International Journal of Innovative
Computing, Information and Control
Volume 3, Number 3, June 2007
 ISSN 1349-4198
pp. 725–736

AN EFFICIENT GLOBAL OPTIMIZATION APPROACH FOR ROUGH SET BASED DIMENSIONALITY REDUCTION

Songbo Tan, Xueqi Cheng and Hongbo Xu
Intelligent Software Department
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100080, P. R. China
tansongbo@software.ict.ac.cn; cxq@ict.ac.cn; hbxu@software.ict.ac.cn

Received January 2006; revised August 2006

Abstract. *The theory of rough set, proposed by Pawlak, provides a formal tool for knowledge discovery from imprecise and incomplete data. Dimensionality reduction (RED) is a crucial problem for rough set based data mining. Unfortunately, it proves to be a NP-hard problem. Many reduction algorithms based on heuristic information have been developed, but they are all based on Boolean space* $\{0,1\}^m$. *In this paper, we introduce the Universal RED problem model, or UniRED, which transforms the discrete attributes reduction problems on Boolean space* $\{0,1\}^m$ *into continuous global optimization problems on real space* E^m. *Based on this transformation, we develop coordinate gradient descent algorithm RED2.1, and coordinate direct descent algorithm RED3.1 for attributes reduction problems. In order to investigate the robustness and efficiency of proposed algorithms, we execute our algorithm RED3.1 on problems from UCI. Meanwhile, the comparison between algorithm RED3.1 and other famous reduct algorithms, such as dynamical reduct algorithm and genetic reduct algorithm, is presented. The experimental results indicate the robustness and efficiency of our algorithms.*
Keywords: Rough set, Dimensionality reduction, Information processing, Knowledge discovery

1. **Introduction.** The rough set theory developed by Pawlak in the early 1980's [1,2] provides a framework for knowledge discovery. Rough set is especially useful for domains where data collected is imprecise and/or incomplete about the domain objects. 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. A reduct is the minimal set of attributes that preserves the indiscernability relation, that is, classification power of the original dataset. The search of a reduct is similar to feature selection problem. All reducts of a dataset can be found by constructing and simplifying the discernibility function [3]. Unfortunately, It has been shown that finding minimal reduct or all reducts are both NP-hard problems.

Therefore, methods to solve these NP-hard problems play an important role in the development of rough set-based data mining. There has been a great interest in designing efficient algorithms to solve these problems. Most algorithms adopt some kind of search algorithm, varying in search direction, search strategy and evaluation measure. These conventional algorithms fall into three categories: the reduction algorithms based on discernibility matrix [4-6,15,16,23], the reduction algorithms based on positive region [7-12] and the reduction algorithms based on random strategy [13,14,17-20].

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 the counts the frequency of other attributes and selects again. This algorithm proceeds until a reduct set is found.

Hu [7,8] proposes a heuristic algorithm based on positive region. In his algorithm, significance of attribute is defined on the basis of dependency coefficient. After greedily adding attributes that mostly increase dependency coefficient of candidate reduct set, the algorithm has a pruning procedure to ensure the minimum of resulting reduct set.

Bazan [17-19] proposed a dynamic reduct algorithm based on random strategy. He argues that part of reduct set is chaotic i.e., it 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 the 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 subsTable created by randomly sampling of a given decision Table.

Nevertheless, all of these reduction methods treat the reduct problem as a discrete optimization problem. In this work, we show how to translate reduct problem into a continuous, unconstrained, global optimization problem. We then show how to use existing global optimization methods, such as coordinate descent methods, to solve this transformed problem. Experimental results indicate that these optimization methods, e.g., coordinate descent methods, achieve significant performance improvement for certain classes of reduct problems.

The rest of this paper is organized as follows: In the next section, we give some preliminaries of rough set theory and then discuss the universal RED (*UniRED*) problem models in real space. Two algorithms based on *UniRED* are proposed in Section 3. Experimental results of RED3.1 are given in Section 4. Finally Section 5 concludes this paper.

2. **UniRed: The Universal Reduction Problem Model.**

2.1. **Preliminaries.**

In this section we provide some basic definitions of rough set theory. Detailed description of the theory can be found in [2,3].

An Information system is a pair $\boldsymbol{A}=(U, A)$, where U is a non-empty, finite set of objects called the *universe* and A is a non-empty finite set of *attributes*. An information system $\boldsymbol{A}=(U,A \cup \{d\})$, where $d \notin A$, is usually called a ***decision Table***. The elements of A we call the *conditional attributes* and d is called *decision attribute*.

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

A ***discernibility function*** can be constructed from discernibility matrix by or-ing($\vee$) all attributes in c_{ij} and then and-ing($\wedge$) all of them together. After simplifying the discernibility function using absorption law, the set of all prime implicants determine the set of all reducts of the information system. However, simplifying discernibility function for reduct is a NP-hard problem.

If we look each attribute as one variable, look each variable of each non-empty c_{ij} as one literal, and look each non-empty c_{ij} as one clause, obviously we can take the discernibility function as a kind of conjunctive normal form (CNF) formulas.

We take the following decision Table for example. The letters a_1, a_2, a_3, a_4 denote conditional attributions and the letter d denotes decision attribute.

Table 1. The 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 our discernibility function f as follows:

$$f = (a_1 \vee a_2 \vee a_3 \vee a_4) \wedge (a_1 \vee a_3) \wedge (a_1 \vee a_4) \wedge (a_1 \vee a_2)$$

Any reduct of A is equal to an assignment of truth values to variables (a_1, a_2, a_3, a_4) that makes the above discernibility function satisfiable (CNF formula). Similarly, a minimal reduct of A is equal to an assignment with minimum truth-values to variables (a_1, a_2, a_3, a_4) that make the above discernibility function (CNF formula) satisfiable.

2.2. **UniRed: The universal reduction problem model.** In order to employ a global optimization to solve the reduction problem, we need to set up a special model that transforms a discrete reduction problem in Boolean space $\{0,1\}^m$ into a continuous reduction problem on real space E^m. In this section, we describe two universal transformation models, i.e., *QuaUniRed* and *AbsUniRed*, for the reduction problem.

2.2.1. *The quadratic universal reduct model (QuaUniRed).* In order to use global optimization techniques for reduction problem on real space, we need the following operations [22] to extend the well-known Boolean DeMorgan Laws into the universal DeMorgan Laws on real space E^m.

Operator Generalization: we first replace Boolean $\vee$ and $\wedge$ operators with real operators $+$ and $\times$, respectively.

Variable Generalization: we extend Boolean vector $\boldsymbol{x}$ on Boolean space $\{0,1\}^m$ to real vector $\boldsymbol{y}$ on real space E^m. The correspondence between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as follows:

$$x_i = \begin{cases} True & \text{if } y_i = 1 \\ False & \text{if } y_i = -1 \\ undefined & \end{cases}$$

which is equivalent to the following:

$$x_i = \begin{cases} True & \text{if } 1 - y_i = 0 \\ False & \text{if } 1 + y_i = 0 \\ undefined & \end{cases}$$

The following transformations are used to derive the objective function in the *QuaUniRed* model.

Literal Transformation: Based on real vector, a literal function is defined for each literal as follows:

$$q_{ij}(y_j) = \begin{cases} (1-y_j)^2 \\ 1 \end{cases}$$

In this quantitative formulation, the solution assignment is coded explicitly in the literal function.

Clause Transformation: A clause function $c_i(y)$ from E^m to E^1 is defined as a product with at most m literal functions:

$$c_i(y) = \prod_{j=1}^{m} q_{ij}(y_j)$$

The clause transformations each clause $c_i(x)$ on Boolean space $\{0,1\}^m$ into a clause function $c_i(y)$ on real space E^m.

The QuaUniRed Model: Using literal and clause functions, a *CNF* formula $F(x)$ on Boolean space is transformed into a continuous *DNF* object function $f(y)$ on real space E^m.

$$f(y) = \sum_{i=1}^{n} c_i(y) = \sum_{i=1}^{n} \prod_{j=1}^{m} q_{ij}(y_j)$$

Whereas, this object function $f(y)$ can only guarantee to find a reduct of conditional attributes A. In fact our goal is to find a minimal reduct of conditional attributes A. Therefore improve above object function formula as follows:

$$f(y) = \sum_{i=1}^{n} c_i(y) = \sum_{i=1}^{n} \prod_{j=1}^{m} q_{ij}(y_j) + \sum_{j=1}^{m} (1+y_j)^2 \tag{1}$$

We take the discernibility function of section 2.1 as an example. Following the above formula, we can obtain the continuous object function $f(y)$ on real space E^m.

$$\begin{aligned} f(y) = {} & (1-a_1)^2(1-a_2)^2(1-a_3)^2(1-a_4)^2 + (1-a_1)^2(1-a_3)^2 \\ & + (1-a_1)^2(1-a_4)^2 + (1-a_1)^2(1-a_2)^2 \\ & + (1+a_1)^2 + (1+a_2)^2 + (1+a_3)^2 + (1+a_4)^2 \end{aligned}$$

But the above improved object function formula still can not be guaranteed to find a minimal reduct of conditional attributes A. For example, given a new discernibility function as follows:

$$\begin{aligned} f(y) = {} & (1-a_1)^2(1-a_2)^2(1-a_3)^2 + (1-a_2)^2(1-a_4)^2 \\ & + (1-a_1)^2(1-a_2)^2 + (1-a_4)^2 \\ & + (1+a_1)^2 + (1+a_2)^2 + (1+a_3)^2 + (1+a_4)^2 \end{aligned}$$

Obviously, we know the minimal reduct of conditional attributes A is $\{a_2,a_4\}$ or $\{a_1, a_4\}$. That is to say the solution of the object function is $(-1,1,-1,1)$ or $(1,-1,-1,1)$. And the value of object function is 8 for each solution. But the solution point $(-1,1,-1,-1)$ can also obtain the same object function value of 8. Accordingly, we can say, the function has three minima. In a practical search for the minima by global optimization, the possibility of finding one of above three solution points is 1/3. Nevertheless, to our knowledge, the solution point $(-1, 1, -1, -1)$ is not the minimal reduct of conditional attributes A, for it cannot satisfy the discernibility function. Consequently, before the acceptance of a minimum of the object function, we should test the current solution point as to whether it satisfies the discernibility function.

2.2.2. *The absolute universal reduct model (AbsUniRed).* . If we replace the above literal transformation by following definition:

$$q_{ij}\left(y_j\right) = \begin{cases} \left|1 - y_j\right| \\ 1 \end{cases}$$

Like the quadratic model, we can deduce our object function formula for *AbsUniRed* model as follows:

$$f\left(y\right) = \sum_{i=1}^{n} c_i\left(y\right) = \sum_{i=1}^{n}\prod_{j=1}^{m} q_{ij}\left(y_j\right) + \sum_{j=1}^{m}\left|1 + y_j\right| \tag{2}$$

We take the discernibility function of section 2.1 as an example. Following the above formula, we can obtain the continuous object function $f(y)$ on real space E^m.

$$\begin{aligned} f\left(y\right) = &\left|1 - a_1\right|\left|1 - a_2\right|\left|1 - a_3\right|\left|1 - a_4\right| + \left|1 - a_1\right|\left|1 - a_3\right| \\ &+ \left|1 - a_1\right|\left|1 - a_4\right| + \left|1 - a_1\right|\left|1 - a_2\right| \\ &+ \left|1 + a_1\right| + \left|1 + a_2\right| + \left|1 + a_3\right| + \left|1 + a_4\right| \end{aligned}$$

3. **Proposed Algorithms.**

3.1. **Optimization method.** Global optimization is also called as global search in some area. There are a lot of global optimization methods developed for real optimization problems, among which, to our knowledge, the most classic and popular global optimization methods are the gradient descent method and the Newton method. But it is very time-consuming to calculate the gradient of object function. The calculation of Hessian matrix used by the Newton method is especially time consuming. Further more, the line search used by the gradient descent method and the Newton method is also very time-consuming. Therefore, the gradient descent method and Newton method are not suiTable for large-scale problems.

To our surprise, the simple coordinate descent method turns out to be the most effective method not only in terms of time but also in terms of space for continuous reduction problems. Hence we base our algorithms on the coordinate descent method. In the following, we will introduce two kinds of coordinate descent methods: the coordinate gradient descent method and the coordinate direct descent method.

3.1.1. *Coordinate gradient descent method.* In each iteration, only one variable y_i is chosen as the descent dimension. Variables other than y_i, i.e., y_j $(j = 1, 2, \ldots, i-1, i+1, \ldots, m)$, remain as constants. Therefore the object function is transformed into a single variable function with respect to the variable y_i:

$$f(y) = a_i\left(1 - y_i\right)^2 + b_i\left(1 + y_i\right)^2 + c_i \tag{3}$$

To our knowledge, the minimum of the single variable function is the point making the derivative of $f(y)$ equal zero.

$$f'(y) = -2a_i\,(1 - y_i) + 2b_i\,(1 + y_i)$$

We set $f(y)$=0, that is,

$$-2a_i\,(1 - y_i) + 2b_i\,(1 + y_i) = 0$$

$$y_i' = \frac{(2a_i - 2b_i)}{(2a_i + 2b_i)} = \frac{(a_i - b_i)}{(a_i + b_i)} \tag{4}$$

y_i' is the minimum point. As a result, in each iteration, what we need to do is compute the value a_i and b_i. Since there is only one term $(1 + y_i)^2$ in the object function with respect to y_i, consequently b_i equals to 1. On the contrary, there may be more than one terms $(1 + y_i)^2$ in the object function with respect to y_i, hereby a_i is range from 0 to ∞. In practice, the value of a_i may overflow, consequently, we cannot use above formula any more. When $a_i \to \infty$, in the view of mathematics, the limit of y_i' is 1. Accordingly, if a_i overflows, we can set y_i' equal 1.

Based on this *QuaUniRed* model and coordinate gradient descent method, *RED2.1* is developed.

3.1.2. *Coordinate direct descent method.* Unlike the above coordinate gradient descent method, our coordinate direct descent method doesn't use the gradient with respect to one variable. As we investigate the object function with respect to one variable y_i, we have a faster and more efficient descent method. Since each variable y_i appears in one clause function a maximum of one time only, the function can be expressed as follows:

$$f(y) = a_i\,|1 - y_i| + b_i\,|1 + y_i| + c_i \tag{5}$$

To our knowledge, the above single variable function takes its minimum value with respect to variable y_i at point either $y_i = 1$ or $y_i = -1$. Thus, if a_i is bigger than or equivalent to b_i, we obtain $y_i = 1$; otherwise, we obtain $y_i = -1$.

If a discernibility function including only one clause is given as follows:

$$|1 - x_1|\,|1 - x_2|\,|1 - x_3|\,|1 - x_4|\,|1 - x_5|\,|1 - x_6|\,|1 - x_7|\,|1 - x_8|\,|1 - x_9|\,|1 - x_{10}|\,|1 - x_{11}|\,|1 - x_{12}|$$

then, if the initial solution is generated as $(-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0.1)$, according to the round off function *RoudOff* () introduced in section 3.2, we can obtain the solution as $(-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1)$. Obviously, the solution satisfies the discernibility function and we can take it as a reduct. Nevertheless, according to our coordinate direct descent method, we calculate a_1 with respect to x_1, and obtain the value 921.6. Since $a_1 = 921.6 > b_1 = 1$, we set x_1=1. Execution of the algorithm in this manner will produce further poor solutions, which may decrease the performance of our algorithm based on coordinate direct descent method. This phenomenon motivates us to define a weight w for all a_i, that is to say, after computing the value of a_i each time, we set $a_i = w * a_i$. Fortunately, this strategy improves in most cases, the performance of our algorithm in our experiment. In our experiment, we set our weight to 0.0001.

Based on the *AbsUniRed* model and coordinate direct descent method, we develop algorithm *RED3.1*. To our surprise, *RED3.1* has a faster convergent speed than *RED2.1*. Therefore we set the maximum search times to $2 * m$ from $10 * m$ which *RED2.1* requires in order to search. As a result, the practical running time of *RED2.1* is about 5 times that of *RED3.1*. Further, *RED3.1* achieves a better performance than algorithm *RED2.1*. Therefore, we employ *RED3.1* to run all datasets from the UCI database [21].

3.2. **Proposed algorithms.**

3.2.1. *Necessary functions.* Before giving the outline of proposed algorithm, we first introduce some necessary functions.

y'=*RoundOff*(y): if y_i >=0 we return 1, else return -1. Therefore, we obtain a binary vector y' in $\{0,1\}^m$. For example, if y is $(0.1, -0.4, 0, 0.5)$, we obtain $y' = (1, -1, 1, 1)$.

y_i'=*RoundOff_OneVar*(y_i): Only perform roundoff operation for one variable y_i. if y_i >=0 we return 1, else return -1.

satflag=*IsSatDisFun*(y'): if the binary vector y' in $\{0,1\}^m$ can satisfy the discernibility function we return TRUE, else return FALSE.

satflag=*IsSatDisFun_OneVar*(y_i'): Only commit satisfaction test for the clauses with respect to variable y_i'. If the binary vector y' in $\{0,1\}^m$ can satisfy the clauses with respect to the variable y_i', we return TRUE, else return FALSE.

f_2=*RealObjFun*(y'): return the sum of 1 is in the binary vector y' in $\{0,1\}^m$. For example, if $y' = (1, -1, 1, 1)$, we return 3; if $y' = (1, -1, 1, -1)$, return 2.

3.2.2. *The RED2.1 algorithm. Initialization*: to start, procedure *LoadData*() loads the decision Table. If the corresponding function gets a variable that satisfies the discernibility function. Then we save the Boolean variable after the procedure *RoundOff*(y). Using function *RealObjFun*(y') we compute the number of selected attributes. At the same time we set the maximum search times to $10 * m$.

Global Search: in this stage, we employ the coordinate descent method to minimize the single variable function with respect to each variable y_i. We then round off the variables. If it can reduce the number of selected attributes and satisfy the discernibility function, we save the round-off variables. Then we select next variable y_{i+1} for minimization.

Termination: in practice, it is rather hard to decide whether it is sufficiently close to a global minimum. But according to our experience, if we set the max search times to $10 * m$, that is to say, if we execute the *for-loop* for $10 * m$ times, it is sufficient close to a global minimum in most cases.

Running time: the running time of the *RED2.1* can be estimated as follows. Given a discernibility function with n clauses, m variables, on average l literals in each clause and on average p clauses with respect to one variable. In one iteration of the *for-loop*, minimizing the single variable function with respect to one variable is of *O(pl)*. The function *RoundOff_OnVar*(y_i) is of $O(1)$. The function *IsSatDisFun_OneVar*$(y_i^{'})$ can be computed in *O(pl)*. The running time of *RealObjFun*(y') is of $O(m)$. The assignment of $y^{'}$ to *best_y* can be done in $O(m)$. Therefore the total running time of *for-loop* is of $O(m(m + m + pl + pl + 1))$, i.e., $O(m(m + pl))$.

3.2.3. *The RED3.1 algorithm.* The *RED3.1* algorithm is similar to the *RED2.1* algorithm. Two main differences from *RED2.1* algorithm are the max search times and the global optimization method. The *RED3.1* algorithm sets the max search times to $2 * m$.

```
Procedures RED2.1 (){
    //load the decision table data
    LoadData();
    //get a vector satisfying discernibility function
    y=Initilization();
    best_y= RoundOff(y);
    MaxSearchNum=10*m;
    best_f=RealObjFun(y');

    for(i=0; i< MaxSearchNum; i++){
        //minimizer
        i=i%m
        minimize f(y) with respect to yi
        yi'= RoundOff_OnVar(yi);
        if(IsSatDisFun_OneVar(yi')){
            f= RealObjFun(y');
            if(best_f>f){
                best_f=f;
                best_y=y';
            }
        }
        else
        //ensure y satisfies the discernibility function
        yi=- yi;
    }
}
```

FIGURE 1. The algorithm of *RED2.1*

In despite of the difference of optimization method between *RED2.1* and *RED3.1*, the time complexity of the two algorithms is equal largely. Therefore the time complexity of *for-loop* for *RED3.1* algorithm is $O(m(m + m + pl + pl + 1))$, namely, $O(m(m + pl))$.

4. **Experiment Results.** In this section, we give experimental results of the attributes reduction algorithm *RED3.1* based on some problems form UCI database [21]. Our algorithm was coded in C and tested using a personal computer with a single AMD dragon 800 MHz processor and 128Mb memory. In order to investigate the robustness and efficiency of our algorithms, we perform a comparison with dynamical reduct [17-20] and genetic reduct [13,14]. We execute the dynamical reduct and genetic reduct in the ROSETTA system [11].

It is worth noticing that, in our experiments, each reduct obtained by algorithm *RED3.1* can satisfy the corresponding discernibility function, i.e., each item in the corresponding discernibility matrix is involved with at least one attribute contained in the reduct.

```
Procedures RED3.1 (){
    //load the decision table data
    LoadData();
    //get a vector satisfying discernibility function
    y=Initilization();
    best_y= RoundOff(y);
    MaxSearchNum=2*m;
    best_f=RealObjFun(y');

    for(i=0; i< MaxSearchNum; i++){
        //minimizer
        i=i%m;
        calculate a_i with respect to y_i;
        a_i = w *a_i;
        if (a_i >= 1 or overflowed)
            y_i =1;
        else   y_i =-1;
        y_i' = RoundOff_OnVar(y_i);
        if(IsSatDisFun_OneVar(y_i')){
            f= RealObjFun(y');
            if(best_f > f){
                best_f = f;
                best_y= y';
            }
        }
        else
        //ensure y satisfies the discernibility function
        y_i=- y_i;
    }
}
```

Figure 2. The algorithm of *RED3.1*

In the case of incomplete datasets, such as an echo cardiogram, we complete it by using the conditional mean complete algorithm. If the attribute with missing values is numerical attribute, we substitute missing values with the mean value of all observed values for that attribute; for a 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 discretize continuous datasets, 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.

Table 2 reports the results of the comparison between *RED3.1* and dynamical reduct. twenty datasets are randomly selected from UCI database. The dynamical reduct algorithm produces many different reducts for a given dataset with different support rate. Thus in order to compare, we select the reduct with the highest support rate when all reducts support rate are smaller than 100%, or select the minimal reduct when at least one reduct's support rate equals 100%. Since it is rather time consuming to run a dynamical reduct, we only select datasets whose objects numbers are smaller than 300. At the same time, the ROSETTA system cannot support the execution of a dynamical reduct on datasets with larger than 20 attributes.

Table 2 indicates that the dynamical reduct is at least one order magnitude slower than our algorithm *RED3.1*. For some large datasets, such as mofn-3-7-10, new-thyroid and Breast Cancer Data, the time required by dynamical reduct is about 100 times larger than that of our algorithm *RED3.1*. Among the twenty datasets, five datasets such as voting, the dynamical reducts finds smaller reduct than our algorithm *RED3.1*; six datasets, such as Wine, the dynamical reduct finds larger reducts than our algorithm *RED3.1*. Therefore, we conclude that dynamical reduct obtains a similar reduct with our algorithm *RED3.1*, but it is much more time dependent than our algorithm.

TABLE 2. Comparison between *RED3.1* and dynamical reduct

Dataset	Is Consistent	Instance	Original Attribute	*RED3.1*		*Dynamic Reduct*	
				Reduct Attribute	Time (Sec.)	Reduct Attribute	Time (Sec.)
Voting	YES	300	16	7	3.845	6	168
Zoo	YES	67	16	4	0.16	4	3
Lymphography	YES	108	18	6	0.42	5	2
BreastCancer Data	YES	191	9	8	0.38	8	42
mofn-3-7-10	YES	300	10	7	0.681	7	141
parity5+5	YES	100	10	5	0.1	5	6
monk1	YES	124	6	3	0.11	3	9
monk2	YES	169	6	6	0.17	6	22
monk3	YES	122	6	4	0.1	4	9
echocardiogram	NO	87	7	7	0.06	6	4
labor-neg	YES	40	16	3	0.1	2	1
wine	YES	118	13	4	1.031	5	14
new-thyriod	YES	215	5	4	0.41	5	49
glass	YES	142	9	6	0.64	9	13
heart	YES	180	13	9	0.781	8	39
hungrarian	YES	196	13	3	0.931	5	41
Hepatitis	YES	103	19	6	0.47	7	15
liver-disorder	YES	230	6	4	0.721	6	58
cars	YES	261	8	1	1.17	1	94
post-operative	NO	60	8	8	0.02	8	2

Table 3 reports the results of comparison between *RED3.1* and a genetic reduct. Seventeen datasets are randomly selected from the UCI database. The genetic reduct algorithm produces many different size reducts for a given dataset with 100% support rate. Thus in order to compare, we select the smallest reduct. For the first three datasets, which are waveform-40, soybean-Small and LungCancerData, the genetic reduct algorithm takes longer than our algorithm *RED3.1*. And for the last three datasets the genetic reduct algorithm achieves better reducts than our algorithm *RED3.1* by spending less time than our algorithm *RED3.1*. For the remainder datasets, our algorithm *RED3.1* takes longer time than the genetic reduct, but finds a smaller reduct than the genetic reduct. Our algorithm *RED3.1* is a litter slower than the genetic reduct, but performs better.

Table 3. Comparison between *RED3.1* and genetic reduct

Dataset	Is Consistent	Instance	Original Attribute	*RED3.1*		*Genetic Reduct*	
				Reduct Attribute	Time (Sec.)	Reduct Attribute	Time (Sec.)
waveform-40	YES	300	40	5	53.106	8	104.0
soybean-Small	YES	31	35	2	0.09	2	1.0
LungCancerData	YES	32	56	6	0.33	4	2.0
wine	YES	118	13	4	1.031	5	1.0
tokyo1	YES	479	44	7	88.177	11	17.0
audiology	YES	150	69	14	1.201	26	1.0
satimage	YES	550	36	10	140.822	15	1
Allhypo	NO	1866	29	12	22.412	18	3.0
vehicle	YES	564	18	8	45.115	13	1.0
waveform-21	YES	300	21	5	15.622	9	1.0
pima	YES	512	8	6	4.686	8	< 1.0
horse_colic	YES	300	22	3	9.293	5	2.0
hungrarian	YES	196	13	3	0.931	8	< 1.0
glass	YES	142	9	6	0.64	9	< 1.0
sonar	YES	138	60	7	12.578	6	9
primary-org	NO	226	17	15	1.712	14	< 1.0
heart	YES	180	13	9	0.781	8	< 1.0

5. **Conclusions.** In this paper, we introduce the Universal RED problem model, or *UniRED*, which transforms the discrete dimensionality reduction problems on Boolean space $\{0,1\}^m$ into continuous global optimization problems on real space E^m. Based on coordinate descent methods, we develop two algorithms *RED2.1* and *RED3.1* for attributes reduction problems. We run our algorithm *RED3.1* on data from UCI [21]. The experimental results indicate the robustness and efficiency of our algorithms.

We believe that this research only scratches the surface of what can be achieved with *UniRED*. Future work is to seek new techniques to enhance the performance of *UniRED* and to apply it to new domain datasets, such as text datasets and gene expression datasets both of which contain a large number of features (or attributes).

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., *Knowledge Discovery in Databases: An Attribute-oriented Rough Set Approach*, Ph.D. Thesis, Regina University, 1995.

[8] Hu, X. H. and N. Cercone, Learning in relational databases: A rough set approach, *International Journal of Computational Intelligence*, vol.11, no.2, pp.232-338, 1995.

[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, nos.1-2, pp.77-103, 1998.

[11] The ROSETTA homepage [http://www.idi.ntnu.no/∼aleks/rosetta/], Norwegian University of Science and Technology, Department of Computer and Information Science.

[12] Lui, S., *Research on Rough Set Theory in Knowledge Discovery*, Ph.D. Thesis, Institute of Computer Technology, China Academy of Science, China, 2003.

[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 1999:130. http://www.idi.ntnu.no/∼staalv/dev/thesis.ps.gz.

[15] Johnson, D. S., Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences,* vol.9, pp.256-278, 1974.

[16] Nguyen, S. H. and H. S. Nguyen, *Some Efficient Algorithms for Rough Set Methods*, 1996.

[17] 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*, L. Polkowski and A. Skowron (eds), Chapter 17, pp.321-365, 1994.

[18] 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.

[19] Bazan, J. G., *Dynamic Reducts and Statistical Inference*. 1996.

[20] Polkowski, L. and A. Skowron (eds), *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, vol.18 of Studies in Fuzziness and Soft Computing, Physica-Verlag, Heidelberg, Germany, 1998.

[21] Merz, C. J. and P. Murphy, *UCI Repository of Machine Learning Database*, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[22] Gu, J., Global optimization for satisfiability (SAT) problem, *IEEE Transactions on Knowledge and Data Engineering*, vol.6, no.3, pp.361-381, 1994.

[23] Tan, S., H. Xu and J. Gu, Efficient algorithms for attributes reduction problem, *International Journal of Innovative Computing, Information & Control*, vol.1, no.4, pp.767-777, 2005.