12,13}; {3,6,8,11,13,14}; {3,6,8,12,13,14}. | {6,13} |
| data_2 | {22,23}; {12,21,35}; {4,12,23,35}; {1,6,23,35}; {8,12,23,35}; {2,12,23,35}; {2,12,23,25}; {12,23,25,35}; {12,23,28,35}; {1,5,10,12,23}; {1,10,12,23,35}; {1,5,10,23,35}. | $\phi$ |
| data_3 | {1,2,3,4,6,8,10}; {1,2,4,5,6,8,10}; {1,2,4,6,7,8,10}; {1,2,4,6,8,9,10}. | {2,4,6,8,10} |

data_1: Zoo; data_2: Soybean_small; data_3: Poker_hand_training_true.

TABLE VII
NUMBER OF RDG LAYER, TOTAL PATH AND RUNNING TIME.

| Data set | objects | attributes | RDG layers | total path | time (s) |
|---|---|---|---|---|---|
| data_1 | 101 | 17 | 14 | $1.17 \times 10^{10}$ | 0.015 |
| data_2 | 47 | 36 | 99 | $1.80 \times 10^{96}$ | 0.031 |
| data_3 | 25010 | 11 | 9 | 144 | 90.901 |

data_1: Zoo; data_2: Soybean_small; data_3: Poker_hand_training_true.

## VII. CONCLUSION

To compute attribute reducts from a decision system, the algorithm CARHS was proposed in this paper. CARHS is based on the concept of *reduct discernibility graph* (RDG). Once the RDG is created, the problem of computing attribute reductions is transformed into the one to look for *non-absorbable complete paths* in the RDG. Since the total path of RDG is always too huge to be examined one by one, four heuristic rules (ME, FD, BLB, SJ) are developed and employed in CARHS. These heuristic rules can ensure that every *absorbable complete path* may be blocked as early as possible, and all of the *complete paths* in the search result are necessary for computing attribute reduct. Thus CARHS may find several attribute reducts with high efficiency while avoiding the problem of combination explosion, which is the key problem of other algorithms such as in [2]-[6].

Compared with our past work in [7], there are several highlights in this paper. *First*, to construct RDG, the algorithm CRFDAS, as part of the algorithm CARHS, has been proposed. CRFDAS can directly compute *reduct family of discernibility attribute set* without needing compute *family of discernibility attribute set* or *discernibility matrix*, although the concept of the former is based on the latter two. So CRFDAS not only can reduce its own time complexity and space complexity, but also can increase the efficiency of the algorithm CARHS. *Second*, the heuristic rules *Stranger Join* has been developed, so that the algorithm CARHS can be completed; *Third*, all of the novel concepts have been illustrated by some simple examples in this paper so that they can be more easily understood than in [7]. *Fourth*, a novel description method in Tab. III has been developed to describe the course of heuristic depth-first search. In this special way, one can easily understand and analysis the algorithm CARHS, the role of the heuristic rules, and the reason why a path can pass through a node or be blocked. *Finally*, the algorithm CARHS is applied to three classic data sets from the UCI database [9]. The experiment shows that CARHS can be applied to compute attribute reducts very quickly for large data sets even with ten thousands of objects. The contribution rates of the different heuristic rules are also compared with each other in this experiment.

TABLE VIII
CONTRIBUTION RATE OF HEURISTIC RULES FOR CUTTING BRANCHES.

| Data set | Member Exclusive | Friend Dissuade | Block_layer Block |
|---|---|---|---|
| data_1 | 45.3% | 24.2% | 30.5% |
| data_2 | 85.7% | 10.7% | 3.6% |
| data_3 | 70.7% | 29.3% | 0 |

data_1: Zoo; data_2: Soybean_small; data_3: Poker_hand_training_true.

The experiment also reveals several future research directions: 1) since the construction of RDG consumes most part of time, we can deeply research on it to shorten the running time; 2) since the heuristic rule *Friend Dissuade* is too strong to possibly block some useful paths, we may develop other heuristic rules so as to avoid missing any attribute reduct, and further find the minimal attribute reducts; 3) since CARHS can be applied only to information systems with numerical and no missing value, an information system with non-numerical and missing value should be transformed to the one with numerical and no missing value, so the transformation may be added as a preliminary step of CARHS to expand its scope of application.

## ACKNOWLEDGMENT

This work was partly supported by the Natural Science Foundation of Fujian Province of China (Grant No. 2010J01346 and 2010J05142), the National Natural Science Foundation of China (Grant No. 60975076 and 61003014/F020101).

## REFERENCES

[1] Z. Pawlak. "Rough sets". International Journal of Information and Computer Science. 1982, (11), pp. 341-356.
[2] Wong SKM, W. Ziarko W, "On optimal decision rules in decision tables," Bulletin of Polish Academy of Sciences, 1985, 33(11/12), pp. 693-696.
[3] Wang GY, Yu H, Yang DC, "Decision table reduct based on conditional information entropy," Chinese Journal of Computers, 2002, 25(7), pp. 759-766.
[4] Yang M, Ni WW, Sun ZH.,"Novel model for minimal attributes reductions," Journal of Southeast University (Natural Science Edition), 2004, 34(5), pp. 604-608 (in Chinese with English abstract).
[5] Zhu JH, Li HB, Pan F., "Knowledge-reduction based on GA and fuzzy-rough set," Computer Emulation, 2007, 24(1), pp. 89,119 (in Chinese with English abstract).
[6] A. Skowron, C. Rauszer, R. Slowinski. "Intelligent dicision support handbook of applications and advances of the rough sets theory", Dordrecht: Kluwer Academic Publishers, 1992, pp. 331-362.
[7] Minghui Shi, Fei Chao, Min Jiang, et al. "Approach to Computing Attribute Reductions for Decision System Based on Heuristic Graph Search", International Conference on Intelligent Computing and Intelligent Systems (ICIS2011), 2011, pp. 636-639.
[8] Walczak B, Massart D.L., "Rough sets theory," Chemometrics and Intelligent Laboratory Systems 1999, pp. 47: 1-16.
[9] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/