Turk J Elec Eng & Comp Sci, Vol.19, No.4, 2011, © TÜBİTAK
doi:10.3906/elk-1008-726

A logic method for efficient reduction of the space complexity of the attribute reduction problem

Mehmet HACIBEYOĞLU[1], **Fatih BAŞÇİFTÇİ**[2,*], **Şirzat KAHRAMANLI**[1]
[1] *Department of Computer Engineering, Faculty of Engineering and Architecture, Selçuk University, 42003 Selçuklu, Konya-TURKEY*
e-mail: {hacibeyoglu, sirzat}@selcuk.edu.tr
[2] *Department of Electronic and Computer Education, Faculty of Technical Education, Selçuk University, 42003 Selçuklu, Konya-TURKEY*
e-mail: basciftci@selcuk.edu.tr

Received: 31.08.2010

Abstract

The goal of attribute reduction is to find a minimal subset (MS) R of the condition attribute set C of a dataset such that R has the same classification power as C. It was proved that the number of MSs for a dataset with n attributes may be as large as $\binom{n}{n/2}$ and the generation of all of them is an NP-hard problem. The main reason for this is the intractable space complexity of the conversion of the discernibility function (DF) of a dataset to the disjunctive normal form (DNF). Our analysis of many DF-to-DNF conversion processes showed that approximately $(1 - 2/\binom{n}{n/2} \times 100)\,\%$ of the implicants generated in the DF-to-DNF process are redundant ones. We prevented their generation based on the Boolean inverse distribution law. Due to this property, the proposed method generates $0.5 \times \binom{n}{n/2}$ times fewer implicants than other Boolean logic-based attribute reduction methods. Hence, it can process most of the datasets that cannot be processed by other attribute reduction methods.

Key Words: *Information system, dataset, attribute reduction, feature selection, discernibility function, computational complexity, reduct*

1. Introduction

In most *information systems* such as data mining, decision support techniques for pattern recognition and neural networks training the *data tables*, called *datasets*, are used. A basic problem for many practical applications of information systems is the selection of a *minimal subset of attributes* (MSA) sufficient for the classification of objects in the considered dataset [1-3]. This problem, known as *attribute reduction* or *feature selection*, was addressed in many studies, and some approaches based on different reductions of *discernibility matrices* (DMs)

*Corresponding author: Department of Electronic and Computer Education, Faculty of Technical Education, Selçuk University, 42003 Selçuklu, Konya-TURKEY or *discernibility functions* (DFs) have been developed [2,4,5]. Attribute reduction provides the following benefits for processing datasets: reducing the dimensionality of feature space, improving the efficiency and precision of data classification rules, speeding up the data mining algorithms, facilitating the data collection process, and reducing the amount of memory needed for storing the datasets [2,5-7]. For instance, the application of attribute reduction to the dataset "Lung Cancer" [8], with 56 attributes and 32 objects, showed that this dataset could be classified with only 4 attributes from the original 56. That is to say, this dataset could be reduced from 56 to 4 columns and classified by rules with only 4 conditions instead of 56. Due to the mentioned benefits, attribute reduction is widely used for preprocessing the datasets used in many fields, including data mining, decision support systems, knowledge acquisition and discovery, pattern recognition, machine learning, text categorization, customer relationship management, intrusion detection, weather forecast, economic forecasts, fault diagnosis, and forecasting [2,6,7].

It was shown that the number of MSAs for a dataset may be as large as $\binom{n}{n/2}$ [2,5]. Usually these MSAs have different cardinalities, and those of the least cardinality are called *reducts* [2,3,5]. Every dataset may be compactly described by any of its reducts. Since the MSA representing a reduct can be recognized only in comparison to other MSAs, all MSAs should be generated for selecting a reduct. To the best of our knowledge, the generation of all MSAs is possible only by DM-based attribute reduction, which is unfortunately an intractable NP-hard problem [9,10]. Therefore, there have only been a few studies of the DM-based attribute reduction problem. Particularly, in [11], a DM-based algorithm was explained. According to this algorithm, the attribute with the highest frequency in the DF is added to the reduct candidate and all clauses in the DF containing this attribute are removed. When all clauses have been removed, the algorithm returns a reduct [12]. An iterative approach similar to that explained in [11] was proposed in [13]. In each iteration, the attribute with the highest frequency in the DM is selected and all elements involved with this attribute are removed from the DM. The algorithm is iterated until a reduct is found. In [14], a heuristic approach was proposed, in which the rough set operations are implemented by bitwise ones. This approach allows a reduction in the time needed for finding an attribute subset. Although all 3 mentioned approaches are DM-based, they are heuristic; therefore, they do not guarantee the optimality of the results. In [1], the concept of *strong compressibility* was introduced. It was applied to the minimized DF ($DF_{\min}$) in conjunction with an *expansion algorithm.* But as was stated in [5], this approach can only be efficient for small datasets. In [15], a reduct generation algorithm was proposed. It was based on transformations of a discrete dimensionality reduction problem in Boolean space to a continuous global optimization problem in real space. The experimental results showed that this approach is considerably faster than *dynamic* r*educt* [16,17] or *genetic reduct* [18] approaches. In [4], a reduct construction algorithm was proposed. According to this approach, a DM is considered as a *matrix of a system of linear equations* and is minimized by the *Gaussian method.* Then, by uniting the elements of the minimized DM, a reduct is generated. Unfortunately, neither [15] nor [4] contains an estimation of the *space complexity* (the amount of memory required) of the proposed algorithm.

By analyzing many DFs for *disjunctive normal form* (DNF) conversion processes, we observed that the mentioned hardness (complexity) of generating MSAs is mainly caused by the *redundant implicants* (RIs) that occur in these processes. Therefore, we developed an approach preventing the generation of RIs during the DF-to-DNF conversion. It allows us to reduce the *worst-case space complexity* of this conversion by a factor of $D_x = 0.5 \times \binom{n}{n/2} = O(10^{(0.3\times n)-1})$. In order to experimentally prove the correctness and efficiency of the approach, we processed all datasets with no more than 64 attributes from the *UCI repository* [8], and the results were compared with those generated by the well-known exact attribute reduction program *RSES* [19]. For most of the datasets, both our approach and *RSES* generated the same results. But for the 7 datasets given in Table 4, *RSES* failed due to an overflow of the 4 GB of memory, while our approach generated the results by using no more than one-eighth of this memory space.

The rest of this paper is organized as follows. In Section 2, the concept of a discernibility function is given. In Section 3, a DF-to-DNF conversion with prevention of the generation of redundant implicants is explained. In Section 4, the estimation of the efficiency of the proposed approach and the results of experiments on different datasets are given. The paper is concluded in Section 5.

2. The discernibility function of a dataset

2.1. Generating the bit-based discernibility function for a dataset

Consider a dataset $S = (U,C)$, where $U = \{u_1, ..., u_m\}$ and $C = \{a_1, ..., a_n\}$. Each $u \in U$ is called an *object* or *instance*, and each $a \in C$ is called an *attribute* or *feature*. Each attribute $a_j \in C$ has a value set (domain) of $V_a = \{a_j(u_i)\}_{i=1}^{m}$, where $a_j(u_i)$ is the value of attribute a_j on object u_i. The DM of S is an $m \times m$ matrix, the entry H_{ik} of which is obtained as follows [2,3,9,10]:

$$H_{ik} = V(\forall a_j : a_j(u_i) \neq a_j(u_k), j \in \{1, 2, \ldots, n\} \text{ and } i, k \in \{1, 2, \ldots, m\}), \tag{1}$$

where H_{ik} as defined by Eq. (1) is a *propositional logic clause* (PLC) of the following form [2,3].

$$H_{ik} = h_{ikj} = h_{ik1} \vee h_{ik2} \vee \ldots \vee h_{ikj} \vee \ldots \vee h_{ikn}, \tag{2}$$

where $h_{ikj} = a_j$ if $a_j(u_i) \neq a_j(u_k)$ and $h_{ikj} = 0$ if $a_j(u_i) = a_j(u_k)$.

Let us introduce a binary variable b_{ikj} such that:

$$b_{ikj} = 1 \text{ if } a_j(u_i) \neq a_j(u_k) \text{ and } b_{ikj} = 0 \text{ if } a_j(u_i) = a_j(u_k). \tag{3}$$

Eq. (3) allows us to use the following *bit-based clause* (BBC) instead of a PLC as in Eq. (2).

$$B_{ik} = b_{ik1} b_{ik2} \ldots b_{ikj} \ldots b_{ikn} \tag{4}$$

As is seen from the comparison of Eqs. (2) and (4), Eq. (4) has been written with the $\vee$ (OR) signs removed. This is because such a representation of a BBC provides compact storing and processing of the DFs composed from BBCs. However, the presence of this sign between the neighbor components of any BBC must always be taken into consideration when it is subjected to any logic operation. In order to transform a PLC into a corresponding BBC and vice versa, the association between Eqs. (2) and (4) can be fixed by a data structure like the following one.

Struct_PLC-BBC { Unsigned a$_1$:1; Unsigned a$_2$:1; . . . ; Unsigned a$_n$:1; }.

This means that if attribute a_j is present in a PLC, then the value of the j^{th} bit in the corresponding BBC is to be 1 else 0 and vice versa. The DF for a dataset with $U = \{u_1, ..., u_m\}$ and $C = \{a_1, ..., a_n\}$ may be generated by the following algorithm:

Algorithm_1 (U, m, n): Generate_BBCs *//m and n are the numbers of objects and attributes, respectively*

```
Begin
  DF = $\emptyset$
  For i = 1 to m
  { For k = i + 1 to m
    { $B_{ik} = \{0\}^n$
      For j = 1 to n
      { If $a_j(u_i) \neq a_j(u_k)$ then $B_{ik}[j] = 1$
        End If }
       DF = DF $\cup$ $B_{ik}$ }
  }
  Return (DF)
End
```

Algorithm_1 has the *space* and *time complexities* $O(m^2)$ and $O(n \times m^2)$ [5,20], respectively. The time complexity $O(n \times m^2)$ can be reduced to $O(m^2)$ by parallel processing of all attributes within each object, which is possible with binary-encoded values of attributes.

2.2. Minimization of a discernibility function

Most BBCs generated by a procedure like *Algorithm_1* are usually redundant and should be removed [1-3,20,21]. This can be done based on the principle of *expand and eliminate*, generating all clauses and then eliminating redundant ones, according to which the BBCs are compared pair by pair and those BBCs absorbed by other ones are eliminated [1,20,21]. The *minimized DF*, denoted by $DF_{\min}$, is constructed by simply connecting the remaining (irredundant) BBCs by | (*bitwise OR*) operation signs.

Example 1. Let us construct the $DF_{\min}$ for the dataset given by Table 1.

Table 1. Example of a dataset.

C U	a_1	a_2	a_3	a_4	a_5
u_1	1	2	0	1	4
u_2	0	1	2	3	4
u_3	1	2	1	1	1
u_4	0	1	1	5	1

1. *Generating the DF*

$DF = \{B_{12}, B_{13}, B_{14}, B_{23}, B_{24}, B_{34}\} = \{$*11110, 00101, 11111, 11111, 00111, 11010*$\}$

The PLCs represented by these BBCs are as follows:

$H_{12} = a_1 \vee a_2 \vee a_3 \vee a_4$; $H_{13} = a_3 \vee a_5$; $H_{14} = a_1 \vee a_2 \vee a_3 \vee a_4 \vee a_5$;
$H_{23} = a_1 \vee a_2 \vee a_3 \vee a_4 \vee a_5$; $H_{24} = a_3 \vee a_4 \vee a_5$; $H_{34} = a_1 \vee a_2 \vee a_4$.

2. *Minimization of the DF*

BBCs B_{14}, B_{23}, and B_{24} are absorbed by BBC B_{13}, and BBC B_{12} is absorbed by BBC B_{34}. Therefore, in the results, only BBCs B_{13} = 00101 and B_{34} = 11010 remain. Consequently, $DF_{\min} = \{B_{13}, B_{34}\} = \{$*00101, 11010*$\}$.

This expression can be generalized for $Q \leq \binom{n}{n/2}$ BBCs as follows:

$$DF_{\min} = |_{q=1}^{Q} B_q, \tag{5}$$

where | is the bitwise OR operation sign. The *bit-based DNF* of a $DF_{\min}$ can be generated as follows:

$$DNF_{BB} = |_{q=1}^{Q} E(B_q), \tag{6}$$

where $E(B_q)$ is the set of *unit bit vectors* generated by projecting BBC B_q onto nonzero bit-positions [3,22] as follows:

$$E(B_q) = \{Pr_j(B_q) | b_j = 1\}, \tag{7}$$

where $Pr_j(B_q) = 00 \ldots b_j \ldots 0$ and b_j is the value of the jth bit of B_q. For instance, for B_{13} and B_{34} given above, $E(B_{13}) = E(00101) = \{Pr_3(B_{13}), Pr_5(B_{13})\} = \{00100, 00001\}$, and $E(B_{34}) = E(11010) = \{Pr_1(B_{34}), Pr_2(B_{34}), Pr_4(B_{34})\} = \{10000, 01000, 00010\}$, respectively. The propositional logic representation of this DNF_{BB} is $DNF = H_{13} \wedge H_{34} = (a_3 \vee a_5) \wedge (a_1 \vee a_2 \vee a_4)$. A $DF_{\min}$ obtained by Eqs. (5-7) will be processed by *Algorithm_2*, given in Subsection 3.3.

2.3. The estimation of the worst-case number of redundant implicants

In order to estimate the worst-case number of redundant implicants, we have to estimate the memory space used by the abovementioned *expand and eliminate* principle [1,20,21]. For this aim, let us introduce the *concept of the weight* of BBC B_q, defined as the number of 1s in it and denoted by $W(B_q)$. If a $DF_{\min}$ to be converted to DNF_{BB} consists of Q BBCs with the average weight w per BBC, then w^Q bitwise additions will be needed for this conversion. Since each of these additions will produce one implicant, the total number of such implicants is to be w^Q. The numerical analysis of w^Q shows that it reaches its maximum possible value at $w = n/2$ and $Q_{\max} = \binom{n}{n/2}$. Consequently, the *worst-case space complexity* [23,24] of the $DF_{\min}$-to-DNF conversion is $SC_{WC} = (n/2)^{\binom{n}{n/2}}$. On the other hand, the number of prime implicants of a $DF_{\min}$ of n attributes cannot exceed the value $SC_{PI} = \binom{n}{n/2}$ [2,5]. That is, the worst-case number of RIs generated in the process of a $DF_{\min}$-to-DNF conversion may be as large as:

$$N_{RI} = SC_{WC} - SC_{PI} = (n/2)^{\binom{n}{n/2}} - \binom{n}{n/2}. \tag{8}$$

Since the value of N_{RI} obtained by Eq. (8) changes over a very wide range depending on n, in Figure 1, the values of N_{RI} are shown using the *decimal logarithmic scale.* As is seen from Figure 1, for $n = 7$ and $n = 10$, N_{RI} may reach values of orders of 10^{22} and 10^{176}, respectively. This is to say that even a memory with $2^{64} = 10^{19.27}$ address space may theoretically be overflowed during the processing of datasets with $n \geq 7$. In order to avoid this negativity, it is necessary to prevent the generation of RIs.

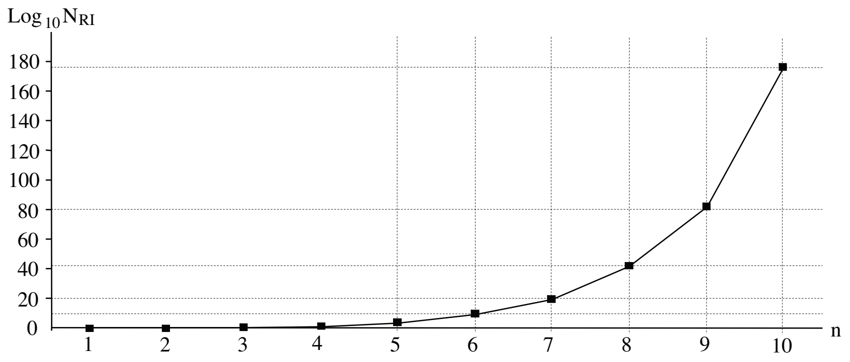

Figure 1. The dependency of N_{RI} on n in the decimal logarithmic scale.

We observed that there are 2 types of RIs: *predetectable* and *postdetectable* RIs. While all predetectable RIs may be identified without being generated, the postdetectable ones may be detected and deleted (ignored) only after they are generated. The analysis of many $DF_{\min}$-to-DNF conversions shows that usually only a small number of RIs are postdetectable.

3. The algorithm of $DF_{\min}$-to-DNF conversion for preventing the generation of RIs

3.1. The basic structure of the algorithm

The algorithm of the conversion of a $DF_{\min}$ to a DNF is based on the following iterative implementation of Eq. (6):

$$\left.\begin{aligned} &F_0 = \{\{0\}^n\} \\ &F_q = F_{q-1}|E(B_q), \forall q = 1, 2, \ldots, Q, \\ &DNF_{BB} = F_Q \end{aligned}\right\} \tag{9}$$

where DNF_{BB} is the bit-based representation of the DNF and F_q is the state of the DNF_{BB} in the q^{th} iteration. F_q is formed by the *bitwise Cartesian summing* of F_{q-1} and $E(B_q)$. Each new component generated by Eq. (9) is separated by a comma from those already generated. Such a representation of F_q, F_{q-1} and $E(B_q)$ allows us to look at each of them as a set and perform the logic operations on the elements of these sets. For instance, according to this representation, the function $G = x|y|z$ is to be considered as the set $G = \{x, y, z\}$. Such a look at equation $F_q = F_{q-1} \mid E(B_q)$ from Eq. (9) allows us to compute it simply, as follows:

$$F_q = \{x|z : x \in F_{q-1} \text{ and } z \in E(B_q)\}, \tag{10}$$

where x is a bit-vector of the weight $1 \le W(x) \le n$ and z is a bit vector always of the weight $W(z) = 1$ (a unit BBC).

3.2. Preventing the generation of predetectable RIs and ignoring the postdetectable RIs

For computation of F_q while preventing the generation of predetectable RIs, we use the following relation that may exist between F_{q-1} and $E(B_q)$.

Let us represent $F_{q-1} = x|y$ as $F_{q-1} = \{x,\ y\}$ and $E(B_q) = z|v$ as $E(B_q) = \{z,\ v\}$, where each $x,y \in F_{q-1}$ is a bit-vector of the weight $1 \le k \le n$ and each $z,v \in E(B_q)$ is a unit bit vector of the weight 1. In order to determine which components of the result of the *Cartesian bitwise summing* of x and y with z and v are redundant, we use the following system of rules.

$$\left.\begin{array}{l}(x|z = x \text{ or } x|v = x) \text{ and } (y|z \neq y \text{ and } y|v \neq y) \rightarrow F_{q-1}|E(B_q) = x \cup y|(z,v)\\[1em](y|z = y \text{ or } y|v = y) \text{ and } (x|z \neq x \text{ and } x|v \neq x) \rightarrow F_{q-1}|E(B_q) = y \cup x|(z,v)\\[1em](x|z = x \text{ or } x|v = x) \text{ and } (y|z = y \text{ or } y|v = y) \rightarrow F_{q-1}|E(B_q) = x \cup y\end{array}\right\} \tag{11}$$

As is seen from Eq. (11), the computation of the expression $F_{q-1}|E(B_q)$ may be reduced to the following:

$$\forall x \in F_{q-1}\text{: } x\&B_q \neq \{0\}^n \rightarrow x|E(B_q) = x, \tag{12}$$

where & is the bitwise conjunction (AND) operation sign. Eq. (12) states that if there is an implicant $x \in F_{q-1}$ such that $x \ \&\ B_q \neq 0$, then it will occur as a bit vector generated by the operation $x|E(B_q)$ and will absorb the other bit vectors generated by this operation. Therefore, each $x \in F_{q-1}$ satisfying the condition $x \ \&\ B_q \neq \{0\}^n$ should be considered as a part of the final result without summing it with $E(B_q)$. For storing such parts of the result, we will use set V_{q1}, obtained as follows:

$$V_{q1} = \{x \in F_{q-1} | x\&B_q \neq \{0\}^n\}. \tag{13}$$

The rest of set F_{q-1} is obtained as follows:

$$V_{q2} = F_{q-1} - V_{q1}. \tag{14}$$

That is, $E(B_q)$ is to be Cartesian-summed with only set V_{q2}.

$$T_q = V_{q2}|E(B_q) \tag{15}$$

The postdetectable RIs may occur in the process of computations in Eq. (15). Each of them may be recognized and ignored by the following rule:

$$T_q = T_q \cup \omega_{ji} \Leftrightarrow \overline{\exists} v \in V_{q1}\text{: } v|\omega_{ji} = v, \tag{16}$$

where $\omega_{ji} = v_j \mid e_i$ such that $v_j \in V_{q2}$, $e_i \in E(B_q)$, $j \in \{1,2,\ldots,|V_{q2}|\}$, and $i \in \{1,2,\ldots,|E(B_q)|\}$.

3.3. The algorithm of $DF_{\min}$-to-DNF conversion with relevant implicants

Remembering that the iterative $DF_{\min}$-to-DNF_{BB} conversion is realized by Eq. (9), prevention of predetectable RIs is realized by Eqs. (13) and (14), and the ignoring of postdetectable RIs is realized by Eqs. (15) and (16). In the example below, we give a procedure implementing these formulas.

Example 2. Convert to DNF the minimized discernibility function $DF_{PL} = (b \vee c \vee d) \wedge (d \vee e) \wedge (a \vee d) \wedge (a \vee b \vee c \vee e)$, the bit-based representation of which is to be generated on the following structure.

Struct. Example 2. { Unsig.a:1; Unsig.b:1; Unsig.c:1; Unsig.d:1; Unsig.e:1; }

That is, $B_1 = \varphi_1 : (b \vee c \vee d) \rightarrow 01110, ;_2 = \varphi_1 : (d \vee e) \rightarrow 00011; B_3 = \varphi_1 : (a \vee d) \rightarrow 10010$; and $B_4 = \varphi_1 : (a \vee b \vee c \vee e) \rightarrow 11101$, where φ_1 is an operator transforming a given PLC to the appropriate BBC on *Struct. Example 2.* Consequently, $DF_{\min} = \{01110, 00011, 10010, 11101\}$. Since $Q = 4$, according to Eq. (9), the DNF_{BB} will be computed in 4 iterations.

Iteration 1: $q = 1, B_1 = 01110$; $F_0 = \{\{0\}^5\} = \{00000\}$.

1.1. Intersect all $x \in F_0$ with B_1 : $00000 \& 01110 = \{0\}^5$.

1.2. Divide F_0 into 2 sets such that $V_{11} = \{x \in F_0 : x \& B_1 \neq \{0\}^5\}$ and $V_{12} = F_0 - V_{11}$:

$V_{11} = \emptyset; V_{12} = F_0 - V_{11} = \{00000\}$.

1.3. Generate, one by one, the elements of $T_1 = V_{12} \mid E(B_1)$. Compare each new generated element with the elements of set V_{11}. Ignore those absorbed by any element of set V_{11} : $E(B_1) = \{01000, 00100, 00010\}; T_1 = \{00000\} \mid \{01000, 00100, 00010\} = \{01000, 00100, 00010\}$. There is no element in T_1 absorbed by the elements of V_{11}.

1.4. Compute $F_1 = V_{11} \cup T_1 : = \{01000, 00100, 00010\}$.

Iteration 2: $q = 2, B_2 = 00011$; $F_1 = \{01000, 00100, 00010\}$.

2.1. Intersect all $x \in F_1$ with B_2: $01000 \& 00011 = \{0\}^5$; $00100 \& 00011 = \{0\}^5$; $00010 \& 00011 \neq \{0\}^5$.

2.2. Divide F_1 into 2 sets such that $V_{21} = \{x \in F_1 : x \& B_2 \neq \{0\}^5\}$ and $V_{22} = F_1 - V_{21}$:

$V_{21} = \{00010\}; V_{22} = F_1 - V_{21} = \{01000, 00100\}$.

2.3. Generate, one by one, the elements of $T_2 = V_{22} \mid E(B_2)$. Compare each new generated element with the elements of set V_{21}. Ignore those absorbed by any element of set V_{21} : $E(B_2) = \{00010, 00001\}; T_2 = \{01000, 00100\} \mid \{00010, 00001\} = \{$ ~~01010~~ $, 01001,$ ~~00110~~$, 00101\} = \{01001, 00101\}$. Here, 01010 and $00110 \in T_2$ are ignored as absorbed by $00010 \in V_{21}$.

2.4. Compute $F_2 = V_{21} \cup T_2 : F_2 = \{00010, 01001, 00101\}$.

Iteration 3: $q = 3, B_3 = 10010$; $F_2 = \{00010, 01001, 00101\}$.

3.1. Intersect all $x \in F_2$ with B_3: $00010 \& 10010 \neq \{0\}^5$; $01001 \& 10010 = \{0\}^5$; $00101 \& 10010 = \{0\}^5$.

3.2. Divide F_2 into 2 sets such that $V_{31} = \{x \in F_2 : x \& B_3 \neq \{0\}^5\}$ and $V_{32} = F_2 - V_{31}$:

$V_{31} = \{00010\}; V_{32} = F_2 - V_{31} = \{01001, 00101\}$.

3.3. Generate, one by one, the elements of $T_3 = V_{32} \mid E(B_3)$. Compare each new generated element with the elements of set V_{31}. Ignore those absorbed by any element of set V_{31} : $E(B_3) = \{10000, 00010\}; T_3 = \{01001, 00101\} \mid \{10000, 00010\} = \{11001,$ ~~01011~~$, 10101,$ ~~00111~~$\} = \{11001, 10101\}$. Here, 01011 and $00111 \in T_3$ are ignored as absorbed by $00010 \in V_{31}$.

3.4. Compute $F_3 = V_{31} \cup T_3 : F_3 = \{00010, 11001, 10101\}$.

Iteration 4: $q = 4, B_4 = 11101$; $F_3 = \{00010, 11001, 10101\}$.

4.1. Intersect all $x \in F_3$ with B_4: *00010 & 11101* $= \{0\}^5$; *11001 &* 11101 $\neq \{0\}^5$; *10101 &* 11101 $\neq \{0\}^5$.

4.2. Divide F_3 into 2 sets such that $V_{41} = \{x \in F_2 : x \,\&\, B_4 \neq \{0\}^5\}$ and $V_{42} = F_3 - V_{41}$:

$V_{41} = \{11001,\ 10101\}$; $V_{42} = F_3 - V_{41} = \{00010\}$.

4.3. Compute, one by one, the elements of $T_4 = V_{42} \mid E(B_4)$. Compare each new generated element with the elements of set V_{41}. Ignore those absorbed by any element of set V_{41} : $E(B_4) = \{10000,\ 01000,\ 00100,\ 00001\}$; $T_4 = \{00010\} \mid \{10000,\ 01000,\ 00100,\ 00001\} = \{10010,\ 01010,\ 00110,\ 00011\}$. There is no element in T_4 to be ignored as absorbed by the elements of set V_{41}.

4.4. Compute $F_4 = V_{41} \cup T_4$: $F_4 = \{11001,\ 10101,\ 10010,\ 01010,\ 00110,\ 00011\}$.

That is, $DNF_{BB} = F_4 = \{11001,\ 10101,\ 10010,\ 01010,\ 00110,\ 00011\}$.

Based on *Struct. Example 2*, set F_4 is to be transformed into the set of MSAs as follows:

$R = \varphi_2 : DNF_{BB} = \varphi_2 : \{11001,\ 10101,\ 10010,\ 01010,\ 00110,\ 00011\} \rightarrow \{\{a,b,e\},\ \{a,c,e\},\ \{a,d\},\ \{b,d\},\ \{c,d\},\ \{d,e\}\}$, where φ_2 is an operator transforming the bit-based implicants into MSAs.

The structured *Algorithm_2*, given below, is a more detailed and formalized form of the procedure by which Example 2 was solved.

```
Algorithm_2 ($DF_{\min}, Q, n$) //The algorithm converting $DF_{\min}$ to $DNF_{BB}$
Begin
   $F_0 = \{0\}^n$; $|F_0| = 1$
   For q = 1 to Q
    { Algorithm_3 ($F_{q-1}, B_q$)
       Algorithm_4 ($B_q, V_{q1}, V_{q2}$) // $V_{q1}$ and $V_{q2}$ are the sets generated by
                                            Algorithm_3 given below
      $F_q = V_{q1} \cup T_q$ }            // $T_q$ is a set generated by Algorithm_4 given below
   Return ($F_q$)
End
```

The subprocedure *Algorithm_3*, used in *Algorithm_2* and given below, generates the set of implicants V_{q1} while preventing the generation of predetectable RIs according to Eqs. (13) and (14).

```
Algorithm_3 ($F_{q-1}, B_q$) // The algorithm generating implicants while preventing predetectable RIs
Begin
   $V_{q1} = \emptyset$; $V_{q2} = \{0\}^n$
   For i = 1 to $|F_{q-1}|$
   { Select $x_i \in F_{q-1}$; $\lambda = x_i \,\&\, B_q$
      If $\lambda \neq \{0\}^n$ then $V_{q1} = V_{q1} \cup x_i$
      Else $V_{q2} = V_{q2} \cup x_i$ }
   Return ($V_{q1}, V_{q2}$)
End
```

The work of *Algorithm_3* has been demonstrated by the pairs of steps (1.1, 1.2), (2.1, 2.2), (3.1, 3.2), and (4.1, 4.2) in Example 2.

The subprocedure *Algorithm_4*, used in *Algorithm_2* and given below, generates implicants according to Eq. (15) and ignores the postdetectable RIs according to Eq. (16).

```
Algorithm_4 ($B_q, V_{q1}, V_{q2}$) // The algorithm generating implicants while ignoring postdetectable RIs
Begin
    $E(B_q) = \{Pr_j(B_q) \underline{|} d_j = 1, j = 1,2,\ldots, N\}$; $T_q = \emptyset$
    For j = 1 to $|V_{q2}|$
    { Select $v_j \in V_{q2}$
      For i = 1 to $|E(B_q)|$
    { Select $e_i \in E(B_q)$; $\omega_{ji} = v_j \mid e_i$
      $P = T_q$; $T_q = T_q \cup \omega_{ji}$
      For k = 1 to $|V_{q1}|$
    { Select $v_k \in V_{q1}$
      If $v_k \,\&\, \omega_{ji} = v_k$ then $T_q = P$; Break } } }
    Return ($T_q$)
End
```

The work of *Algorithm_4* has been demonstrated by steps 1.3, 2.3, 3.3, and 4.3 in Example 2.

4. Estimation of the efficiency of algorithm _2 (convert $\boldsymbol{DF}_{\min}$ to $\boldsymbol{DNF}_{BB}$)

4.1. The efficiency of Algorithm _2 at worst-case space complexity

As mentioned in Subsection 2.3, the $DF_{\min}$ with the worst-case space complexity is that composed from $\binom{n}{n/2}$ clauses, each of which contain exactly $n/2$ attributes. This is because n elements can compose the sets Z_1, $Z_2, \ldots, Z_{n/2}, \ldots, Z_n$ such that the cardinality of Z_i is $\binom{n}{i}$ and $max(\binom{n}{1}, \binom{n}{2}, \ldots, \binom{n}{n/2}, \ldots, \binom{n}{n}) = \binom{n}{n/2}$. This is to say, the set of maximal cardinality is $Z_{n/2}$, containing $\binom{n}{n/2}$ clauses. Therefore, the bit-based representation of such a $DF_{\min}$ has $D_{DF} = (n/2) \times \binom{n}{n/2}$ 1s. The number of 1s per attribute is obtained as follows:

$$D = D_{DF}/n = (n/2) \times \binom{n}{n/2}/n = \binom{n}{n/2}/2 = 0.5 \times \binom{n}{n/2}. \tag{17}$$

In other words, the average number of presences of each attribute $x \in$ C in such a $DF_{\min}$ is $D = 0.5 \times \binom{n}{n/2}$. According to the Boolean law $(x \vee F_i) \wedge (x \vee F_j) = F_iF_j$ [20], only the first appearance of x is to be included in the result of the expansion of the expression $(x \vee F_1) \wedge (x \vee F_2) \wedge \ldots \wedge (x \vee F_z) = x \vee F_1F_2 \ldots F_z$,. Therefore, subprocedure *Algorithm_3* of *Algorithm_2* omits all multiplications of implicants containing attribute x with all clauses containing this attribute. Due to this reduction, *Algorithm_2* generates only the irredundant implicants, the total number of which is $D = 0.5 \times \binom{n}{n/2} = O(10^{(0.3xn)-1})$ times less than the total number of all implicants (redundant and irredundant) generated by an algorithm based on *expand and eliminate* principle. The dependency of the order of D on the number of attributes is given in Table 2.

Table 2. The dependency of the efficiency of the algorithm on the number of attributes.

n	10	20	30	40	50	60	70	80	90	100	110	120
D	10^2	10^5	10^8	10^{11}	10^{14}	10^{17}	10^{20}	10^{23}	10^{26}	10^{29}	10^{32}	10^{35}

As is seen from this Table, with each ,n increased by 10, the efficiency of the algorithm increases approximately by a factor of 10^3.

4.2. The efficiency of the algorithm for the $DF_{\min}$-to-DNF conversion at a space complexity other than the worst case

Since the $DF_{\min}$ with the worst-case space complexity is only one of 2^{2^n} possible Boolean functions [25], the probability of the occurrence of such a $DF_{\min}$ is $1/2^{2^n}$, which for $n \geq 5$ becomes negligibly small. Therefore, it is necessary to obtain the probabilities of occurrences of clauses of sizes 1 to n separately. Unfortunately, we did not do this analytically. Hence, we estimated the efficiency of *Algorithm_2* based on the results of experiments performed on several datasets from the UCI repository [8]. In Table 2, the expected coefficients of reduction of the space complexities for several datasets are given. The coefficient D for every dataset was obtained in accordance with Eq. (17), as follows:

$$D = W(DF_{BB})/n,$$

where $W(DF_{BB})$ is the total number of 1s in the DF_{BB} (in all BBCs) of the dataset considered. Recall that m, n, and Q denote the number of objects in the given dataset, the number of attributes of this dataset, and the number of BBCs in the minimized form of the DF for that dataset, respectively. The values of D for several datasets from the UCI repository [8] are given in Table 3.

Table 3. The expected coefficients of the reduction of the space complexity for several datasets.

Dataset	m	n	Q	$W(DF_{BB})$	D
Shuttle	43,500	9	8	26	2.9
Zoo	101	17	14	64	3.8
Diabet	768	8	15	66	8.3
Austra	690	14	23	139	9.9
Mushroom	8124	22	30	198	9.0
Anneal	798	38	55	279	7.3
Heart	270	8	68	430	53.8
Lymn	148	18	154	725	40.3
Ionesphere	351	34	235	5690	167.3
Lung Cancer	32	56	331	8588	153.4
Spectf Heart	187	44	2096	80,726	1834.7
Sonar	208	60	2004	98,969	1649.5
Statlog (Landsat Satellite)	4435	36	13,473	340,955	9471.0

As is seen from Tables 2 and 3, the values for D in Table 3 are considerably less than those in Table 2. The reason for this is the fact that Table 3 includes not all theoretically possible BBCs, but only those that are contained in the $DF_{\min}$ for the corresponding dataset. For instance, in Table 2, for a dataset with $n = 60$, D takes a value of the order of 10^{17}, while dataset Sonar with $n = 60$, given in Table 3, takes a value of the order of 10^3. This is due to the fact that in Table 2, only $2004/\binom{60}{30} = 1.69 \times 10^{-14}$ parts of possible BBCs exist. Since $1.69 \times 10^{-14} \times 10^{17} = 1.69 \times 10^3 \times 10^3$, the value of D for dataset Sonar is in a good accordance with that obtained for a dataset with $n = 60$ (Table 2).

By using *Algorithm_2*, we processed many datasets from the UCI repository that could not be processed by other algorithms due to memory overflows. Some of these datasets are given in Table 4.

Table 4. Examples of datasets that could be processed only by *Algorithm_2*.

Dataset	Number of Attributes/ Instances/ Classes	Number of Minimal Subsets /Reducts/ Size of a Reduct	Memory Space Used (MB)	Time Elapsed (s)
Spectf Heart	44/187/56	26,454/1/2	2.7	21.120
Dna	57/106/2	6,259,767/1/3	265	492,726
Statlog (Landsat Satellite)	36/4435/7	1,088,611/1145/5	295	16,320
Lung Cancer	56/32/3	9,007,859/6/4	500	1,044,210
Annealing	38/798/6	275/6/6	2.8	2.4
Ionosphere	33/351/2	4257/6/2	3.2	1.6
Sonar	60/208/2	31,844/168/2	1	27.6

The computations of the data given in Table 4 were performed by a computer with an Intel Core2Quad@2.83 GHz processor, 4 GB of memory, and Microsoft XP Professional Edition OS.

5. Conclusion

The problem of finding all MSAs for datasets of information systems is known to be NP-hard. This is due to the intractable space complexity of the DF-to-DNF conversion used as the main transformation in the attribute reduction. The analysis of the DF-to-DNF conversion showed that the mentioned complexity is mainly caused by redundant implicants occurring in this process. Therefore, we developed an algorithm preventing the generation of redundant implicants. We showed that by using this algorithm, the computational complexity of the DF-to-DNF conversion may be reduced thousands and even millions of times, allowing us to process many datasets that cannot be processed by other methods. However, it may also be unsuccessful for some large-sized datasets. We will further attempt to apply this method in conjunction with partitioning the DF to be converted to the DNF.

Acknowledgement

This work was supported by the Selçuk University Scientific Research Projects Coordinatorship, Konya, Turkey.

References

[1] J.A. Starzyk, D.E. Nelson, K. Sturtz, “A mathematical foundation for improved reduct generation in information systems”, Journal of Knowledge and Information Systems, Vol. 2, pp. 131-146, 2000.

[2] J. Komorowski, L. Polkowski, A. Skowron, “Rough set: a tutorial”, http://folli.loria.fr/cds/1999/library/pdf/skowron.pdf, 112 pages, 1999.

[3] R. Jensen, Q. Shen, "Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches", IEEE Trans. on Knowledge and Data Engineering, Vol. 16, pp. 1457-1471, 2004.

[4] Y. Yao, Y. Zhao, "Definability matrix simplification for constructing attribute reducts", Information Sciences, Vol. 179, pp. 867-882, 2009.

[5] X. Wang, Y. Jie, X. Teng, W. Xia, R. Jensen, "Feature selection based on rough sets and particle swarm optimization", Pattern Recognition Letters, Vol. 28, pp. 459-471, 2007.

[6] W.Q. Shang, H.K. Huang, H.B. Zhu et al., "A novel feature selection algorithm for text categorization", Expert Systems with Applications, Vol. 33, pp. 1-5, 2007.

[7] Y. Matsumoto, J. Watada, "Knowledge acquisition from time series data through rough sets analysis", International Journal of Innovative Computing, Information and Control, Vol. 5, pp. 4885-4897, 2009.

[8] http://archive.ics.uci.edu/ml/, last access date: 7 May 2010.

[9] A. Skowron, C. Rauszer, "The discernibility matrices and functions in information systems", Fundamenta Informaticae, Vol. 15, pp. 331-362, 1991.

[10] A. Skowron, "The rough sets theory and evidence theory", Fundamenta Informaticae, Vol. 13, pp. 245-262, 1990.

[11] A. Øhrn, J. Komorowski, A. Skowron, R. Synak, "The design and implementation of a knowledge discovery toolkit based on rough sets: the Rosetta system", in Rough Sets in Knowledge Discovery I: Methodology and Applications, L. Polkowski and A. Skowron, Eds., Heidelberg, Physica-Verlag, pp. 376-399, 1999.

[12] R. Jensen, Q. Shen, "Rough set-based feature selection: a review", http://cadair.aber.ac.uk/dspace/handle/2160/490, 52 pages, 2007.

[13] J. Wang, J. Wang, "Reduction algorithms based on discernibility matrix: the ordered attributes method", Journal of Computer Science & Technology, Vol. 16, pp. 489-504, 2001.

[14] W. Chen, S. Tseng, T. Hong, "An efficient bit-based feature selection method", Expert Systems with Applications, Vol. 34, pp. 2858-2869, 2008.

[15] S. Tan, X. Cheng, H. Xu, "An efficient global optimization approach for rough set based dimensionality reduction", International Journal of Innovative Computing, Information and Control, Vol. 3, pp. 725-736, 2007.

[16] J.G. Bazan, "A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables", in Rough Sets in Knowledge Discovery I: Methodology and Applications, L. Polkowski and A. Skowron, Eds., Heidelberg, Physica-Verlag, pp. 321-365, 1994.

[17] J.G. Bazan, A. Skowron, 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] S. Vinterbo, A. Øhrn, "Minimal approximate hitting sets and rule templates", International Journal of Approximate Reasoning, Vol. 25, pp. 123-143, 2000.

[19] http://logic.mimuw.edu.pl/∼rses/, last access date: 13 March 2010.

[20] R.K. Brayton, G.D. Hachtel, C.T. McMullen, A. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Boston, Kluwer Academic Publishers, 1984.

[21] A.A. Malik, R.K. Brayton, A.R. Newton, A. Sangiovanni-Vincentelli, “Reduced offsets for minimization of binary-valued functions”, IEEE Transactions on Computer-Aided Design, Vol. 10, pp. 413-426, 1991.

[22] Y. Korshunov, The Mathematical Basics of the Cybernetics, Moscow, Energiya Publishers, 1980 (in Russian).

[23] D.M. Giovanni, Synthesis and Optimization of Digital Circuits, New York, McGraw-Hill, 1994.

[24] R. Johnsonbaugh, M. Schaefer, Algorithms, Pearson Education Inc., 2004.

[25] V.A. Gorbatov, The Basics of Discrete Mathematics, Moscow, Visshaya Shkola Publishers, 1986 (in Russian).