

DFA-Based Incremental Algorithm for Simplifying the Binary Discernibility Matrix

IVAN PIZA-DAVILA¹, GUILLERMO SANCHEZ-DIAZ^{2,*}, MANUEL S. LAZO-CORTES³, AND
NELVA N. ALMANZA-ORTEGA⁴

¹Tlaquepaque, Jalisco, 45604, Department of Electronics, Systems, and Informatics, Instituto Tecnológico y de Estudios Superiores de Occidente, Mexico

²San Luis Potosí, SLP, 78290, Faculty of Engineering, Universidad Autónoma de San Luis Potosí, Mexico

³Tlalnepantla de Baz, Edo. Mex. 54070, Division of Graduate Studies and Research, Tecnológico Nacional de México/IT Tlalnepantla, Mexico

⁴Ciudad de México, 03940, Support services directorate for the consolidation of the scientific and humanistic community, Investigadoras e Investigadores por México/SECIHTI, Mexico

Abstract

Feature selection methods in Rough Set Theory and Logical Combinatorial Pattern Recognition commonly rely on simplified binary discernibility matrices to represent the condition attributes that distinguish objects from different decision classes. However, the original discernibility matrix involves a large number of redundant object comparisons, which severely limits computational efficiency and scalability. Although several algorithms have been proposed to compute the simplified matrix, most of them first construct the full discernibility matrix and then simplify it, making these approaches impractical for large-scale datasets. In this paper, we introduce ISBMA, an incremental algorithm based on a deterministic finite automaton that directly constructs the simplified binary discernibility and discernibility/similarity matrices without explicitly generating the full matrix. Experimental comparisons with the fastest reported method demonstrate that ISBMA significantly reduces redundant computations, requires substantially less memory, and achieves superior runtime performance, remaining feasible on large decision tables where existing approaches fail.

Keywords *feature selection; incremental computation; large-scale data; pairwise comparison matrix; rough set theory; testor theory*

1 Introduction

Feature selection is the process of identifying a reduced subset of relevant attributes for classification tasks (Webb, 2002). In pattern recognition and machine learning (Webb, 2002; Hang, 2024), it is a staple of practice as it contributes not only to the improvement in model interpretability but also to the reduction in computational cost and, in many scenarios, the improvement of predictive performance (Gao et al., 2024). The proposed methods for feature selection based on the concepts have been grounded on several theoretical frameworks, namely Rough Set Theory (RST) (Pawlak, 1982) and Logical Combinatorial Pattern Recognition (LCPR) (Ruiz-Shulcloper

*Corresponding author. Email: guillermo.sanchez@uaslp.mx.

and Abidi, 2002). A broad selection of algorithms has been suggested in this context (Yao and Zhao, 2008; Ni et al., 2020; Sanchez-Diaz et al., 2022; Lefebvre-Lobaina and Ruiz-Shulcloper, 2023), indicating importance and progress on this subject since then (Wang et al., 2019; Zhou et al., 2019). Rather than operating directly on decision systems or original datasets, many feature selection algorithms rely on a binary discernibility matrix, which encodes attribute-wise differences between objects and significantly reduces the number of comparisons required during the selection process (Felix and Ushio, 1999; Lazo-Cortes et al., 2001). In both RST and LCPR, this matrix—or its simplified version—is a fundamental computational structure underlying numerous selection techniques. The simplified binary discernibility matrix, also referred to as the basic matrix (Yao and Zhao, 2009; Lazo-Cortes et al., 2001; Xu et al., 2010), eliminates redundant object comparisons and thereby accelerates the execution of feature selection procedures.

Within RST (Pawlak, 1982, 1992), feature selection is commonly formulated in terms of reducts and constructs, which capture different preservation criteria over the original decision table. A reduct focuses on maintaining the ability to distinguish objects belonging to different decision classes, whereas a construct extends this notion by additionally preserving selected similarity relations among objects within the same class (Susmaga, 2014; Susmaga and Slowinski, 2015; Gonzalez-Diaz et al., 2024). In LCPR, irreducible testors serve an analogous role by identifying minimal subsets of attributes that retain discriminative power (Lazo-Cortes et al., 2001; Sanchez-Diaz et al., 2022). All these approaches benefit from the use of the simplified binary discernibility matrix or basic matrix as a compact and informative representation of attribute relevance.

Discernibility matrices encode object differences on an attribute-by-attribute basis, and the amount of distinguishable information they contain has been shown to be closely related to the importance of the corresponding attributes (Liang et al., 2022). Despite their usefulness, relatively few efforts have focused explicitly on simplifying the binary discernibility matrix itself (Wei et al., 2019). In LCPR, the basic matrix is obtained from the dissimilarity matrix through specific reduction procedures (Lazo-Cortes et al., 2001), whereas in RST, simplification methods based on algebraic and combinatorial properties of the discernibility matrix were proposed in (Zhi and Miao, 2002) and later improved in (Wang and Ma, 2006). The most recent contribution in this line of research is the algorithm introduced in (Zhang et al., 2018), which employs deterministic finite automata (DFA) to compute the simplified binary discernibility matrix more efficiently than earlier methods. This approach represents a significant advance in terms of computational efficiency and has been shown to outperform previous algorithms based on matrix simplification rules.

Nevertheless, the construction of the binary discernibility matrix itself becomes infeasible when applied to large-scale datasets (Li et al., 2023). As the number of objects increases, memory overflow may occur (Liang et al., 2022), and the conversion of the full discernibility matrix into conjunctive or disjunctive normal forms becomes computationally prohibitive (Zhang et al., 2019). These limitations severely restrict the applicability of existing methods in modern data-intensive scenarios. Motivated by these scalability challenges, this paper proposes an incremental version of the deterministic finite automaton introduced in (Zhang et al., 2018). Unlike the original approach, the proposed algorithm constructs the simplified binary discernibility matrix directly, without generating or storing the full discernibility matrix, thereby reducing both memory usage and redundant computations.

The remainder of this paper is organized as follows. Section 2 introduces the fundamental concepts of reducts and constructs. Section 3 presents the proposed incremental algorithm.

Section 4 describes the experimental setup and benchmarking methodology. Section 5 discusses the experimental results, and Section 6 concludes the paper.

2 Preliminaries

Within the Rough Set Theory (RST) framework, data is typically represented using a decision table (Pawlak and Skowron, 2007). Let S_d denote a decision table consisting of a non-empty set of objects U , called the universe, and a set of attributes A . We assume that $A = A^* \cup \{d\}$, where A^* denotes the set of condition attributes used to describe the objects in the universe, and d is the decision attribute, which assigns each object to a specific decision class. This work focuses exclusively on consistent decision tables, meaning that the set A^* is sufficient to distinguish between objects belonging to different decision classes.

Suppose we have a decision table S_d and a non-empty subset $B \subseteq A^*$. The supervised indiscernibility relation, denoted by $IND(B, d)$, pairs elements (x, y) from the universe U such that, for every attribute $a \in B$, either the values of a on x and y coincide, or the corresponding decision values $d(x)$ and $d(y)$ are the same. Thus, two objects are considered supervised indiscernible if they cannot be distinguished using the attributes in B unless their decision values differ. This relation is formally defined as:

$$IND(B, d) = \{(x, y) \in U \times U : \forall a \in B [a(x) = a(y)] \vee [d(x) = d(y)]\}. \quad (1)$$

A subset of attributes R is called a reduct if it satisfies two conditions. First, the supervised indiscernibility relation induced by R must coincide with that defined by the full set of condition attributes A^* , meaning that R preserves the same discriminative power as A^* . Second, R must be minimal: removing any attribute $a \in R$ causes this equivalence to be lost. In this sense, every attribute in R is essential. Formally, these conditions can be expressed as:

$$IND(R, d) = IND(A^*, d). \quad (2)$$

$$\forall a \in R, IND((R - \{a\}), d) \neq IND(A^*, d). \quad (3)$$

A reduct retains the same discriminative power as the full set of condition attributes for distinguishing between decision classes, and none of its attributes is redundant. The notion of reduct adopted in this work is based on the comparison of objects from different decision classes, following the supervised indiscernibility relation introduced in (Pawlak, 1992).

On the other hand, the similarity relation $SIM(B, d)$ relates pairs of elements (x, y) from the universe U that share the same decision value, $d(x) = d(y)$, and coincide in at least one attribute $a \in B$, that is, $a(x) = a(y)$. This relation is defined as:

$$SIM(B, d) = \{(x, y) \in U \times U : [d(x) = d(y)] \wedge \exists a \in B [a(x) = a(y)]\}. \quad (4)$$

This relation indicates that if a pair of objects belongs to $SIM(B, d)$, then both objects are in the same decision class and are indistinguishable with respect to at least one attribute in the set B (Susmaga, 2014). In the same context, a subset of attributes C is called a construct if it satisfies three conditions (Susmaga and Slowinski, 2015). First, the supervised indiscernibility relation induced by C coincides with that of the full set A^* . Second, the similarity relation induced by C also coincides with that of A^* . Finally, C is minimal, in the sense that removing

any attribute $a \in C$ causes at least one of these two properties to fail. These conditions can be summarized as follows:

$$IND(C, d) = IND(A^*, d). \quad (5)$$

$$SIM(C, d) = SIM(A^*, d). \quad (6)$$

$$\forall a \in C, IND((C - \{a\}), d) \neq IND(A^*, d) \vee SIM((C - \{a\}), d) \neq SIM(A^*, d). \quad (7)$$

Constructs are free of redundant attributes. Consequently, both constructs and reducts consist of attributes that are jointly sufficient and individually necessary (Lazo-Cortes et al., 2024).

2.1 Binary Discernibility (Discernibility/Similarity) Matrix

A discernibility matrix, as introduced in (Skowron and Rauszer, 1992), serves as a matrix-based representation that stores sets of condition attributes capable of distinguishing objects in the universe. The binary discernibility matrix (Felix and Ushio, 1999) is a binary presentation of the original discernibility matrix, which is commonly used for reduct construction.

Given a decision table S_d , let $M = |A^*|$, and let $A^* = \{a_1, a_2, \dots, a_M\}$ be the set of condition attributes. For a pair of objects $o_i, o_j \in U$, we define the attribute-wise discernibility value associated with attribute a_k as

$$\delta_k(i, j) = \begin{cases} 1, & \text{if } a_k(o_i) \neq a_k(o_j). \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

The binary comparison row associated with the pair (o_i, o_j) is then defined as

$$r_{i,j} = (\delta_1(i, j), \delta_2(i, j), \dots, \delta_M(i, j)). \quad (9)$$

Thus, $r_{i,j}[k] = \delta_k(i, j)$ represents whether objects o_i and o_j differ in the condition attribute a_k . The binary discernibility matrix M_{BDM} is formed by the rows $r_{i,j}$ obtained from pairs of objects belonging to different decision classes, that is, from pairs satisfying $d_i \neq d_j$. Therefore,

$$M_{BDM}(i, j) = r_{i,j}, \quad \text{for } d_i \neq d_j. \quad (10)$$

The maximum number of possible pairwise comparison rows is given by $R_M = |U|(|U| - 1)/2$, where $|U|$ is the number of objects in the decision table. For the binary discernibility matrix M_{BDM} , only interclass comparisons are included, so the actual number of rows may be smaller depending on the distribution of objects among decision classes. These matrices perform comparisons exclusively between objects that belong to different decision classes. A value of 0 indicates equality of the corresponding attribute values, while a value of 1 denotes inequality.

In contrast, for constructs, the associated binary matrix is derived from the full discernibility table. This type of matrix, introduced in (Lazo-Cortes et al., 2015), includes comparisons between all pairs of objects, regardless of whether they belong to the same or different classes, that is, both interclass and intraclass comparisons.

The matrix CM_{BDM} represents the binary discernibility/similarity matrix associated with S_d . The row associated with a pair of objects $o_i, o_j \in U$ is defined componentwise as

$$CM_{BDM}(i, j)[k] = \begin{cases} \delta_k(i, j), & \text{if } d_i \neq d_j. \\ 1 - \delta_k(i, j), & \text{if } d_i = d_j. \end{cases} \quad k = 1, 2, \dots, M. \quad (11)$$

Equivalently, if $\mathbf{1}$ denotes the all-one binary row of length M , then

$$CM_{BDM}(i, j) = \begin{cases} r_{i,j}, & \text{if } d_i \neq d_j. \\ \mathbf{1} - r_{i,j}, & \text{if } d_i = d_j. \end{cases} \quad (12)$$

Binary discernibility (and discernibility/similarity) matrices typically contain redundant information. Consequently, a simplified matrix obtained by applying absorption laws is used instead of the full binary matrix (Zhi and Miao, 2002; Wang and Ma, 2006; Lazo-Cortes et al., 2015). In Testor Theory, this reduced representation is known as the basic matrix (Lazo-Cortes et al., 2001). Hereafter, the term simplified binary matrix refers to either the simplified discernibility or discernibility/similarity matrix. Let MDS_{BDM} denote either a binary discernibility matrix or a binary discernibility/similarity matrix, and let r_p denote the p -th row of MDS_{BDM} . For two rows r_p and r_q , we write $r_q \leq r_p$ if $r_q[k] \leq r_p[k]$, for every k .

Moreover, we write $r_q < r_p$ if $r_q \leq r_p$ and $r_q \neq r_p$, that is:

$$r_q < r_p \iff (r_q[k] \leq r_p[k], \text{ for every } k) \text{ and } (r_q[j] < r_p[j], \text{ for at least one } j). \quad (13)$$

A nonzero row r_p of MDS_{BDM} is called non-essential if there exists a nonzero row r_q of MDS_{BDM} such that $r_q < r_p$ (Rodriguez-Diez et al., 2020). In this case, r_p is said to be absorbed by r_q .

Definition 1. The simplified binary matrix $SMDS_{BDM}$ of MDS_{BDM} is the matrix formed by the nonzero rows of MDS_{BDM} that are not non-essential. Repeated rows are represented only once. Therefore, $SMDS_{BDM}$ can be regarded as a set of rows, and the order of its rows is irrelevant.

Thus, the simplified binary matrix is obtained by removing zero rows, repeated rows, and non-essential rows from the binary discernibility matrix (Yao and Zhao, 2009) or from the binary discernibility/similarity matrix (Lazo-Cortes et al., 2015). Several algorithms have been proposed to compute the simplified binary matrix (Zhi and Miao, 2002; Wang and Ma, 2006; Lazo-Cortes et al., 2001). The most efficient algorithm reported to date for this task is based on a deterministic finite automaton (Zhang et al., 2018). One of the main drawbacks of these algorithms is that they require as input the full binary discernibility (or discernibility/similarity) matrix, which must be fully computed and stored in advance. However, modern information systems often involve large volumes of data and frequent updates, making the construction and storage of such matrices infeasible in practice. The process requires considerable memory resources (Liang et al., 2022), and in many cases, the matrix does not fit in main memory (Zhang et al., 2019).

The space complexity of constructing the binary comparison matrix is $O(MN^2)$, where M is the number of features and N is the number of objects in the universe. For datasets containing tens of thousands of objects or more, the storage of the full matrix will lead to memory allocation failures (or out-of-memory errors). In this work, we adopt the deterministic finite automaton introduced in (Zhang et al., 2018) as the basis for the incremental construction of the simplified binary matrix.

2.2 Deterministic Finite Automaton

Deterministic finite automata are a fundamental concept in the theory of computation and provide a convenient formalism for modeling finite-state processes. In particular, a deterministic finite automaton (DFA) defines a finite-state machine that processes an input string symbol by symbol and decides whether to accept or reject it by reaching either an accepting state or a

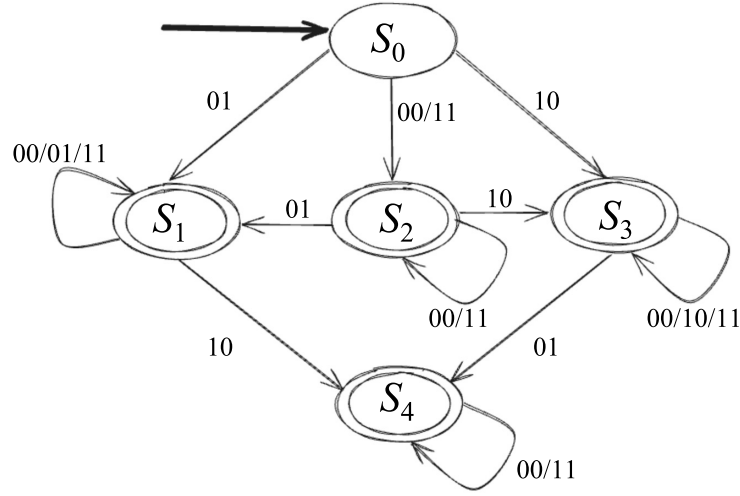


Figure 1: Deterministic finite automaton for row relations.

non-accepting state (Sipser, 2013). A DFA is described by a tuple $(Q, \Sigma, \delta, S_0, F)$, where Q is a finite and non-empty set of states, Σ is a finite set of input symbols (the input alphabet), δ is the transition function defining how the automaton moves between states upon reading input symbols, $S_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting (or final) states (Zhang et al., 2018). The symbols e_p^m and e_q^m are used as input symbols for the deterministic finite automaton to represent relations between rows in a binary discernibility matrix M_{BDM} , where p and q index the rows r_p and r_q of M_{BDM} , and m denotes the column index.

A deterministic finite automaton for row relations in a binary discernibility matrix, denoted by DFA_{row} , is defined as a tuple $(Q, \Sigma, \delta, S_0, F)$, where: $Q = \{S_0, S_1, S_2, S_3, S_4\}$ is a finite set of states; Σ is the input alphabet consisting of binary symbols of the form e_p^m and e_q^m , where $1 \leq m \leq |A^*|$ and $1 \leq p, q \leq R_M$; δ is the transition function; $S_0 \in Q$ is the initial state and $F = \{S_1, S_2, S_3, S_4\}$ is the set of accepting states. An illustration of this automaton is shown in Figure 1.

The deterministic finite automaton described in Figure 1 can be applied, without modification, to determine relationships between rows of a binary discernibility/similarity matrix as well.

Considering any two distinct rows r_p and r_q of M_{BDM} or CM_{BDM} , the automaton shown in Figure 1 operates according to the following states:

- S_0 is the initial state.
- S_1 indicates that row r_q is non-essential, since $r_p < r_q$.
- S_2 indicates that rows r_p and r_q are equal, that is, $r_p = r_q$.
- S_3 indicates that row r_p is non-essential, since $r_q < r_p$.
- S_4 indicates that rows r_p and r_q are neither equal nor related by absorption.

In simple terms, the deterministic finite automaton is used to control the comparison between two binary rows generated from object pairs. It detects when equality or absorption relations occur, allowing the algorithm to stop the comparison without performing unnecessary additional computations. Furthermore, states S_1 and S_3 indicate to the algorithm that one of the compared rows is non-essential. State S_2 indicates that the compared rows are equal.

Algorithm 1 The proposed ISBMA algorithm based on a deterministic finite automaton.

Require: A row $r_{i,j}$ (resulting from comparing $o_i, o_j \in U$), a list RL of currently essential rows

Ensure: A list of updated essential rows

```

1: if ( $isZeroRow(r_{i,j}) == True$ ) then
2:   return  $RL$ 
3: else
4:    $r_p \leftarrow First(RL)$ 
5:    $foundSubrow \leftarrow False$ 
6:   while ( $r_p \neq null$ ) and ( $foundSubrow == False$ ) do
7:      $state \leftarrow DFA(r_{i,j}, r_p)$ 
8:     if ( $state == S_1$ ) or ( $state == S_2$ ) then
9:        $r_p \leftarrow Erase(RL, r_p)$ 
10:    else
11:      if ( $state == S_3$ ) then
12:         $foundSubrow \leftarrow True$ 
13:      end if
14:       $r_p \leftarrow Next(RL, r_p)$ 
15:    end if
16:  end while
17:  if ( $foundSubrow == False$ ) then
18:     $Append(RL, r_{i,j})$ 
19:  end if
20:  return  $RL$ 
21: end if

```

3 Proposed Algorithm

The proposed algorithm, called ISBMA (Incremental Simplified Binary Matrix Algorithm), is shown in Algorithm 1. It updates the current list of essential rows each time a new binary row $r_{i,j}$ is generated from a pair of objects $o_i, o_j \in U$, with $i < j$. Algorithm 2 embeds this update in the complete pairwise traversal of the decision table. If the generated row $r_{i,j}$ is not already represented in the row list and is not non-essential with respect to a row in the list, it is added to the list; otherwise, it is discarded. Conversely, if a stored row r_k becomes non-essential after comparison with $r_{i,j}$, then r_k is removed from the list.

Function $isZeroRow(r_{i,j})$ returns true if the input row $r_{i,j}$ contains only zeros, and function $First(RL)$ returns the first row stored in RL . The function $DFA(r_{i,j}, r_p)$ returns the state reached by the deterministic finite automaton in Figure 1 after comparing the newly generated row $r_{i,j}$ with the stored row r_p , in that order. The methods $Erase(RL, r_p)$ and $Append(RL, r_{i,j})$, together with the function $Next(RL, r_p)$, are constant-time linked-list operations: $Erase(RL, r_p)$ deletes r_p and returns the next row (or null if r_p was the last row), $Append(RL, r_{i,j})$ inserts a new node containing $r_{i,j}$ at the end of the list, and $Next(RL, r_p)$ returns the row following r_p (or null if none exists).

To compare a generated row $r_{i,j}$ with the rows currently stored in RL , ISBMA uses the deterministic finite automaton in Figure 1. If a nonzero row $r_{i,j}$ is processed when RL is empty, it is added to the list because $foundSubrow$ remains *false*. If $r_{i,j}$ is the second row, it is compared with the first one; if either row is non-essential with respect to the other, the non-essential row is

Algorithm 2 General algorithm for processing the objects of a decision table.

Require: A decision table S_d and comparison type CT (interclass or combined interclass/intraclass).

Ensure: A simplified binary discernibility matrix or discernibility/similarity matrix SDM .

```

1:  $RL \leftarrow EmptyList()$ 
2:  $i \leftarrow 1$ 
3: while ( $i < |U|$ ) do
4:    $j \leftarrow i + 1$ 
5:   while ( $j \leq |U|$ ) do
6:      $k \leftarrow 1$ 
7:     while ( $k \leq |A^*|$ ) do
8:       if ( $CT == inter$ ) and ( $d_i == d_j$ ) then
9:          $r_{i,j}[k] \leftarrow 0$ 
10:      else if ( $CT == inter$ ) then
11:         $r_{i,j}[k] \leftarrow M_{BDM}(i, j)[k], a_k \in A^*$ 
12:      else
13:         $r_{i,j}[k] \leftarrow CM_{BDM}(i, j)[k], a_k \in A^*$ 
14:      end if
15:       $k \leftarrow k + 1$ 
16:    end while
17:     $RL \leftarrow ISBMA(r_{i,j}, RL)$ 
18:     $j \leftarrow j + 1$ 
19:  end while
20:   $i \leftarrow i + 1$ 
21: end while
22:  $SDM \leftarrow List\_to\_Matrix(RL)$ 
23: return  $SDM$ 

```

removed. Otherwise, RL contains two rows that are essential relative to each other. Thus, each newly generated row is compared only with the rows currently in RL , although some stored rows may become non-essential after the new row is introduced.

At the end of each comparison process, one of the following situations occurs:

1. One or more rows in RL are non-essential with respect to $r_{i,j}$ and are removed from the list.
2. The row $r_{i,j}$ is non-essential with respect to an existing row and is therefore discarded.
3. Otherwise, $r_{i,j}$ is added to the list RL .

This incremental construction of the simplified binary matrix avoids unnecessary comparisons with rows that, a priori, are known not to be part of the final matrix. For the interclass comparison type, pairs satisfying $d_i = d_j$ generate the zero row and are therefore discarded by *isZeroRow*.

In the general algorithm, $ISBMA(r_{i,j}, RL)$ returns the updated row list after processing the row generated by comparing objects $o_i, o_j \in U$, $List_to_Matrix(RL)$ converts the final row list into a simplified binary discernibility or discernibility/similarity matrix, and $EmptyList()$ returns an empty row list. The following proposition establishes the correctness of the proposed algorithm by showing that the row list maintained by $ISBMA$ contains exactly the rows of the simplified binary matrix associated with the rows generated so far.

Proposition 1. *Let MDS_{BDM} be either the binary discernibility matrix or the binary discernibility/similarity matrix associated with a decision table, and let $SMDS_{BDM}$ be its simplified binary matrix. If all rows of MDS_{BDM} are generated by the pairwise comparisons of objects in the decision table, then the matrix obtained by applying *List_to_Matrix* to the final row list returned by ISBMA is exactly $SMDS_{BDM}$.*

Proof. Let $r_1, r_2, \dots, r_T$ be an arbitrary ordering of the rows generated from the pairwise comparisons of objects in the decision table. Let $R_t = \{r_1, r_2, \dots, r_t\}$ denote the set of rows generated after the first t comparisons, and let RL_t be the row list maintained by ISBMA after processing these t rows. We prove, by induction on t , that RL_t contains exactly the nonzero rows of R_t that are not non-essential, with repeated rows represented only once. In other words, RL_t is the simplified binary matrix associated with the rows generated up to step t .

For $t = 0$, no row has been processed and the list RL_0 is empty. Therefore, the claim holds trivially.

Assume now that the claim holds after processing the first $t - 1$ rows. We prove that it also holds after processing the new row r_t . By the induction hypothesis, RL_{t-1} contains exactly the nonzero rows of R_{t-1} that are not non-essential, with no repeated rows.

If r_t is a zero row, then it cannot belong to the simplified binary matrix by definition. The algorithm discards it, and therefore $RL_t = RL_{t-1}$, which remains the simplified binary matrix of R_t .

Suppose now that r_t is not a zero row. The algorithm compares r_t with each row r_p currently stored in RL_{t-1} by using the deterministic finite automaton. There are four relevant cases.

If the automaton determines that $r_t < r_p$, then r_p becomes non-essential with respect to the enlarged set R_t , because it is absorbed by the newly generated row r_t . Consequently, r_p must not belong to the simplified binary matrix of R_t , and the algorithm correctly removes it from RL .

If the automaton determines that $r_t = r_p$, then both rows represent the same information. Since repeated rows are represented only once in the simplified binary matrix, keeping only one copy is sufficient. The algorithm removes the stored copy and, if no smaller row absorbing r_t is found later, inserts r_t at the end. Thus, the resulting set of rows remains unchanged with respect to duplicates.

If the automaton determines that $r_p < r_t$, then the new row r_t is non-essential, because it is absorbed by a row already present in RL_{t-1} . In this case, the algorithm sets *foundSubrow* to true and does not append r_t to the list. Therefore, no non-essential row is added.

Finally, if the automaton determines that r_t and r_p are neither equal nor related by absorption, then neither row makes the other non-essential. Hence, no modification is required for this pair.

After the comparison loop terminates, two situations are possible. If *foundSubrow* is true, then there exists a row $r_p \in RL_{t-1}$ such that $r_p < r_t$. Hence, r_t is non-essential and is not added to the list. Moreover, no unprocessed row in RL_{t-1} needs to be removed: if there were a row $r_s \in RL_{t-1}$ such that $r_t < r_s$, then, by transitivity, $r_p < r_s$, which would make r_s non-essential in RL_{t-1} , contradicting the induction hypothesis. If *foundSubrow* remains false, then r_t has been compared with all rows in RL_{t-1} . In this case, every row in RL_{t-1} that is absorbed by r_t has been removed, and r_t itself is appended to the list. Therefore, RL_t contains exactly the nonzero rows of R_t that are not non-essential, with repeated rows represented only once.

By induction, the claim holds for every $t = 0, 1, \dots, T$. In particular, after all rows of MDS_{BDM} have been generated and processed, the final list RL_T contains exactly the rows of

$SMDS_{BDM}$. Since $List_to_Matrix(RL_T)$ only converts this list into matrix form, the matrix obtained from the final row list returned by ISBMA is exactly $SMDS_{BDM}$. Moreover, because the simplified binary matrix is defined as a set of essential rows, with repeated rows represented only once, the final result is independent of the order in which the rows are processed. $\square$

Complexity analysis

As mentioned in Section 2.1, to obtain the simplified binary matrix, the *DFA*-based algorithm first constructs the full binary matrix. This matrix contains $O(N^2)$ binary rows, where N is the number of objects, and M is the number of features. Therefore, the construction and storage of the full binary matrix require $O(MN^2)$ time and $O(MN^2)$ space, respectively.

After the full matrix has been constructed, the simplification step removes zero rows, repeated rows, and rows for which a sub-row exists. This requires comparing rows of length M . In the best case, each row can be discarded after a constant number of row comparisons, and the simplification step requires $O(MN^2)$ time. In the worst case, no row is absorbed by another row, and each row must be compared with all the remaining rows. Since the full matrix contains $O(N^2)$ rows, this leads to $O(N^4)$ row comparisons, each one requiring $O(M)$ time. Thus, the worst-case time complexity of this step is $O(MN^4)$. Consequently, the overall time complexity of the *DFA*-based construct-then-simplify procedure ranges from $O(MN^2)$ to $O(MN^4)$, while its space complexity is always $O(MN^2)$.

The proposed algorithm avoids explicitly constructing and storing the full binary matrix. Instead, each newly generated row is compared only with the rows currently stored in the simplified row list RL . The new row is added to RL only if it is not absorbed by a row already stored in the list, and it may also remove previously stored rows that are absorbed by the new row.

Let R denote the maximum number of rows stored in RL during the execution of the algorithm. Since $O(N^2)$ rows are generated and each row may be compared with at most R stored rows, the time complexity of *ISBMA* is $O(MN^2R)$. Its space complexity is $O(MR)$, because at most R binary rows of length M are stored at any time. In the best case, $R = O(1)$, and the algorithm requires $O(MN^2)$ time and $O(M)$ space. In the worst case, $R = O(N^2)$, and the algorithm requires $O(MN^4)$ time and $O(MN^2)$ space.

Therefore, the main advantage of ISBMA is that its time and space requirements depend on the number of rows that remain in the simplified row list during execution, rather than on the full binary matrix. When R is much smaller than N^2 , as observed in the experimental results, the proposed algorithm substantially reduces memory usage and improves running time. The next section compares both algorithms in terms of running time and memory consumption.

4 Experimental Results

In this section, we evaluate the computational performance and scalability of the *DFA*-based algorithm (Zhang et al., 2018) and the proposed ISBMA algorithm using two groups of real decision tables obtained from UCI datasets (Kelly et al., Accessed June 10, 2025). The first group consists of benchmark decision tables originally used to evaluate the *DFA*-based algorithm. Tables 1 and 2 report results for the comparison criteria $M_{BDM}(i, j)$ and $CM_{BDM}(i, j)$, defined in Equations 10 and 12, respectively, and provide a baseline comparison under controlled conditions.

The second group contains real-world UCI datasets of increasing size: medium (approximately 10,000 objects), large (up to 100,000 objects), and very large (more than 100,000 ob-

Table 1: Comparison of the DFA-based algorithm and ISBMA on benchmark decision tables from (Zhang et al., 2018) using interclass comparisons.

<i>(a) Matrix dimensions and row counts</i>					
Dataset	O-DS	C-DS	R-BDM	R-SBDM	MR-SBDM
Anneal	898	39	161,297	62	87
Autos	205	26	16,187	29	43
Breastw	699	11	110,378	20	36
Connect-4	67,557	43	1,133,893,847	43	697
Credita	690	16	117,581	23	64
Dermatology	366	35	53,501	1,124	1,621
Heartstatlog	270	14	18,000	68	68
Laborneg	40	17	364	67	67
Lungcancer	32	57	337	327	327
Promoters	106	59	5,565	2,761	2,761

<i>(b) Memory and runtime results</i>					
Dataset	M-BDM	M-SBDM	RT-DFA	RT-ISBMA	RATE
Anneal	15,412 KB	2,868 KB	1.232	0.330	5.37
Autos	2,996 KB	2,228 KB	0.027	0.015	1.34
Breastw	8,628 KB	2,228 KB	0.103	0.043	3.87
Connect-4	> 12.21* GB	119.6 MB	NA	3,308.167	> 102.11
Credita	9,344 KB	1,964 KB	0.260	0.055	4.76
Dermatology	5,984 KB	2,220 KB	1.925	0.834	2.70
Heartstatlog	2,656 KB	2,224 KB	0.037	0.014	1.19
Laborneg	2,224 KB	2,224 KB	0.001	0.001	1.00
Lungcancer	2,224 KB	2,224 KB	0.008	0.004	1.00
Promoters	2,224 KB	2,224 KB	0.286	0.253	1.00

jects). Results obtained with the same comparison criteria are reported in Tables 3 and 4. These experiments assess runtime performance, memory requirements, and the practical scalability of constructing simplified binary matrices. All experiments were conducted on a computer running Linux Mint 22.3 Zena, equipped with an 8-core Ryzen 7 7000-series processor operating at 3.8 GHz and 16 GB of RAM. All algorithms were implemented in the C programming language.

In all tables, the following notation is used:

- **O-DS** and **C-DS** denote the number of objects and the number of features (columns) in the decision table, respectively.
- **R-BDM** and **M-BDM** denote the number of rows in the binary discernibility (or discernibility/similarity) matrix and the amount of RAM used by the DFA-based algorithm, respectively.
- The symbol * indicates the last memory reading recorded before available RAM was exhausted and the program was terminated.
- **R-SBDM**, **MR-SBDM**, and **M-SBDM** denote the number of rows in the simplified binary matrix, the maximum number of rows stored while generating it, and the amount of RAM used by ISBMA, respectively.

Table 2: Comparison of the DFA-based algorithm and ISBMA on benchmark decision tables from (Zhang et al., 2018) using combined interclass and intraclass comparisons.

<i>(a) Matrix dimensions and row counts</i>					
Dataset	O-DS	C-DS	R-BDM	R-SBDM	MR-SBDM
Anneal	898	39	402,753	152	249
Autos	205	26	20,910	30	48
Breastw	699	11	283,259	10	10
Connect-4	67,557	43	2,281,940,346	43	1,114
Credita	690	16	237,412	13	13
Dermatology	366	35	66,795	1,954	2,391
Heartstatlog	270	14	36,795	10	19
Laborneg	40	17	780	19	33
Lungcancer	32	57	496	486	486
Promoters	106	59	5,565	2,674	2,674
<i>(b) Memory and runtime results</i>					
Dataset	M-BDM	M-SBDM	RT-DFA	RT-ISBMA	RATE
Anneal	34,228 KB	2,996 KB	4.156	0.914	11.42
Autos	3,248 KB	2,228 KB	0.040	0.019	1.46
Breastw	16,692 KB	2,228 KB	0.197	0.074	7.49
Connect-4	> 12.6* GB	119.5 MB	NA	3,644.696	> 105.48
Credita	16,768 KB	2,048 KB	0.234	0.086	8.19
Dermatology	7,252 KB	2,220 KB	3.274	1.850	3.27
Heartstatlog	3,960 KB	2,224 KB	0.057	0.018	1.78
Laborneg	2,224 KB	2,224 KB	0.001	0.001	1.00
Lungcancer	2,224 KB	2,224 KB	0.008	0.004	1.00
Promoters	2,424 KB	2,224 KB	0.364	0.303	1.09

- **RT-DFA** denotes the runtime in seconds of the DFA-based algorithm, including construction of the full binary matrix and its subsequent simplification. **NA** indicates that execution did not complete because of insufficient memory.
- **RT-ISBMA** denotes the runtime in seconds required by ISBMA to construct the simplified binary matrix directly.
- **RATE** denotes the ratio between the RAM used by the DFA-based algorithm and that used by ISBMA.

For the benchmark datasets on which both algorithms complete, ISBMA is never slower than the DFA-based algorithm. Under interclass comparisons, the runtime speedup $RT-DFA/RT-ISBMA$ ranges from 1.00 to approximately 4.73, whereas under combined interclass and intraclass comparisons it ranges from 1.00 to approximately 4.55. The *Connect-4* dataset exposes the main scalability limitation of the DFA-based approach: in both comparison schemes, its memory use exceeds 12 GB and execution does not complete, whereas ISBMA completes using about 120 MB. The corresponding DFA/ISBMA memory-use ratios exceed 102 and 105, respectively.

Table 3: Comparison of the DFA-based algorithm and ISBMA on UCI datasets of increasing size using interclass comparisons.

<i>(a) Dataset and matrix dimensions</i>						
Dataset	Type	O-DS	C-DS	R-BDM	R-SBDM	MR-SBDM
Ta-Ba-Pr	medium	6,819	96	1,451,780	348	348
Mushroom	medium	8,124	23	16,478,528	30	59
As-Ma-Le	medium	9,546	8	44,859,258	1	1
Ro-Oc-Es	medium	10,129	19	34,373,047	303	305
Adult	large	32,561	15	193,829,519	8	20
Pr-Cl-Cl	large	35,311	7	556,879,767	2	2
Ban-Mar	large	45,211	17	211,147,458	58	80
Sec-Mus	large	61,069	21	921,109,728	207	375
Rt-iot2022	vlarge	123,117	85	3,019,187,203	4	336
Cl-Da-On-Sh	vlarge	165,474	14	8,252,658,876	2	17
Onl-Ret	vlarge	541,909	8	23,953,744,288	3	3

<i>(b) Memory and runtime results</i>					
Dataset	M-BDM	M-SBDM	RT-DFA	RT-ISBMA	RATE
Ta-Ba-Pr	231.76 MB	27,512 KB	407.428	12.339	8.42
Mushroom	1.03 GB	9,336 KB	54.028	8.083	111.31
As-Ma-Le	2.8 GB	5,496 KB	10.622	4.451	511.14
Ro-Oc-Es	1.06 GB	9,440 KB	34.615	7.199	112.41
Adult	12.13 GB	22,776 KB	136.423	58.977	532.88
Pr-Cl-Cl	> 13.12* GB	15,208 KB	NA	38.310	> 863.29
Ban-Mar	13.2 GB	34,596 KB	601.366	125.835	532.88
Sec-Mus	> 13.16* GB	55,812 KB	NA	970.675	> 235.91
Rt-iot2022	> 14.54* GB	419.18 MB	NA	9,706.670	> 34.70
Cl-Da-On-Sh	> 12.06* GB	104.77 MB	NA	1,482.850	> 115.18
Onl-Ret	> 13.27* GB	193.96 MB	NA	8,598.130	> 68.47

For clarity, the following abbreviated dataset names are used in Tables 3 and 4: Assessing Mathematics Learning in Higher Education (As-Ma-Le), Room Occupancy Estimation (Ro-Oc-Es), Bank Marketing (Ban-Mar), Product Classification and Clustering (Pr-Cl-Cl), Secondary Mushroom (Sec-Mus), Clickstream Data for Online Shopping (Cl-Da-On-Sh), and Online Retail (Onl-Ret).

The scalability difference becomes more pronounced in the second group. Under interclass comparisons, the DFA-based algorithm completes 6 of the 11 datasets, whereas ISBMA completes all 11; among the datasets completed by both methods, the runtime speedup ranges from approximately 2.31 to 33.02. Under combined interclass and intraclass comparisons, the DFA-based algorithm completes only the four medium-sized datasets and fails on every large and very large dataset, while ISBMA again completes all 11; for the four datasets completed by both methods, the speedup ranges from approximately 1.21 to 10.36. The memory results show the same pattern: the DFA/ISBMA memory-use ratio exceeds 500 on several datasets and reaches a lower bound greater than 868 in one case. Thus, as dataset size increases, memory consumption

Table 4: Comparison of the DFA-based algorithm and ISBMA on UCI datasets of increasing size using combined interclass and intraclass comparisons, as defined in Equation 12.

<i>(a) Dataset and matrix dimensions</i>						
Dataset	Type	O-DS	C-DS	R-BDM	R-SBDM	MR-SBDM
Ta-Ba-Pr	medium	6,819	96	21,794,191	349	349
Mushroom	medium	8,124	23	16,517,098	50	77
As-Ma-Le	medium	9,546	8	698,987	5	5
Ro-Oc-Es	medium	10,129	19	34,373,047	17	17
Adult	large	32,561	15	336,254,971	13	15
Pr-Cl-Cl	large	35,311	7	66,535,938	1	1
Ban-Mar	large	45,211	17	810,746,798	16	16
Sec-Mus	large	61,069	21	943,571,118	97	202
Rt-iot2022	vlarge	123,117	85	12,134,568,386	8	344
Cl-Da-On-Sh	vlarge	165,474	14	18,756,457,199	3	18
Onl-Ret	vlarge	541,909	8	269,711,616,629	5	5

<i>(b) Memory and runtime results</i>					
Dataset	M-BDM	M-SBDM	RT-DFA	RT-ISBMA	RATE
Ta-Ba-Pr	3.29 GB	27,768 KB	494.846	407.428	118.71
Mushroom	2.07 GB	9,464 KB	172.753	16.677	218.89
As-Ma-Le	2.85 GB	5,496 KB	11.505	5.129	519.10
Ro-Oc-Es	3.2 GB	9,592 KB	39.781	14.040	334.60
Adult	> 13.15* GB	22,776 KB	NA	121.259	> 577.59
Pr-Cl-Cl	> 13.2* GB	15,208 KB	NA	41.417	> 868.47
Ban-Mar	> 13.16* GB	34,312 KB	NA	258.290	> 383.81
Sec-Mus	> 11.22* GB	55,784 KB	NA	704.263	> 201.28
Rt-iot2022	> 15.14* GB	419.18 MB	NA	15,297.056	> 36.13
Cl-Da-On-Sh	> 10.42* GB	104.64 MB	NA	2,238.280	> 99.62
Onl-Ret	> 14.21* GB	193.96 MB	NA	14,581.440	> 73.26

becomes the decisive limitation of the construct-then-simplify strategy, whereas the incremental construction used by ISBMA remains feasible.

5 Discussion

The experimental behavior is consistent with the complexity analysis in Section 3. The main limitation of the DFA-based algorithm is not the finite-automaton simplification itself, but the need to construct and store the full binary matrix before simplification. ISBMA removes this bottleneck by comparing each newly generated row only with the rows currently retained in *RL*. In the experiments, the intermediate and final row lists remain much smaller than the full matrix, which explains both the lower memory requirements and the runtime improvements observed as the datasets grow.

The simplified matrix produced by ISBMA is a preprocessing structure, but it is also a compact representation that preserves the information required for several downstream combi-

natorial tasks. For example, irreducible testors can be computed from this type of matrix (Piza-Davila et al., 2017) and used for feature selection in supervised classification. In addition, the formal relationship between irreducible testors and minimal transversals (Alba-Cabrera et al., 2019) connects this representation with hypergraph-based problems. Consequently, improving the efficiency of simplified-matrix construction also increases the practical feasibility of these downstream tasks on large or computationally demanding datasets.

6 Conclusions

This work introduced ISBMA, an incremental algorithm that directly constructs simplified binary discernibility and discernibility/similarity matrices without materializing the corresponding full binary matrix. The correctness result shows that this incremental procedure produces exactly the same simplified matrix as the conventional construct-then-simplify approach, while the experiments demonstrate substantially lower memory requirements and improved runtime scalability. ISBMA remains applicable to decision tables with hundreds of thousands of objects in cases where the DFA-based algorithm cannot complete because of memory limitations. Its incremental formulation also provides a natural basis for processing newly generated object comparisons without reconstructing the full binary matrix, making it suitable for data-intensive applications that require scalable discernibility-based representations.

Supplementary Material

To improve reproducibility, a .zip file, available at the following URL <https://drive.google.com/file/d/175Ypx-eyCc34tx9B2kPyofcguZDTyzKi/view?usp=sharing>, includes all of our source code and sample data for the implementations. including a README that provides details on how to install and run our program to test the algorithms on the sample data. Restrictions apply to the availability of the real data, which were used under license for this study.

Acknowledgments

The authors thank the editor, associate editor, and referees for their constructive comments which has led to significant improvement of this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

- Alba-Cabrera E, Godoy-Calderon S, Lazo-Cortes M, Martinez-Trinidad J, Carrasco-Ochoa J (2019). On the relation between the concepts of irreducible testor and minimal transversal. *IEEE Access*, 7: 82809–82816.
- Felix R, Ushio T (1999). Rough sets-based machine learning using a binary discernibility matrix. In: *Proceedings of the Second International Conference on Intelligent Processing and Manu-*

- facturing of Materials*, John A Meech, Marcello M Veiga, Michael H Smith, Steven R LeClair (eds.), 299–305.
- Gao W, Cao Y, Dong X (2024). Risk assessment of water inrush from coal seam floor with a pca-rst algorithm in chenmanzhuang coal mine. *Water*, 16 (22): 1–16.
- Gonzalez-Diaz Y, Martinez-Trinidad J, Carrasco-Ochoa J, Lazo-Cortes M (2024). An algorithm for computing all rough set constructs for dimensionality reduction. *Mathematics*, 12: 1–19.
- Hang L (2024). *Machine Learning Methods*. Tsinghua university press, China.
- Kelly M, Longjohn R, Nottingham K (Accessed June 10, 2025). The uci machine learning repository, <https://archive.ics.uci.edu>.
- Lazo-Cortes M, Carrasco-Ochoa J, Martinez-Trinidad J, Sanchez-Diaz G (2015). Computing constructs by using typical testor algorithms. In: *Proceedings of the Mexican Conference on Pattern Recognition*, JA Carrasco-Ochoa, JF Martinez-Trinidad, JH Sossa-Azuela, JA Olvera-López, Fazel Familli (eds.), 24–27. LNCS Springer.
- Lazo-Cortes M, Ruiz-Shulcloper J, Alba-Cabrera E (2001). An overview of the evolution of the concept of testor. *Pattern Recognition*, 34: 753–762.
- Lazo-Cortes M, Sanchez-Diaz G, Almanza-Ortega N (2024). Shortest-length and coarsest-granularity constructs vs. reducts: An experimental evaluation. *International Journal of Approximate Reasoning*, 170: 1–14.
- Lefebvre-Lobaina J, Ruiz-Shulcloper J (2023). Regularsearch, a fast performance algorithm for typical testors computation. *Information Sciences*, 649: 1–11.
- Li X, Xiao H, Tang J (2023). New variable precision reduction algorithm for decision tables. *IEEE Access*, 11: 42701–42712.
- Liang B, Zhang H, Lu Z, Zhang Z (2022). Indistinguishable element-pair attribute reduction and its incremental approach. *Mathematical Problems in Engineering*, 2022: 1–16.
- Ni P, Zhao S, Wang X, Chen H, Li C, Tsang E (2020). Incremental feature selection based on fuzzy rough sets. *Information Sciences*, 536: 185–204.
- Pawlak Z (1982). Rough sets. *International Journal of Information and Computer Science*, 11: 341–356.
- Pawlak Z (1992). *Rough Sets, Theoretical Aspects of Reasoning About Data*. Springer Dordrecht, United Kingdom.
- Pawlak Z, Skowron A (2007). Rudiments of rough sets. *Information Sciences*, 177: 3–27.
- Piza-Davila I, Sanchez-Diaz G, Lazo-Cortes M, Rizo-Dominguez L (2017). A cuda-based hill-climbing algorithm to find irreducible testors from a training matrix. *Pattern Recognition Letters*, 95: 22–28.
- Rodriguez-Diez V, Martinez-Trinidad J, Carrasco-Ochoa J, Lazo-Cortes M, Olvera-Lopez A (2020). Minreduct: A new algorithm for computing the shortest reducts. *Pattern Recognition Letters*, 138: 177–184.
- Ruiz-Shulcloper J, Abidi M (2002). *Logical Combinatorial Pattern Recognition: A Review*. Transworld Research Networks, India.
- Sanchez-Diaz G, Lazo-Cortes M, Aguirre-Salado C, Piza-Davila I, Garcia-Contreras P (2022). A review of algorithms to computing irreducible testors applied to feature selection. *Artificial Intelligence Review*, 55: 6607–6628.
- Sipser M (2013). *Introduction to the Theory of Computation*. Thomson, USA.
- Skowron A, Rauszer C (1992). *The Discernibility Matrices and Functions in Information Systems*. Kluwer Academic, The Netherlands.
- Susmaga R (2014). Reducts and constructs in classic and dominance-based rough sets approach.

- Information Sciences*, 271: 45–64.
- Susmaga R, Slowinski R (2015). Generation of rough sets reducts and constructs based on inter-class and intra-class information. *Fuzzy Sets and Systems*, 274: 124–142.
- Wang M, Wu C, Wang L, Xiang D, Huang X (2019). A feature selection approach for hyperspectral image based on modified ant lion optimizer. *Knowledge-Based Systems*, 168: 39–48.
- Wang X, Ma Y (2006). Research on space complexity of binary discernibility matrix. *Journal of Tianjing University of Science & Technology*, 21: 50–53.
- Webb A (2002). *Statistical Pattern Recognition*. Wiley, USA.
- Wei W, Song P, Liang J, Wu X (2019). Accelerating incremental attribute reduction algorithm by compacting a decision table. *International Journal of Machine Learning and Cybernetics*, 10: 2355–2373.
- Xu Z, Yang B, Qian W, Shu W (2010). An efficient algorithm for Pawlak reduction based on simplified discernibility matrix. *Fuzzy Information and Engineering*, 4: 433–443.
- Yao Y, Zhao Y (2008). Attribute reduction in decision theoretic rough set models. *Information Sciences*, 178: 3356–3373.
- Yao Y, Zhao Y (2009). Discernibility matrix simplification for constructing attribute reducts. *Information Sciences*, 179: 867–882.
- Zhang N, Gao X, Yu T (2019). Heuristic approaches to attribute reduction for generalized decision preservation. *Applied Sciences*, 9: 1–20.
- Zhang N, Li B, Zhang Z, Guo Y (2018). A quick algorithm for binary discernibility matrix simplification using deterministic finite automata. *Information*, 9: 1–18.
- Zhi T, Miao D (2002). The binary discernibility matrix’s transformation and high efficiency attributes reduction algorithm’s conformation. *Computer Science*, 49: 883–896.
- Zhou H, Zhang Y, Zhang Y, Liu H (2019). Feature selection based on conditional mutual information: Minimum conditional relevance and minimum conditional redundancy. *Applied Intelligence*, 49: 883–896.