Artificial Intelligence Review (2022) 55:6607–6628
https://doi.org/10.1007/s10462-022-10162-z

A review of algorithms to computing irreducible testors applied to feature selection

Guillermo Sanchez-Diaz[1] · Manuel S. Lazo-Cortes[2] · Carlos A. Aguirre-Salado[1] · Ivan Piza-Davila[3] · Jorge P. Garcia-Contreras[1]

Published online: 15 March 2022

Abstract
Feature selection is an important task in the areas of pattern recognition and data mining. Various approaches to feature selection have been developed. In particular, this paper focuses on the algorithms for computing irreducible testors, which have been used to solve feature selection problems. The calculation of irreducible testors is an expensive computational process; the complexity of the algorithms to calculate the complete set of irreducible testors exponentially depends on the number of characteristics that describe the objects in the problem. To improve the execution time of these algorithms, different alternatives have been developed, such as parallel implementations, hardware-software implementation, rearrangement of the data, as well as heuristics to generate just an irreducible testor or a subset of the entire set of irreducible testors, among other strategies. This paper presents a review of the literature on irreducible testors, with the aim of providing a guide for researchers working in the areas of pattern recognition and data mining, interested in feature selection, using heterogeneous data and possibly missing data.

Keywords Irreducible testor · Typical testor · Feature selection · Testor

1 Introduction

Feature selection is the process of selecting relevant features for classification tasks. This process is capable of identifying a reduced number of features that best represent data (Webb 2002).

Supervised feature selection is often employed to increase efficiency of classification methods. Its fundamental principle is the correlation between a feature and its class label, or the feature relevance. Available feature selection techniques follow many

✉ Guillermo Sanchez-Diaz
guillermo.sanchez@uaslp.mx

[1] Universidad Autonoma de San Luis Potosi, Av. Dr. Manuel Nava 8, 78290 San Luis Potosi, Mexico

[2] TecNM/Instituto Tecnologico de Tlalnepantla, Av. Instituto Tecnologico s/n, Col. La Comunidad, 54070 Tlalnepantla de Baz, Mexico

[3] Instituto Tecnologico y de Estudios Superiores de Occidente, Periferico Sur Manuel Gomez Morin 8585, 45604 Tlaquepaque, Jalisco, Mexico

different approaches, including the following: heuristics and metaheuristics (Freeman et al. 2015; Kohavi and Jhon 1997; Saeys et al. 2004), multi-objective optimization (Mierswa and Michael 2006), clustering-based methods (Witten and Tibshirani 2010), and correlation-based (Hall 2000).

This work is focused on irreducible testors applied to feature selection problems. Chegis and Yablonskii (1955) developed the first formulation of the concept of testor, also known as *test*. They associated testors to the problem of detection of faults in electrical circuits. With the aim of solving classification problems in Geology, Yu. I. Zhuravlev (Dmitriev et al. 1966) connected the idea of testor to pattern recognition. Since then, testor theory has been developed closely linked to Logical Analysis of Data (Chikalov et al. 2012) and Logical Combinatorial Pattern Recognition (Ruiz-Shulcloper and Abidi 2002). In addition, testor theory continued evolving through applications in the control and diagnosis of faults in contact networks and combinatorial circuits, see for example (Armstrong 1966; Chikalov et al. 2012; Eldred 1959; Roth 1966), where the term *test* is more common than testor. Nevertheless, in this paper, as pointed out earlier, we will focus on problems regarding feature selection for classification.

From this perspective, a testor is a subset of features that allows differentiating objects from different classes. Classical formulation of testor was focused on object descriptions by Boolean features or Boolean comparison criteria between them. This work is focused on algorithms that rely on the classical formulation. Nevertheless, the concept of testor has been extended and generalized in several ways (Lazo-Cortes et al. 2001; Valev and Zhuravlev 1991). Other closely related ideas have been proposed, such as non-reducible descriptors (Valev and Sankur 2004).

In Logical Combinatorial Pattern Recognition (Ruiz-Shulcloper and Abidi 2002), feature selection is addressed using irreducible testors, which are particularly useful when object descriptions contain both qualitative and quantitative values.

Irreducible testors have been used to solve several problems in feature selection. Some examples are the following: estimation of stellar parameters with remotely sensed data (Santos et al. 2004), natural-disaster texts classification (Carrasco-Ochoa and Martinez-Trinidad 2004), evaluation of the relevance of features in differential diagnosis of diseases (Ortiz-Posadas et al. 2001), medical electrodiagnostic using pattern recognition tools (Lopez-Perez et al. 1997), determination of risk factors associated to pregnant Mexican women (Torres et al. 2006), document clustering in research literature (Li and Zhu 2011), identification of risk factors associated to Transfusion Related Acute Lung Injury (TRALI) (Torres et al. 2014), feature selection in breast cancer cells (Gallegos et al. 2016), feature selection on pap-smear data (Rojas-Delgado 2016), determination of the impact of contents in subjects under vocational training (Morales-Escobar et al. 2017), identification of risk factors in medical pathologies (Gallegos-Acosta 2018), etc.

In addition, irreducible testors have been widely used in voting and Kora classifier algorithms (Aslanyan et al. 2015; Ruiz-Shulcloper and Lazo-Cortes 1999); for dimensionality reduction on image databases (Ochoa et al. 2008); in text categorization (Pons-Porrata et al. 2007); for reduction of neural network models (Vazquez and Godoy-Calderon 2007) and automatic summarization of documents (Pons-Porrata et al. 2003), among other applications.

Nevertheless, the computation of the whole set of irreducible testors requires exponential time (Valev and Asaithambi 2003). Due to the computational complexity required to calculate irreducible testors, a wide variety of algorithms have been proposed since their inception.

This work presents a review of algorithms that compute irreducible testors. In Sect. 2, the theoretical background of irreducible testors including basic definitions is shown. Sect. 3 introduces a classification of algorithms to compute irreducible testors. In this section both exhaustive and heuristics algorithms are presented, some that calculate the whole set of irreducible testors, as well as others that compute either a subset of them, only one irreducible testor, or only those having a minimal length. In Sect. 4 several properties and attributes of the algorithms and two related issues are discussed. Finally, conclusions are presented.

2 Theoretical background

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

Most of the algorithms that calculate irreducible testors use a dissimilarity matrix, denoted by DM, instead of working directly with the training sample. The rows of DM are obtained from feature-by-feature comparison between every pair of objects belonging to different classes.

The columns of DM correspond to the features in the training sample.

Every row in DM contain only 0 and 1 values. 0 denotes that the compared objects are equal or similar enough with respect to the corresponding feature, whereas 1 means the opposite.

Definition 1 Let r_j be a row in DM, and let F_j be the associated feature subset (corresponding to ones in r_j). r_j is a super-row of r_l, if $F_l \subset F_j$.

The sub matrix obtained from DM after removing all the super-rows and repeated rows is called a basic matrix (BM).

Most of the algorithms that compute irreducible testors operate with BM because, in general, BM contains much fewer rows than DM and they share the same set of irreducible testors (Piza-Davila et al. 2017).

Let $T \subseteq R$ be a subset of features. BM^T denotes the sub matrix of BM that contains only the columns corresponding to features in T. As mentioned before, a testor is a feature subset capable to differentiate any object from any other belonging to another class. So, formally, we can define a testor as follows.

Definition 2 A feature subset T is a testor of TS, if there are no zero-rows in BM^T (Piza-Davila et al. 2017). Any row containing only zeros is called a zero-row.

In order to reach the concept of an irreducible testor, let us consider the following definitions.

Definition 3 Let T be a testor of TS. A feature $x_j \in T$, is called a non-removable feature of T, if after removing from BM^T the column corresponding to x_j, there exists a zero-row in $BM^{T-\{x_j\}}$. Otherwise, x_j is called a removeable feature (Piza-Davila et al. 2017).

Definition 4 A feature subset T is called an irreducible (or typical) testor of TS, if T is a testor of TS and each feature $x_i \in T$ is a non-removable feature of T (Piza-Davila et al. 2017).

In summary, a testor is a subset of features that allows differentiating any two objects in the training sample as long as they belong to different classes. We call it an irreducible testor if each of its features is essential in the subset. If we refer it to the dissimilarity matrix, or even better to the basic matrix, a testor is associated with a set of columns for which no zero-row appears and each column is essential to maintain that property.

For some algorithms the following definition is important.

Definition 5 Let C be a feature subset of TS. We say that a feature $x_j \notin C$ contributes to C if the number of zero rows in the sub-matrix $BM^{C\cup\{x_j\}}$ is less than the number of zero rows in BM^C.

A concept similar to irreducible testor is the non-reducible descriptor (Asaithambi and Valev 2004). A descriptor of a certain pattern in a particular class is a sequence of values of its characteristics that makes it different from the patterns in the remaining classes. A non-reducible descriptor (NRD) is a descriptor that consists of the least number of characteristics. It means that if any value present in a NRD is ignored, the resulting sequence will no longer be a descriptor.

3 Classification of the algorithms to compute irreducible testors

There are different approaches to design algorithms to compute irreducible testors. In this paper, we consider two types of algorithms: exhaustive/exact and heuristic/approximate.

3.1 Exhaustive algorithms

These algorithms were designed to compute the whole set of irreducible testors. In this category, we include an algorithm that only calculates irreducible testors of minimum length, exclusively. For these algorithms, sequential and parallel implementations have been developed.

Both sequential and parallel algorithms use theoretical properties to prune the power set of the feature set and/or to generate combinations of features as candidates for testors. The common strategy is to ignore some combinations of features to avoid analyzing some subsets, that will not lead us to an irreducible testor.

Sequential algorithms process combinations of features one at a time. The next feature combination is derived from the previous one until no more irreducible testors can be generated or until no further feature combinations can be evaluated.

The search performed by sequential algorithms is implemented by going through the Power Set of Features (PSF) or by building combinations with the rows and columns of cells containing 1 in the basic matrix (BMC).

3.1.1 Algorithms to compute the whole set of irreducible testors

Algorithms without pruning criteria Chegis and Yablonskii proposed the first exact algorithm to construct the set of irreducible testors of a *TS* (Chegis and Yablonskii 1955). Basically, the algorithm consists of building a Boolean logical function $F(TS)$ so that for each pair of rows r_1, r_2 labeled with different decisions, an elemental disjunction $(x_{i_1} \vee \cdots \vee x_{i_l})$ is generated, where i_1, ..., i_l are index values and $r_1 \neq r_2$ (notice that all variables are Boolean). $F(TS)$ describes a function $f(x_1, ..., x_n)$ that is a product of elementary disjunctions for all pairs of rows labeled with different decisions. It can be seen that $F(TS)$ takes the value 1 if and only if the corresponding subset of columns forms a testor for *TS*. To get an explicit description of all irreducible testors, the formula $F(TS)$ should be converted to the reduced disjunctive normal form (by multiplying the elemental disjunctions and applying absorption where possible); the resulting formula is a disjunction of elementary conjunctions of the form $(x_{j_1} \wedge \cdots \wedge x_{j_p})$. Each of these conjunctions describes an irreducible testor, and the complete set of conjunctions corresponds to the set of all irreducible testors.

Algorithms that perform a PSF search The first published algorithms that adopted a PSF search were: BT (Ruiz-Shulcloper et al. 1985), REC (Jimenez-Jacinto 1995), CER (Ayaquica-Martinez and Jimenez-Jacinto 1997) and NRD (Asaithambi and Valev 2004). These algorithms establish a specific order over the power set of features to analyze the generated combinations. Table 1 was taken from (Santiesteban-Alganza and Pons-Porrata 2003), and shows the ordering followed by algorithms that perform a PSF search.

BT algorithm follows an ascending order over the power set of features. In general terms, the first candidate processed by the BT algorithm is $\{x_n\}$, followed by $\{x_{n-1}\}, \{x_{n-1}, x_n\}, \{x_{n-2}\}, \ldots$, until reaching $\{x_1, x_2, \ldots, x_n\}$. This algorithm verifies whether the current feature combination C_i is a testor.

If C_i is a testor, then the irreducibility property is verified. Every irreducible testor found is stored. Depending on the conclusion about C_i, the corresponding pruning rule is applied to generate the next candidate C_{i+1}. There are two different pruning rules: 1) C_i is not a testor, and 2) C_i is a testor regardless of whether it is irreducible or not. BT algorithm does not analyze any intermediate combination between C_i and C_{i+1}.

In the same publication where BT algorithm was introduced, a dual algorithm -called TB- was described too (Ruiz-Shulcloper et al. 1985). This algorithm is based on traversing the power set of features in the opposite direction (descending) using pruning rules

Table 1 Examples of power-set traversals, using features x_1, x_2, and x_3

Ascending		Descending		Breadth		Deep/Lexicographic	
Subset	Binary	Subset	Binary	Subset	Binary	Subset	Binary
$\{\emptyset\}$	000	$\{x_1, x_2, x_3\}$	111	$\{\emptyset\}$	000	$\{\emptyset\}$	000
$\{x_3\}$	001	$\{x_1, x_2\}$	110	$\{x_1\}$	100	$\{x_1\}$	100
$\{x_2\}$	010	$\{x_1, x_3\}$	101	$\{x_2\}$	010	$\{x_1, x_2\}$	110
$\{x_2, x_3\}$	011	$\{x_1\}$	100	$\{x_3\}$	001	$\{x_1, x_2, x_3\}$	111
$\{x_1\}$	100	$\{x_2, x_3\}$	011	$\{x_1, x_2\}$	110	$\{x_1, x_3\}$	101
$\{x_1, x_3\}$	101	$\{x_2\}$	010	$\{x_1, x_3\}$	101	$\{x_2\}$	010
$\{x_1, x_2\}$	110	$\{x_3\}$	001	$\{x_2, x_3\}$	011	$\{x_2, x_3\}$	011
$\{x_1, x_2, x_3\}$	111	$\{\emptyset\}$	000	$\{x_1, x_2, x_3\}$	111	$\{x_3\}$	001

similar to those from BT. TB algorithm is no longer mentioned in the subsequent literature. For this reason, we will not refer to it again in this review.

REC algorithm traverses the power set of features in descendant order. For this, the set $\{x_1, x_2, \ldots, x_n\}$ is the first candidate to be verified, followed by $\{x_1, x_2, \ldots, x_{n-1}\}$, $\{x_1, x_2, \ldots, x_{n-2}, x_n\}$ up to $\{x_n\}$. REC algorithm defines an accumulated error function to evaluate each candidate. If this error function returns 0, then there is no confusion among features of objects, resulting possibly in a new irreducible testor found. If so, it is then stored and a pruning rule is applied to generate the next candidate. The main drawback of REC is that it works directly over the training sample (instead of using the basic matrix), handling a huge amount of superfluous information.

On the other hand, CER algorithm considers the lexicographic order over the power set of features. For this order, the first combination to analyze is $\{x_1\}$, followed by $\{x_1, x_2\}, \ldots, \{x_1, x_2, \ldots, x_n\}, \{x_1, x_3\}$, until reach $\{x_{n-1}, x_n\}, \{x_n\}$.

CER evaluates whether the current candidate C_i is a testor and, if so, it verifies if this testor is irreducible for its storage. Taking into account the lexicographic order, when C_i is a testor (irreducible or not), all subsequent feature combinations that are super sets of C_i are pruned. The next candidate C_{i+1} is generated by either adding a new feature or replacing one or more of the features in C_i.

In the case of non-reducible descriptors, each object in *TS* is considered individually as if it were the only representative of its class, and it is contrasted with all the objects of the other classes. The NRD algorithm establishes an in-depth search on the power set of features. This algorithm works directly with *TS*. The first combination parsed contains the values of the current object O_i.

Considering four features x_1, x_2, x_3, x_4 and a current object $O_i = (1, 1, 0, 0)$, NRD builds a tree that contains this combination as the root node. The second level in the tree contains the combinations $(\diamond, 1, 0, 0)$, $(1, \diamond, 0, 0)$, $(1, 1, \diamond, 0)$ and $(1, 1, 0, \diamond)$, where $\diamond$ indicates that the corresponding characteristic will not be considered. Thus, the search path would be $(\diamond, 1, 0, 0)$, $(\diamond, \diamond, 0, 0)$, $(\diamond, \diamond, \diamond, 0)$, $\cdots$, up to $(1, 1, 0, \diamond)$. If the current candidate is not a descriptor, all child nodes in the tree will be dropped. If the current combination is a descriptor and it does not have child nodes, or none of its child nodes are descriptors, it is checked to see if it does not include any descriptor already stored, and if so, it is saved.

Improvements implemented to BT, REC and CER algorithms A column and row rearrangement of *BM* can be considered as the major improvement of BT, REC and CER algorithms. For example, in BT algorithm, a rearrangement of rows and columns so that the greatest number of zeros is found at the upper right section of *BM* resulted in a significant reduction in execution time (Sanchez-Diaz and Lazo-Cortes 2002) because it increased the pruning capability while traversing the search space. A similar behavior is shown by REC algorithm (Sanchez-Diaz 1997).

A convenient arrangement of either *BM* or *DM* could be implemented for algorithms CER and NRD respectively, to improve their performance.

Algorithms that build only potential combinations A drawback of the early algorithms developed is that they always incorporate a new feature x_j into a combination-candidate C_i and then check testor and irreducibility conditions; later they take pruning decisions, if applicable, and the power set traverse continues.

Some algorithms developed later eliminate this disadvantage, namely : LEX (Santiesteban-Alganza and Pons-Porrata 2003), CT-EXT (Sanchez-Diaz and Lazo-Cortes 2007) and BR (Lias-Rodriguez and Pons-Porrata 2009). Before adding a new feature x_j to a combination C_i, these algorithms verify if this new feature can lead to generating an irreducible

testor candidate (or just a testor), using saved information of the features that make up the current combination and the properties of the new feature to be added.

These algorithms follow a lexicographic order in the power set of features. At the beginning, a convenient rearrangement of *BM* that consists in placing the row with the least number of 1s at the top of *BM* is carried out. Furthermore, all the columns containing 1 on the first row are translated to the leftmost columns in *BM*. These algorithms incorporate some premises to avoid analyzing feature combinations that will not generate any irreducible testor. These premises allow developing a search for non-exclusionary features (NEF), a search for contributing features (CTF) and a recursive search for non-excluding features (RNE).

NEF consists of finding a feature $x_p \notin C_i$, which allows all the features of the next combination C_{i+1} to be non-removable features (see definition 3), including x_p.

CTF leads to find a feature $x_p \notin C_i$, which allows decreasing the number of zero rows in $BM^{C_{i+1}}$ when including x_p (see definition 5).

Finally, RNE works similar to NEF, but it creates recursively subsets of features excluding those that will not be non-removable features of the next combination C_{i+1}.

Additionally, these algorithms use some pruning techniques on the power set of features. These prunings are based on the elimination of the gap (GAP) and the reduction of columns by avoiding unnecessary supersets of features (USF). The gap of C_i is the feature with the highest index whose consecutive characteristic in C_i is not its consecutive characteristic (column) in *BM*. The pruning property based on gap elimination avoids a large number of verifications, specifically of subsequent combinations to C_i which cannot be irreducible testors.

On the other hand, USF pruning allows eliminating subsequent super sets of the current combination C_i, which resulted in a testor (irreducible or not), because these super sets will not generate any irreducible testor.

Table 2 summarizes the premises and pruning strategies of LEX, CT-EXT and BR algorithms.

Improvements implemented to CT-EXT and BR algorithms Several improvements over CT-EXT and BR algorithms have been implemented, among which we can highlight the following: Fast-CT-EXT (Sanchez-Diaz et al. 2012) and Fast-BR (Lias-Rodriguez and Sanchez-Diaz 2013).

These algorithms incorporated a binary representation of both the *BM* and the irreducible testor list that made it possible to replace integer summations and comparisons with bitwise operators.

Besides, as a pre-processing step, Fast-BR discards the computation of irreducible testors using duplicated columns of *BM*, if such applies, i.e., if there exist irreducible testors involving duplicate columns, new irreducible testors will be generated replacing these columns.

There exists another adaptation of BR algorithm, as presented in (Gonzalez-Guevara et al. 2015), which develops adaptations of learning strategies to compare combinations of

Table 2 Premises and pruning strategies of LEX, CT-EXT and BR algorithms

Algorithm	Premises	Prunings
LEX	NEF	GAP
CT-EXT	CTF	USF
BR	RNE	GAP

features. These strategies use both globally and locally learned information to construct the next combinations of features.

Algorithms that perform a BMC search The algorithms that perform a BMC search use properties and rules to generate candidate sets from the elements (1's) of *BM*, to verify whether the combinations of characteristics associated with these sets satisfy the definition of irreducible testor. In some cases, testors are built and then the irreducible-testor property is verified; in other cases, the sets are built and, if they are testors then they are already irreducible. The algorithms that follow this strategy are the following: CC (Aguila and Ruiz 1984), CT (Bravo 1983) and YYC (Alba-Cabrera et al. 2014).

These algorithms use the following definitions.

Definition 6 Let $a_{ij} = a_{pq} = 1$ elements of *MB*. a_{ij} and a_{pq} are compatible elements if $a_{iq} = a_{pj} = 0$ (Aguila and Ruiz 1984; Valev and Radeva 1993).

Definition 7 The set $ES = \{a_{i_1 j_1}, a_{i_2 j_2}, \ldots, a_{i_t j_t}\}$, generated by elements of *BM* is called a compatible set, if either $t = 1$ or each pair of elements of *ES* is a pair of compatible elements (Aguila and Ruiz 1984; Valev and Radeva 1993).

Definition 8 Let $a_{ij} = 1$ and $a_{sj} = 0$ elements of a column of *MB*. If there is an element $a_{sp} = 1$, $j \neq p$, then a_{ij} and a_{sp} are complementary elements (Bravo 1983).

Definition 9 The set $YS = \{a_{i_1 j_1}, a_{i_2 j_2}, \ldots, a_{i_t j_t}\}$ generated by elements of *BM* is called a complementary set, if either $t = 1$ or each pair of elements of *YS* is a pair of complementary elements (Bravo 1983).

CC algorithm takes elements of *BM* to generate sets of compatible elements. At every step, if possible, it incorporates new elements allowing the set to remain a compatible set.

Whenever a new element cannot be included in the set, it is checked whether the combination of features associated with the current set is a testor; only that condition is verified because all the elements of a compatible set meet the condition of being essential, if the set is a testor.

If the current set is an irreducible testor, then it is stored. Iteratively, the last element of the set that can be replaced, is substituted by the next compatible element that has not been processed. This algorithm establishes an order from left to right and from top to bottom on the elements of *BM*, to find compatible elements. When all the elements in *BM* have been processed, the algorithm ends.

CT algorithm behaves similarly to CC algorithm. CT builds sets of complementary elements, using the elements (1's) of *BM*. At each iteration, new elements of *BM* that make the current candidate remain a complementary set are incorporated.

If no more elements can be added to the current set, then it is checked whether the combination of features associated with the current candidate is irreducible, because by construction the elements of a complementary set generate a testor.

YYC algorithm starts by obtaining all the irreducible testors implicit in the first row of the basic matrix *BM*. After that, each subsequent row of *BM* is analyzed one by one. Each step triggers an update on the known set of irreducible testors. If a subset of columns was previously labeled as an irreducible testor, and the new row has, at least one value 1 in any of those columns, then the sub-matrix defined by the updated column set does not have any row exclusively formed by zeros, and therefore this subset of features remains a testor (an

irreducible testor). Otherwise, if the new row shows only zeros in the columns of a previously known irreducible testor, then some new columns must be included -if possible- in order to preserve the testor condition. The candidate columns to be included can only be those with a value 1 at the new row. Including any 1-valued column to the column subset will certainly preserve its testor condition, however, to also be irreducible the sub-matrix delimited by those columns and rows must, under some row and column rearrangement, include a compatible set. Combinations that are no longer irreducible testors are eliminated. The algorithm finishes after the last row in *BM* is processed.

Improvements implemented in CC, CT and YYC algorithms Similar to the algorithms that implement a PSF search, a reordering of columns and rows of BM may result in a performance improvement of CT and YYC algorithms.

In CT algorithm, a row with the least amount of 1s is selected as the first row of *BM*. Columns are arranged from left to right, in descending order with respect to the number of 1s (Sanchez-Diaz 1997).

For YYC algorithm, *BM* rows are sorted in ascending order of the number of ones in each row; also the Hamming distance was used as a second criterion (Piza-Davila et al. 2018).

A strategic ordering of *BM* allows the pruning rules of these algorithms to generate fewer candidates, considerably reducing the execution time of the algorithms (Sanchez-Diaz 1997; Alba-Cabrera et al. 2014).

On the other hand, CC algorithm incorporates locks to avoid analyzing *BM* elements that generate an irreducible testor previously obtained. This is motivated by the fact that the algorithm is vulnerable to generating the same combination of columns more than once, corresponding to different compatible sets. These locks are implemented using some properties of the elements in *BM*.

Parallel algorithms Parallel versions of algorithms have also been developed, both performing PSF and BMC searches. These parallel implementations are based on BT, REC, CC and CT algorithms (Sanchez-Diaz 1997).

Parallelization is carried out by either dividing the set of columns of *BM* or sectioning the power set of characteristics. The first approach is to split the set of columns of *BM* to allocate them across multiple threads of the CPU. When all CPU threads finished calculating the irreducible testors using some algorithm mentioned above, all subsets of found features are merged. In some of these algorithms, it is then verified whether the merged sets constitute irreducible testors or not. The second approach consists of sectioning the power set of characteristics. Each CPU thread is assigned one or more sections, to calculate the irreducible testors. All found irreducible testors are merged after all sections are processed. No further steps are required in this strategy.

In addition, there is a parallel version of the YYC (Piza-Davila et al. 2018) algorithm. The procedure also consists of dividing the *BM*'s column set. Unlike other parallel algorithms, this one only processes columns that have a one in the first row of *BM*.

The number of threads is usually bounded by the number of logical cores available on the CPU.

Hardware implementations Hardware architectures implemented in a Field Programmable Gate Array (FPGA) device have been proposed. These implementations for generating irreducible testors include the Brute force approach (Cumplido et al. 2006), the BT algorithm (Rojas et al. 2012) and the CT-EXT algorithm (Rodriguez-Diez et al. 2015).

These FPGA implementations consist of at least three modules: 1) candidate generator, b) BM storage, and c) testor identification. The candidate generator module produces candidates in the form of tuples to be evaluated by the BM storage module. The BM module

stores the basic matrix in the form of rows, by using logical components to perform bit-wise operations in a single clock cycle. Lastly, the testor identification module determines whether the processed candidate is an irreducible testor or not.

3.1.2 Algorithms to compute an irreducible testor or a subset of irreducible testors

A greedy algorithm Moshkov Chikalov et al. (2012) considers a greedy algorithm *A* that builds a testor for a given training sample *TS*. During each step, this algorithm chooses a feature which separates the maximum number of pairs of objects with different decisions unseparated during the previous steps, and adds this feature to the generated testor. The algorithm finishes when all pairs of objects with different decisions are separated. This algorithm finds a testor but does not guarantee that it is irreducible.

Computing irreducible testors of minimum length Some authors highlight the principle of minimum description length (Rissanen 1978) when addressing certain problems. One goal of this principle applied to feature selection is to select the minimum number of features from a data set. For example, in decision rule generation algorithms, minimum length feature subsets are desirable (Piza-Davila et al. 2020). This principle leads us to consider the subset of irreducible testors of minimum length.

There is an algorithm designed to calculate all irreducible testors of minimum length, MLIT (Piza-Davila et al. 2020). MLIT algorithm, like most of the algorithms reported for computing the whole set of irreducible testors, operates over the basic matrix, instead of the training sample. It carries out a breadth-first search over the power set of features (PSF) instead of the traditional depth-first search, performed by various exhaustive algorithms like Fast-CT-EXT and Fast-BR.

By running a BFS, MLIT algorithm finds irreducible testors or it develops pruning techniques on the power set of features, keeping only potential candidates in a queue data structure, while discarding combinations that will not lead to any irreducible testors. MLIT uses the same CTF premise as CT-EXT algorithm and a premise that does not allow adding features in current combination if these features will not generate any irreducible testor eventually.

In general terms, MLIT algorithm builds feature combinations in non-decreasing order of length. It never builds a candidate longer than a minimum-length irreducible testor. For each feature combination built, it is verified whether it satisfies the testor property or not. 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 previously.

As in other algorithms, a preprocessing of the basic matrix is performed. This preprocessing finds a row with the least amount of ones, and each column containing 1 at this row is moved to the left in the basic matrix. Every candidate built by the algorithm contains a feature represented in the mentioned set of columns; otherwise, the candidate leads us to a non-testor.

3.2 Heuristic algorithms

Given the existence of problems for which there is no need to generate the whole set of typical testors, heuristic algorithms have been developed to find only a subset of the whole set of irreducible testors, using a certain optimization criterion.

There exist heuristics implemented as Genetic Algorithms GAs (Sanchez-Diaz et al. 1999; Torres et al. 2014), Uni-variate Marginal Distribution Algorithm UMDA (Alba et al.

2000), Particle Swarm Optimization PSO (Borja-Cazales and Diaz-Garcia 2014) and Hill-Climbing algorithm HC (Sanchez-Diaz et al. 2014).

The goal of heuristics GAs , UMDA and PSO is to reach a global maximum, in this particular case it refers to the whole set of typical testors. Unlike them, HC consider an individual irreducible testor as a local maximum.

GAs define a fitness function that returns one of three distinct categorical values depending on if the current individual is either a testor, an irreducible testor or non-testor. They implement crossover and mutation operators to generate new combinations from current ones.

Heuristics UMDA, PSO and HC define a fitness function that evaluates how far or near a combination C_i is from being an irreducible testor. Different operators are used by these heuristics to generate new combinations. The operator to be selected is in terms of the value returned by the fitness function.

3.2.1 Parallel implementations of heuristics algorithms

There are some parallel implementations of HC algorithm: Parallel-Hill Climbing PHC uses Java threads (Piza-Davila et al. 2015), and CUDA-Hill Climbing CHC takes advantage of the cores available in the graphic card (Piza-Davila et al. 2017).

PHC implements a random mutation operator that either: a) adds a feature to an individual that is not a testor, b) removes a feature from an individual that holds the testor property, or c) adds and removes features from an individual that holds the irreducible testor property. Besides, PHC handles a binary representation of the structures and uses a digital-search tree to eliminate repeated irreducible testors found.

On the other hand, CUDA-HC uses all internal threads supported by the GPU. Besides, this implementation employs a Bloom filter to handle duplicate irreducible testors efficiently.

4 Discussion

This section illustrates the trend of research works relative to testor theory and applications, and provides some comparison charts that summarize properties and attributes of published algorithms for computing irreducible testors. In addition, this section includes some discussion regarding how to identify beforehand which algorithm would be most suitable for a given matrix, and proposes to explore a specific published platform.

4.1 Properties and attributes of the algorithms

A tag cloud engine called WordCloud Generator based on natural language processing techniques (available at https://monkeylearn.com/word-cloud/) for data deepending and visualization was utilized (Tonkin and Tourte 2016). The seed text used for building tag clouds was obtained from abstracts of 50 validated papers found in a Scopus search with the term “testor”.

In Figs 1a and 1b, two word-clouds to explore graphically the relevance of particular topics of interest referring to testors are shown.

Figure 1a depicts the most prominent words related to testor theory in the following way: typical testor was found 53 times with a relevance of 0.99; feature selection was found

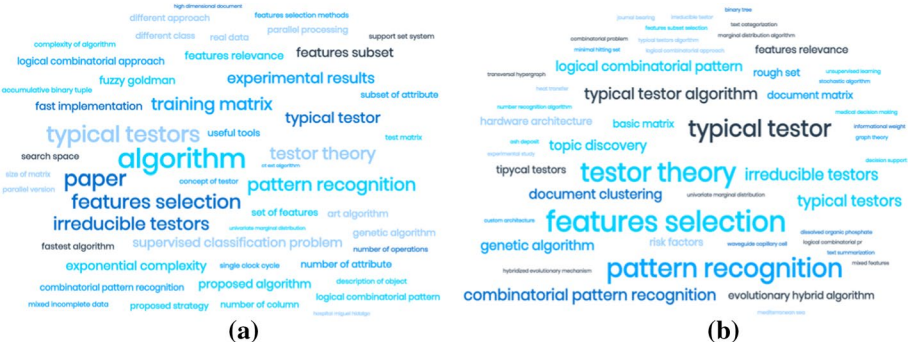

Fig. 1 Word cloud built on literature about testors indexed in Scopus: **a** Based on abstract's data, and **b** Based on author's keywords

31 times with a relevance of 0.451; irreducible testors was found 23 times with a relevance of 0.431; testor theory was found 20 times with a relevance of 0.392; training matrix was found 19 times with a relevance of 0.353.

Relevance was calculated as the inverse document frequency of the term (TF-IDF) in an unrelated corpus of text. That is, it is a relative frequency that increases proportionally to the number of times a term appears in a document and is counterbalanced by the number of documents in the corpus that contain the word. This leads us to compensate the fact that some words are more common than others. Word size in Figures 1 a) and 1 b) is calculated by multiplying frequency by relevance metric (McMahon et al. 2021).

Figure 1 b) presents the most relevant terms found in the keywords of papers related to testor theory. In this search, feature selection was found 21 times with a relevance of 0.998; testor theory was found 15 times with a relevance of 0.713; pattern recognition was found 15 times with a relevance of 0.523; typical testor was found 11 times with a relevance of 0.333; logic-combinatorial pattern recognition was found 4 times with a relevance of 0.285.

In Figure 2, it can be observed a positive trend of works published on testor theory and applications since 1995; this trend includes a temporary erratic behavior. The plot can be divided into four stages: 1^{st} period: 1995-2003, 2^{nd} period: 2003-2009, 3^{rd} period: 2009-2014, and 4^{th} period: 2014-2020.

There have been over thirty proposals of algorithms to compute irreducible testors to date. Some of them are brand new algorithms, while other are modifications to existing ones. This review does not include algorithms published either in languages other than English or in non-international forums. Published algorithms for computing irreducible testors are summarized in Table 3. In this table, the following is shown for each algorithm: a) publication reference, b) name, c) whether it finds the whole set of irreducible testors or only a subset, and d) the type of search: along the power set of features (PSF) or over rows and columns of BM (BMC).

In Table 4, dimensions of rows and columns of *BM* for which the use of the algorithm is suitable, and the relative speed of algorithms are both shown.

Each basic matrix employed to test the algorithms was labeled by the number of features as: small (from 1 to 50), middle (from 51 to 100) and sizeable (more than 100 features). Additionally, each BM was labeled by the number of rows as: low level (1 to 500), average level (501 to 5000) and upper level (more than 5000 rows).

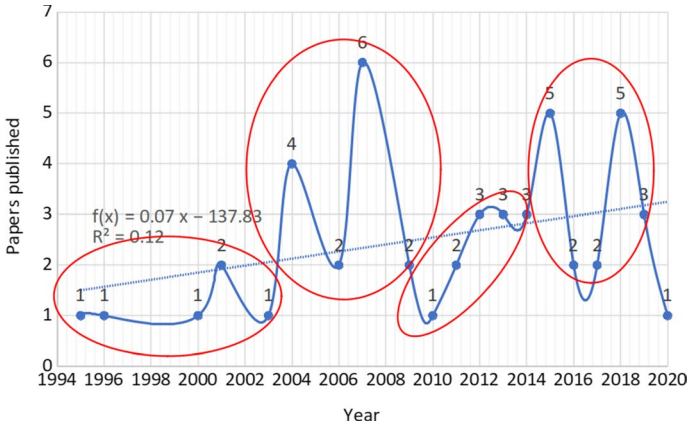

Fig. 2 Amount of papers appearing on Scopus published during 1995-2020

Furthermore, the relative speed reported in the publications is classified in this table as: slow, medium, fast and very fast. Even though this categorization considered the results originally reported by the authors, recent publications (Lias-Rodriguez and Sanchez-Diaz 2013; Piza-Davila et al. 2015, 2017, 2018, 2020) have enabled us to affirm that the fastest algorithms in calculating the entire set of irreducible testors are Fast-BR and Parallel YYC, whereas Parallel-HC and CUDA-HC are the best ones if we wish to find a subset of them, and MLIT is the most appropriate if we are interested in only minimum-length irreducible testors. However, according to Alba-Cabrera et al. (2013), the performance of an algorithm for computing irreducible testors can be quite diverse because it usually depends on the characteristics of the matrices in such a way that, for each algorithm, we can design a particular matrix ad hoc so that the algorithm performs best; however, those matrices could have theoretical value only.

Table 5 shows the journals or conferences where papers regarding irreducible testors were published. This table includes the following: editorial, impact factor (obtained from Journal Impact Factor 2020) and the number of works published in each journal or conference.

In addition, Table 6 shows the distribution of the papers corresponding to the affiliation (and country) of the first author.

Lastly, Table 7 shows the relationship between the size of a basic matrix and the performance of each of the algorithms that compute irreducible testors. We noticed that the number of rows is not as relevant as the number of columns. Nevertheless, in some cases it affects the algorithm performance.

The size of a basic matrix is categorized as follows:

(a) Tiny, the matrix contains less than 20 columns
(b) Small, the number of columns ranges from 20 to 50
(c) Medium, the number of columns ranges from 50 to 70
(d) Large, the number of columns ranges from 70 to 100
(e) Huge, the matrix contains at least 100 columns.

Table 3 Summary of proposed heuristics and exhaustive algorithms, sorted chronologically

Publication	Algorithm	Purpose of finding	Search type
Bravo (1983)	CT	All irreducible testors	BMC
Aguila and Ruiz (1984)	CC	All irreducible testors	BMC
Ruiz-Shulcloper et al. (1985)	BT - TB	All irreducible testors	PSF
Jimenez-Jacinto (1995)	REC	All irreducible testors	PSF
Ayaquica-Martinez and Jimenez-Jacinto (1997)	CER	All irreducible testors	PSF
Sanchez-Diaz (1997)	Improved REC	All irreducible testors	PSF
Sanchez-Diaz (1997)	Improved CT	All irreducible testors	BMC
	Parallel BT		PSF
	Parallel REC		PSF
	Parallel CC		BMC
	Parallel CT		BMC
Sanchez-Diaz et al. (1999)	GA	Subset of irreducible testors	PSF
Alba et al. (2000)	UMDA	Subset of irreducible testors	PSF
Sanchez-Diaz and Lazo-Cortes (2002)	Improved BT	All irreducible testors	PSF
Santiesteban-Alganza and Pons-Porrata (2003)	LEX	All irreducible testors	PSF
Asaithambi and Valev (2004)	NRD	All non-reducible descriptors	PSF
Cumplido et al. (2006)	FPGA Brute force	All irreducible testors	PSF
Sanchez-Diaz and Lazo-Cortes (2007)	CT-EXT	All irreducible testors	PSF
Lias-Rodriguez and Pons-Porrata (2009)	BR	All irreducible testors	PSF
Sanchez-Diaz et al. (2012)	Fast-CT-EXT	All irreducible testors	PSF
Rojas et al. (2012)	FPGA BT	All irreducible testors	PSF
Lias-Rodriguez and Sanchez-Diaz (2013)	Fast-BR	All irreducible testors	PSF
Alba-Cabrera et al. (2014)	YYC	All irreducible testors	BMC
Torres et al. (2014)	GA	Subset of irreducible testors	PSF
Borja-Cazales and Diaz-Garcia (2014)	PSO	Subset of irreducible testors	PSF
Sanchez-Diaz et al. (2014)	HC	Subset of irreducible testors	PSF
Piza-Davila et al. (2015)	Parallel HC	Subset of irreducible testors	PSF

Table 3 (continued)

Publication	Algorithm	Purpose of finding	Search type
Gonzalez-Guevara et al. (2015)	Adjustment BR	All irreducible testors	PSF
Rodriguez-Diez et al. (2015)	FPGA CT-EXT	All irreducible testors	PSF
Piza-Davila et al. (2017)	CUDA HC	Subset of irreducible testors	PSF
Piza-Davila et al. (2018)	Improved YYC	All irreducible testors	BMC
Piza-Davila et al. (2020)	MLIT	All minimal irreducible testors	PSF

Table 4 Summary of suitable dimensions of *BM* and relative speed reported for each algorithm

Algorithm	Number of features	Number of rows in *BM*	Speed
BT-TB	Small	Low level	Slow
REC	Small	Low level	Slow
CER	Small	Low level	Slow
NRD	Small	Low level	Slow
Improved BT	Small	Low level	Slow
Improved REC	Small	Low level	Slow
LEX	Middle	Average level	Fast
CT-EXT	Middle	Average level	Fast
BR	Middle	Average level	Fast
Fast-CT-EXT	Middle	Average level	Fast
Fast-BR	Middle	Upper level	Very fast
Adjustment BR	Middle	Upper level	Fast
CT	Small	Low level	Slow
CC	Small	Low level	Slow
YYC	Middle	Average level	Fast
Improved CT	Small	Low level	Slow
Improved YYC	Middle	Upper level	Very fast
MLIT	Sizeable	Upper level	Very fast
Parallel BT	Small	Average level	Middle
Parallel REC	Small	Average level	Middle
Parallel CC	Small	Average level	Middle
Parallel CT	Small	Average level	Middle
FPGA Brute force	Small	Low level	Middle
FPGA BT	Small	Average level	Fast
FPGA CT-EXT	Middle	Average level	Fast
GA	Middle	Average level	Middle
UMDA	Middle	Average level	Middle
PSO	Middle	Average level	Middle
HC	Middle	Upper level	Fast
Parallel HC	Sizeable	Upper level	Very fast
CUDA HC	Sizeable	Upper level	Very fast

According to the execution time reported in the experiments carried out, the performance of each algorithm is cataloged as follows:

(a) Excellent, it took seconds or few minutes to finish in all the cases.
(b) Good, it took minutes to finish the execution for most of the matrices.
(c) Average, it took few hours to finish the execution for several matrices.
(d) Regular, it took hours or days to finish the execution for most of the matrices.
(e) Poor, it does not finish the execution for several matrices.

Table 5 Summary of number of papers published on testor theory by journal or conference

Journal or conference	Editorial or organization	Impact factor	Papers published
Pattern Recognition	Elsevier	7.196	2
Information Sciences	Elsevier	5.91	2
Expert Systems with Applications	Elsevier	5.452	3
Information Processing and Management	Elsevier	4.787	1
IEEE Access	IEEE Press	4.098	3
Frontiers in Marine Science	Frontiers	3.661	1
Fuzzy Sets and Systems	Elsevier	3.305	1
Pattern Recognition Letters	Elsevier	3.255	3
Applied Intelligence	Springer	2.882	1
Int. J. of Computational Intelligence Systems	Atlantis Press	1.838	1
Int. J. of Pattern Recognition and Artificial Intelligence	World Scientific	1.375	2
International Journal of Bio-Medical Computing	Elsevier	n/a	1
Journal of Software	IAP	n/a	1
Iberoamerican Congress on Pattern Recognition	LNCS Springer	n/a	12
Mexican Conference on Pattern Recognition	LNCS Springer	n/a	4
Mexican Int. Conf. on Artificial Intelligence	LNAI Springer	n/a	4
Int. Conf. on Int. Data Eng. and Automated Learning	LNCS Springer	n/a	2
European Conf. on Principles and Practice of KDD	LNCS Springer	n/a	1
Int. Work. on Art. Intelligence and Pattern Recognition	LNCS Springer	n/a	1
Int. Conf. on Communication Software and Networks	LNCS Springer	n/a	1
Int. Conf. on Reconfigurable Computing and FPGAs	IEEE Press	n/a	1
Advances in Intelligent Systems and Computing	Springer	n/a	1
Asia-Pacific Conf. on Information Processing	IEEE Press	n/a	1
Total			50

4.2 Evaluating the efficiency of irreducible-testor computing algorithms

As can be seen from the overview shown here, a lot of effort has gone into designing increasingly fast algorithms to compute irreducible testors from a matrix. However, the problem of deciding a priori which algorithm will have the best performance for a given matrix remains unsolved. It could be conjectured that for each algorithm there is at least one matrix for which that algorithm is the most efficient. It is in this direction that the work of Alba-Cabrera et al. (2016) must be highlighted. Without being able to describe exactly how, it can be said that the behavior of the algorithms depends on the number of rows, columns, the distribution of ones and zeros in the matrix, the arrangement of the rows and columns, among other factors. In Alba-Cabrera et al. (2016) a theoretical framework and a practical strategy for intentionally building certain test matrices is presented. The theoretical basis of this platform consists of a set of operators that the authors define and that allow designing basic matrices with controlled dimensions and for which the number of irreducible testors is known a priori. The use of this platform can constitute a contribution to make the algorithm evaluation more efficient.

Table 6 Number of papers published per first-author affiliation

First-author affiliation	Country	Papers published
Instituto Nacional de Astrofísica, Óptica y Electrónica	Mexico	14
Universidad San Francisco de Quito	Ecuador	6
Universidad de Oriente	Cuba	4
Instituto Tecnológico y de Estudios Superiores de Occidente	Mexico	4
Centro de Investigación en Computación, IPN	Mexico	3
Universidad Autónoma de Aguascalientes	Mexico	3
Beijing University of Chemical Technology	China	2
Universidad Central de Las Villas	Cuba	2
Universidad Autónoma del Estado de Hidalgo	Mexico	2
Universidad Autónoma de San Luis Potosí	Mexico	1
Pondicherry Engineering College	India	1
Ministry of Science, Technology and Environment	Cuba	1
Universidad Tecnológica de La Habana	Cuba	1
Universidad de Camagüey	Cuba	1
Universidad de Ciego de Avila	Cuba	1
Université de Toulon	France	1
Centro de Investigación y Estudios Avanzados, IPN	Mexico	1
Escuela Superior de Cómputo, IPN	Mexico	1
Universidad de Guadalajara	Mexico	1
Total		50

4.3 Convergent theories

As mentioned above, ever since the origin of testor theory, a strong association between the computation of irreducible testors and the calculation of the prime implicants of a logical function has been revealed. In addition, within the framework of rough set theory, emerged in 1991 (Pawlak 1991), the concept of reduct has been formalized, which turns out to be closely related to irreducible testors, as can be seen in Lazo-Cortés et al. (2015). Additionally, a detailed exposition of the relationship between logic, testor theory and rough set theory can be found in Chikalov et al. (2012). On the other hand, a new point of convergence with testor theory has been formalized in 2019 (Alba-Cabrera et al. 2019), namely, the minimal transversal of a hypergraph. All of the above gives more relevance to the present work because it provides a comparison framework to evaluate related algorithms in various areas, and favors the integration of the best ideas in new algorithms on related areas. Future work would be to expand the review to cover the areas mentioned on this section from their own perspectives, analyzing their points of agreement as well as what would be more useful, unraveling the approaches studied in from any of these areas that have not yet been explored from the others yet.

Table 7 Relationship between algorithms and basic matrices according to performance and characteristics

Algorithm	Size of basic matrix				Huge
	Tiny	Small	Medium	Large	
BT	Excellent	Good	Regular	Poor	Poor
REC	Excellent	Good	Regular	Poor	Poor
CER	Excellent	Good	Regular	Poor	Poor
DT	Excellent	Good	Regular	Poor	Poor
Improved BT	Excellent	Good	Regular	Poor	Poor
Improved REC	Excellent	Good	Regular	Poor	Poor
LEX	Excellent	Good	Regular	Poor	Poor
CT-EXT	Excellent	Good	Regular	Poor	Poor
BR	Excellent	Excellent	Good	Regular	Poor
Fast-CT-EXT	Excellent	Excellent	Good	Regular	Poor
Fast-BR	Excellent	Excellent	Good	Regular	Poor
Adjustment BR	Excellent	Excellent	Good	Regular	Poor
CT	Excellent	Good	Regular	Poor	Poor
CC	Excellent	Good	Regular	Poor	Poor
YYC	Excellent	Good	Regular	Poor	Poor
Improved CT	Excellent	Good	Regular	Poor	Poor
Improved YYC	Excellent	Excellent	Good	Poor	Poor
MLIT	Excellent	Excellent	Good	Regular	Regular
Parallel BT	Excellent	Good	Regular	Poor	Poor
Parallel REC	Excellent	Good	Regular	Poor	Poor
Parallel CC	Excellent	Good	Regular	Poor	Poor
Parallel CT	Excellent	Good	Regular	Poor	Poor
FPGA Brute force	Excellent	Excellent	Regular	Poor	Poor
FPGA BT	Excellent	Excellent	Regular	Poor	Poor
FPGA CT-EXT	Excellent	Excellent	Regular	Poor	Poor
GA	Good	Good	Good	Regular	Poor
UMDA	Good	Good	Good	Regular	Poor
PSO	Good	Good	Good	Regular	Poor
HC	Excellent	Excellent	Excellent	Good	Regular
Parallel HC	Excellent	Excellent	Excellent	Good	Good
CUDA HC	Excellent	Excellent	Excellent	Good	Good

5 Conclusions

Ever since Zhuravlev and his collaborators adapted Chegis and Yablonskii tests to solve pattern recognition problems, testors and especially irreducible testors have spread in various ways. Many algorithms that find all the irreducible testors or some set of them have been developed or improved since then.

With the aim of obtaining irreducible testors more efficiently, different search techniques on the power set of features as well as different orderings across the search space and various ways to produce candidate feature combinations from the basic matrix have been explored.

Several research works regarding the computation of irreducible testors have been published in international journals. Some other works have been published in conference proceedings in regular series, such as Springer and IEEE. This subject has been approached by researchers from various universities and research centers throughout the world.

In practical terms, we could recommend to use either Fast-BR or Parallel YYC if you want to calculate all the irreducible testors of a matrix, Parallel HC or CUDA HC if you want to find any subset of them, and MLIT if you want to calculate only minimum-length irreducible testors.

Technological advances have made it possible to develop algorithms that calculate irreducible testers in increasingly complex problems. However, the problem remains an interesting one due to the high computational complexity required by the calculations. There are no results yet that allow us to know in advance which algorithm would perform best on solving a given problem. Therefore, we are sure that publications on this topic will continue to appear.

Declarations

Conflict of interest All authors declare no conflict of interest in this study.

References

Aguila L, Ruiz J (1984) MB algorithm for k-valued information elaboration on pattern recognition problems. Ciencias Mat J V(3):89–101

Alba-Cabrera E, Godoy-Calderon S, Ibarra-Fiallo J (2016) Generating synthetic test matrices as a benchmark for the computational behavior of typical testor-finding algorithms. Pattern Recogn Lett 80:46–51

Alba-Cabrera E, Godoy-Calderon S, Lazo-Cortés MS, Martínez-Trinidad JF, Carrasco-Ochoa JA (2019) On the relation between the concepts of irreducible testor and minimal transversal. IEEE Access 7:82809–82816

Alba-Cabrera E, Ibarra-Fiallo J, Godoy-Calderon S (2013) A theoretical and practical framework for assessing the computational behavior of typical testor-finding algorithms. In: 18th Iberoamerican congress CIARP. LNCS, vol. 8258. Springer, New York, pp 351–358

Alba-Cabrera E, Ibarra-Fiallo J, Godoy-Calderon S, Cervantes-Alonso F (2014) YYC: a fast performance incremental algorithm for finding typical testors. In: 19th Iberoamerican congress CIARP. LNCS, vol. 8827. Springer, New York, pp 416–423

Alba E, Santana R, Ochoa A, Lazo M (2000) Finding typical testors by using an evolutionary strategy. In: Proceedings of V Iberoamerican workshop on pattern recognition, pp 267–278

Armstrong DB (1966) On finding a nearly minimal set of fault detection tests for combinatorial logic nets. IEEE Trans Electron Comput 15:66–73

Asaithambi A, Valev V (2004) Construction of all non-reducible descriptors. Pattern Recogn 37(9):1817–1823

Aslanyan L, Ryazanov V, Sahakyan H (2015) Testor and logic separation in pattern recognition. Math Problems Comput Sci 44:33–41

Ayaquica-Martinez I, Jimenez-Jacinto V (1997) A new algorithm of outer scale for the generation of typical testors. In: Proceedings of the Iberoamerican workshop on pattern recognition (TIARP 97), pp 141–148

Borja-Cazales D, Diaz-Garcia M (2014) Differential evolution and particle swarm algorithms for calculate typical testors. BSc thesis, National Polytechnic Institute, Mexico

Bravo A (1983) Algorithm CT for compute of typical test of a k-valued matrix. Ciencias Mat J IV(2):123–144

Carrasco-Ochoa J, Martinez-Trinidad J (2004) Feature selection for natural disaster texts classification using testors. In: Proc. of fifth int. conf. on intelligent data engineering and automated learning. LCNS, vol. 3177. Springer, New York, pp 424–429

Chegis IA, Yablonskii SV (1955) On tests for electric circuits. Uspieji Matematicheskij Nauk 4(66):182–184

Chikalov I, Lozin V, Lozina I, Moshkov M, Nguyen HS, Skowron A, Zielosko B (2012) Three approaches to data analysis: test theory, rough sets and logical analysis of data, vol 41. Springer Science & Business Media, New York, pp 3–38

Cumplido R, Carrasco-Ochoa A, Feregrino C (2006) On the design and implementation of a high performance configurable architecture for testor identification. In: Proc. 10th Iberoamerican congress CIARP. LNCS, vol 4225. Springer, New York, pp 665–673

Dmitriev A, Zhuravlev I, Krendeliev F (1966) About mathematical principles and phenomena classification. Diskretni Analiz 7:3–15
Eldred RD (1959) Test routines based on symbolic logic statements. J ACM 6(1):33–36
Freeman C, Kulic D, Basir O (2015) An evaluation of classifier-specific filter measure performance for feature selection. Pattern Recogn 48(5):1812–1826
Gallegos-Acosta A (2018) Identification of risk factors in medical pathologies by means of feature selection characteristics. MsC. Thesis, Autonomous University of Aguascalientes, Mexico
Gallegos A, Torres D, Alvarez F, Torres A (2016) Feature subset selection and typical testors applied to breast cancer cells. Res Comput Sci 121:151–163
Gonzalez-Guevara V, Godoy-Calderon S, Alba-Cabrera E, Ibarra-Fiallo J (2015) A mixed learning strategy for finding typical testors in large datasets. In: 20th Iberoamerican congress CIARP. LNCS, vol 9423. Springer, New York, pp 716–723
Hall MA (2000) Correlation-based feature selection for discrete and numeric class machine learning, In: Proc. 17th int. conf. machine learning, pp 359–366
Jimenez-Jacinto V (1995) Feature selection with the algorithm REC. BSc. thesis on applied mathematics and computation, UNAM, Mexico
Journal Impact Factor (2020) Journal citation reports science edition. Clarivate analytics
Kohavi R, Jhon G (1997) Wrappers for feature subset selection. Artif Intell 97:273–324
Lazo-Cortes M, Ruiz-Shulcloper J, Alba-Cabrera E (2001) An overview of the evolution of the concept of testor. Pattern Recogn 34(4):753–762
Lazo-Cortés MS, Martínez-Trinidad JF, Carrasco-Ochoa JA, Sanchez-Diaz G (2015) On the relation between rough set reducts and typical testors. Inf Sci 294:152–163
Li F, Zhu Q (2011) Dcoument clustering in research literature based on NMF and testor theory. J Softw 6(1):78–82
Lias-Rodriguez A, Pons-Porrata A (2009) BR: a new method for computing all typical testors. In: 14th Iberoamerican congress CIARP. LNCS, vol. 5856. Springer, New York, pp 433–440
Lias-Rodriguez A, Sanchez-Diaz G (2013) An algorithm for computing typical testors based on gaps and reduction of columns. Int J Pattern Recognit Artif Intell 27(8):1–18
Lopez-Perez S, Lazo-Cortes M, Estrada-Garcia H (1997) Medical electrodiagnostic using pattern recognition tools. In: Proceedings of the Iberoamerican workshop on pattern recognition (TIARP 97), pp 237–244
McMahon A, Lewis E, Buniello A, Cerezo M, Hall P, Sollis E, Parkinson H, Hindorff L, Harris L, MacArthur J (2021) Sequencing-based genome-wide association studies reporting standards. Cell Genom 1(1):1–29
Mierswa I, Michael W (2006) Information preserving multiobjective feature selection for unsupervised learning. In: Proceedings of the genetic and evolutionary computation conference. ACM Press, pp 1545–1552
Morales-Escobar J, Roblero-Aguilar S, Guevara-Cruz E, Orozco-Aguirre H (2017) The use of typical testors for determinate the impact of contents in subjects of vocational training. Oper Res J 3:299–304
Ochoa J, Valdes M, Moctezuma I, Ayala C (2008) Dimension reduction in image databases using the logical combinatorial approach. In: Innovations and advances techniques in systems, computing sciences and software engineering. Springer, New York, pp 260–265
Ortiz-Posadas M, Martinez-Trinidad J, Ruiz-Shulcloper J (2001) A new approach to differential diagnosis of diseases. Int J Biomed Comput 40(3):179–185
Pawlak Z (1991) Rough sets: theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht, MA
Piza-Davila I, Sanchez-Diaz G, Aguirre-Salado C, Lazo-Cortes M (2015) A parallel hill-climbing algorithm to generate a subset of irreducible testors. Appl Intell 42:622–641
Piza-Davila I, Sanchez-Diaz G, Lazo-Cortes M, Rizo-Dominguez L (2017) A CUDA-based hill-climbing algorithm to find irreducible testors from a training matrix. Pattern Recogn Lett 95:22–28
Piza-Davila I, Sanchez-Diaz G, Lazo-Cortes M, Noyola-Medrano C (2018) Enhancing the performance of YYC algorithm useful to generate irreducible testors. Int J Pattern Recognit Artif Intell 32(1):1–18
Piza-Davila I, Sanchez-Diaz G, Lazo-Cortes M, Villalon-Turrubiates I (2020) An algorithm for computing minimum-length irreducible testors. IEEE Access 8:56312–56320
Pons-Porrata A, Ruiz-Shulcloper J, Berlanga-Llavori R (2003) A method for the automatic summarization of topic-based clusters of documents. In: Proceedings of VIII Iberoamerican conference on pattern recognition. LNCS, vol 2905. Springer, New York, pp 596–603
Pons-Porrata A, Gil-Garcia R, Berlanga-Llavori R (2007) Using typical testors for feature selection in text categorization. In: Proceedings of XII Iberoamerican conference on pattern recognition. LNCS, vol 4756. Springer, New York, pp 643–652
Rissanen R (1978) Modeling by shortest data description. Automatica 14(5):465–471
Rodriguez-Diez V, Martinez-Trinidad F, Carrasco-Ochoa J, Lazo-Cortes M, Feregrino-Uribe C, Cumplido R (2015) A fast hardware software platform for computing irreducible testors. Expert Syst Appl 42:9612–9619

Rojas A, Cumplido R, Carrasco-Ochoa A, Feregrino C, Martínez-Trinidad J (2012) Hardware-software platform for computing irreducible testors. Expert Syst Appl 39:2203–2210
Rojas-Delgado J (2016) Feature selection on pap-smear data using heuristic information. Cuban J Inf Sci 10(2):73–88
Roth JP (1966) Diagnosis of automata failures: a calculus and a method. IBM J Res Dev 10:278–291
Ruiz-Shulcloper J, Abidi M (2002) Logical combinatorial pattern recognition: a review. In: Recent research developments in pattern recognition. Transword Research Networks, India, pp 133–176
Ruiz-Shulcloper J, Bravo-Martinez A, Aguila-Feros L (1985) Algorithms BT and TB for compute all typical testors. Ciencias Mat J VI:11–18
Ruiz-Shulcloper J, Lazo-Cortes M (1999) Mathematical algorithms for the supervised classification based on fuzzy partial precedence. Math Comput Model 29:111–119
Saeys Y, Degroeve S, Van de Peer Y (2004) Digging into acceptor splice site prediction: an iterative feature selection approach. In: Proceedings of principles and practice of knowledge discovery in databases, pp 386–397
Sanchez-Diaz G (1997) Develoment and programming of efficient algorithms (sequential and parallel) for generated typical testors of a basic matrix. MsC. thesis, Autonomous University of Puebla, Mexico
Sanchez-Diaz G, Diaz-Sanchez G, Mora-Gonzalez M, Piza-Davila I, Aguirre-Salado C, Huerta-Cuellar G, Reyes-Cardenas O, Cardenas-Tristan A (2014) An evolutionary algorithm with acceleration operator to generate a subset of typical testors. Pattern Recogn Lett 41:34–42
Sanchez-Diaz G, Lazo-Cortes M (2007) CT-EXT: an algorithm for computing typical testor set. In: 11th Iberoamerican congress CIARP. LNCS, vol. 4756. Springer, New York, pp 506–514
Sanchez-Diaz G, Lazo-Cortes M (2002) Modifications to algorithm BT for improve their execution time. Ciencias Mat J 20(2):129–136
Sanchez-Diaz G, Lazo-Cortes M, Fuentes-Chavez O (1999) Genetic algorithm to calculate typical testors of minimal cost. In: IV Iberoamerican symposium on pattern recognition, pp 207–212
Sanchez-Diaz G, Lazo-Cortes M, Piza-Davila I (2012) A fast implementation for the typical testor property identification based on an accumulative binary tuple. Int J Comput Intell Syst 5(6):1025–1039
Santiesteban-Alganza Y, Pons-Porrata A (2003) LEX: a new algorithm to generate typical testors. Ciencias Mat J 21(1):85–95
Santos J, Carrasco A, Martinez J (2004) Feature selection using typical testors applied to estimation to stellar parameters. Comput Sistemas J 8(1):15–23
Tonkin E, Tourte G (2016) Working with text: tools, techniques and approaches for text mining. Chandos Publishing, Cambridge, MA
Torres D, Torres A, Cuellar F, Torres M, Ponce-de-Leon E, Pinales F (2014) Evolutionary computation in the identification of risk factors, case of TRALI. Expert Syst Appl 41(3):831–840
Torres D, Torres A, Ponce-de-Leon E (2006) Genetic algorithm and typical testors in feature subset selection problem. In: Proceedings of sixth Iberoamerican conference on systemics, cybernetics and informatics, pp 1–5
Valev V, Asaithambi A (2003) On computational complexity of non-reducible descriptors. In: Proceedings of the IEEE international conference on information reuse and integration, pp 208–211
Valev V, Radeva P (1993) On the determining of non-irreducible descriptors for multidimensional pattern recognition problems. Pattern Recogn Image Anal 3(3):258–265
Valev V, Sankur B (2004) Generalized non-reducible descriptors. Pattern Recogn 37(9):1809–1815
Valev V, Zhuravlev Y (1991) Integer-valued problems of transforming the training tables in k-valued code in pattern recognition problems. Pattern Recogn 24(4):283–288
Vazquez R, Godoy-Calderon S (2007) Using testor theory to reduce the dimension of neural network models. Res Comput Sci 28:93–103
Webb A (2002) Statistical pattern recognition. Wiley, New York, pp 305–360
Witten DM, Tibshirani R (2010) A framework for feature selection in clustering. J Am Stat Assoc 105(490):713–726