

## EFFICIENT ALGORITHMS FOR ATTRIBUTES REDUCTION PROBLEM

SONGBO TAN

Graduate School, Chinese Academy of Sciences  
Software Department, Institute of Computing Technology  
Chinese Academy of Sciences, P. O. Box 2704, Beijing 100080, P. R. China  
tansongbo@software.ict.ac.cn

HONGBO XU

Software Department, Institute of Computing Technology  
Chinese Academy of Sciences, P. O. Box 2704, Beijing 100080, P. R. China  
hbxu@software.ict.ac.cn

JUN GU

Software Department, Institute of Computing Technology  
Chinese Academy of Sciences, P. O. Box 2704, Beijing 100080, P. R. China  
Department of Computer Science  
Hong Kong University of Science and Technology, Hong Kong  
eecs@263.net

Received July 2004; revised February 2005

**ABSTRACT.** *The theory of rough set, proposed by Pawlak, provides a formal tool for knowledge discovery from imprecise and incomplete data. Attributes reduction is a crucial problem for rough set based data mining. Unfortunately, finding minimal reduct turns out to be a NP-hard problem. In this paper, we propose a heuristic reduction algorithm HeuriRed and a complete heuristic reduction algorithm HeuriComRed based on discernibility matrix. The time and space requirements of two proposed algorithms are both  $O(|A| * |U|^2)$ . In our experiment, in order to investigate the robustness, efficiency and completeness of proposed algorithms, we execute our algorithms HeuriRed, HeuriComRed and genetic reduct on some problems from UCI repository. The experimental results not only indicate the robustness and efficiency of our algorithms, but also provide proofs for the completeness of our algorithm HeuriComRed.*

**Keywords:** Rough Set, Attributes Reduction, Discernibility Matrix, Data Mining

1. **Introduction.** The rough set theory was developed by Pawlak in the early 1980's [1,2] and has been applied successfully in a lot of domains, such as machine learning, knowledge discovery, and expert systems. It provides powerful tools for data analysis and data mining from imprecise and ambiguous data.

Reduct is the most important concept in rough set-based data mining. Given a distinguished feature, called decision, the notion of decision reduct is constructed over, so called, discernibility matrix, where information about all pairs of objects with different decision values is stored. A reduct is any minimal (in the sense of inclusion) subset of non-decision features (conditions), which discern all such pairs, necessary to be considered, e.g., with

respect to proper decision of classification of new cases. Unfortunately, it has been shown that finding minimal reduct is a NP-hard problem. Hence heuristic algorithm is always employed to find an approximation reduct. We say a reduct is complete if the deletion of any attribute of a reduct will make at least one pair of objects with different decision attribute values indiscernible.

Therefore, efficient methods to solve this NP-hard problem play an important role in the development of rough set-based data mining. In this paper, we propose a fast heuristic attribution reduction algorithm *HeuriRed* whose time complexity and space complexity are both  $O(|A| * |U|^2)$ . ( $|A|$  denotes the number of attributes and  $|U|$  denotes the number of objects of information system). Unfortunately, the algorithm *HeuriRed* is not complete. In order to obtain complete reducts for all datasets, a complete heuristic reduction algorithm *HeuriComRed* is proposed whose time complexity and space complexity are also both  $O(|A| * |U|^2)$ . In our experiment, our algorithms achieve significant performance in comparison to other reduction algorithm, e.g., the *genetic reduct* algorithm [13,14].

The rest of this paper is organized as follows: In the next section, we briefly overview the previous work. Section 3 gives some preliminaries of rough set theory that simplify our discussion. Our algorithms *HeuriRed* and *HeuriComRed* are introduced in Section 4. Experimental results of *HeuriRed* and *HeuriComRed* are given in Section 5. Finally Section 6 concludes this paper.

**2. Prior Work.** There are many algorithms have been developed for reduct problem. In this section, we briefly review some algorithms for reduct problem. The conventional algorithms fall into two categories: the reduction algorithms based on heuristic information [4-11,15] and the reduction algorithms based on random strategies [13,14,16-19]. Nevertheless, all of these algorithms cannot ensure to find complete reducts for all datasets.

Johnson's strategy [15] is based on Johnson approximation algorithm for computing minimal prime implicant of any Boolean function in conjunctive normal form (CNF) formula. The main idea of the algorithm is to find an attribute discerning the largest number of pairs of objects, i.e., attribute most often occurring in the entries of discernibility matrix. This algorithm proceeds until a reduct set is found. The time complexity of this algorithm is  $O(|A|^2 |U|^2)$  and the space complexity of this algorithm is  $O(|A| |U|^2)$ . Note that the algorithm is not complete.

Jue Wang [4] proposes an attributes reduction algorithm based on attributes significance in discernibility matrix. He defines the attributes significance as attributes frequency in discernibility matrix. Hence he regards the number of occurrences of each attribute as the significance of each attribute. First he selects the attribute  $a$  with the largest frequency, and deletes the elements involved with the selected attribute  $a$  in discernibility matrix. Then count the frequency of other attributes and select again. This algorithm proceeds until a reduct set is found. Very like Johnson's algorithm, the time complexity of this algorithm is also  $O(|A|^2 |U|^2)$  and the space complexity of this algorithm is also  $O(|A| |U|^2)$ . Note that the algorithm is also not complete.

By making use of attribute frequency information in discernibility matrix, Keyun Hu [5,6] develops a feature ranking mechanism. Based on the mechanism, a heuristic reduct algorithm is proposed. The time complexity of this algorithm is  $O((|A| + \log |U|) |U|^2)$  and the space complexity of this algorithm is  $O(|A| |U|^2)$ . Like above two algorithms, the algorithm is not complete.

X. Hu et al. [7] proposed a new rough sets model and redefined the core attributes and reducts based on relational algebra to take advantages of the very efficient set-oriented database operations. With this new model and new definitions, they presented two new algorithms to calculate core attributes and reducts for feature selections. However, the time complexity of their reduct algorithm is at least  $O(|A|^2|U|)$  even if the database systems provide both hashing and indexing mechanism.

After the introduction of four heuristic algorithms, we continue to review two random reduct algorithms.

Vinterbo [13,14] formulates the rough set based attributes reduction as minimal hitting set problem. Further he defines an  $r$ -approximate hitting set as a set that intersects at least a fraction  $r$  of given sets. Also, approximations of reducts from rough set theory are defined by means of minimal  $r$ -approximate hitting sets. Then a genetic algorithm is devised to compute  $r$ -approximate hitting sets. The time complexity of this algorithm is  $O(|A|^2|U| \log |U|)$  and the space complexity of this algorithm is  $O(|U|)$ . Obviously, reducts obtained by this algorithm are not guaranteed to be complete.

Bazan [16-19] thinks above methods are not taking into account the fact that part of reduct set of is chaotic i.e. is not stable in randomly chosen samples of a given decision table. Therefore he develops a method for selection of feature (attribute) sets relevant for extracting laws from data. These sets of attributes are called dynamical reducts. Dynamical reducts are in some sense the most stable reducts of given decision table, i.e., they are the most frequently appearing reducts in subtables created by randomly sampling of a given decision table. Computing dynamically reduct can be extremely computationally intensive, even for moderately size decision table. This algorithm is quite stable in most cases, but it is still not a complete reduction algorithm.

**3. Preliminaries.** In this section we provide some basic definitions of rough set theory. Detail description of the theory can be found in [3].

*Information system* is a pair  $\mathbf{A}=(U, A)$ , where  $U$  is a non-empty, finite set of objects called the *universe* and  $A$  is a non-empty finite set of *attributes* i.e.  $a : U \rightarrow V_a$  for  $a \in A$ , where  $V_a$  is usually called a *decision value set* of  $a$ . An information system  $\mathbf{A}=(U, A \cup \{d\})$ , where  $d \notin A$ , is usually called a *decision table*. The elements of  $A$  are called the *conditional attributes* and  $d$  is called *decision attribute*.

Decision table is *consistent* iff there are no objects with the same values of conditional attributes and different decision values.

*Indiscernability relation*  $IND(B)$  is defined for any subset  $B \subseteq A$  ( $B \subseteq A \cup \{d\}$ ) as follows:

$$IND(B) = \{(x, y) \in U \times U : \text{for every } a \in B \ a(x) = a(y)\}$$

*Reduct* is any subset  $B \subseteq A$  ( $B \subseteq A \cup \{d\}$ ) such that  $IND(B) = IND(A)$  and  $IND(B - \{a\}) \neq IND(A)$  for any  $a \in B$ . Hence it is enough to consider only the attributes from the reduct to distinguish objects in  $U$ . By  $RED(A)$  we denote the set of all reducts of  $A$ . A *minimal reduct* of  $A$  is an element of  $RED(A)$  minimal in order of cardinality.

The *discernibility matrix* of an information system is a symmetric  $|U| \times |U|$  matrix with entries  $c_{ij}$  defined as  $\{a \in A / a(x_i) \neq a(x_j)\}$  if  $d(x_i) \neq d(x_j)$ ,  $\phi$  otherwise.

**4. Proposed Algorithms.** Before give the outline of proposed algorithms, we first introduce some necessary denotations, variables and functions.

*A Data Structures*

1 *Be-Covered* (or *NormMat*) Matrix

Assume the dataset is consistent. We can regard the reduction problem as a special set-covering problem in which the elements of ground set are the non-empty items of discernibility matrix and the subsets are the attributes of dataset. Therefore, if we take the non-empty items of discernibility matrix as rows, then we can construct a so-called *Be-Covered* (or *NormMat*) Matrix, in which each row records the attributes covering a non-empty item. The first column of *NormMat* denotes the number of attributes covering each non-empty item of discernibility matrix. In order to illustrate *NormMat*, we take a simple dataset as an example.

The letters  $a_1, a_2, a_3, a_4$  denote conditional attributes and the letter  $d$  denotes decision attribute.

TABLE 1. Simple decision table

$U$	$a_1$	$a_2$	$a_3$	$a_4$	$d$
1	TRUE	TRUE	FALSE	FALSE	0
2	TRUE	FALSE	TRUE	FALSE	0
3	FALSE	FALSE	TRUE	TRUE	1
4	FALSE	TRUE	TRUE	FALSE	1

Then we can obtain the discernibility matrix as follows:

$$\begin{pmatrix} \phi & \phi & a_1 a_2 a_3 a_4 & a_1 a_3 \\ \phi & \phi & a_1 a_4 & a_1 a_2 \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \end{pmatrix} \tag{1}$$

Then we obtain our *NormMat* as following formula (2) and (3):

$$\begin{pmatrix} 4 & a_1 & a_2 & a_3 & a_4 \\ 2 & a_1 & a_3 & \phi & \phi \\ 2 & a_1 & a_4 & \phi & \phi \\ 2 & a_1 & a_2 & \phi & \phi \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} 4 & 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & \phi & \phi \\ 2 & 1 & 4 & \phi & \phi \\ 2 & 1 & 2 & \phi & \phi \end{pmatrix} \tag{3}$$

In above matrix (3) 1, 2, 3 and 4, excluding in the first column, denote the attributes of dataset. In our algorithms, discernibility matrix is unnecessary, because we can directly construct all four matrixes from the decision table.

2 Covering Matrix (*CovMat*)

On the other hand, if we take attributes of dataset as rows, then we can construct a so-called *CovMat* matrix, in which each row records the items of discernibility matrix

covered by one attribute. The first column of *CovMat* is the number of items covered by one attribute. Then we obtain *CovMat* of above simple dataset as formula (4):

$$\begin{pmatrix} 4 & 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & \phi & \phi \\ 2 & 1 & 2 & \phi & \phi \\ 2 & 1 & 3 & \phi & \phi \end{pmatrix} \quad (4)$$

in which 1, 2, 3 and 4, excluding the first column, denote non-empty items of discernibility matrix  $a_1a_2a_3a_4$ ,  $a_1a_3$ ,  $a_1a_4$  and  $a_1a_2$  respectively. *CovMat* facilitates to quickly find all non-empty entries covered by one attribute without scanning all entries of discernibility matrix.

### 3 *IsBeCovered* Vector

*IsBeCovered* Vector has the same rows as *NormMat* and has only one column, which indicates whether the corresponding row of *NormMat* is covered by at least one attribute. Each item of *IsBeCovered* Vector has value 0 or other numbers larger than 0, i.e., the NO. of attribute, which denotes the corresponding row of *NormMat* is not covered or already covered. For example, if the *IsBeCovered* Vector is (0 0 0 1) indicates the fourth row of *NormMat*, i.e., the item " $a_1a_2$ " of discernibility matrix, is covered by the attribute 1, i.e.,  $a_1$ . First we initialize the *IsBeCovered* Vector as (0 0 0 0). Then as soon as an attribute is selected, we set items of *IsBeCovered* Vector corresponding to rows covered by selected attribute ( $j$ ) to  $j$ . For example, if we first select the attribute  $a_1$ , then we change the *IsBeCovered* Vector to (1 1 1 1). And if we first select the attribute  $a_2$ , then we change the *IsBeCovered* Vector to (2 0 0 2).

### 4 *CovUncovItem* Vector

*CovUncovItem* vector has the same rows as *CovMat* and has only one column, which records the number of items of discernibility matrix covered by one attribute of dataset and on the same time not covered by any selected attributes. The *CovUncovItem* vector facilitates to pick out the attribute covering the maximum non-empty entries of discernibility matrix, which are so far not covered by any selected attribute.

First we initialize the *CovUncovItem* as the first column of *CovMat*. For example, we obtain initial *CovUncovItem* of above dataset as (4 2 2 2). If we select an attribute, we set the corresponding item of *CovUncovItem* to -1. Then we reduce the item of *CovUncovItem* corresponding to each unselected attribute by the number of items covered by this unselected attribute among all items covered by justly selected attribute but not covered by previously selected attributes.

For example, if we first select attribute  $a_1$ , then  $CovUncovItem(1)=-1$ ;  $CovUncovItem(2)=2-2=0$ ;  $CovUncovItem(3)=2-2=0$ ;  $CovUncovItem(4)=2-2=0$ .

If we first select  $a_2$ , on another case, then  $CovUncovItem(2)=-1$ ;  $CovUncovItem(1)=4-2=2$ ;  $CovUncovItem(3)=2-1=1$ ;  $CovUncovItem(4)=2-1=1$ .

Therefore, when a reduct is found, the items of *CovUncovItem* corresponding to selected attributes equal -1 and other items of *CovUncovItem* equal 0.

### B Necessary Variable

*NormRows*: the number of rows of *NormMat*; *NormColumns*: the number of columns of *NormMat*; *CovRows*: the number of rows of *CovMat*; *CovColumns*: the numbers of columns of *CovMat*; *UnCoverRows*: the number of rows of *NormMat*, which are not covered by selected attributes; *CovRowNum*: the number of those rows in *NormMat* covered by one attribute, and these rows are corresponding to a row of *CovMat*; *CovRow*: the current row of *NormMat* covered by one attribute; *NormColNum*: the number of attributes covering one row of *NormMat*; *NormCol*: the current attribute covering one row of *NormMat*.

### C The HeuriRed Algorithm

*Initialization*: to start, procedure *LoadData()* loads the decision table data. Then according to the decision table, we obtain *NormMat*, *CovMat*, *IsBecovered* and *CovUncovItem* by calling function *Initialization()*.

*Heuristic Reduction*: the reduct of attributes is found by calling the function *HeuriSelAttri()* (See FIGURE 1). In this function *HeuriSelAttri()*, we employ the heuristic information: the number of items of discernibility matrix covered by each unselected attribute, which are not covered by selected attributes. Then *SelectMaxCovAttri()* chooses the attribute *max<sub>j</sub>* that covers the most maximal items of discernibility matrix that are not covered by selected attributes. And then we scan the row of *CovMat* corresponding to justly selected attribute *max<sub>j</sub>*. If the row of *NormMat* corresponding to scanned item of *CovMat* is not covered by selected attribute excluding justly selected attribute *max<sub>j</sub>*, then we scan the row of *NormMat*. For each scanned attribute *j*, if it is not selected, we execute *CovUncovItem(j)*--.

*Termination*: our criterion for termination is very simple, i.e., if *UnCoverRows*=0.

*Running time*: the running time of the *HeuriRed* can be estimated as follows. Given a decision table with  $|U|$  objects and  $|A|$  attributes. Then the running time of *LoadData()* is  $O(|A||U|)$ . And the running time of *Initialization()* is  $O(|A||U|^2+|A||U|^2+|U|^2+|A|)$ , i.e.,  $O(|A||U|^2)$ . *SelectMaxCovAttri()* can be done in  $O(|A|)$ . In each execution of *while-loop* only one row of *CovMat* and rows of *NormMat* covered by justly selected attribute *max<sub>j</sub>*, which are not covered by previously selected attributes (Not including justly selected attribute *max<sub>j</sub>*). Therefore the computation with respect to *CovMat* is at most one time scan of *CovMat* and the computation with respect to *NormMat* is just one time scan of *NormMat*. As a result, the executing time of *while-loop* is  $O(|A||U|^2+|A||U|^2+|A|)$ , i.e.,  $O(|A||U|^2)$ . Hence the total running time is  $O(|A||U|+|A||U|^2+|A||U|^2)$ , i.e.,  $O(|A||U|^2)$ .

*Space requirement*: the main space requirements are the four matrixes: *NormMat*, *CovMat*, *IsBecovered* and *CovUncovItem* which require  $O(|A||U|^2)$ ,  $O(|A||U|^2)$ ,  $O(|U|^2)$  and  $O(|A|)$  respectively. Consequently the space complexity is  $O(|A||U|^2)$ .

### D The HeuriComRed Algorithm

*Initialization*: to start, procedure *LoadData()* loads the decision table data. Then according to the decision table, we obtain *NormMat*, *CovMat*, *IsBecovered* and *CovUncovItem* by calling function *Initialization()*.

*Heuristic Reduction*: the reduct of attributes is found by calling the function *HeuriSelAttri()* (See FIGURE 1).

```

void HeuriSelAttri(){
    UnCoverRows=NormRows;
    while(UnCoverRows!=0){
        // choose max_j that maximize{ CovUncovItem(i)}
        max_j=SelectMaxCovAttri();
        CovUncovItem[max_j]=-1;
        CovRowNum=CovMat[max_j]*CovColumns+0;
        for(j=0;j<CovRowNum;j++){
            CovRow=CovMat[max_j]*CovColumns+(j+1);
            //Only scan rows of NormMat that are not covered by selected attributes
            if(IsBeCovered[CovRow-1]==0){
                IsBeCovered[CovRow-1]=max_j;
                UnCoverRows--;
                NormColNum=NormMat[(CovRow-1)*NormColumns+0];
                for(k=0;k<NormColNum;k++){
                    NormCol=NormMat[(CovRow-1)*NormColumns+(k+1)];
                    if(CovUncovItem[NormCol-1]!=-1)
                        CovUncovItem[NormCol-1]--;
                }
            }
        }
    }
} //End while
}

```

FIGURE 1. The function of *HeuriSelAttri*

```

void HeuriRed (){
    //load the decision table data
    LoadData();
    //Obtain the NormMat, CovMat, IsBeCovered and CovUncovItem
    Initialization();
    //the heuristic attribute selection procedure
    HeuriSelAttri();
    //print the result of reduction
    OutPutResult();
}

```

FIGURE 2. The algorithm of *HeuriRed*

*Redundant Attribute deletion*: by calling function *HeuriDelAttri()* (See FIGURE 3), we check every attribute of reduct whether it can be deleted one by one. For each attribute  $j$  of reduct, if every row  $i$  of *NormMat* whose *IsBeCovered*( $i-1$ ) equals  $j$  can be covered by at least one selected attribute excluding the attribute  $j$ , then we can say the attribute  $j$  can be deleted; otherwise cannot be deleted. If the attribute  $j$  can be deleted, we execute  $CovUncovItem(j-1) = 0$ . Then for every row  $i$  of *NormMat* whose *IsBeCovered*( $i-1$ ) equals  $j$ , we randomly select one selected attribute  $j_1$  which can cover the rows  $i$ , and execute  $IsBeCovered(i-1) = j_1$ .

```

void HeuriDelAttri(){
  for(i=0;i<CovRows;i++){
    if(CovUncovItem[i]==-1){
      //the checking part
      CanBeDel=true;
      CovRowNum = CovMat[i*(CovColumns +1)+0];
      for(j=0;j<CovRowNum;j++){
        CovRow =CovMat[i*(CovColumns +1)+(j+1)];
        if(IsBeCovered[CovRow -1]==(i+1)){
          NormColNum = NormMat[(CovRow -1)*(NormColumns+1)+0];
          num=0;
          for(k=0;k<NormColNum;k++){
            NormCol = NormMat[(CovRow -1)*(NormColumns+1)+(k+1)];
            if((NormCol!=(i+1))&&CovUncovItem[NormCol -1]==-1)
              num++;
          }
          if(num==0){
            CanBeDel=false;
            break;
          }
        }
      }
      //the deleting part
      if(CanBeDel){
        CovUncovItem[i]=0;
        CovRowNum =CovMat[i*(CovColumns+1)+0];
        for(j=0;j<CovRowNum;j++){
          CovRow=CovMat[i*(CovColumns+1)+(j+1)];
          if(IsBeCovered[CovRow-1]==(i+1)){
            NormColNum = NormMat[(CovRow-1)*(NormColumns+1)+0];
            for(k=0;k<NormColNum;k++){
              NormCol =NormMat[(CovRow -1)*(NormColumns+1)+(k+1)];
              if(CovUncovItem[NormCol -1]==-1){
                IsBeCovered[CovRow -1]= NormCol;
                break;
              }
            }
          }
        }
      }
    }
  }
  //if(CovUncovItem[i]==-1){
} //for(i=0;i<CovRows;i++)
}

```

FIGURE 3. The function of *HeuriDelAttri*

*Running time:* the running time of the *HeuriComRed* can be estimated as follows. Given a decision table with  $|U|$  objects and  $|A|$  attributes. Then the running time of *Load-Data()* is  $O(|A||U|)$ . And the running time of *Initialization()* is  $O(|A||U|^2+|A||U|^2+|U|^2+|A|)$ , i.e.,  $O(|A||U|^2)$ .

According to Section C, the executing time of *HeuriSelAttri()* is  $O(|A||U|^2)$ .

The *HeuriDelAttri()* scans the *NormMat* just once and  $|R|$  rows of *CovMat*, i.e.,  $|R|$  is the number of selected attributes, to check whether one selected attribute can be deleted. Hereby the running time of the checking part (See FIGURE 3) is  $O(|A||U|^2)$ . If  $|R'|$  is the number of deleted attributes from the reduct, i.e.,  $|R'| < |R| < |A|$ , then *HeuriDelAttri()* needs to scan *NormMat* at most once and just  $|R'|$  rows of *CovMat*. Consequently the

```

void HeuriComRed (){
    //load the decision table data
    LoadData();
    //Obtain the NormMat, CovMat, IsBecovered and CovUncovItem
    Initialization();
    //the heuristic attribute selection procedure
    HeuriSelAttri();
    //delete the redundant attributes by check one by one
    HeuriDelAttri();
    //print the result of reduction
    OutPutResult();
}

```

FIGURE 4. The algorithm of *HeuriComRed*

running time of the deleting part (See FIGURE 3) is  $O(|A||U|^2)$ . Therefore the total time of *HeuriDelAttri()* is  $O(|A||U|^2 + |A||U|^2)$ , i.e.,  $O(|A||U|^2)$ .

Hence the total running time is  $O(|A||U| + |A||U|^2 + |A||U|^2 + |A||U|^2)$ , i.e.,  $O(|A||U|^2)$ .

*Space requirement:* the main space requirements are the four matrixes: *NormMat*, *CovMat*, *IsBecovered* and *CovUncovItem* which require  $O(|A||U|^2)$ ,  $O(|A||U|^2)$ ,  $O(|U|^2)$  and  $O(|A|)$  respectively. Consequently the space complexity is  $O(|A||U|^2)$ .

**5. Experiment Results.** In this section, we give experimental results of proposed attributes reduction algorithms *HeuriRed* and *HeuriComRed* on some problems from UCI repository [12]. Our algorithms are coded in C and tested on a personal computer with a single AMD 700 MHz CPU.

In case of incomplete datasets, we complete it via conditional mean complete algorithm. If the attribute with missing values is numerical attribute, then we substitute missing values with the mean value of all observed values for that attribute; for string attribute, the missing values are substituted by the “mode” value, i.e., the most frequently occurring value among the observed entries for that attribute.

In order to deal with the datasets with continuous attributes, we adopt the equally frequency binning method, which is an unsupervised and univariate discretization algorithm. Fixing a number of interval  $n$  and investigating the histogram of each attribute,  $n - 1$  cuts are determined so that approximately the same number of attribute values falls into each of the  $n$  intervals. This corresponds to assigning  $n - 1$  cuts such that the area between two neighboring cuts in normalized histogram is as close to  $1/n$  as possible.

In order to investigate the robustness, effectiveness and completeness of our algorithms, we conduct comparison with *genetic reduct* [13-14]. We execute *genetic reduct* in ROSETTA system [8].

The leftmost column of TABLE 2 is the name of the dataset from UCI. The 2<sup>nd</sup>, 3<sup>rd</sup> columns are instances numbers, attribute numbers. The 4<sup>th</sup>, 7<sup>th</sup>, 10<sup>th</sup> columns are reduct attribute numbers generated by *HeuriRed*, *HeuriComRed* and *genetic reduct* algorithm respectively. The 5<sup>th</sup>, 8<sup>th</sup>, 11<sup>th</sup> columns are running time in seconds required by *HeuriRed*, *HeuriComRed* and *genetic reduct* algorithm respectively. The 6<sup>th</sup>, 9<sup>th</sup> columns denote the incompleteness/completeness of reducts obtained by *HeuriRed* and *HeuriComRed*

respectively. Since the *genetic reduct* algorithm produces many different size reducts for a given dataset with 100% support rate. In order to compare fairly, we selected the smallest reduct.

In terms of running time, our algorithm *HeuriComRed* takes approximately the same running time as *HeuriRed* which has a much faster speed than *genetic reduct* in all selected datasets with exception of the dataset, i.e., Auto.

In terms of quality of reduct, in case of four datasets, i.e., soybean-Small, LungCancer-Data, sonar and churn, our algorithm *HeuriRed* has the same size reducts as *genetic reduct*, in case of another three datasets, i.e., voting, letter and auto, our algorithm *HeuriRed* has larger size reducts than *genetic reduct* and has smaller size reducts than *genetic reduct* in other datasets. There are seven datasets whose reducts obtained by *HeuriComRed* are the same size as reducts obtained by *genetic reduct* and for other datasets *HeuriComRed* finds a smaller reduct than *genetic reduct*. Consequently, *HeuriComRed* delivers the best performance and *HeuriRed* ranks the second.

For four datasets, i.e., voting, letter, auto and satimage, *HeuriRed* cannot find complete reducts. In contrast with to *HeuriRed*, *HeuriComRed* finds complete reducts for all datasets.

Therefore we can make our conclusion that our algorithms *HeuriRed* and *HeuriComRed* are both faster and better than *genetic reduct* in most cases. Furthermore, in contrast to *HeuriRed* and *genetic reduct*, our algorithm *HeuriComRed* is a complete attributes reduction algorithm.

TABLE 2. Comparison between our algorithms and *Genetic Reduct*

Dataset	Insta.	Orig. Attri.	<i>HeuriRed</i>			<i>HeuriComRed</i>			<i>Genetic Reduct</i>	
			Red	T(s)	IsCom.	Red	T(s)	IsCom.	Red	T(s)
waveform-40	300	40	5	0.68	YES	5	0.771	YES	8	104.0
wine	118	13	3	0.08	YES	3	0.05	YES	5	1.0
Soybean-Small	31	35	2	0.01	YES	2	0.01	YES	2	1.0
LungCancerData	32	56	4	0.04	YES	4	0.05	YES	4	2.0
tokyo1	479	44	6	1.412	YES	6	1.422	YES	11	17.0
audiology	150	69	12	0.1	YES	12	0.11	YES	26	1.0
Allhypo	1866	29	12	1.662	YES	12	1.672	YES	18	3.0
vehicle	564	18	7	0.951	YES	7	0.961	YES	13	1.0
horse_colic	300	22	3	0.25	YES	3	0.26	YES	5	2.0
waveform-21	300	21	5	0.32	YES	5	0.33	YES	9	1.0
sonar	138	60	6	0.15	YES	6	0.14	YES	6	9
mushroom	400	22	3	0.33	YES	3	0.32	YES	4	4
TicGame	638	9	6	0.15	YES	6	0.16	YES	7	1
Australian	690	14	7	0.711	YES	7	0.68	YES	10	1
banding	138	29	4	0.08	YES	4	0.09	YES	7	2
buggyCrx	552	15	6	0.49	YES	6	0.5	YES	11	1
Churn	1000	20	1	0.831	YES	1	0.871	YES	1	2
Voting	435	16	10	0.3	NO	9	0.3	YES	9	1
Letter	500	16	7	0.881	NO	6	0.92	YES	6	2
Auto	136	25	10	0.07	NO	9	0.08	YES	9	< 1
Satimage	400	36	18	0.57	NO	17	0.66	YES	20	1

**6. Conclusions.** In this paper, based on discernibility matrix, we develop a heuristic reduction algorithm *HeuriRed* and a complete heuristic reduction algorithm *HeuriComRed* for attributes reduction problems. By introducing four matrixes, time and space complexity of our algorithms are both  $O(|A||U|^2)$ . We run *HeuriRed*, *HeuriComRed* and *genetic reduct* on twenty-one problems from UCI repository [21]. The experimental results not only indicate the robustness and effectiveness of our algorithms *HeuriRed* and *HeuriComRed*, but also offer proofs for the completeness of our algorithm *HeuriComRed*.

## REFERENCES

- [1] Pawlak, Z., Rough sets, *International Journal of Computer and Information Science*, vol.11, no.5, pp.341-356, 1982.
- [2] Pawlak, Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, 1991.
- [3] Pal, S. K. and A. Skowron, *Rough Fuzzy Hybridization - A New Trend in Decision-making*, Springer, 1999.
- [4] Wang, J. and J. Wang, Reduction algorithms based on discernibility matrix: the ordered attributes method, *Journal of Computer Science & Technology*, vol.16, no.6, pp.489-504, 2001.
- [5] Hu, K., Y. Lu and C. Shi, *Feature Ranking in Rough Sets*, <http://citeseer.nj.nec.com/587273.html>.
- [6] Hu, K., L. Diao, Y. Lu and C. Shi, *Sampling for Approximate Reduct in Very Large Datasets*, <http://citeseer.nj.nec.com/587308.html>.
- [7] Hu, X., T. Y. Lin and J. Han, A new rough set model based on database systems, *Journal of Fundamental Informatics*, vol.59, pp.135-152, 2004.
- [8] Øhrn, A. and J. Komorowski, ROSETTA - a rough set toolkit for analysis of data, *Proc. of the Third International Joint Conference on Information Sciences*, Durham, NC, USA, pp.403-407, 1997.
- [9] Jelonek, J., K. Krawiec and R. Slowinski, Rough set reduction of attributes and their domains for neural networks, *International Computational Intelligence*, vol.11, no.2, pp.339-347, 1995.
- [10] Guan, J. W. and D. A. Bell, Rough computational methods for information systems, *Artificial Intelligences*, vol.105, no.1-2 pp.77-103, 1998.
- [11] Lui, S., *Research on Rough Set Theory in Knowledge Discovery*, Ph.D. Thesis, Institute of Computer Technology, China Academy of Sciences.
- [12] Merz, C. J. and P. Murphy, *UCI Repository of Machine Learning Database*, <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [13] Vinterbo, S. and A. Øhrn, Minimal approximate hitting sets and rule templates, *International Journal of Approximate Reasoning*, vol.25, no.2, pp.123-143, 2000.
- [14] Vinterbo, S. and A. Øhrn, Minimal approximate hitting sets and rule templates, in *Predictive Models in Medicine: Some Methods for Construction and Adaptation*, Department of Computer and Information Science, NTNU report, vol. 130. <http://www.idi.ntnu.no/~staalv/dev/thesis.ps.gz>, 1999.
- [15] Johnson, D. S., Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences*, pp.256-278, 1974.
- [16] Bazan, J. G., A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables, in *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, Polkowski and Skowron (editors), Physica-Verlag, Heidelberg, Germany, Chapter 17, pp.321-365, 1998.
- [17] Bazan, J. G., A. Skowron and P. Synak, Dynamic reducts as a tool for extracting laws from decision tables, *Proc. of the International Symposium on Methodologies for Intelligent Systems, Lecture Notes in Artificial Intelligence*, Springer-Verlag, vol.869, pp.346-355, 1994.
- [18] Bazan, J. G., *Dynamic Reducts and Statistical Inference*, 1996.
- [19] Polkowski, L. and A. Skowron (editors), *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, Physica-Verlag, Heidelberg, Germany, 1998.