IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 23, 2021, accepted September 8, 2021, date of publication September 13, 2021,
date of current version September 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3112385

RoPM: An Algorithm for Computing Typical Testors Based on Recursive Reductions of the Basic Matrix

JOEL PINO GÓMEZ[1], FIDEL ERNESTO HERNÁNDEZ MONTERO[1], JOEL CHARLES SOTELO[1], JULIO CÉSAR GÓMEZ MANCILLA[2], AND YENNY VILLUENDAS REY[3]

[1]Department of Telecommunications, Technological University of Havana (Cujae), Marianao, Havana 19390, Cuba
[2]Vibrations and Rotor Dynamic Laboratory, ESIME, Instituto Politécnico Nacional (IPN), Mexico City 07738, Mexico
[3]Centro de Innovación y Desarrollo Tecnológico en Cómputo, Instituto Politécnico Nacional (IPN), Mexico City 07700, Mexico

Corresponding authors: Yenny Villuendas Rey (yenny.villuendas@gmail.com) and Joel Pino Gómez (joelpinogomez@gmail.com)

ABSTRACT Feature selection plays an important role in pattern recognition and smart computing. The full set of typical testors constitutes a useful tool for solving feature selection problems, especially those problems in which the objects are described by both quantitative and qualitative features. However, finding the typical testors involves a high computational cost. That is why even the most efficient methods become unsuitable to solve some problems. In this work, a new algorithm was introduced in order to reduce the long runtimes involved in the search of typical testors. The performance of the proposed algorithm was evaluated by means of several tests, which use both real-world and simulation data. MATLAB and Java language on Eclipse SDK platform were used to build the simulation dataset and to perform the tests, respectively. The runtimes achieved by the proposed algorithm were significantly shorter than those obtained by fast-BR and GCreduct (the two fastest algorithms) mainly when the latter ones exhibited excessively long runtimes.

INDEX TERMS Algorithm, feature selection, runtimes, typical testors.

I. INTRODUCTION

Testor theory emerged together with the development of logical mathematical methods for the localization of faults in electrical circuits at the middle of the last century [1]. After that, only few years were needed for this theory to be extended to the solution of classical pattern recognition (PR) problems [2]. Testor theory has proved to be an important tool mainly when working with mixed (quantitative and qualitative) and incomplete data is required. In addition, its development is closely related to the advances achieved in the field of the Logical Combinatorial Pattern Recognition (LCPR) [3].

The most compact form in which a testor can be found is called typical testor (TT). The set of all TTs is a useful tool used mainly for feature selection tasks [4], [5] or clustering [6], [7]. TTs have been used successfully to determine the feature relevance in different supervised classification problems [8]–[15]. Likewise graph learning techniques have impacted on the current advances in digital signal and image processing (e.g., hyperspectral image analysis [16], [17]), typical testors have played an important role in those fields that require the smart processing of mixed and incomplete data.

However, the time required for computing the complete set of TTs can be excessively long in certain situations. This is due to the fact that finding all the typical testors in a given problem involves the evaluation of all sets of features, whereas the relationship between the number of sets of features and the number of features is exponential, as shown by the following expression:

$$C = 2^n - 1$$

where C is the number of set of features and n is the number of features employed to describe the problem. Then, in problems described by a large number of features, the search of all TTs might take long time intervals. Undoubtedly, this is a nonpolynomial problem that can lead to a high computational cost. All the notation included in the paper is provided in Appendix A.

Algorithms addressing the search of all TTs must run as faster as possible [18]–[24]. Various strategies have been

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Gyu Kim.

developed to reach this goal, for example, the application of hardware and software-hardware configurations [25], [26]. In addition, several algorithms have been proposed in order to compute only the minimum-length TTs [27], [28], not the set of all TTs.

Nevertheless, the use of the fastest algorithms, even when they run on powerful computers, does not ensure to solve complex problems in short time (indeed, runtimes could take weeks for some problems [27]).

In this paper, a new algorithm, named Recursive over the Possible Matrices, RoPM, was developed in order to diminish the runtime involved in TTs search procedures. The proposed algorithm performs binary operations over the possible reductions of the basic matrix. This matrix includes essential information resulting from the comparison of the objects involved in the problem.

The new RoPM algorithm allows for the computation of all TTs in short runtimes, regardless the complexity of the problem. In many situations, the proposed algorithm is capable to exhibit better performances than the fastest algorithms (fastBR and GCreduct [20], [21], [24]), especially when achieving a significant reduction of runtimes is required. In particular, GCreduct is an algorithm that allows for the search of reducts (from the rough set theory) and TTs [21], [24]. Reducts and TTs are concepts very close related to each other [21], [27]–[30].

This paper is organized as follows: Section II presents the concepts of the LCPR approach related to the testor theory. Section III provides a brief summary of main works performed in order to find TTs. In Section IV, the operation of the novel algorithm is described in detail. Section V presents a comparative study on the performance achieved by the proposed algorithm through the implementation of several tests. Section VI presents the conclusions and some future works.

II. MAIN CONCEPTS OF THE LCRP APPROACH

A PR problem can be described through intrinsic object features. The term object refers to the elements under study. The term feature represents the variables that describe the objects, usually features describe their main properties. Features can be expressed through variables either quantitative (age, height, weight) or qualitative (skin color, type of hair, religion). Since missing data and partial representations are often involved in some problems, incomplete descriptions should also be considered.

A class is formed by a set of objects that fulfill certain properties. In this paper, we consider each object belongs to one and only one class; that is, each object has just one class label.

An object representation is given by a specific set of features defined as a n-tuple of variables in the following form: $I(O) = (x1(O), \ldots, xn(O))$, where $xi(O) \in Mi, i = 1, \ldots, n$ is the value of the feature xi in the object O and Mi is the admissible set of values of xi.

In the area of supervised classification, a PR problem can be represented in matrix form. Firstly, a learning matrix (LM), where each row represents an object and each column represents a feature, is built up. A LM is comprised by the information of the object classification. This can be achieved by using, for example, subsets of rows for each class, or new columns indicating the membership to the classes, or any other choice [18]. Table 1 shows an example of a LM with only five objects grouped into two classes (K_1, K_2) and characterized through four features.

TABLE 1. A learning matrix (LM) example.

Object	x_1 Age	x_2 Sex	x_3 Education Level	x_4 Historical Records	Class
O_1	19	M	HS	0	K_1
O_2	32	F	BD	1	K_1
O_3	23	M	BD	1	K_1
O_4	28	M	MD	1	K_2
O_5	37	F	HS	0	K_2

An effective comparison between objects can be performed through the application of a different comparison criterion for each feature. A comparison criterion (CC) is defined as follows:

Definition 1. Let Li be a complete ordered set and Mi be the admissible set of values of xi, the function denoted as CC of the values of xi is defined as $Ci : Mi \times Mi \rightarrow Li$, such that:

$$CC_i(xi(O), xi(O)) = \min_{y \in Li}\{y\}$$

if CCi is a dissimilarity CC between values of xi, or

$$CC_i(xi(O), xi(O)) = \max_{y \in Li}\{y\}$$

if CCi is a similarity CC between values of xi for $i = 1, \ldots, n$.

Then, a CC can express the similarity (or dissimilarity) degree between any two values of a variable xi for $i = 1, \ldots, n$. Two examples of boolean dissimilarity CC are:

$$CC_1\,(xs(Oi), xs(Oj)) = \begin{cases} 0 & if\ xs(Oi) \neq\ xs(Oj) \\ 1 & otherwise \end{cases} \quad (1)$$

$$CC_2\,(xs(Oi), xs(Oj)) = \begin{cases} 0 & if\ xs(Oi),\ xs(Oj) \in Ap \\ 1 & otherwise \end{cases} \quad (2)$$

where $xs(Oi)$ and $xs(Oj)$ are the values of feature xs for objects Oi and Oj, respectively, and Ap is a set of values of the variable xs [5]. For example, the CC given by (2) can be used in order to compare the feature x_1 (age) in the example presented in Table 1. Three sets of values (Ap) are used in this example to compare the values of x_1: [under 20], [between 20 and 35], and [over 35]. The remaining features in the example given in Table 1 can be compared by means of (1).

A testor can be defined as the set of features by which all objects that belong to different classes can be differentiated according to the predefined CCs. In a subset τ of columns of

LM, if none row is similar to another row belonging to a different class (according to the boolean CC established for each feature), then the set of features τ is a testor [3], [18], [30]. Given the example presented in Table 1, the set given by features x_2 and x_3 constitutes a testor because it is not possible to find two similar objects that belong to different classes. On the other hand, the set given by features x_3 and x_4 is not a testor because the objects O_1 and O_5, which belong to different classes, are similar. Besides, it can be verified that the set of all testors is the following:

$$\{\{x_1, x_2, x_3, x_4\}, \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \{x_1, x_3\}, \{x_2, x_3, x_4\}, \{x_2, x_3\}\}$$

A testor is an irreducible testor (i.e., a TT) if it stops being a testor when any of its features is removed [5]. Given a testor comprised by the subset τ of columns of LM, if τ stops being a testor when any of its columns is removed, then τ is a TT [3, 18, 30]. For example, given the example presented in Table 1, the set given by features x_1 and x_3 is a TT. However, the set given by features x_2, x_3, and x_4 is not a TT because although when x_4 is removed, the remaining features, x_2 and x_3, represent a testor. Moreover, it can be verified that the set of all TTs is:

$$\{\{x_1, x_3\}, \{x_2, x_3\}\}$$

Another useful matrix, the binary comparison matrix (CM), can be also built up by applying pre-established binary comparison criteria for each feature and making all possible comparisons between the objects of different classes. Each row in a CM contains the result of the comparison between two objects [18]. As an example, Table 2 presents the CM corresponding to the LM presented in the Table 1. The values of feature x_1 were compared by means of (2), whereas (1) was used in order to compare the values of features x_2, x_3, and x_4.

TABLE 2. CM corresponding to the LM presented in Table 1.

Comparison	x_1	x_2	x_3	x_4
O_1 with O_4	1	0	1	1
O_1 with O_5	1	1	0	0
O_2 with O_4	0	1	1	0
O_2 with O_5	1	0	1	1
O_3 with O_4	0	0	1	0
O_3 with O_5	1	1	1	1

The first row of Table 2 is comprised by the boolean vector: [1, 0, 1, 1], which means that objects O_1 and O_4 are similar only for the feature x_2; according to the remaining features, these two objects are dissimilar. The next rows represent the results obtained by the other comparisons.

A useful definition related to the CM is the subrow.

Definition 2. Given any two rows of CM, denoted by *Rp* and *Rq*, *Rq* is a subrow of *Rp* if and only if:

1) in every column where *Rq* presents a '1', *Rp* also presents '1',
2) there is at least one column where *Rp* presents a '1' and *Rq* presents a '0'.

According to these statements, since:

- the second and sixth rows of the CM presented in Table 2 have'1s' at the first ($x1$) and second ($x2$) columns,
- the sixth row has '1s' at the third ($x3$) and fourth ($x4$) columns, and
- the second row has'0s' at such positions,

the second row (1 1 0 0) is a subrow of the sixth row (1 1 1 1). Similarly, it can be seen that the fifth row is a subrow of the first, third, fourth, and sixth rows.

The subrows are also useful for determining the basic rows in CM.

Definition 3. Let *Rb* be a row of CM. If there is not another row being a subrow of *Rb*, then *Rb* is a *basic row* (BR). The matrix formed by the different BRs of a CM is called as *basic matrix* (BM).

Since the only rows with no subrows in the CM presented in Table 2 are the second and fifth rows, they are the only BRs. The resulting BM is:

$$bm = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

It is recommended to obtain the TTs from the BM rather than from the previously mentioned matrices [5]. Usually, the BM is comprised by a number of rows and a density of '1s' that are smaller than those of the CM and the LM, thus the work with the BM is an advantage. Indeed, the starting step of the fastest algorithms for TTs computing involves the work with the BM [18]–[21].

A given set of columns of BM, τ, is said to form a covering if it is not comprised by any row with all '0' elements [20]. Accordingly, the first and second columns, c_1 and c_2, of the BM obtained in the example, are not a covering because the second row they form is an all-0 row. On the contrary, the first, second and third columns, c_1, c_2, and c_3, are a covering since they do not form any all-0 row. This way, it can be verified that the set of covering of such a BM are:

$$\{\{c_1, c_2, c_3, c_4\}, \{c_1, c_2, c_3\}, \{c_1, c_3, c_4\}, \{c_1, c_3\}, \{c_2, c_3, c_4\}, \{c_2, c_3\}\}$$

It should be notice that every covering of the BM is a testor.

Definition 4. Let *Rt* be a row of a BM and $\tau = [x1, x2, \ldots, xn]$ a set of features of the BM. Rt is said to be a *typical row* (TR) of the feature *xi* (such that $1 \leq i \leq n$) with respect to τ, if it has '1' at the column corresponding to *xi* and '0' at the remaining columns of τ.

Regardless the definition of TT and TR, a set of features τ is a TT if:

a) it is a covering
b) for each feature $xi \in \tau$ there is at least one TR of the feature *xi* with respect to τ.

The set of features τ that fulfills the condition b) is called as *sequence of compatible elements* (SCE) [20].

In case of the example's BM, the covering formed by columns c_1, c_2, and c_3 does not comply with condition b) since a TR cannot be found for features 1 and 2. It can be verified that condition b) is fulfilled by only the coverings:

$$\{\{c_1, c_3\}, \{c_2, c_3\}\}$$

It should be noticed that this SCE coincides with the set of TTs obtained previously.

The next section provides a brief summary of the main works performed in order to find TTs.

III. RELATED WORK

Reducing the long runtimes that the most efficient algorithms still require to solve certain problems is the main motivation for the development of new methods of searching TTs. The algorithms developed for computing TTs can be classified in either external or internal scale algorithms [5]. External scale algorithms use lists of feature sets (candidates) to find TTs. The differences between the several external scale algorithm implementations reside mainly in the strategy used for the candidate formation, which involves the work with a number of candidates as lower as possible. With regard to the internal scale algorithms, they use the aforementioned matrices and are focused on finding the conditions that ensure the existence of the TTs without using lists of candidates.

Up to now, the fastest algorithms are external scale algorithms. Among them, the lowest runtimes correspond to those algorithms that select only the features that satisfy specific properties [18]–[21]. Some algorithms select those features that ensure that the candidate meets the testor property at first place, and after that, they verify the typicity property [19]. Nevertheless, the first step of other algorithms is to select those features that guarantee the typicity of the candidate, and afterword, they verify testor property [18], [20], [21]. Two techniques are recognized as the fastest techniques for computing TTs: the algorithms fast-BR and GCreduct [20], [21], [24]. However, the use of these algorithms does not ensure to reach the solution of complex problems in short time (indeed, the work based on powerful personal computers can take days or weeks.) In fact, according to [24], these algorithms take several tens of seconds to solve problems that do not include a large number of features.

The main motivation for developing a new algorithm is to help to reduce the excessive runtimes that the most efficient algorithms still require to solve the most complex problems. The main idea and novelty of this proposal is to use the structure of the basic matrix to construct the candidates list on the fly, while successive reductions of this matrix are performed. The aim of this method is to reduce the number of candidates, the cost involved in testing testor property and typicity, and the runtimes required to solve complex problems.

A detailed description of the proposed method is presented in next section, where an example based on a simple basic matrix is developed in order to facilitate understanding. At this section, a second example constituting a comparison between the number of candidates verified by the proposed algorithm and those verified by the algorithms fast-BR and GCreduct is presented.

IV. RoPM: A NEW ALGORITHM FOR TTs COMPUTING

In this paper, a new algorithm named Recursive over the Possible Matrices (RoPM) is proposed for computing TTs. This algorithm is an external scale algorithm that works in a recursive way. The algorithm rearranges the BM on each iteration, and removes the rows that do not provide useful information and could overload the next iteration. Then, the algorithm performs logical operations that ensure the testors arising. Whenever a testor is found, typicity verification is performed using the original BM. Fig. 2 shows a diagram of algorithm RoPM.

The main function (Fig. 1) reads the file comprised by the input parameter (BM), initializes the input variables, runs the timer, calls the recursive method ***RoPM***, stops the timer when the recursive method ends, and display the results. Two global variables are defined in the main function: the arrays *bm* and *tt*. The matrix *bm* stores the input BM and is used in order to check the typicity of both the candidates and the testors. The matrix *bm* remains unaltered during all iterations. The matrix *tt*, initially empty, will be used to store the TTs. Other two variables are defined: the arrays *mt* and *masc*. The matrix *mt* also stores initially the input BM, but unlike *bm*, *mt* is modified at each iteration. Some features selected from the first row of *mt* determine the candidates. The latter are stored in the vector *masc*, initially empty, which is modified at each iteration as the candidates are updated.

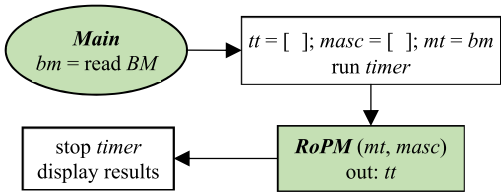

FIGURE 1. Main Function. Flow Diagram.

The recursive main function ***RoPM*** (see Fig. 2) essentially includes eight internal methods: ***And***, ***SCEr***, ***UpdateTT***, ***SetIndexTo0***, ***All0Row***, ***OrdMat***, ***SCEm***, and ***UpdateMT***.

The BM of the example developed in Section II will be used in order to explain how such methods work. This BM is expressed as follows:

$$bm = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

During the algorithm explanation, although some of the parameters used by the methods, for example, *mt*, *tt*, *masc*, and *ind_m*, are binary they will appear in function of the indices of the '1s'. This allows for a more suitable representation of the way the RoPM algorithm works. Then, the initial values of the input parameters for the algorithm, iteration #1, are:

$$mt = \begin{bmatrix} 1 & 2 \\ 3 & \end{bmatrix} masc = []$$

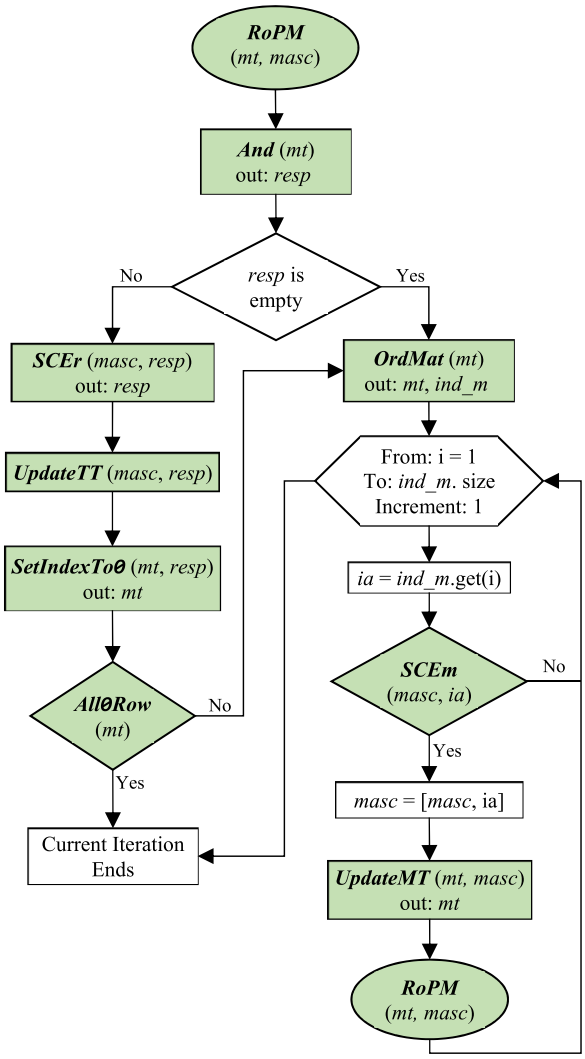

FIGURE 2. Algorithm RoPM. Flow Diagram.

- **Iteration #1:**

Firstly, method ***And*** is executed in order to obtain the indices of the all-ones columns of matrix *mt*(vector *resp*). For the current example, the vector *resp*, obtained at iteration #1, is empty because the same index is not included in all rows of *mt*.

$$resp = []$$

Since the vector *resp* is empty, the method ***OrdMat*** is called. The method ***OrdMat*** determines which row will be the first row in the matrix *mt*. The idea is that this position is occupied by the row with the lowest number of '1s'. If the first row has not the lowest number of '1s', then it is switched with the row with the lowest number of '1s'. Then, the output matrix of ***OrdMat***, that is, *mt*, at iteration #1 is:

$$mt = \begin{bmatrix} 3 & \\ 1 & 2 \end{bmatrix}$$

However, if several rows have the same lowest number of '1s', then, the total number of '1s' of the columns at the positions of the '1s' of these rows, is computed for each of such rows. The row yielding the highest result will occupy the first position in matrix *mt*.

The other output of method ***OrdMat***, the vector *ind_m,* is comprised by the indices of the positions of the '1s' in the resulting first row of matrix *mt*. Such indices are arranged in descending order, according to the number of '1s' in the column identified by the corresponding index. Thus, at iteration #1 the vector *ind_m* is:

$$ind\_m = [3]$$

At the end of method ***OrdMat***, a cycle is executed and repeated as many times as the length of the vector *ind_m*. For the current example, iteration #1 runs this loop only one time, since the vector *ind_m* is comprised by only one index. Inside this loop, the method ***SCEm*** will verify the typicity of the feature set defined by both the vector *masc* and the current index of vector *ind_m*. This procedure is always applied to the matrix *bm*. Since the vector *masc* is null and the vector *ind_m* has just one index at iteration #1, the method ***SCEm*** works with only one column of the matrix *bm*(the third column):

$$mct(:, 3) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The output of method ***SCEm*** is "*true*" at iteration #1, which indicates that the typicity condition is fulfilled.

Whenever the typicity condition is fulfilled, this index of the vector *ind_m* is included in the vector *masc*. In case of the typicity condition is not fulfilled, the analysis is repeated for a new index of vector *ind_m*. When this cycle ends, the current iteration also ends. Thus, at the end of iteration #1 the vector *masc* is:

$$masc = [3]$$

Afterwards, the method ***UpdateMT*** is called in order to perform the following actions: (a) removing the rows of matrix *mt* that have '1' at the position given by such an index of vector *ind_m;* (b) turning into all-zeros columns the columns of matrix *mt* at the positions of the indices that were included in vector *masc* during previous cycles. Since at this stage of iteration #1, just one index has been included in vector *masc*, the first action is the only applied to matrix *mt*. Hence, at the end of this iteration, the matrix *mt* is:

$$mt = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

In this case, the application of ***UpdateMT*** resulted in a new matrix *mt* with one row less. At each iteration, successive reductions of *mt* are performed by working with both methods ***UpdateMT*** and ***SetIndexTo0*** (the latter is explained below). These reductions occur in matrix *mt* as many times as possible. That is why the proposed algorithm was named *Recursive over the Possible Matrices* (RoPM).

Iteration #1 ends when a new iteration of the method ***RoPM*** is called. For the iteration #2, the inputs parameters are:

$$mt = \begin{bmatrix} 1 & 2 \end{bmatrix} masc = [3]$$

- **Iteration #2:**

The method ***And*** is executed and the result (vector *resp*) is:

$$resp = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

which matches the only row included in matrix *mt*, as expected. Since the vector *resp* is not empty, the method ***SCEr*** is called.

The method ***SCEr*** verifies the typicity of the feature sets defined by both the vector *masc* and each index of vector *resp*. This procedure always runs on the matrix *bm*. At this iteration and according to the features included in both the vector *masc* and the vector *resp*, the method ***SCEr*** evaluates two subsets of columns of matrix *bm*: the subset given by the third and the first column, and the subset given by the third and the second column, that is:

$$bm(:, [3\ 1]) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad bm(:, [32]) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

At this iteration, the method ***SCEr*** performs the corresponding verification which reveals that both sets of features fulfill the typicity condition. Thus, vector *resp* includes two indices at the method ***SCEr*** output:

$$resp = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

Each index in vector *resp* at the method ***SCEr*** output, as well as the indices of vector *masc*, form a new TT that is stored (***UpdateTT***). Then, the TTs are updated at iteration #2, which yields the following matrix *tt*:

$$tt = \begin{bmatrix} 3 & 1 \\ 3 & 2 \end{bmatrix}$$

The '1s' in the columns of matrix *mt*, which are identified by the indices of vector *resp*, are switched to '0s' (***SetIndexTo0***). Hence, at iteration #2, the matrix *mt* at the method ***SetIndexTo0*** output is:

$$mt = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Then, the method ***All0Row*** is executed. This method makes the current iteration end because matrix *mt* includes a null row vector. If matrix *mt* does not include any null row vector, then the method ***OrdMat*** is called.

At this stage, since every preceding iteration has already been completed, the method ***RoPM*** ends as well. Finally, according to the matrix *tt,* the resulting set of TTs is:

$$TT = \begin{bmatrix} x1 & x3 \\ x2 & x3 \end{bmatrix}$$

An advantage of the algorithm RoPM is the low number of candidates that this algorithm works with. This can be revealed, for example, through a BM from [21], denoted here as *BMa*, and defined as:

$$BMa = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

It should be noticed that the algorithm fast-BR removes the repeated columns from the BM. Then, columns #4 and #5 are removed from *BMa*, resulting on the following BM:

$$BMr(\text{fast\_BR}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

In this case, the pairs of columns #2 and #4, and #3 and #5 of *BMa* are represented in matrix *BMr* by the columns #2 and #3, respectively. Besides, columns #6 and #7 of *BMa* are represented in matrix *BMr* by the columns #4 and #5, respectively.

Table 3 presents the candidates that are verified by the algorithms GCreduct, fast-BR and RoPM. The TTs are highlighted in bold-type text. The candidate #11 of the algorithm fast-BR represents a pseudo-TT, since it comes from the *BMr*. Then, the TTs are computed from this pseudo-TT.

As shown in Table 3, GCreduct evaluated 24 subsets, fast-BR evaluated 12 subsets plus the computation of TTs from the pseudo-TT, and the proposed RoPM evaluated just 9 subsets. These results reveal that the number of candidate subsets evaluated by fast-BR and GCreduct algorithms is higher than the number of candidates evaluated by the RoPM algorithm. However, it should be noticed that processing fewer candidates does not necessarily guarantee that an algorithm is the fastest; other important factors, such as the way by which the testor and typicity properties are verified, determine the runtime of an algorithm as well.

V. PERFORMANCE EVALUATION

Several experiments were implemented for evaluating the performance of the RoPM algorithm. The experiments were aimed to evaluate its with respect the two fastest algorithms: fast-BR and GCreduct [24]. The comparison with the recent algorithms presented in [27], [28] was not performed because they do not deliver the full set of TTs. The tests were conducted on a laptop Alienware, model 17R3, Intel(R) Core (TM) i7-6700HQ CPU @2.60 GHZ (8 CPUs) processor, 16 GB RAM, Windows 10 Pro 64 bits operating system. The algorithms were programmed in Java through the software Eclipse SDK, version 4.13.

A. TESTS WITH REAL-WORLD DATASETS

The first test was carried out by using ten BMs available in the machine learning repository of the Informatics Sciences University (UCI), from Havana, Cuba [31]. Such BMs come from true study cases and are widely used on different works on Pattern Recognition, artificial intelligence and machine learning. Table 4 shows the name and the search space of each BM, as well as the candidates solutions evaluated by the algorithms fast-BR, GCreduct and RoPM. The lowest numbers of candidates evaluated are highlighted in bold type text.

TABLE 3. Candidates verified by each algorithm.

#	GCreduct	fast-BR	RoPM
1	$\{x_1\}$	$\{x_1\}$	$\{x_1\}$
2	$\{x_1\ x_2\}$	$\{x_1\ x_2\}$	$\{x_1\ x_7\}$
3	$\{x_1\ x_2\ x_3\}$	$\{x_1\ x_3\}$	$\{x_1\ x_7\ x_6\}$
4	$\{x_1\ x_2\ x_3\ x_4\}$	$\{x_1\ x_4\}$	$\{x_1\ x_7\ x_6\ x_2\}$
5	$\{x_1\ x_2\ x_3\ x_5\}$	$\{x_1\ x_5\}$	$\{\boldsymbol{x_1\ x_7\ x_6\ x_2\ x_3}\}$
6	$\{x_1\ x_2\ x_3\ x_6\}$	$\{x_1\ x_2\ x_3\}$	$\{\boldsymbol{x_1\ x_7\ x_6\ x_2\ x_5}\}$
7	$\{\boldsymbol{x_1\ x_2\ x_3\ x_6\ x_7}\}$	$\{x_1\ x_2\ x_4\}$	$\{x_1\ x_7\ x_6\ x_4\}$
8	$\{x_1\ x_2\ x_4\}$	$\{x_1\ x_2\ x_5\}$	$\{\boldsymbol{x_1\ x_7\ x_6\ x_4\ x_3}\}$
9	$\{x_1\ x_2\ x_5\}$	$\{x_1\ x_2\ x_3\ x_4\}$	$\{\boldsymbol{x_1\ x_7\ x_6\ x_4\ x_5}\}$
10	$\{x_1\ x_2\ x_5\ x_6\}$	$\{x_1\ x_2\ x_3\ x_5\}$	
11	$\{\boldsymbol{x_1\ x_2\ x_5\ x_6\ x_7}\}$	$\{\boldsymbol{x_1\ x_2\ x_3\ x_4\ x_5}\}$	
12	$\{x_1\ x_3\}$	$\{x_2\}$	
13	$\{x_1\ x_3\ x_4\}$		
14	$\{x_1\ x_3\ x_4\ x_5\}$	pseudo-TT:	
15	$\{x_1\ x_3\ x_4\ x_6\}$	$\{\boldsymbol{x_1\ x_2\ x_3\ x_4\ x_5}\}$	
16	$\{\boldsymbol{x_1\ x_3\ x_4\ x_6\ x_7}\}$	⇩	
17	$\{x_1\ x_3\ x_5\}$	original-TTs:	
18	$\{x_1\ x_3\ x_6\}$	$\{\boldsymbol{x_1\ x_2\ x_3\ x_6\ x_7}\}$	
19	$\{x_1\ x_3\ x_6\ x_7\}$	$\{\boldsymbol{x_1\ x_2\ x_5\ x_6\ x_7}\}$	
		$\{\boldsymbol{x_1\ x_3\ x_4\ x_6\ x_7}\}$	
20	$\{x_1\ x_4\}$	$\{\boldsymbol{x_1\ x_4\ x_5\ x_6\ x_7}\}$	
21	$\{x_1\ x_4\ x_5\}$		
22	$\{x_1\ x_4\ x_5\ x_6\}$		
23	$\{\boldsymbol{x_1\ x_4\ x_5\ x_6\ x_7}\}$		
24	$\{x_2\}$		
Total	**24**	**12**	**9**

Table 4 reveals that RoPM evaluates the lowest number of candidates in all cases. Indeed, this number of candidates is always much lower than the number of candidates evaluated by the algorithms fast-BR and GCreduct. However, as mentioned in Section IV and presented in Table 5, this does not necessarily mean that RoPM is the fastest at all. The Table 5 shows the runtimes achieved by the three algorithms for these BMs.

In Table 5, the density of '1s' (rate of the number of '1s' and the number of BM items), the BM order (number of rows per number of columns), the number of TTs found, and runtimes, are presented. The lowest runtimes per test are highlighted in bold type text. Tests were repeated several times because in many cases the runtimes were very short

TABLE 4. Number of candidates evaluated. UCI BMs.

	Datasets		Candidates Evaluated		
#	Name	S. space	fast-BR	GCreduct	RoPM
1	Connect-4	$4.40\text{x}10^{12}$	$9.87\text{x}10^{8}$	$1.33\text{x}10^{9}$	$\mathbf{4.50\text{x}10^{2}}$
2	QSAR-bg	$2.20\text{x}10^{12}$	$3.66\text{x}10^{6}$	$7.51\text{x}10^{6}$	$\mathbf{4.80\text{x}10^{2}}$
3	Diabetes	$5.63\text{x}10^{14}$	$8.24\text{x}10^{7}$	$1.42\text{x}10^{8}$	$\mathbf{1.31\text{x}10^{5}}$
4	Dermatol	$1.72\text{x}10^{10}$	$2.31\text{x}10^{7}$	$1.47\text{x}10^{8}$	$\mathbf{1.93\text{x}10^{5}}$
5	Flags	$5.37\text{x}10^{8}$	$7.07\text{x}10^{6}$	$1.19\text{x}10^{7}$	$\mathbf{4.00\text{x}10^{4}}$
6	Sponge	$3.52\text{x}10^{13}$	$3.59\text{x}10^{5}$	$1.47\text{x}10^{7}$	$\mathbf{1.30\text{x}10^{4}}$
7	Student-p	$4.29\text{x}10^{9}$	$2.06\text{x}10^{8}$	$9.97\text{x}10^{8}$	$\mathbf{1.60\text{x}10^{6}}$
8	Student-m	$4.29\text{x}10^{9}$	$1.21\text{x}10^{8}$	$6.37\text{x}10^{8}$	$\mathbf{1.25\text{x}10^{6}}$
9	Lung-cr	$7.21\text{x}10^{16}$	$8.41\text{x}10^{7}$	$2.22\text{x}10^{9}$	$\mathbf{5.77\text{x}10^{6}}$
10	Cylinder	$5.50\text{x}10^{11}$	$1.21\text{x}10^{6}$	$2.15\text{x}10^{7}$	$\mathbf{3.70\text{x}10^{4}}$

TABLE 5. Algorithm performance. UCI BMs.

	Datasets			Algorithms Performance		
#	Den.	BM order	TT	Time (s)		
				f-BR	GCr	RoPM
1	0.04	406x42	35	162.8	40.11	**0.045**
2	0.08	40x41	256	0.268	0.181	**0.005**
3	0.27	2212x49	77349	20.96	17.60	**2.463**
4	0.34	1103x34	112708	3.786	11.52	**1.318**
5	0.35	390x29	23543	0.796	0.601	**0.201**
6	0.38	68x45	10992	0.046	0.411	**0.026**
7	0.41	8158x32	851584	**114.9**	770.8	157.6
8	0.43	6904x32	679121	**57.61**	401.7	102.8
9	0.46	237x56	4183355	**7.302**	108.4	8.157
10	0.50	1147x39	23534	**0.239**	3.011	0.351

and then, no accurate conclusions could be made. When QSAR-Biodeg, Diabetes, Dermatology, Flags, Sponge, and Cylinder datasets were used, the algorithms were applied ten times. When Connect-4, Student-por, Student-mat, and Lung-cancer datasets were used, the algorithms were applied five times. The runtimes presented in Table 5 are the mean values of the execution times incurred by each algorithm, per dataset.

Fig. 3 shows a bar chart with the performance achieved by the three algorithms. In this figure, horizontal axis corresponds to the BM identification number and the vertical axis corresponds to each average runtime expressed as the percentage of the sum of the average runtime attained by the three algorithms.

Both Table 5 and Fig. 3 reveal that fast-BR was the fastest algorithm for four BMs, it was the second fastest algorithm for two BMs, and it was the slowest algorithm for the remaining four BMs. GCreduct was the second fastest algorithm for four BMs, and it exhibited the worst results for the resting six BMs. The algorithm RoPM was the fastest one for six

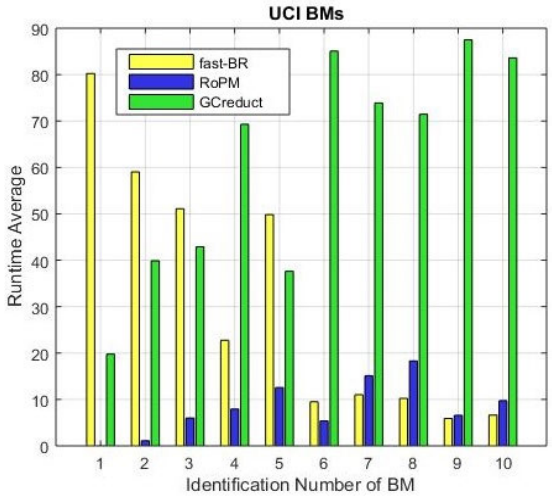

FIGURE 3. Algorithm Performance in Function of the Average Runtimes. UCI BMs.

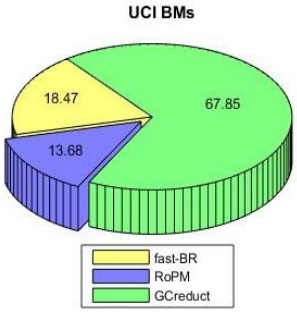

FIGURE 4. Algorithm Performance in Function of the Total Average Runtimes. UCI BMs.

BMs, it achieved the second most outstanding times for the remaining four BMs.

Fig. 3 also shows that when the algorithm RoPM did not achieve the best performance, it exhibited a low percentage of average runtime; actually, this algorithm achieved a percentage of average runtime below the 20% for all BMs. The algorithms fast-BR and GCreduct obtained percentage of average runtime below the 20% for five and one BMs, respectively. With regard to the highest percentages, two algorithms, GCreduct and fast-BR, exhibited percentages of average runtime above the 50% for six and three BMs, respectively.

Fig. 4 shows the total average runtime (sum of the average runtimes from the work with the ten BMs) of each algorithm as a percentage of the sum of the total average runtimes achieved by the three algorithms. The shortest average runtime, 13.68 %, was achieved by the algorithm RoPM; the algorithms fast-BR and GCreduct achieved the 18.47 % and 67.85 %, respectively.

Nevertheless, such results do not allow for assessing the conditions that lead an algorithm to reach the solution of a problem in the shortest runtime. The next section is a deeper analysis of the performance behavior of the three algorithms under different conditions of application.

TABLE 6. Number of TT. in UD BMs.

Typical Testors						
Density	Dimension					
	30x25	40x30	60x40	100x45	500x50	700x55
0.05	5	11	2313	7401	$2x10^4$	$1x10^5$
0.10	500	1048	$1x10^6$	$1x10^6$	$1x10^7$	$1x10^8$
0.15	1307	7998	$2x10^6$	$1x10^6$	$6x10^7$	$3x10^8$
0.20	1063	4029	$3x10^6$	$5x10^5$	$2x10^7$	$1x10^8$
0.25	1002	1002	$2x10^6$	$6x10^5$	$2x10^7$	$2x10^8$
0.30	1351	3583	$1x10^6$	$5x10^5$	$1x10^7$	$7x10^7$
0.35	2317	6949	$7x10^5$	$2x10^5$	$1x10^7$	$3x10^7$
0.40	1437	3449	$4x10^5$	$1x10^5$	$4x10^6$	$1x10^7$
0.45	1135	3350	$2x10^5$	$1x10^5$	$3x10^6$	$1x10^7$
0.50	1149	2373	$1x10^5$	$1x10^5$	$1x10^6$	$6x10^6$
0.55	1296	2306	$7x10^4$	$7x10^4$	$1x10^6$	$3x10^6$
0.60	737	1550	$4x10^4$	$3x10^4$	$6x10^5$	$1x10^6$
0.65	650	2095	$2x10^4$	$3x10^4$	$5x10^5$	$1x10^6$
0.70	617	1318	7200	$2x10^4$	$2x10^5$	$7x10^5$
0.75	490	1269	4940	$1x10^4$	$1x10^5$	$3x10^5$
0.80	248	491	3603	6966	$6x10^4$	$1x10^5$
0.85	243	425	2117	4640	$2x10^4$	$6x10^4$
0.90	213	346	905	1533	$1x10^4$	$2x10^4$
0.95	300	423	616	924	2400	9831

B. TESTS USING UNIFORM-DISTRIBUTED BASIC MATRICES

In this section, the results obtained from the implementation of new tests, which could lead to a better performance characterization of the algorithms, are presented. Specifically, such results were achieved by using uniform-distributed (UD) BMs. The BMs were built up by means of the software MATLAB. The work with six different BM dimensions, 30×25, 40×30, 60×40, 100×45, 500×50, and 700×55, was performed. 114 BMs (19 matrices per dimension) having different density of '1s', from 0.05 to 0.95 with step 0.05, were built up.

Table 6 shows the number of TT obtained for each matrix. In this table the largest numbers of TT are more frequently obtained at low densities and high dimensions. In cases of the number of TTs exceeded ten thousand, this number was rounded by defect and expressed in scientific notation.

Table 6 reveals that the highest amounts of TTs of all dimensions are found at densities of '1s' between 0.10 and 0.25.

Tables 7, 8, and 9 show the runtimes (in seconds) achieved by the three algorithms for BMs with densities within the intervals: [0.05 0.25], [0.3 0.6], and [0.65 0.95], respectively. In the first columns of these tables, the BM dimensions 30×25, 40×30, 60×40, 100×45, 500×50, and 700×55

TABLE 7. Runtimes in UD BMs. Density: [0.05 0.25].

		Density:				
		0.05	0.10	0.15	0.20	0.25
Dim.	Alg.	Time (s)				
1	f-BR	0.121	0.022	0.009	**0.004**	**0.003**
	GCr	**0.0001**	0.015	0.009	**0.004**	0.004
	RoPM	0.001	**0.004**	**0.005**	**0.004**	0.004
2	f-BR	18.406	0.029	0.043	**0.010**	**0.003**
	GCr	0.765	0.025	0.090	0.022	0.005
	RoPM	**0.002**	**0.008**	**0.018**	0.013	0.005
3	f-BR	67.449	261.08	54.542	25.277	6.641
	GCr	23.657	576.90	642.75	149.09	128.92
	RoPM	**0.025**	**2.299**	**3.744**	**5.429**	**3.689**
4	f-BR	5371.3	486.19	37.517	2.470	1.578
	GCr	3964.4	521.86	102.72	5.526	5.673
	RoPM	**0.116**	**3.203**	**3.869**	**1.225**	**1.300**
5	f-BR	$>12x10^4$	51914.2	11227.8	736.69	299.43
	GCr	$>12x10^4$	37802.1	15185.7	1301.4	773.24
	RoPM	**1.198**	**145.005**	**638.507**	**298.819**	**273.55**
6	f-BR	$>17x10^4$	$>12x10^5$	77603.8	6398.64	**3271.6**
	GCr	$>17x10^4$	$>12x10^5$	130715.8	11477.4	10582.9
	RoPM	**11.154**	**3821.18**	**4813.03**	**2612.67**	3657.98

TABLE 8. Runtimes in UD BMs. Density: [0.3 0.6].

		Density:						
		0.30	0.35	0.40	0.45	0.50	0.55	0.60
Dim.	Alg.	Time (s)						
1	f-BR	**0.003**	**0.003**	**0.002**	**0.002**	**0.002**	**0.002**	**0.001**
	GCr	**0.003**	0.005	**0.002**	**0.002**	**0.002**	**0.002**	**0.001**
	RoPM	0.004	0.006	0.004	0.004	0.004	0.003	0.002
2	f-BR	**0.005**	**0.009**	**0.004**	**0.004**	**0.003**	**0.003**	0.003
	GCr	0.009	0.014	0.006	0.005	**0.003**	0.010	**0.002**
	RoPM	0.009	0.012	0.007	0.007	0.006	0.004	0.004
3	f-BR	2.135	**0.981**	**0.404**	**0.164**	**0.079**	**0.044**	**0.025**
	GCr	23.38	8.628	3.225	1.512	0.566	0.166	0.081
	RoPM	**1.894**	1.251	0.829	0.395	0.222	0.127	0.081
4	f-BR	**0.783**	**0.297**	**0.135**	**0.091**	**0.059**	**0.039**	**0.021**
	GCr	4.793	1.225	2172	0.323	2124	0.165	0.061
	RoPM	1.086	0.535	0.296	0.229	0.201	0.123	0.064
5	f-BR	**55.21**	**33.27**	**8.904**	**5.284**	**1.950**	**1.229**	**0.566**
	GCr	167.5	282.3	36.83	35.64	6.594	10.08	2.605
	RoPM	117.4	82.17	33.28	21.81	10.34	6.152	3.170
6	f-BR	**500.0**	**147.7**	**48.18**	**23.76**	**10.79**	**5.193**	**2.207**
	GCr	$3x10^5$	511.1	158.3	92.74	39.04	23.37	11.09
	RoPM	$1x10^3$	439.5	192.8	110.3	55.87	27.82	14.04

are denoted by numbers from 1 to 6 respectively. Several executions per matrix were performed for the shortest runtimes, in order to ensure the reliability of the results; in such cases the average runtime is presented. For example, in case of 30 × 25 matrices, each algorithm was executed 30 times per matrix and then, the average value was computed. The shortest runtime attained from the work with each BM is highlighted in bold type text.

Results presented in Table 7 reveal that the best performances for the density interval [0.05 0.25] were more often achieved by the algorithm RoPM. Specifically, RoPM stands out when both fast-BR and GCreduct incur their longest delays. In fact, runs of Fast-BR and GCreduct had to stop at 36, 48 or 345 hours in some cases (Table 7), due to their delay compared to the runtimes of algorithm RoPM.

Results presented in Tables 8 and 9 reveal that the algorithms fast-BR and GCreduct tends to reach the shortest runtimes for the density interval [0.3 0.95]. Specifically, the algorithm GCreduct stands out at low dimensions and high densities (see Table 9). However, the runtimes achieved by the three algorithms are in general shorter than one minute for density intervals showed in Tables 8 and 9. The runtimes become more significant only for BMs with dimension 700 × 55 and densities between 0.3 and 0.4. In particular, the algorithm GCreduct incur in a significative delay (more than 7 hours) for the BM with density 0.3 and dimension 700 × 55.

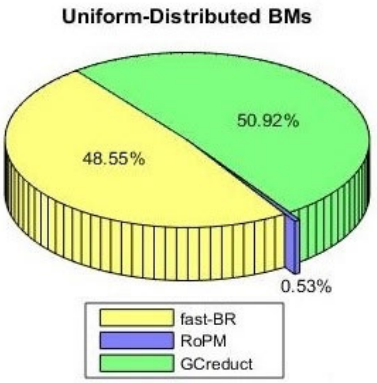

FIGURE 5. Algorithm Performance in Function of the Total Runtimes. UD BMs.

Fig. 5 shows the time taken by each algorithm to complete all the tests that used the UD BMs, expressed as a percentage of the sum of the time taken by the three algorithms. The algorithm RoPM only required 0.53 % of the total time to complete the tests, whereas algorithms fast-BR and GCreduct each resulted in about 50% of that time. This is essentially due to the fact that in cases of large dimension BMs with low density of '1s' (a problem with very high complexity), by far the best performance was achieved by the algorithm RoPM and the use of algorithms fast-BR and GCreduct became unsuitable.

TABLE 9. Runtimes in UD BMS. Density: [0.65 0.95].

		Density:						
		0.65	0.70	0.75	0.80	0.85	0.90	0.95
Dim.	**Alg.**	**Time (s)**						
1	f-BR	**0.001**	**0.001**	**0.001**	0.001	0.001	0.001	0.001
	GCr	**0.001**	**0.001**	**0.001**	**3x10^{-4}**	**3x10^{-4}**	**2x10^{-4}**	**2x10^{-4}**
	RoPM	0.002	**0.001**	**0.001**	0.001	8x10^{-4}	6x10^{-4}	6x10^{-4}
2	f-BR	0.003	**0.002**	0.002	0.002	0.002	0.002	0.001
	GCr	**0.002**	**0.002**	**0.001**	**0.001**	**0.001**	**3x10^{-4}**	**3x10^{-4}**
	RoPM	0.004	0.003	0.003	**0.001**	**0.001**	0.001	0.001
3	f-BR	**0.014**	**0.010**	**0.006**	**0.004**	0.004	0.003	0.003
	GCr	0.039	0.011	**0.006**	0.005	**0.003**	**0.001**	**0.001**
	RoPM	0.042	0.016	0.010	0.007	0.006	0.003	0.002
4	f-BR	**0.017**	**0.011**	**0.011**	**0.006**	0.006	0.004	0.004
	GCr	0.034	0.025	0.014	0.007	**0.003**	**0.002**	**0.001**
	RoPM	0.047	0.036	0.023	0.013	0.007	0.003	0.002
5	f-BR	**0.433**	**0.146**	**0.083**	**0.028**	**0.017**	**0.009**	0.007
	GCr	1.094	0.374	0.391	0.117	0.033	**0.009**	**0.002**
	RoPM	2.394	1.046	0.563	0.230	0.122	0.042	0.020
6	f-BR	**1.404**	**0.571**	**0.224**	**0.071**	**0.039**	**0.015**	0.010
	GCr	5.089	1.847	0.740	0.211	0.080	0.021	**0.006**
	RoPM	10.05	4.364	1.803	0.641	0.298	0.108	0.058

C. STATISTICAL ANALYSIS

This section presents a complementary analysis of the algorithm performance resulting from the tests carried out on the UD BMs (section VB). This work consisted in the statistical analysis of the positions occupied by the algorithms according to the average times consumed during the tests. Two nonparametric statistical tests were performed: Friedman [32] test and Holm's post-hoc [33] test. A significance level $\alpha = 0.05$ for a 95% of confidence was established for the hypothesis tests. Two hypotheses were established: a null hypothesis stating that there is no difference between the positions attained by the algorithms, and an alternative hypothesis stating that there are differences between such positions.

The results of the implementation of both tests (Friedman and Holm) are shown in Table 10. The p-values highlighted in bold type text represent the cases of the null hypothesis was rejected. Similarly, the rankings highlighted in bold type text identify the algorithm that achieved the first position as result of the tests.

For densities of '1s' between 0.05 and 0.20, by far the best performance was achieved by means of the RoPM algorithm (see Table 10). This density interval is the most critical since it is the interval that yields the highest number of TTs (see Table 6). In addition, RoPM algorithm achieved the second highest statistical ranking for densities of '1s' equal to 0.25, 0.30, 0.35, 0.55, and 0.95. Besides, according to the Holm's post-hoc test, no significant differences between the performance of RoPM and those of fast-BR and GCreduct are reached for densities of '1s' equal to 0.25, 0.30, 0.35, and 0.55.

TABLE 10. Statistical analysis of the algorithm performance.

	Friedman test				Holm's post-hoc test			
		ranking				unadjusted p-value		
Density	**p-value**	**f-BR**	**GCr**	**RoPM**	**≤**	**f-BR vs RoPM**	**GCr vs RoPM**	**GCr vs f-BR**
0.05	**0.0155**	2.833	2.000	**1.167**	0.025	**0.0039**	0.1489	0.1489
0.10	**0.0107**	2.583	2.417	**1.000**	0.050	**0.0061**	**0.0141**	0.7728
0.15	**0.0039**	2.083	2.917	**1.000**	0.025	0.0606	**0.0009**	0.1489
0.20	**0.0302**	1.833	2.833	**1.333**	0.025	0.3865	**0.0094**	0.0833
0.25	**0.0421**	**1.500**	2.833	1.667	0.017	0.7728	0.0433	0.0209
0.30	**0.0478**	**1.250**	2.667	2.083	0.025	0.1489	0.3123	**0.0141**
0.35	**0.0057**	**1.000**	2.833	2.167	0.025	0.0433	0.2482	**0.0015**
0.40	**0.0226**	**1.083**	2.417	2.500	0.050	**0.0141**	0.8852	**0.0209**
0.45	**0.0226**	**1.083**	2.417	2.500	0.050	**0.0141**	0.8852	**0.0209**
0.50	**0.0302**	**1.167**	2.167	2.667	0.025	**0.0094**	0.3865	0.0833
0.55	**0.0208**	**1.083**	2.583	2.333	0.025	0.0304	0.6650	**0.0094**
0.60	**0.0137**	**1.250**	1.833	2.917	0.025	**0.0039**	0.0606	0.3123
0.65	**0.0076**	**1.250**	1.750	3.000	0.025	**0.0024**	0.0304	0.3865
0.70	**0.0226**	**1.2500**	1.9167	2.8333	0.025	**0.0061**	0.1124	0.2482
0.75	**0.0372**	**1.4167**	1.7500	2.8333	0.025	**0.0141**	0.0606	0.5637
0.80	0.1298	1.5833	1.7500	2.6667	0.017	0.7728	0.0606	0.1124
0.85	0.1298	2.0000	1.4167	2.5833	0.017	0.3123	0.0433	0.3123
0.90	0.1298	2.0000	1.4167	2.5833	0.017	0.0433	0.0606	0.8852
0.95	**0.0107**	2.5833	**1.0000**	2.4167	0.050	0.7728	**0.0141**	**0.0061**

Although RoPM did not achieve the first position very often, this algorithm reached the best performance by far when long runtimes are required to compute the TTs. Then, this statistical analysis is confirming the results presented in previous sessions.

The following suggestions can be very useful for the solution of certain problems:

- To use RoPM for densities of '1s' lower than 0.2
- To use fast-BR for densities of '1s' between 0.25 and 0.75
- To use any algorithm for densities of '1s' higher than 0.75 and lower than 0.95.
- To use GCreduct for densities of '1s' higher or equal to 0.9

Consequently, and on the basis of the results achieved by the statistical tests, it is possible to state that the method RoPM is suitable to be used for basic matrices with low and high densities (less than 0.25 and between 0.75 and 0.95).

These suggestions must be followed carefully, because they have been stated as result of the tests performed on UD BMs. This may change for other BM distributions.

VI. CONCLUSION

This paper presents a new technique, the algorithm RoPM, which allows for the computation of all TTs.

Different tests carried out by means of BMs of the UCI machine learning repository [31] revealed that in many cases the algorithm RoPM is capable of performing better than the already-published two fastest algorithms: fast-BR and GCreduct.

In addition, other tests were implemented by means of randomly-generated UD BMs with different densities of '1s' and dimensions, in order to achieve a better characterization of the algorithm's performance. The results revealed the ability of the algorithm RoPM to find the TTs in short runtimes, regardless the complexity of the problem. Although RoPM was not the fastest algorithm in all cases, the results showed that it can significantly reduce the runtimes required for computing the TTs. In fact, the cases where both algorithms fast-BR and GCreduct became unsuitable were those when RoPM showed its greatest power. Thus, in cases of density of '1s' lower than 0.25, the use of RoPM is very advisable.

The RoPM algorithm constitutes a significant contribution to reduce the high computational costs associated with the nonpolynomial problem of finding the TTs.

APPENDIX A

Nomenclature used in the paper, in alphabetical order:

Notations	Meanings
Ap	Set of values of certain feature
BD	Bachelor Degree
BM	Basic Matrix
bm	Variable that stores the BM
BMs	Basic Matrices
BR	Basic Row
BRs	Basic Rows
C	Number of sets of features
CC	Comparison Criterion
CC_k	Comparison Criterion k
CCs	Comparison Criteria
CM	Comparison Matrix
Den.	Density of '1s'
F	Female
fast-BR	Faster Boolean Recursive algorithm
f-BR	fast-BR
GCr	GCreduct
GCreduct	Algorithm for computing all reducts based on Gap elimination and attribute Contribution
HS	High School
Kr	Class r
LCPR	Logical Combinatorial Pattern Recognition
LM	Learning Matrix
M	Male
masc	Variable that stores the candidates
MD	Master Degree
mt	Variable that stores the modified BM
n	Number of features
Oi	Object i
PR	Pattern Recognition
RoPM	Recursive over the Possible Matrices
Rp	Row p
S. space	Search Space
SCE	Sequence of Compatible Elements
TR	Typical Row
TT	Typical Testor
tt	Variable that stores the TTs
TTs	Typical Testors
UCI	Informatics Sciences University of Cuba
UD BMs	Uniform-distributed BMs
x_j	Feature j
τ	Set of features

ACKNOWLEDGMENT

Authors would like to thank Guillermo Sanchez Diaz, Vladimir Rodriguez Diez, Manuel S. Lazo Cortés, and Alexsey Lias Rodríguez for sharing the datasets from the Informatics Sciences University and the codes of algorithms fast-BR and GCreduct. They also thank the Instituto Politécnico Nacional (Secretaría Académica, COFAA, SIP, ESIME, and CIDETEC), the CONACyT, and SNI for their economic support to develop this work.

REFERENCES

[1] I. A. Cheguis and S. V. Yablonskii, "About testors for electrical outlines," *Uspieji Matematicheskij Nauk*, vol. 4, no. 66, pp. 182–184, 1955.
[2] A. N. Dmitriev, Y. I. Zhuravlev, and F. P. Krendeliev, "About mathematical principles of objects and phenomena classification," *Diskretni Analiz*, vol. 7, pp. 3–11, 1966.
[3] M. Lazo-Cortés, J. Ruiz-Shulcloper, and E. Alba-Cabrera, "An overview of the evolution of the concept of testor," *Pattern Recognit.*, vol. 34, no. 4, pp. 753–762, 2001.
[4] J. F. Martínez-Trinidad and A. Guzmán-Arenas, "The logical combinatorial approach to pattern recognition, an overview through selected works," *Pattern Recognit.*, vol. 34, no. 4, pp. 741–751, Apr. 2001.
[5] J. Ruiz-Shulcloper, "Pattern recognition with mixed and incomplete data," *Pattern Recognit. Image Anal.*, vol. 18, no. 4, pp. 563–576, Dec. 2008.
[6] A. Pérez-Suárez, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, "A review of conceptual clustering algorithms," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 1267–1296, Aug. 2019.
[7] Y. Reyes-González, U. de las Ciencias Informáticas, N. Martínez-Sánchez, A. Díaz-Sardiñas, M. D. L. C. Patterson-Peña, U. de las Ciencias Informáticas, U. de las Ciencias Informáticas, and U. de las Ciencias Informáticas, "Conceptual clustering: A new approach to student modeling in intelligent tutoring systems," *Revista Facultad de Ingeniería Universidad de Antioquia*, vol. 87, pp. 70–76, Jun. 2018.
[8] J. A. Carrasco-Ochoa and J. F. Martínez-Trinidad, "Feature selection for natural disaster texts classification using testors," in *Intelligent Data Engineering and Automated Learning* (Lecture Notes in Computer Science), vol. 3177. Berlin, Germany: Springer, 2004.
[9] F. Li and Q. Zhu, "Dcoument clustering in research literature based on NMF and testor theory," *J. Softw.*, vol. 6, no. 1, pp. 78–82, 2011.
[10] A. Gallegos, D. Torres, F. Álvarez, and A. Torres, "Feature subset selection and typical testors applied to breast cancer cells," *Res. Comput. Sci.*, vol. 121, no. 1, pp. 151–163, Dec. 2016.
[11] M. D. Torres, A. Torres, F. Cuellar, M. D. L. L. Torres, E. P. D. León, and F. Pinales, "Evolutionary computation in the identification of risk factors. Case of TRALI," *Expert Syst. Appl.*, vol. 41, no. 3, pp. 831–840, Feb. 2014.

[12] M. D. T. Soto, A. T. Soto, D. A. B. Aranda, N. C. Munoz, E. E. P. de Leon Senti, and C. E. Velazquez Amador, "Suicidal tendency neural identifier in university students from aguascalientes, Mexico," in *Proc. 14th Latin Amer. Conf. Learn. Technol. (LACLO)*, Oct. 2019, pp. 387–392.

[13] K. Muenala, J. Ibarra-Fiallo, and M. Intriago-Pazmiño, "Study of the number recognition algorithms efficiency after a reduction of the characteristic space using typical testors," in *New Knowledge in Information Systems and Technologies* (Advances in Intelligent Systems and Computing), vol. 930, Á. Rocha, H. Adeli, L. Reis, and S. Costanzo, Eds., La Toja, Spain. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-16181-1_82.

[14] J. P. Gómez, F. E. H. Montero, and J. C. G. Mancilla, "Variable selection for journal bearing faults diagnostic through logical combinatorial pattern recognition," in *Progress in Artificial Intelligence and Pattern Recognition* (Lecture Notes in Computer Science), vol. 11047, Y. H. Heredia, V. M. Núñez, and J. R. Shulcloper, Eds., La Habana, Cuba. Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-030-01132-1_34.

[15] B. H. R. Orfe, M. M. Luque, Y. R. González, N. M. Sánchez, W. L. Jiménez, and M. H. Perez, "Computational tool to support the process of teaching and learning in the discipline of artificial intelligence," in *Cross Reality and Data Science in Engineering* (Advances in Intelligent Systems and Computing), M. Auer and D. May, Eds. Athens, GA, USA. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-52575-0_75.

[16] G. Shi, F. Luo, Y. Tang, and Y. Li, "Dimensionality reduction of hyperspectral image based on local constrained manifold structure collaborative preserving embedding," *Remote Sens.*, vol. 13, no. 7, p. 1363, Apr. 2021.

[17] B. Rasti, D. Hong, R. Hang, P. Ghamisi, X. Kang, J. Chanussot, and J. A. Benediktsson, "Feature extraction for hyperspectral imagery: The evolution from shallow to deep: Overview and toolbox," *IEEE Geosci. Remote Sens. Mag.*, vol. 8, no. 4, pp. 60–88, Dec. 2020.

[18] Y. Santiesteban-Alganza and A. Pons-Porrata, "LEX: Un nuevo algoritmo para el Cálculo de los testores típicos," *Revista Ciencias Matemáticas*, vol. 21, no. 1, pp. 85–95, 2003.

[19] 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, 2012.

[20] A. Lias-Rodríguez 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, p. 18, 2013.

[21] 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, pp. 111–123, Apr. 2018.

[22] 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, p. 15, 2018.

[23] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. S. Lazo-Cortés, "Fast-BR vs. fast-CT_EXT: An empirical performance study," in *Pattern Recognition* (Lecture Notes in Computer Science), vol. 10267, J. Carrasco-Ochoa, J. Martínez-Trinidad, and J. Olvera-López, Eds., Huatulco, Mexico. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-59226-8_13.

[24] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. S. Lazo-Cortés, "The impact of basic matrix dimension on the performance of algorithms for computing typical testors," in *Pattern Recognition* (Lecture Notes in Computer Science), vol. 10880, J. Martínez-Trinidad, J. Carrasco-Ochoa, J. Olvera-López, and S. Sarkar, Eds., Puebla, Mexico. Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-319-92198-3_5.

[25] A. Rojas, R. Cumplido, J. A. Carrasco-Ochoa, C. Feregrino, and J. F. Martínez-Trinidad, "FPGA-based architecture for computing testors," in *Intelligent Data Engineering and Automated Learning—IDEAL* (Lecture Notes in Computer Science), vol. 4881, H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao, Eds., Birmingham, U.K. Berlin, Germany: Springer, 2007, doi: 10.1007/978-3-540-77226-2_20.

[26] 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.

[27] I. Piza-Dávila, G. Sánchez-Díaz, M. S. Lazo-Cortés, and I. Villalón-Turrubiates, "An algorithm for computing minimum-length irreducible testors," *IEEE Access*, vol. 8, pp. 56312–56320, 2020.

[28] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, M. S. Lazo-Cortés, and J. A. Olvera-López, "MinReduct: A new algorithm for computing the shortest reducts," *Pattern Recognit. Lett.*, vol. 138, pp. 177–184, Oct. 2020.

[29] M. S. Lazo-Cortés, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and G. Sanchez-Diaz, "Computing constructs by using typical testor algorithms," in *Pattern Recognition* (Lecture Notes in Computer Science), vol. 9116, J. Carrasco-Ochoa, J. Martínez-Trinidad, J. Sossa-Azuela, J. O. López, and F. Famili, Eds., Mexico City, Mexico. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-19264-2_5.

[30] 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.

[31] Machine Learning Repository. Center for Machine Learning and Intelligent Systems. Informatics Sciences University (UCI). Accessed: Feb. 16, 2021. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.php

[32] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, 1940.

[33] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Statist.*, vol. 6, pp. 65–70, 1979.

JOEL PINO GÓMEZ was born in Pinar del Río, Cuba, in 1984. He received the B.S. degree in telecommunications and electronics engineering from the University of Pinar del Río (UPR), Pinar del Río, in 2008.

He worked as a Professor at UPR, until 2014. Since 2014, he has been working as a Researcher and a Professor at the Technological University of Havana (Cujae), Havana, Cuba. He is the author of five papers. His research interests include signal processing, machine condition monitoring, hybrid and incomplete data management, feature selection, and classification.

Mr. Pino Gómez is a member of the Cuban Association for Pattern Recognition (ACRP).

FIDEL ERNESTO HERNÁNDEZ MONTERO was born in Pinar del Río, Cuba, in 1972. He received the B.S. degree in telecommunications and electronics from the University of Pinar del Río, Pinar del Río, in 1995, the M.S. degree in digital systems from the Technological University of Havana (Cujae), Cuba, in 2000, and the Ph.D. degree in industrial electronics and automation from Mondragon University, Spain, in 2006.

He worked at the University of Pinar del Río, from 1995 to 2014. Since 2015, he has been working as a Researcher and a Professor at the Department of Telecommunications and Telematics, Cujae, Cuba. Since 2018, he has been working as the Vice Dean for the Research and Postgraduate Studies, Faculty of Telecommunications and Electronics, Cujae. He has contributed to the publications of several books and more than 20 articles. His research interests include machine condition monitoring, signal processing, human gait analysis, and machine learning.

Dr. Hernández Montero is a member of the Iberoamerican Association for Pattern Recognition (IAPR) and the Cuban Association for Pattern Recognition (ACRP).

JOEL CHARLES SOTELO was born in Santiago de Cuba, Cuba, in 1986. He received the B.S. degree in informatics engineering from Informatics Sciences University (UCI), Havana, Cuba, in 2009.

He has been working as a Software Developer, since 2010, and has been an Associate Researcher at the Technological University of Havana (Cujae), Havana, since 2017. He is the author of one paper. His research interests include image processing, geographic information systems, computer graphics, computer simulation, and artificial intelligence.

JULIO CÉSAR GÓMEZ MANCILLA was born in Colima, México, in 1955. He received the B.Sc. degree (*magna cum laude*) in industrial engineering from the Universidad Autónoma de Guadalajara (UAG), México, in 1979, the master's degree in manufacturing from AOTS, Tokyo, Japan, in 1983, and the M.Sc. and Ph.D. degrees from Washington University in St. Louis (WashU), MO, USA, in 1987 and 1992, respectively.

In WashU, he was a Research Assistant to Prof. Andrew D. Dimarogonas. In industry, from 1979 to 1982, he worked at Burroughs Computer Machines Ltd., as a Project Leader of the 3/6MB-computer-floppy drive and commissioned to the Glenrothes plant in England; then, was an Industrial Engineering Manager at HYLSA, Mina Las Encinas, Colima, Mexico, the largest private Mexican iron mine-steel manufacturer. Till 1996, he worked as a Research Project Leader with the Electrical Research Institute, IIE, Cuernavaca, México. He spent the following invited sabbatical years at: Ecole Centrale de Lyon, Lyon, France (2003–2004); Universidad Carlos III, Madrid, Spain, in 2007; Triste University, UNITS, Italy (2013–2014); and has had invited research stays, one to three months each, at: Cornel Fracture Group, Cornell University, NY, USA, in 1993; WashU, from 1996 to 2007; Patras University, Greece, in 1998; and Université Aix in Provence, France, in 2017. Since 1996, he has been the Chair Professor of mechanical engineering with the Instituto Politécnico Nacional, IPN, México City, where he founded and equipped the ESIME Vibration and Rotor Dynamics Laboratory. He has authored/coauthored over 70 papers on international journals and conferences, has supervised six Ph.D. and 35 M.Sc. thesis, and advised several international exchange students, mostly from France. Since 1993, he has been with the National Researchers System, SNI-CONACyT, Mexico. Since 1997, he has advised-directed six graduate students, who have won a prize in the National Contests for Research Thesis. His main research interests include mechanical design, rotor dynamics, fault detection and diagnosis by vibration monitoring, mechatronics applied to bearing lubrication, turbo machine modeling and simulation, fracture mechanics turbo machine modeling and simulation, and fracture mechanics. His other distinctions are: a member for life of Tau-Beta-Pi Honor Society, Missouri Chapter, USA, in 1987; an editor and reviewer for several international indexed journals and science committees; and the Representative Mexico at the IFToMM Rotor dynamics Technical Committee.

YENNY VILLUENDAS REY was born in Ciego de Ávila, Cuba, in 1983. She received the B.S. and M.S. degrees in computer science from the University of Ciego de Ávila, Cuba, in 2005 and 2007, respectively, and the Ph.D. degree in technical sciences from the Universidad Central ''Marta Abreu'' de Las Villas, in 2014.

Since 2015, she has been working as a Researcher and a Professor at CIDETEC, Instituto Politécnico Nacional, Mexico. She is the author of two books and more than 20 articles. Her research interests include data analysis, hybrid and incomplete data management, classification, bio-inspired algorithms, and soft computing.

Dr. Villuendas Rey is a member of the Cuban Association for Pattern Recognition (ACRP). She received the National Prize to the Scientific Research, the National Mention to the Best Young Research in Technical Sciences, and the National Prize to the Best Student in Technical Sciences, all from the Cuban Academy of Sciences, in 2015, 2014, and 2005, respectively. In addition, she received the Annual Rector Award to the Best Researcher from the Rector of the University of Ciego de Ávila, in 2012, 2013, and 2014, respectively. She was a recipient of the National Researcher System (SNI) level C from CONACYT, Mexico.