Received January 24, 2020, accepted March 3, 2020, date of publication March 19, 2020, date of current version March 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982133

An Algorithm for Computing Minimum-Length Irreducible Testors

IVÁN PIZA-DÁVILA[1], GUILLERMO SÁNCHEZ-DÍAZ[2], MANUEL S. LAZO-CORTÉS[3], AND IVÁN VILLALÓN-TURRUBIATES[1], (Senior Member, IEEE)
[1]Instituto Tecnologíco y de Estudios Superiores de Occidente, Tlaquepaque 45604, México
[2]Faculty of Engineering, Universidad Autónoma de San Luis Potosí, San Luis Potosí 78210, México
[3]TecNM/Instituto Tecnológico de Tlalnepantla, Tlalnepantla de Baz 54070, México

Corresponding author: Iván Piza-Dávila (hpiza@iteso.mx)

ABSTRACT In pattern recognition, the elimination of unnecessary and/or redundant attributes is known as feature selection. Irreducible testors have been used to perform this task. An objective of the Minimum Description Length Principle (MDL) applied to feature selection in pattern recognition and data mining is to select the minimum number of attributes in a data set. Consequently, the MDL principle leads us to consider the subset of irreducible testors of minimum length. Some algorithms that find the whole set of irreducible testors have been reported in the literature. However, none of these algorithms was designed to generate only minimum-length irreducible testors. In this paper, we propose the first algorithm specifically designed to calculate all minimum-length irreducible testors from a training sample. The paper presents some experimental results obtained using synthetic and real data in which the performance of the proposed algorithm is contrasted with other state-of-the-art algorithms that were adapted to generate only irreducible testers of minimum length.

INDEX TERMS Feature selection, MDL principle, minimum-length irreducible testors, testor.

I. INTRODUCTION

Feature selection is a significant task in supervised classification and other pattern recognition problems focused on removing irrelevant and/or redundant features [22]. It is a process of selecting subsets of the entire set of features to optimally reduce the feature space according to certain evaluation criteria [2].

The aim of dimension reduction is to find a minimum set of attributes that preserves all the essential information of the training sample for pattern recognition or data mining domains [23], [40], [45], [48], [49].

There are several problems described by a considerable amount of data, in which is desirable to find an optimal (or near to this) feature subset which maximizes the classification performance and minimizes the number of selected features [10], [15], [41], [44]. Among other feature selection developed techniques, testors have been focused on this purpose; as well as the reducts from the rough set theory, which have been related to a certain type of testors called typical or irreducible [19].

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

The concept of testor in pattern recognition problems was introduced by Dmitriev *et al.* [12]. They defined a testor as a subset of features that allows differentiating objects from different classes. A testor is called irreducible if it is not a superset of another testor. So, an irreducible testor has the same capability as the entire feature set to discern between objects belonging to different classes. This is the reason why irreducible testors have been used for solving several problems related to supervised pattern recognition (see for example [11], [20], [25], [29], [47]).

The computation of the whole set of irreducible testors has exponential complexity with respect to the number of features [46]. In addition, the number of irreducible testors found from a dataset could be outsize. Then, finding a subset of irreducible testors that meet any specific property is desirable.

Some algorithms for data compression, noise elimination, and pattern candidate generation, are based on the principle of Minimum Description Length [31]. One goal of this principle applied to feature selection in pattern recognition and data mining is selecting the minimal number of features from a dataset. Besides, in decision rules generation algorithms, minimum-length features subsets are desirable. These rules would be simpler and easier to find. So, in practical applications, minimum-length descriptions are preferred (see for example [1], [5], [6], [13], [14], [18], [24], [30], [42]

Consequently, the principle of Minimum Description Length leads us to consider the subset of irreducible testors of minimum length. Obviously, all minimum length testors are irreducible but the reciprocal in general is not true.

Within the framework of rough set theory, algorithms have been developed and are still being developed to calculate the minimum length reducts and/or subsets of reducts that meet certain optimality criteria [4], [8], [9]. However, no algorithm specifically developed for calculating the minimum length testors is reported in the literature.

In this paper, we propose the first algorithm specifically designed for computing all minimum length irreducible testors from a training sample. The proposed algorithm, MLIT, does not search the entire set of irreducible testors, but only calculates those of minimum length.

The rest of this paper is structured as follows: in section II we briefly comment about related work; section III contains the theoretic background, section IV details our proposal; in section V, we show and discuss the experimental results, and finally our conclusions constitute section VI.

II. RELATED WORK

There exist several methods for computing irreducible testors from a training sample. Some of them are exhaustive algorithms and other heuristics.

The exhaustive algorithms take exponential time in the number of features. Some examples of these algorithms include the following: Lex [39], all-NRD [7], Fast-CT-EXT [37], YYC [3], Fast-BR [21], and Parallel-YYC [27]. These algorithms can be easily adapted to return only minimum-length irreducible testors. This is possible if the algorithm guarantees that the entire set of irreducible testors was found.

On the other hand, heuristics take polinomial time in a fixed number of generations. Some state-of-the-art heuristics include the following: HC [38] and PHC [26]. For large and/or dense datasets, these heuristics have shown very good performance. However, the output of a heuristic is only a subset of the entire set of irreducible testors. Therefore, the minimum-length irreducible testors taken from the output might not be actual minimum-length irreducible testors in a global context.

Thus, the irreducible testors of minimum-length from the output of a heuristics might not be all of them or even not be minimum-length irreducible testors.

There exists no algorithm reported in the literature designed just to find minimum-length irreducible testors.

III. BACKGROUND

Let $TS = \{O_1, O_2, \cdots, O_m\}$ be a training sample containing m objects, distributed in c classes K_i, $i = 1, \cdots, c$, described in terms of n features $R = \{x_1, x_2, \cdots, x_n\}$, where $x_i \in R$ can takes a qualitative or quantitative value.

Let DM be a dissimilarity matrix, whose rows are obtained from feature-by-feature comparison between every pair of objects belonging to different classes, using a criterion D_i, which returns 0 or 1, if the compared values of the corresponding feature are similar or dissimilar, respectively [35].

Definition 1 [35]: Let p and q be two rows of DM. We say that p is a sub-row of q, if: $\forall\, j[q_j = 0 \Rightarrow p_j = 0]$ and $\exists_i[p_i = 0\ AND\ q_i = 1], j \in \{1, \cdots, n\}$. A row r of DM is called basic if no row in DM is a sub-row of r. The submatrix of DM that only contains all its basic rows (without repetitions), is called a basic matrix (denoted BM)

Let $T \subseteq R$ be a subset of features. BM^T denotes the submatrix of BM only containing all columns corresponding to features in T.

Definition 2 [28]: Let p be a row of BM^T; we say that p is a zero row if it contains only zeros.

Definition 3 [28]: A set T is a testor of TS, if there are no zero rows in BM^T

Definition 4 [28]: Let $x_i \in T$, x_i is called a non-removable feature of T if there exists a row p in BM^T such that if we eliminate from BM^T the column corresponding to x_i, the remaining row p is a zero row of $BM^{T-\{x_i\}}$. Otherwise, x_i is called a removable feature.

Definition 5 [28]: A set T is called an irreducible testor of TS if T is a testor of TS and each feature $x_i \in T$ is a non-removable feature of T.

Definition 6 A set T' is called minimum-length irreducible testor of TS if T' is an irreducible testor of TS and $|T'| \leq |T|$ for every testor T of TS.

Definition 7 [36]: Let $T \subseteq R$, $x_j \in R$, $x_j \notin T$. Besides, let $zr(T)$ be the number of zero rows of T. The feature x_j contributes with T if and only if $zr(T \cup \{x_j\}) < zr(T)$.

From these definitions, the following two propositions are quite immediate [36].

Proposition 1: Let $T \subseteq R$, $x_j \in R$, $x_j \notin T$. If x_j does not contribute with T, then $T \cup \{x_j\}$ will not generate any irreducible testor.

Proposition 2: Let $T \subseteq R$. If $zr(T) = 0$ then T is a testor.

We introduce the following proposition to determine a priori if a zero row will prevail after adding any of the remaining features to the current combination.

Proposition 3: Let $Q = \{x_{i_1}, \ldots, x_{i_q}\}$, $Q \subset R$, and let $x_s \in R$, $x_s \notin Q$. If there exists a zero-row p in BM^Q such that $p_{x_s} = p_{x_{s+1}} = \ldots = p_{x_n} = 0$ then $Q \cup \{x_s\}$ will not generate any testor.

Proof 1: Since there exists a zero-row in BM^Q, Q is not a testor. By hypothesis, if we consider $BM^{Q\cup\{x_s\}}$ the row p prevails as a zero row, even if we add any feature in $\{x_{s+1}, \ldots, x_n\}$. So, we can conclude that $Q \cup \{x_s\}$ will not generate any testor.

IV. PROPOSED ALGORITHM

This section introduces an algorithm that finds the entire set of minimum-length irreducible testors. Depending on the density of the input basic matrix, this algorithm selects one of two different search methods. We say that a basic matrix is sparse if its density is not greater than 0.30. This number was obtained experimentally, as will be discussed later.

Like most of the algorithms reported for computing irreducible testors, these algorithms operate over the basic matrix, instead of the training sample [28]. They both carry out a breadth-first search over the power set of features instead of the depth-first search, performed by various exhaustive algorithms like Fast-CT-EXT and Fast-BR.

In general terms, the algorithm builds feature combinations in non-decreasing order of length. It never builds a combination longer than a minimum-length irreducible testor.

For each feature combination built, it is verified whether it satisfies the testor property or not (see Definition 3).

The first feature combination holding this property is a minimum-length irreducible testor since it cannot be a superset of an irreducible testor which has not been found yet (see Definitions 5 and 6).

As in [36], a preprocessing of the basic matrix is performed before calling any of these algorithms in order to improve performance. This preprocessing finds a row with the least amount of ones (r_{min}) and each column c_i where $r_{min}[i] = 1$ is moved to the left in the basic matrix. The features are relabelled according to this new arrange. f_{min} denotes the number of ones in r_{min}.

Every combination built by any of these algorithms contains a feature in the range $[x_0, \cdots, x_{f_{min}}]$ at its first position because otherwise r_{min} would be a zero-row, leading us to a non testor (see Definition 3).

A. IN-PLACE SEARCH BASED ON NEXT COMBINATION CALCULATION

If the basic matrix is not sparse, this technique iteratively produces the next feature combination from the current one, by modifying the value of one or more features in the input combination (see Algorithm 1). The sequence of combinations follows the lexicographical order of feature indexes, such that the combination size ranges from 1 to $|T|$ where T is the first testor found.

Example 1: Sequences of combinations for $n = 5$ and $f_{min} = 3$, grouped by length s, are:
$s = 1 : \{x_1\}, \{x_2\}, \{x_3\}$
$s = 2 : \{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \{x_1, x_5\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_2, x_5\}, \{x_3, x_4\}, \{x_3, x_5\}$
$s = 3 : \{x_1, x_2, x_3\}, \{x_1, x_2, x_4\}, \{x_1, x_2, x_5\}, \{x_1, x_3, x_4\}, \{x_1, x_3, x_5\}, \{x_1, x_4, x_5\}, \{x_2, x_3, x_4\}, \{x_2, x_3, x_5\}, \{x_2, x_4, x_5\}, \{x_3, x_4, x_5\}$
$s = 4 : \{x_1, x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_5\}, \{x_1, x_3, x_4, x_5\}, \{x_2, x_3, x_4, x_5\}$
$s = 5 : \{x_1, x_2, x_3, x_4, x_5\}$

As can be seen in Example 1, every combination is built as an increasing sequence of feature indexes. That is, if $T = \{x_{f1}, \cdots, x_{fs}\}$ is a combination of s features, the following holds: $f_1 < f_2 < \ldots < f_s$.

If $T_L = \{x_{l_1}, x_{l_2}, \cdots, x_{l_s}\}$ is the last combination of s features in the enumeration, then for $s \geq 2$, $l_s = n$, and $l_j = l_{j+1} - 1$ for every $j = 1, \cdots, s-1$ if $s \geq (n + 1 - f_{min})$, but for $j = 2, \cdots, s-1$ if $s \leq (n - f_{min})$

Besides, for all s, $l_1 = min\{f_{min}, n - s + 1\}$.

The next-combination method (NC) searches for the rightmost feature x_{c_i} in the current combination $\{x_{c_1}, x_{c_2}, \cdots, x_{c_s}\}$, that fulfils $c_i < l_i$. If such feature x_{c_i} exists, then c_i increases by one and, for each $j > i$, $x_{c_j} = x_{1+c_{j-1}}$.

Otherwise, the current combination is actually the last combination of length s. Therefore, the first combination of the next length is built as: $\{x_1, \ldots, x_s\}$

B. SEARCH WITH PRUNING BASED ON FEATURE AND ROW CONTRIBUTIONS

If the basic matrix is sparse, we apply another strategy; it prunes the search space required by the lexicographical order by keeping in a queue data-structure only potential combinations (see Algorithm 2), i.e. it discards combinations that will not lead us to any irreducible testors. In sparse matrices, non-potential combinations are very common. By using propositions 1 and 3, we can determine if the current combination will not lead us to any irreducible testor after adding new features. We refer to this strategy as pruning based on feature and row contributions ($PFRC$).

In Algorithm 2, the invocation $Cont(T; x_f)$ returns true only if the number of zero-rows in $BM^{T \cup \{x_f\}}$ is less than that in BM^T. The invocation $ZeroRowPrevails(T, x_f)$ returns true if there exists at least one zero-row in BM^T that will prevail in $BM^{T \cup T'}$, being $T' = \{x_f, x_{f+1} \cdots x_n\}$. In order to determine this in constant time, a boolean matrix AZR is built a priori. AZR is just an auxiliary matrix that contains cumulative information about the existence of prevailing zero rows to the right of a cell.

Each cell (row, col) is filled as follows:

$$AZR(row, col) = \begin{cases} 1 & \text{if } \exists\, k,\ col \leq k \leq n, \\ & \text{such that } BM(row, k) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Thus, if for each zero-row r in BM^T, $AZR(r, x_f) = 1$, then $ZeroRowAccFound(T, x_f)$ returns false; otherwise, it returns true.

Example 2:

$$BM = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{1}$$

$$AZR = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2}$$

In Example 2, $AZR(2, 2) = 1$ because $BM(2, k) = 1$, for $k = 3$ and $k = 6$. Similarly, $AZR(3, 5) = 0$ because $BM(3, k) = 0$, for $5 \leq k \leq 6$.

Following the steps of Algorithm 1, the first combinations built are $\{x_1\}$ and $\{x_2\}$. None of them is a testor, so both are enqueued. The first combinations dequeued are $\{x_1\}$ and $\{x_2\}$. We determine if each of the following combinations is either a testor or potential: $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_1, x_4\}$, $\{x_1, x_5\}$, $\{x_1, x_6\}$, $\{x_2, x_3\}$, $\{x_2, x_4\}$, $\{x_2, x_5\}$ and $\{x_2, x_6\}$.

Combination $\{x_1, x_2\}$ is not a testor because there is a zero row (row 4) in $BM^{\{x_1,x_2\}}$. Therefore, we test if $\{x_1, x_2\}$ is a potential combination. We can see that $\{x_2\}$ contributes with $\{x_1\}$ because there are fewer zero-rows in $BM^{\{x_1,x_2\}}$ than in $BM^{\{x_1\}}$. Besides, no zero-row in $BM^{\{x_1\}}$ prevails in $BM^{\{x_1,x_2\}\cup T'}$, being $T' \subseteq \{x_3, x_4, x_5, x_6\}$. Therefore, $\{x_1, x_2\}$ is enqueued. x_3 and x_4 contributes with $\{x_1\}$, but combinations $\{x_1, x_3\}$ and $\{x_1, x_4\}$ are not a testor, because there are zero-rows in $BM^{\{x_1,x_3\}}$ and $BM^{\{x_1,x_4\}}$ respectively. Both combinations are enqueued.

On the other way, x_5 and x_6 contributes with combination $\{x_1\}$. However, for these combinations, $AZR(3, 5) = 0$ and $AZR(3, 6) = 0$. Like $BM(3, col = 1) = 0$, then any of these features will generate a testor with $\{x_1\}$. Both combinations are not enqueued. The next combination dequeued is $\{x_2\}$. We determine if each of the following combinations is a testor: $\{x_2, x_3\}$, $\{x_2, x_4\}$, $\{x_2, x_5\}$ and $\{x_2, x_6\}$. Only combination $\{x_2, x_6\}$ is a testor, so it is added to the final list, and MinLength is set to 2. The next combination dequeued is $\{x_1, x_2\}$. Since its length is no less than MinLength it is discarded (it will not produce a minimal-length testor) and the queue is emptied.

Figure 1, shows the general behavior of proposed algorithm.

The input of algorithms 1 and 2 is the following: a Basic Matrix BM which columns are arranged in a way that the first f_{min} columns contain 1 at r_{min}, and the number of features at r_{min}. The output of both algorithms is a list L containing all the minimum-length irreducible testors of the input matrix.

Algorithm 1 In-place search based on next combination calculation.

```
NC-MiLIT (BM, $f_{min}$)
    Let $L$ be an empty list of feature combinations
    Let $T \leftarrow \{x_1\}$
    Let $MinLength \leftarrow 0$
    While $MinLength = 0$ OR $MinLength \geq |T|$
        If $IsTestor(T, BM)$
            $L \leftarrow L \cup \{T\}$
            $MinLength \leftarrow |T|$
        $T \leftarrow NextCombination(T, f_{min})$
    Return $L$
```

Algorithm 2 Search with pruning based on feature and row contrubitions.

```
PFRC-MiLIT(BM, $f_{min}$)
    Let $L$ be an empty list of feature combinations
    Let $Q$ be an empty queue of feature combinations
    Let $MinLength \leftarrow 0$
    For $f \leftarrow 1$ to $f_{min}$
        Let $c \leftarrow \{x_f\}$
        $Q.enqueue(c)$
    While $Q$ is not empty
        Let $T \leftarrow Q.dequeue()$
        If $MinLength > 0$ AND $|T| > MinLength$
            $Q.empty()$
        Else
            If $IsTestor(T, BM)$
                $L \leftarrow L \cup \{T\}$
                $MinLength \leftarrow |T|$
            Else
                Let $i \leftarrow |T|$
                Let $nf \rightarrow NextFeature(T[i], f_{min})$
                For $f \leftarrow nf$ to $F$
                    If $Cont(T, x_f)$ and
                        not $ZeroRowPrevails(T, x_f)$
                        $Q.enqueue(T \cup \{x_f\})$
    Return $L$
```

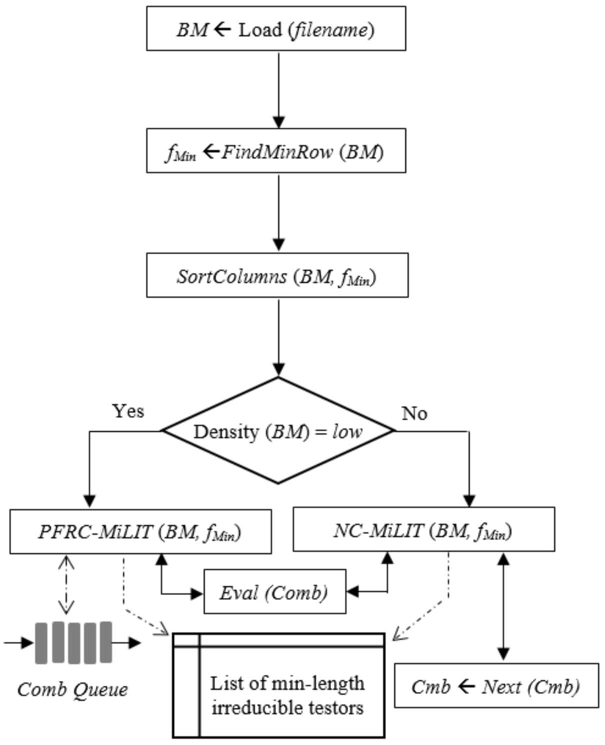

FIGURE 1. Flowchart with the behavior of proposed algorithm.

Let T be the first testor found. The output of the algorithm proposed is a list containing every testor of length $|T|$. By proposition 4, T is an irreducible testor. By corollary 1, $|T|$ is the minimum length of any testor of BM. By corollary 2, we can conclude that all the testors found by the algorithm proposed are minimum-length irreducible testors.

Proposition 4: Let T be the first testor of TS found by the algorithm MLIT, then T is an irreducible testor.

Proof 2: Since the search strategy of the algorithm is breadth-first, from feature subsets with lower cardinal to subsets with greater cardinal; if some subset of T is a testor, it would have been previously found by the algorithm. Then, being T a testor without subsets that are testors, T is an irreducible testor.

Corollary 1: Let T be the first testor found by the algorithm MLIT, then $|T|$ is the minimum-length of the irreducible testors of TS.

Corollary 2: Let T be the first testor found by the algorithm MLIT, then the algorithm MLIT stops searching minimum length irreducible testors when the search among the subsets of cardinal equal to $|T|$ ends.

Remark 1: The length of the irreducible testors computed from a given basic matrix cannot be determined beforehand. If a depth-first search (DFS) is used, in the worst case all the irreducible testors must be computed in order to determine which ones have minimal-length. This would happen if testors are found in decreasing order of length.

On the other hand, when a breadth-first search (BFS) is applied, the combinations are processed in non-decreasing order of length, thus, the algorithm is certain that the first testor found has minimal length.

If a DFS-based algorithm process the BM described in equation 1, with the same pruning criteria utilized in example 2, the following 18 combinations are processed, where NT, TE and IT stand for No Testor, Testor and Irreducible Testor, respectively: $\{x1\}$ (NT), $\{x1, x2\}$ (NT), $\mathbf{\{x1, x2, x5\}}$ **(IT)**, $\{x1, x2, x6\}$ (TE), $\{x1, x3\}$ (NT), $\{x1, x3, x4\}$ (NT), $\mathbf{\{x1, x3, x5\}}$ **(IT)**, $\{x1, x3, x6\}$ (NT), $\{x1, x4\}$ (NT), $\mathbf{\{x1, x4, x5\}}$ **(IT)**, $\mathbf{\{x1, x4, x6\}}$ **(IT)**, $\{x2\}$ (NT), $\{x2, x3\}$ (NT), $\mathbf{\{x2, x3, x5\}}$ **(IT)**, $\{x2, x3, x6\}$ (NT), $\{x2, x5\}$ (NT), $\{x2, x5, x6\}$ (TE), $\mathbf{\{x2, x6\}}$ **(IT)**. Notice that the only minimal-length testor was found at the end.

On the other hand, when a BFS-based algorithm is utilized, 9 combinations are processed, as is previously showed in example 2.

C. COMPLEXITY ANALYSIS

The term that will dominate the equation expressing the running time of any of algorithms 1 and 2 corresponds to the loop in charge of building and processing combinations. Hence, we start this analysis by calculating the number of combinations built by Algorithm 1.

As mentioned above, both algorithms carry out a BFS, which enable them to process combinations in increasing order of length. Let k be the length of the first testor found. Algorithm 1 finishes after all the combinations of length k are processed. Equation 3 expresses the number of combinations processed by the algorithm as the sum of the first k binomial coefficients for fixed n:

$$C(n, k) = \sum_{i=1}^{k} \binom{n}{i} \tag{3}$$

If we plot Equation 3 for a fixed two-digit n and every k from 1 to n, we obtain a logistic curve (see Figure 2). Equation 4 expresses an approximation of the number of combinations built as a logistic function, setting 2^N as the curve's maximum value, and 2^{N-1} as the sigmoid's midpoint. Clearly, as k becomes smaller, the denominator grows and, therefore, the number of combinations reduces dramatically.

$$C(n, k) = \frac{2^N}{1 + e^{-(k-2^{N-1})}} \tag{4}$$

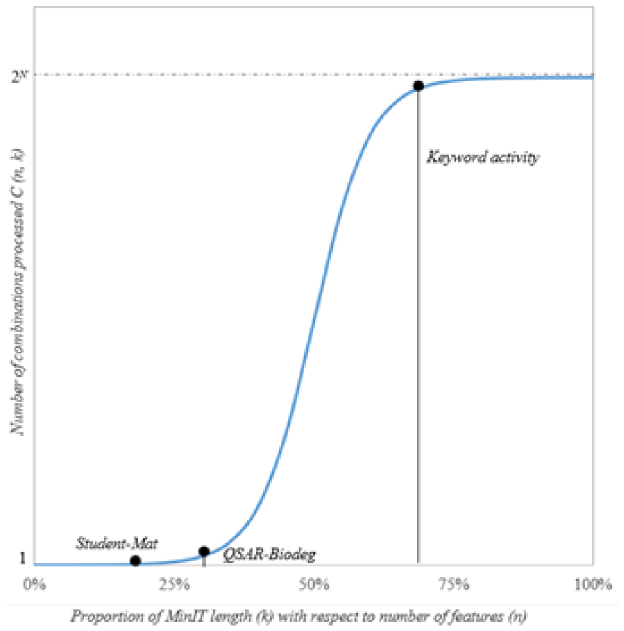

FIGURE 2. **Proportion of MinIT length (k) with respect to number of features (n).**

Figure 2 also depicts the number of combinations processed by the algorithm with respect to the total number (Equation 3), using three of the datasets reported in Table 1: *Student-Mat, QSAR-Biodeg* and *Keyword activity*. There, we can appreciate that when the length of the minimum testors is less than a quarter of the number of features, the number of combinations built and processed is negligible with respect to the power set of features.

Algorithm 1 builds and evaluates every combination of length 1 to k. To do this, it calls function *NextCombination* in charge of producing the next combination in terms of the current one. Function *isTestor* takes $O(r * k)$ time since it searches for a zero-row in the basic matrix considering only k columns. Function *NextCombination* takes $O(k)$ time since it modifies at most $|T|$ values at the current combination (either from left to right or viceversa), and $|T|$ is upper bounded by k. Equation 5 expresses the running time of Algorithm 2.

$$\begin{aligned} T_{NC-MiLIT}(r, n, k) &= C(n, k) * (O(r * k) + O(k)) \\ &= O((r * k) * C(n, k)) \end{aligned} \tag{5}$$

Algorithm 2 builds and evaluates only potential combinations of length 1 to k. To do this, the algorithm holds a queue data-structure. Those combinations that will definitely never lead us to a testor are not enqueued. Both functions that determine this –*Cont* and *ZeroRowPrevails*– take $O(r * k)$ time. Therefore, Equation 5 represents an upper bound of the untime, where every combination is potential. However, in sparse matrices, a huge amount of combinations will never lead us to a testor because current zero-rows will remain the same. Equation 6 expresses a lower bound of the runtime, where we assume that, for each length $l = 1, 2, \ldots, k - 1$,

TABLE 1. Basic matrices with small amount of columns.

Dataset	Dimensions	Density	IT	MiLIT	length	GCR	PYYC	MF $(algorithm)^*$
Chess(KR vs KP)	29×36	0.03	4	4	29	**0.0**	0.016	0.009 (P-M)
Keyword activity	26×36	0.03	3	1	25	0.556	**0.0**	0.002 (P-M)
Connect-4	406×42	0.05	35	10	24	89.3	**0.046**	0.97 (P-M)
QSAR-Biodeg	40×41	0.08	256	2	13	0.2	**0.016**	0.137 (P-M)
Diabetes	2212×49	0.27	77,349	1	1	26.8	21.298	**0.021** (P-M)
Landsat	9815×36	0.33	12,412,798	6	14	14857.53	13,646.115	**13,622.496** (N-M)
Dermatology	1103×34	0.34	112,708	137	6	19.5	9.594	**0.332** (N-M)
German credit	223×20	0.35	846	1	2	0.006	0.032	**0,002** (N-M)
Flags	390×29	0.35	23,543	1	1	0.826	0.890	**0.001** (N-M)
Sponge	68×45	0.38	10,992	1	1	0.518	0.078	**0.001** (N-M)
Student-Por	8158×32	0.41	851,584	1	7	2459.180	1,034.666	**1.538** (N-M)
Student-Mat	6904×32	0.43	679,121	2	6	1408.169	444.153	**0.26** (N-M)
Lung cancer	237×56	0.46	4,183,355	112	4	175.565	41.767	**0.002** (N-M)
Cylinder bands	1147×39	0.50	23,534	2	2	6.054	3.172	**0.004** (N-M)
Waveform	14172×21	0.50	86,977	6218	9	**2.829**	44.752	13.662 (N-M)

the algorithm chooses only one combination from the set of at most n possible combinations; for instance: $< x_1 >$, $< x_1, x_2 >$, $< x_1, x_2, x_3 >$, …, $< x_1, x_2, \ldots, x_k >$. The average runtime of Algorithm 2 is somewhere in between Equations 5 and 6.

$$T_{PFRC-MiLIT}(r, n, k) = O((r * k) * (n * k)) \qquad (6)$$

Regarding to the space complexity, the present analysis does not consider the memory required by the input matrix and the list of irreducible testors returned. Algorithm 1 is an in-place search which means that it does not require extra memory to perform its work, except for the current combination. Therefore, its space complexity is $O(n)$. In contrast, Algorithm 2 requires a queue data-structure to discard combinations with no future. In addition, this algorithm requires a $(r \times n)$ matrix employed by function *ZeroRowAccFound* to determine if the current combination has future. The space complexity is in terms of the maximum size of the queue. As k approaches to $\frac{n}{2}$, the number of combinations of length k becomes larger. Equation 7 expresses the space complexity of Algorithm 2, assuming $k < \frac{1}{2}n$.

$$S_{PFRC-MiLIT}(r, n, k) = O((r * n) + n * \binom{n}{k}) \qquad (7)$$

V. EXPERIMENTS AND RESULTS

We carried out some experiments with the aim of comparing the efficiency of the algorithm proposed with other algorithms in the state of the art, namely, GCreduct [32] and Parallel-YYC [27]. We have selected these two algorithms for being recent and the fastest ones in finding the entire set of irreducible testors in most of the basic matrices employed. We added to both algorithms a process that finds and selects only minimum-length irreducible testors from the output. This process consumes negligible time with respect to the total execution time.

Remark 2: In table 1, the original Chess dataset has 3196 objects, indeed. However, the input of our algorithm is a basic matrix. In this particular case, the basic matrix contains only 29 rows. Recalling Definition 1, the basic matrix contains only the basic rows from the dissimilarity matrix which, in turn, is obtained from feature-by-feature comparison between every pair of objects at the training set.

Table 1 contrasts the run time of the proposed algorithm with GCreduct and Parallel-YYC using some datasets presented in [32]. The description of the table is as follows:

a) Dataset is the name of the original training sample; b) dimensions of each basic matrix is expressed as rows per columns; c) density is calculated as the number of ones divided by the total number of cells of the basic matrix; d) Column titled IT denotes the maximum number of irreducible testors that can be found from each training matrix. e) MiLIT stands for the number of minimal-length irreducible testors; f) GCR means GCreductR; g) length refers to minimal-length; for some basic matrices, this length is 1; this means that they have at least a column containing only ones. h) PYYC means Parallel-YYC; i) MF denotes the algorithm proposed (MiLIT Finder) and the name of the strategy employed is in brackets. The last three columns show the runtime in seconds for the corresponding algorithm. In tables 1, 2, and 3, on MF(algorithm)*, P-M refers to PFRC-MiLIT, N-M refers to NC-MiLIT.

For each row, the cell containing the shortest runtime is highlighted.

Tables 2 and 3 present similar comparisons, but using sintetics basic matrices obtained from previously works as [21] and [26], respectively. OOM indicates an Out of Memory Error occurred during the execution, whereas NF means that the algorithm did not finish its execution in less than a week.

All algorithms were implemented in Java. The experiments were performed on a 3.00 GHz Intel Xeon E5-1607 processor with 8.0 GB of memory, running Windows 8 (64 bits).

As can be seen in Tables 1, 2 and 3; in very few cases, the algorithm proposed was outperformed by either GCreduct or Parallel-YYC. However, the difference in running time in such cases was in the order of the milliseconds, which is negligible if we consider the running time achieved with large basic matrices.

Previously, we conducted an experiment, to select a threshold that would allow us to discern when to use one strategy

TABLE 2. Basic matrices with medium dimensions.

Dimensions	Density	IT	MLIT	length	GCR	PYYC	MF $(algorithm)^*$
150×70	0.61	4,299,547	10	3	66.2	41.892	**0.003** (N-M)
150×90	0.61	17,597,374	19	3	124.5	72.143	**0.006** (N-M)
1000×100	0.70	166,254,922	11,413	4	1235.1	3227.515	**1.381** (N-M)
500×200	0.74	593,808,700	894	3	44842.08	OOM	**0.214** (N-M)
1500×120	0.75	313,440,173	37,484	4	16104.6	OOM	**5.494** (N-M)

TABLE 3. Basic matrices used by heuristics.

Dimensions	Density	IT	MLIT	length	GCR	PYYC	MF $(algorithm)^*$
1215×105	0.33	NF	467	4	NF	OOM	**0.461** (N-M)
200×600	0.84	293,274,709	2,368	2	27019.9	OOM	**0.032** (N-M)

TABLE 4. Contrasting run times of strategies 1 and 2 with basic matrices of different densities.

Basic matrix $(rows \times cols)$	Density	MLIT	In-place search NC-MiLIT	Search with pruning PFRC-MiLIT
Keyword activity (26×36)	0.03	1	1,611.55	0.002
Chess(KR vs KP) (29×36)	0.03	4	1,397.89	0.009
QSAR-Biodeg (40×41)	0.08	2	325.53	0.137
MB_{21} (1215×105)	0.33	467	0.461	1.692
Dermatology (1103×34)	0.34	137	0.332	0.821
Student-Por (8158×32)	0.41	1	1.538	76.732
Student-Mat (6904×32)	0.43	2	0.26	31.779
Cylinder-bands (1147×39)	0.50	2	0.004	0.005
MB_{22} (200×600)	0.84	2,368	0.032	0.205

or the other. For this, we selected some training samples, with different densities in their basic matrices and execute the two strategies. Table 4 contains the samples selected for this experiment, sorted according to their density, likewise, the table shows the runtimes obtained for each strategy.

From this experiment, we selected 0.30 as threshold to consider if a matrix is sparse or not, and consequently choose the PFRC strategy or the NC, respectively. This result is compatible with others reported in related papers [32]–[34].

The algorithm proposed has proved to have a good performance no matter the density of the input basic matrix. We can conclude that if the input basic matrix has low density (<0.3), the search with pruning based on feature and row contributions, $(PFRC)$, performs very well because it discards quickly combinations that will never lead us to an irreducible testor. Otherwise, the in-place search based on next combination, (NC), is the best option because it does not consume time in creating objects and never runs out of memory.

A. DISCUSSION

In most of the datasets used in experiments, the length of minimum irreducible testors are much less than half of the number of columns, as is described in sub-section IV-C.

As can be seen in Table 1, the length of the minimum irreducible testor is 1 in some cases. This means that these basic matrices have at least a column containing only ones.

As shown in tables 2 and 3, for some datasets both algorithms YYC and GCreduct took hours or days to finish their execution because they have to find the entire set of irreducible testors to detemine which ones are of minimal-length.

Pruning techniques introduced in section IV-B reduced dramatically the search space in sparse matrices, but they require extra memory to keep only potential combinations.

Finally, the experiments carried out also proved that, if we follow the lexicographical order of feature indexes, we can find minimum-length irreducible testors much faster than running the fastest algorithms based on depth-first searches.

B. PERFORMANCE ON CLASSIFICATION OF MiLIT

This work is not focused on assessing the performance of minimal-length irreducible testors on classification tasks. Nevertheless, a comparative study in this regard can be found in [43]. This article shows the effectiveness of minimal-length reducts over variable-length reducts for object classification purposes. If the dissimilarity matrix has no zero-rows, the algorithm proposed in [43] can be adapted easily to find irreducible testors.

This algorithm performs a Depth-First search (DFS) to find reducts. When a new reduct is found and its length is shorter than the reducts found before, all these reducts are replaced with the new reduct. In the worst case, all reducts have to be computed, i.e., the DFS traverses the entire search space.

VI. CONCLUSION

Computing the set of all the irreducible testors of a training sample is a problem of exponential complexity, and therefore it consumes a great amount of computation time. Hence, it makes sense to consider finding some subset, instead of finding all of them. Among those subsets, we can think of calculating only those minimum-length testors. As far as we know, although there are a number of algorithms published to compute all irreducible testors, there is no previous publication that contains a proposal like the one presented here, which is specifically designed to get the shortest testors.

To reduce execution times, our proposal takes into account the density of 1's of the basic matrix, adapting the strategy to whether that matrix is more or less dense; which is intuitively related to the length of the shortest testors.

The proposed algorithm proved to be a viable option, since it finds all the shortest testors of a training sample and achieves it in a considerably shorter time than the modified variants of the algorithms that find them all. Minimum-length testors could be an option of support sets for decision-rule based algorithms.

Although deep learning based methods have been very popular in pattern recognition due to efficient formulas behind them [16], it is commonly known that parameters (such as batch size [17]) should be chosen carefully and training sets should be created carefully. Also, appropriate activation, loss and optimization functions should be chosen. However, there is not any clear way how to chose these parameers and functions in different cases. Therefore, feature selection is still an open issue and significant task in traditional and deep learning based methods.

REFERENCES

[1] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Syst. Appl.*, vol. 139, Jan. 2020, Art. no. 112824.

[2] E. Alba-Cabrera, S. Godoy-Calderon, and J. Ibarra-Fiallo, "Generating synthetic test matrices as a benchmark for the computational behavior of typical testor-finding algorithms," *Pattern Recognit. Lett.*, vol. 80, pp. 46–51, Sep. 2016.

[3] E. Alba-Cabrera, J. Ibarra-Fiallo, S. Godoy-Calderon, and F. Cervantes-Alonso, "YYC: A fast performance incremental algorithm for finding typical testors," in *Proc. Iberoamer. Congr. Pattern Recognit. (CIARP)*, Puerto Vallarta, Mexico, vol. 8827, 2014, pp. 416–423.

[4] Q. A. Al-Radaideh and M. A. Al-Abrat, "An arabic text categorization approach using term weighting and multiple reducts," *Soft Comput.*, vol. 23, no. 14, pp. 5849–5863, Jul. 2019.

[5] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Syst. Appl.*, vol. 116, pp. 147–160, Feb. 2019.

[6] B. Asadi and V. Varadharajan, "Towards a robust classifier: An MDL-based method for generating adversarial examples," 2019, *arXiv:1912.05945*. [Online]. Available: http://arxiv.org/abs/1912.05945

[7] A. Asaithambi and V. Valev, "Construction of all non-reducible descriptors," *Pattern Recognit.*, vol. 37, no. 9, pp. 1817–1823, Sep. 2004.

[8] A. Bar, A. Kumar, and P. S. V. S. Prasad, "Finding optimal rough set reduct with a search algorithm," in *Proc. 8th Int. Conf. Pattern Recognit. Mach. Intell. (PReMI)*, Tezpur, India, vol. 11941, Dec. 2019, pp. 317–327.

[9] A. Bar and S. Prasad, "Multiple reducts computation in rough sets with applications to ensemble classification," in *Proc. ICETIT*, in Lecture Notes in Electrical Engineering, vol. 605, New Delhi, India, Sep. 2019, pp. 449–461.

[10] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using UMAP," *Nature Biotechnol.*, vol. 37, no. 1, pp. 38–44, Jan. 2019.

[11] J. A. Carrasco-Ochoa and J. F. Martínez-Trinidad, "Feature selection for natural disaster texts classification using testors," in *Proc. Int. Data Eng. Automated Learn. (IDEAL)*, in Lecture Notes in Computer Science, vol. 3177, Exeter, U.K., Aug. 2004, pp. 424–429.

[12] A. Dmitriev, I. Zhuravlev, and F. Krendeliev, "About mathematical principles and phenomena classification," (in Russian), *Diskretni Analiz*, vol. 7, no. 1, pp. 3–15, 1966.

[13] N. E. I. Karabadji, I. Khelf, H. Seridi, S. Aridhi, D. Remond, and W. Dhifli, "A data sampling and attribute selection strategy for improving decision tree construction," *Expert Syst. Appl.*, vol. 129, pp. 84–96, Sep. 2019.

[14] J. Fang, H. Ouyang, L. Shen, E. R. Dougherty, and W. Liu, "Using the minimum description length principle to reduce the rate of false positives of best-fit algorithms," *EURASIP J. Bioinf. Syst. Biol.*, vol. 2014, no. 1, Dec. 2014, Art. No. 13.

[15] D. Fernandez, C. Gonzalez, D. Mozos, and S. Lopez, "FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images," *J. Real-Time Image Process.*, vol. 16, no. 5, pp. 1395–1406, Oct. 2019.

[16] E. Goceri, "Formulas behind deep learning success," in *Proc. Int. Conf. Appl. Anal. Math. Model.*, Istanbul, Turkey, Jun. 2018, pp. 1–4.

[17] E. Goceri and A. Gooya, "On the importance of batch size for deep learning," in *Proc. Int. Conf. Math.*, Istanbul, Turkey, Jun. 2018, pp. 1–6.

[18] M. Hansen and B. Yu, "Model selection and the principle of minimum description length," *J. Amer. Stat. Assoc.*, vol. 96, pp. 746–774, Jun. 2001.

[19] M. S. Lazo-Cortés, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and G. Sanchez-Diaz, "On the relation between rough set reducts and typical testors," *Inf. Sci.*, vol. 294, pp. 152–163, Feb. 2015.

[20] F. Li, Q. Zhu, and X. Lin, "Topic discovery in research literature based on non-negative matrix factorization and testor theory," in *Proc. Asia–Pacific Conf. Inf. Process.*, Shenzhen, China, Jul. 2009, pp. 266–269.

[21] A. Lias-Rodrìguez and G. Sanchez-diaz, "An algorithm for computing typical testors based on elimination of gaps and reduction of columns," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 27, no. 8, Dec. 2013, Art. no. 1350022.

[22] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. London, U.K.: Chapman & Hall, Jan. 2007.

[23] M. M. Mafarja and S. Mirjalili, "Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection," *Soft Comput.*, vol. 23, no. 15, pp. 6249–6265, Aug. 2019.

[24] J. I. Myung, D. J. Navarro, and M. A. Pitt, "Model selection by normalized maximum likelihood," *J. Math. Psychol.*, vol. 50, no. 2, pp. 167–179, Apr. 2006.

[25] M. R. Ortíz-Posadas, J. F. Martínez-Trinidad, and J. Ruíz-Shulcloper, "A new approach to differential diagnosis of diseases," *Int. J. Bio-Med. Comput.*, vol. 40, no. 3, pp. 179–185, Jan. 1996.

[26] I. Piza-Davila, G. Sanchez-Diaz, C. A. Aguirre-Salado, and M. S. Lazo-Cortes, "A parallel hill-climbing algorithm to generate a subset of irreducible testors," *Appl. Intell.*, vol. 42, no. 4, pp. 622–641, Jun. 2015.

[27] I. Piza-Davila, G. Sanchez-Diaz, M. S. Lazo-Cortes, and C. Noyola-Medrano, "Enhancing the performance of YYC algorithm useful to generate irreducible testors," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 32, no. 1, Jan. 2018, Art. no. 1860001.

[28] I. Piza-Davila, G. Sanchez-Diaz, M. S. Lazo-Cortes, and L. Rizo-Dominguez, "A CUDA-based hill-climbing algorithm to find irreducible testors from a training matrix," *Pattern Recognit. Lett.*, vol. 95, pp. 22–28, Aug. 2017.

[29] A. Pons-Porrata, R. Gil-García, and R. Berlanga-Llavori, "Using typical testors for feature selection in text categorization," in *Proc. 12 th Iberoamer. Congr. Pattern Recognit. (CIARP)*, in Lecture Notes in Computer Science, Valparaiso, Chile, vol. 4756, Nov. 2007, pp. 643–652.

[30] J. Quinlan, "The minimum description length and categorical theories," in *Proc. 11th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, Jul. 1994, pp. 233–241.

[31] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, Sep. 1978.

[32] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. S. Lazo-Cortés, "A new algorithm for reduct computation based on gap elimination and attribute contribution," *Inf. Sci.*, vol. 435, no. 1, pp. 111–123, Apr. 2018.

[33] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, M. Lazo-Cortés, C. Feregrino-Uribe, and R. Cumplido, "A fast hardware software platform for computing irreducible testors," *Expert Syst. Appl.*, vol. 42, no. 24, pp. 9612–9619, Dec. 2015.

[34] A. Rojas, R. Cumplido, J. Ariel Carrasco-Ochoa, C. Feregrino, and J. F. Martínez-Trinidad, "Hardware–software platform for computing irreducible testors," *Expert Syst. Appl.*, vol. 39, no. 2, pp. 2203–2210, Feb. 2012.

[35] J. Ruiz-Shulcloper, A. Soto, and A. Fuentes, "Characterization of the typical testor concept in terms of a notable set of columns," (in Spanish), *Revista Ciencias Matematicas*, vol. 1, pp. 123–134, 1980.

[36] G. Sanchez-Diaz and M. S. Lazo-Cortes, "CT EXT: An external escale algorithm for generated typical testors," in *Proc. 12th Iberoamer. Cong. Pattern Recognit. (CIARP)*, in Lecture Notes in Computer Science, Valparaiso, Chile, vol. 4756, Nov. 2007, pp. 506–514.

[37] G. Sanchez-Diaz, M. Lazo-Cortes, and I. Piza-Davila, "A fast implementation for the typical testor property identification based on an accumulative binary tuple," *Int. J. Comput. Intell. Syst.*, vol. 5, no. 6, pp. 1025–1039, Nov. 2012.

[38] G. Sanchez-Diaz, G. Diaz-Sanchez, M. Mora-Gonzalez, I. Piza-Davila, C. A. Aguirre-Salado, G. Huerta-Cuellar, O. Reyes-Cardenas, and A. Cardenas-Tristan, "An evolutionary algorithm with acceleration operator to generate a subset of typical testors," *Pattern Recognit. Lett.*, vol. 41, pp. 34–42, May 2014.

[39] Y. Santiesteban-Alganza and A. Pons-Porrata, "LEX: A new algorithm for calculating typical testors," (in Spanish), *Revista Ciencias Matematicas*, vol. 21, no. 2, pp. 85–95, 2003.

[40] A. Saxena, K. Saxena, and J. Goyal, "Hybrid technique based on DBSCAN for selection of improved features for intrusion detection system," in *Advances in Intelligent Systems and Computing*. Singapore: Springer, vol. 841, Jan. 2019, pp. 365–377.

[41] G. I. Sayed, A. E. Hassanien, and A. T. Azar, "Feature selection via a novel chaotic crow search algorithm," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 171–188, Jan. 2019.

[42] H. Si, J. Zhou, Z. Chen, J. Wan, N. N. Xiong, W. Zhang, and A. V. Vasilakos, "Association rules mining among interests and applications for users on social networks," *IEEE Access*, vol. 7, pp. 116014–116026, Aug. 2019.

[43] J. Sil and A. K. Das, "Variable length reduct vs. Minimum length reduct—A comparative study," *Procedia Technol.*, vol. 4, pp. 58–68, Feb. 2012.

[44] D. Singh and B. Singh, "Hybridization of feature selection and feature weighting for high dimensional data," *Appl. Intell.*, vol. 49, no. 4, pp. 1580–1596, 2019.

[45] M. Tsagris, V. Lagani, and I. Tsamardinos, "Feature selection for high-dimensional temporal data," *BMC Bioinf.*, vol. 19, no. 1, pp. 1–17, Dec. 2018.

[46] V. Valev and A. Asaithambi, "On computational complexity of non-reducible descriptors," in *Proc. 5th IEEE Int. Conf. Inf. Reuse Integr.*, Las Vegas, NV, USA, Oct. 2003, pp. 208–211.

[47] V. Valev and J. I. Zhuravlev, "Integer-valued problems of transforming the training tables in K-valued code in pattern recognition problems," *Pattern Recognit.*, vol. 24, no. 4, pp. 283–288, Jan. 1991.

[48] M. Wang, C. Wu, L. Wang, D. Xiang, and X. Huang, "A feature selection approach for hyperspectral image based on modified ant lion optimizer," *Knowl.-Based Syst.*, vol. 168, pp. 39–48, Mar. 2019.

[49] H. Zhou, Y. Zhang, Y. Zhang, and H. Liu, "Feature selection based on conditional mutual information: Minimum conditional relevance and minimum conditional redundancy," *Appl. Intell.*, vol. 49, no. 3, pp. 883–896, Mar. 2019.

IVÁN PIZA-DÁVILA received the B.S. degree in computer systems from the Instituto Tecnologico de Colima, Mexico, in 2000, and the M.S. and Ph.D. degrees in computer science from the Centro de Investigacion y de Estudios Avanzados del IPN, Mexico, in 2002 and 2006, respectively. He is currently a full-time Professor–Researcher with the Instituto Tecnológico y de Estudios Superiores de Occidente, Guadalajara, México. He has published several articles in journals and conference proceedings. His research interests include pattern recognition, machine learning, and parallel algorithms.

GUILLERMO SÁNCHEZ-DÍAZ received the B.S. and M.S. degrees in computer science from the Autonomous University of Puebla, Mexico, in 1995 and 1997, respectively, and the Ph.D. degree in computer science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. He is currently a full-time Professor–Researcher with the Universidad Autónoma de San Luis Potosí, México. He has published several articles in journals and conference proceedings. His research interests include pattern recognition, machine learning, and data mining. He has served regularly in the program committees of international conferences. He has also been a Reviewer for some international journals in his fields.

MANUEL S. LAZO-CORTÉS received the B.S. degree in mathematics and the Ph.D. degree in mathematical sciences from the University de las Villas, Cuba, in 1979 and 1994, respectively. He is currently a full-time Professor with the TecNM/Instituto Tecnológico de Tlalnepantla, México. He has published several articles in journals, chapter books, and conference proceedings. His research interests include pattern recognition and machine learning. He has served regularly in the program committees of international conferences. He has also been a Reviewer for some international journals in his fields.

IVÁN VILLALÓN-TURRUBIATES (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Centro de Investigación y de Estudios Avanzados (CINVESTAV), in 2007. He is currently a full-time Titular Professor and a Researcher with the Department of Electronics, Systems and Informatics, Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO), Tlaquepaque, Mexico. He is a Coordinator of the master's degree program in computer systems, from 2019 to 2022. His research is focused in applications of signal and image processing to multispectral and hyperspectral remote sensing data, and holds several publications in journals, international conferences, and research books on these topics.