IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 25, 2019, accepted May 27, 2019, date of publication June 12, 2019, date of current version July 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922231

On the Relation Between the Concepts of Irreducible Testor and Minimal Transversal

EDUARDO ALBA-CABRERA[1], SALVADOR GODOY-CALDERON[2], MANUEL S. LAZO-CORTÉS[3], J. FCO. MARTÍNEZ-TRINIDAD[4], AND JESÚS A. CARRASCO-OCHOA[4]

[1]Colegio de Ciencias e Ingenierías, Universidad San Francisco de Quito, Quito 170901, Ecuador
[2]Centro de Investigación en Computación, Instituto Politécnico Nacional, Ciudad de México 07738, Mexico
[3]SEP/SES/TecNM/Instituto Tecnológico de Tlalnepantla, Mexico 54070, Mexico
[4]Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla 72840, Mexico

Corresponding author: Salvador Godoy-Calderon (sgodoyc@cic.ipn.mx)

This work was supported in part by the Universidad San Francisco de Quito through the Project under Grant 5512, and in part by the Instituto Politécnico Nacional, México, particularly through the Project under Grant SIP-20196094.

ABSTRACT This paper aims at studying the relationship between two rather relevant theoretic fields such as Graph Theory and Testor Theory, deepening in the unexploited relation between the concepts of Minimal Transversal and Irreducible Testor. First, the classic definitions of each concept are provided, and then the relation between them is shown and formalized. Some of the immediate consequences of this relation, in terms of the duality property of transversals and about the equivalence of the result of two specific algorithms, one from each field, are discussed. Finally, we also discuss several future research directions that arise from the relationship between the concepts of Minimal Transversal And Irreducible Testor, and the way in which those directions can potentially benefit the development of theory and algorithms for solving different practical problems in both areas.

INDEX TERMS Testor theory, graph theory, irreducible testor, minimal transversal, hitting set.

I. INTRODUCTION

Testor Theory is a useful tool for feature selection and evaluation in Pattern Recognition. It has been used for solving a wide variety of practical problems like medical diagnosis [1], text categorization [2], document summarization [3], document clustering [4], etc. The concept of irreducible testor [5] has been extended and generalized in several ways, allowing researchers to develop algorithms which, despite of their high complexity, have helped tackling new interesting practical problems with the same theoretical framework [6]–[9].

Graph Theory, on the other hand, has been one of the most relevant fields of discrete mathematics during the last decades. Its versatility for modeling a wide range of phenomena has taken it to an elite position among research areas. A particularly useful concept in this field is that of minimal transversal [10]. Artificial intelligence, reliability theory, database theory, integer programming, and learning theory, are among the relevant areas where this concept has been applied [11], [12]. However, algorithms for computing minimal transversals also have a high time complexity and therefore, several different algorithms have been proposed for this task [13], [14].

By deepening in the convergence between these two fields, we look forward into contributing to a broader and unified understanding of both. Also, since each field has been developed independently from each other, we observe that each field can benefit from some concepts and techniques from the other one.

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

II. THEORETICAL FRAMEWORK FROM TESTOR THEORY

Let U be a partition of objects, described by a set of n features and grouped into k disjoint classes (i.e. a training sample). By using feasible comparison criteria, a matrix $A = [a_{ij}]_{m\times n}$ can be constructed to hold the information about the comparison of all objects belonging to different classes in U. That matrix is known as pairwise comparison matrix. Henceforth, we will refer to it as comparison matrix.

If Boolean comparison functions are used for constructing a comparison matrix, then each $a_{ij} \in \{0, 1\}$. When an element $a_{ij} = 1$, it means that objects within pair i have different values in feature j, while $a_{ij} = 0$ is interpreted as objects within pair i have similar values in feature j. In that case, the matrix is called a Boolean difference matrix. In all of the

following, whenever we use a difference matrix we will be referring to a Boolean difference matrix.

Let A be a difference matrix, $\mathcal{R}_A = \{a_1, \ldots, a_m\}$ be the set of rows in A, and $\mathcal{F}_A = \{x_1, \ldots x_n\}$ be the set of features used to describe objects. Also, let $A|_T$, with $T \subseteq \mathcal{F}_A$,the sub-matrix obtained by eliminating from A all columns whose features do not belong to the set T. Then, we have the following:

Definition 1: A subset of features $T \subseteq \mathcal{F}_A$, is called a testor in A if the sub-matrix $A|_T$ does not have any row composed exclusively by 0s. The set of all testors in A is denoted by $\psi(A)$.

According to the previous definition, a *testor* is a subset of features capable of describing all objects in the training sample, without causing confusion among objects belonging to different classes.

Definition 2: A testor T is called an irreducible testor in A if it is minimal with respect to the inclusion relation; that is, if no proper subset of T is a testor in A. The set of all irreducible testors in A is denoted by $\psi^*(A)$.

The above definition implies that no feature can be removed from T without loosing its testor nature. Consequently, for each $x_j \in T$, being T an irreducible testor, there must be at least one row in $A|_T$, such that it has a 1 in the column of x_j and 0 in all other columns in T. Such row is called a typical row of x_j with respect to T, and it seems evident that all features in an irreducible testor, must have at least a typical row.

The information needed for finding irreducible testors in a difference matrix can be shrunk into a much smaller structure. A row r_p in a difference matrix is considered a sub-row of another row r_q if each position of r_p has a value less than or equal to the corresponding value in r_q and there is at least one position where r_p has a value strictly less than the corresponding one in r_q.

Definition 3: A row r_p in a difference matrix A is called a basic row if there is not another row in A being a sub-row of r_p. A matrix B is called a basic matrix from a difference matrix A if it contains all and exclusively all the basic rows of a difference matrix without repetition.

Evidently, a basic matrix B has equal or less rows than the original difference matrix A. However, the set of irreducible testors embedded within a basic matrix B, $\psi^*(B)$, is exactly the same as the one embedded within the difference matrix A, $\psi^*(A)$; this is proved in [15] through the following theorem.

Theorem 1: Let A be a difference matrix and B its corresponding basic matrix, then $\psi^*(A) = \psi^*(B)$.

Since a basic matrix has fewer rows but the same irreducible testors, this explains why most of algorithms for computing irreducible testors are run on basic matrices instead of difference matrices.

III. THEORETICAL FRAMEWORK FROM HYPERGRAPH THEORY

In graph theory, a hypergraph is a generalization of the traditional graph concept. A hypergraph is defined as follows:

Definition 4: A hypergraph $\mathcal{H}$ is an ordered pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of objects, and $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\}$ is a covering of $\mathcal{V}$, i.e. a family of subsets of $\mathcal{V}$ such that $\mathcal{E}_i \neq \emptyset$ $(i = 1, \ldots, m)$ and $\bigcup_{i=1}^{m} \mathcal{E}_i = \mathcal{V}$.

The elements of $\mathcal{V}$ are called *vertices* or *nodes*, while the elements of $\mathcal{E}$ are called *hyperedges* of the hypergraph $\mathcal{H}$.

From the above definition it is clear that a hypergraph can be seen as a generalization of a graph, but without the restriction of an edge (hyperedge) connecting only two vertices.

Definition 5: The incidence matrix A of $\mathcal{H}$ is an $n \times m$ matrix $A = [a_{ij}]_{n \times m}$ whose rows and columns correspond to the vertices and the hyperedges of $\mathcal{H}$ respectively, in such a way that $a_{ij} = 1$ if $v_i \in \mathcal{E}_j$ and $a_{ij} = 0$ otherwise.

Definition 6: A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is called simple if for every pair $(\mathcal{E}_i, \mathcal{E}_j)$, $\mathcal{E}_j \subseteq \mathcal{E}_i \Rightarrow j = i$.

When a hypergraph $\mathcal{H}$ is simple the sets of vertices represented by the hyperedges of $\mathcal{H}$ form a Sperner family, (i.e. A family of sets in which none of the sets is contained in another one).

Definition 7: Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph. A set $\tau \subseteq \mathcal{V}$ is called a transversal (or, hitting set) of $\mathcal{H}$, if it intersects all its hyperedges, i.e., $(\forall \mathcal{E}_i \in \mathcal{E})\ \tau \cap \mathcal{E}_i \neq \emptyset$. A transversal τ is called minimal if no proper subset τ' of τ is a transversal of $\mathcal{H}$.

Definition 8: The transversal hypergraph $Tr(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is the family of all minimal transversals of $\mathcal{H}$.

IV. RELATION

The previous sections present self-explained concepts about both testor and hypergraph theories. By analizing definitions of basic matrix, incidence matrix and simple hypergraph, we first established the following auxiliar theorem:

Theorem 2: A transposition over the incidence matrix of a simple hypergraph, results in a matrix that fulfills all required properties to be a basic matrix.

The proof of this theorem is immediate from definitions 3, 6 and 7. However, the result it enunciates is very important, since the resulting basic matrix can then be used to find irreducible testors, and the complete set of irreducible testors from that matrix turns out to be the transversal hypergraph of the original hypergraph.

We now formalize the relation between the concepts of Minimal Transversal and Irreducible Testor with the following theorem:

Theorem 3: Let $\psi^*(B) = \{\tau_1, \ldots, \tau_s\}$ be the complete family of irreducible testors from a basic matrix B. Let $Tr(\mathcal{H})$ be the transversal hypergraph for a simple hypergraph $\mathcal{H}$ whose incidence matrix is exactly the transposed matrix of B, B^t, then $\psi^*(B) = Tr(\mathcal{H})$.

Proof: Let $\mathcal{F}_A = \{x_1, \ldots x_n\}$ be the set of features in a basic matrix B, and let $\psi^*(B) = \{\tau_1, \ldots, \tau_s\}$ be the family of irreducible testors in B. By definition, all rows in B must be basic rows, it means that they are pairwise incomparable. Since B is a Boolean matrix, B^t can also be interpreted as the incidence matrix for some hypergraph $\mathcal{H}$. When doing so, it immediately comes to mind the fact that, since B is

formed exclusively by incomparable rows, then $\mathcal{H}$ must be a simple hypergraph. Now, by definition, each τ_j is a subset of features such that, $B|\tau_j$ contains no zero rows. Therefore, when interpreting B as an incidence matrix, it seems evident that all hyperedges in $\mathcal{H}$ are incident on at least one vertex, consequently each $\tau_j \in \psi^*(B)$ must be a transversal of $\mathcal{H}$. Also, τ_j is minimal, since irreducible testors are always minimal. In conclusion, since $\psi^*(B)$ contains all minimal transversal of $\mathcal{H}$, by definition it is precisely the transversal hypergraph of $\mathcal{H}$. Now, by definition, $Tr(\mathcal{H}) = \{\tau_1, \ldots, \tau_s\}$ is the family of all minimal transversals in $\mathcal{H}$, which means that each $\tau_j \in Tr(\mathcal{H})$ is a subset of hyperedges always incident on at least one vertex of $\mathcal{H}$. Within the incidence matrix of $\mathcal{H}$, each set τ_j determines a sub-matrix that has no zero rows, which is precisely the required condition so that τ_j is a testor in $\mathcal{H}$. Since all transversals are known to be minimal, then each τ_j is an irreducible testor in $\mathcal{H}$, and $Tr(\mathcal{H})$ is the complete family of irreducible testors in the incidence matrix of $\mathcal{H}$. □

The last theorem expresses the equivalence between computing minimal transversals of a simple hypergraph $\mathcal{H}$ and computing irreducible testors from the transposed incidence matrix of $\mathcal{H}$ (i.e., the corresponding basic matrix).

A. SOME IMPLICATIONS OF THE RELATION

The most immediate implication of Theorem 3 is the fact that both tasks, the computation of irreducible testors and the computation of minimal transversals, can be performed by any known algorithm originally designed for one task or the other. Algorithms from both fields can be comparatively tested under the same conditions. However, for really exploiting the relation between the concepts of Minimal Transversal and Irreducible Testor, a deeper analysis should be performed. Concepts, techniques and even application areas known to one field but not explored by the other are the cornerstone for building new fields with unified frameworks.

Another rather relevant consequence of the relation lies on the *duality property* [16] which states that $Tr\,(Tr\,(\mathcal{H})) = \mathcal{H}$, and has the following implications:

1) The transversal hypergraph of a simple hypergraph is also a simple hypergraph (which we will identify as the *dual hypergraph*).
2) The transversal hypergraph of the dual hypergraph is the original hypergraph.

Based on *duality property*, two important properties (lemmas 12 and 13) in the field of Testor Theory emerge immediately as follows.

If a basic matrix B and its complete family of irreducible testors $\psi^*(B) = \{\tau_1, \ldots, \tau_s\}$ are known, then a matrix can be constructed with all the irreducible testors written in standard notation (i.e. each row i is a binary sequence where an entry 1 at position j means that feature $x_j \in \tau_i$ and an entry 0 means the opposite). Such matrix will be called the irreducible testor matrix of B and will be denoted as $\left[\psi^*(B)\right]$. Rows in $\left[\psi^*(B)\right]$ represent irreducible testors in B, while columns represent descriptive features for the studied objects. Observe that $\left[\psi^*(B)\right]$ has a row for each irreducible testor in B, and exactly the same number of columns than B. Since by definition all irreducible testors are minimal, then all rows in the testor matrix are incomparable, and therefore the following lemmas are true:

Lemma 1: $\left[\psi^*(B)\right]$ *is also a basic matrix.*

Lemma 2: $\left[\psi^*\left(\left[\psi^*(B)\right]\right)\right] = B$, *except for some row rearrangement.*

The aforementioned implications, emerged from Theorem 2, can potentially benefit the field of Testor Theory by suggesting a proved method for validating algorithms for computing irreducible testors. Also, a complete benchmark for these algorithms can be defined with a carefully studied family of test matrices [17].

Following the opposite view, in the field of Testor Theory it is common knowledge that algorithms for computing irreducible testors follow two main strategies: internal and external. The general strategy for internal algorithms is to iteratively select some entries from the basic matrix and using them to construct irreducible testor candidates [18].

On the other hand, external algorithms induce an order over the power set of features used to describe objects, and then use some strategy, guided by the logical properties of that order, to find the family of irreducible testors [19].

After a thorough review in state of the art research works, it seems that the concept of external algorithm is not present at all into Graph Theory, consequently it represents a new strategy that can be followed by trnansversal-computing algorithms.

V. COMPARATIVE EXAMPLE

In this section, we select a representative well-known algorithm from each field, we briefly analyze their fundamental strategies, and then re-write them both to fit into a unified framework. The general structure of this framework is a search algorithm over the power set of hypervertices (or descriptive features), and in both rewritten algorithms a list is used as the main data structure to traverse the search space. The required input is a binary matrix, which represents the incidence matrix of a simple hypergraph (for the transversal algorithm), or the transposition of a basic matrix (for the irreducible testor algorithm). By presenting both algorithms rewritten in this unified framework, we hope to contribute to the understanding of some of the most immediate consequences of the presented the relation between the concepts of Minimal Transversal and Irreducible Testor.

A. THE KAVVADIAS-STAVROPOULOS ALGORITHM (KS)

The Transversal Hypergraph Generation Problem, as known in the field of Graph Theory, is the problem of generating the set $Tr(\mathcal{H})$ of all minimal transversals of a given hypergraph $\mathcal{H}$.

Several algorithms are known for the above problem. For example, in [10], Berge proposed an algorithm that has been a reference point for most of the subsequent proposals. Khachiyan [20] proposed another algorithm that notably

outperforms Berge's algorithm. Truly remarkable because its performance enhancement is the well known algorithm proposed by Kavvadias and Stavropoulos [16], whose incorporation of the *generalized vertex* concept allows it to successfully handle larger data problems.

The Kavvadias-Stavropoulos algorithm (*KS* algorithm) is based on a hyperedge wise incremental computation of the transversal hypergraph. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_m\}$, the *KS* algorithm starts by constructing $Tr((\mathcal{V}, \mathcal{E}_1))$ and then, incrementally complements the result with the transversals obtained from each subsequent hyperedge. The algorithm ends when the transversals from $\mathcal{E}_m$ have been generated and incorporated to the final result. Following that strategy, all the hyperedges of $\mathcal{H}$ need to be analyzed before the final result (the transversal hypergraph of $\mathcal{H}$), can be delivered, and to overcome the potentially exponential memory requirement of other algorithms, Kavvadias-Stavropoulos heavily rely on the following definition of a generalized vertex:

Definition 9: Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph. The set $\chi \subseteq \mathcal{V}$ is a generalized vertex of $\mathcal{H}$ if all vertices in χ belong to exactly the same hyperedges of $\mathcal{H}$.

By appropriately exploiting the concept of generalized vertex, and by structuring its algorithm in a depth-first manner, the *KS* algorithm achieves high performance. Also, since the *KS* algorithm operates in a generate-and-forget fashion, its memory requirements are greatly reduced. To avoid having to wait until the algorithm ends to see some results, Kavvadias-Stavropoulos reshape their algorithm to define a tree of minimal transversals in a depth-first fashion. In order to accomplish that, they incrementally analyze each hyperedge and label all generalized vertices according to the elements that the currently analyzed hyperedge has, under the exact columns of that generalized vertex. Consequently, when analyzing a new incidence matrix row, each previously known generalized vertex χ is assigned a label as follows:

- type α if the new row has only 0s under all columns of χ
- type β if the new row has only 1s under all columns of χ
- type γ if the new row has any combination of 0s and 1s under the columns of χ.

Lastly, to avoid testing generalized vertex combinations that will not produce a minimal transversal, Kavvadias-Stavropoulos rely on the following definition:

Definition 10: Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph and let τ be a minimal transversal of the partial hypergraph $\mathcal{H}_i$ (i.e. up to row i of the incidence matrix). A generalized vertex $v \subseteq V \setminus \tau$ is an appropriate vertex for τ, at level i, if no other vertex in $\tau \cup \{v\}$ except v can be removed and the remaining set is still a transversal of $\mathcal{H}_i$.

Let K_α, K_β and K_γ denote the number of generalized vertices of each respective type. The *KS* algorithm uses the following auxiliary functions:

– *Label* (v_i, j) assigns an α, β or γ label to vertex v_i according to the analysis of the jth row in the incidence matrix.
– $\alpha(v_i)$, $\beta(v_i)$ and $\gamma(v_i)$ are predicate functions that identify v_i as an α, β or γ vertex.
– *Split* (v) separates a vertex v_i of type γ into its two parts: the generalized vertex containing all columns with 0's in the currently analyzed row of the incidence matrix, and the generalized vertex containing all columns with 1's.
– *Recombine*$(\mathcal{V})$ receives a set of γ-type generalized vertices, which were already procesed by the function *Split* and returns a set with all possible combinations of their parts. Note that one, and only one, of the resulting combinations will only contain parts with 0s, that element is called the *Zero* vertex.
– *GetZero*(T) receives a set of generalized vertices and returns the *Zero* vertex.
– *Complete*(T_1, T_2) receives two sets of generalized vertices and returns a new set where the whole set T_1 is treated as a γ-vertex part and it is combined with all the vertices in T_2.
– *Appropriate*(T, i) returns a set with all the appropriate vertices for T in hyperedge e_i.
– *Refine*(T) receives a set of generalized vertices and returns a set containing all possible vertex unions in distinct generalized vertices; in some sense, the Cartesian product of generalized vertices in T.

The *KS* algorithm and its associated auxiliary function KSDescendants are outlined in Algorithm 1 and Function 1, respectively. For more detailed explanation of this algorithm, the reader should refer to [13], [16].

Algorithm 1 Kavvadias-Stavroupoulos

Input: The incidence matrix of a simple hypergraph $\mathcal{H}$
Output: $Tr(\mathcal{H})$
Let T be the set of all generalized vertices of hyperedge e_1
Push [T,1] into *Stack*
While *Stack* not empty do
    [T, j] = Pop(*Stack*)
    Reverse Push all KSDescendants(T, j+1) into *Stack*
endWhile
Final answer is stored in Transversals

B. THE BINARY-RECURSIVE ALGORITHM (BR)

As it has been established, traditional external algorithms for computing irreducible testors search within the power set of features used to describe objects. However, the search process is not an exhaustive search over the power set. Some properties of each tested subset allow the algorithm to infer which other successive subsets, following the established order, are not irreducible testors, and therefore they are not worth be tested. The act of bypassing the test of some subsets is commonly referred to as *jumping*. In general, the order selected to traverse the power set of columns, along with the magnitude of the jumps (the number of non-tested subsets), and the specific procedure applied over a subset to test whether it is an irreducible testor or not, determine the behavior of an external algorithm.

Function 1 KSDescendants (T, j)

Input: A set T of generalized vertices
Output: The set of all descendants of T
Let *Temp1*, *Temp2* and *Zero* be auxiliary empty sets
Label(T, j)
Push all α and β vertices into *Temp1*
Push all γ vertices into *Temp2*
Temp2 = ***Recombine***(***Split***(*Temp2*))
If $K_\beta \neq 0$ then
    Temp1 = ***Complete***(*Temp1*, *Temp2*)
else
    When $K_\gamma \neq 0$ do
        Zero = ***GetZero***(*Temp2*)
        Temp2 = ***Complete***(*Temp1*, *Temp2* \ *Zero*)
    endWhen
    Zero = *Zero* $\cup$ *Temp1*
    Zero = ***Complete***(*Zero*, ***Appropriate***(*Zero*))
    Temp1 = *Temp2* $\cup$ *Zero*
endIf
If $j = m$ then
    Push ***Refine***(*Temp1*) into *Transversals* and Return($\emptyset$)
else
    Return([*Temp1*, *j*])
endIf

The *BR* algorithm [19] starts by inducing the $\preceq_{BR}$ order over the power set of features as follows:

Let T_1, T_2, and X be ordered sets such that $T_1, T_2 \subseteq X$. Also, let the function *first*(T), which returns the first element in T, and finally let $\preceq_{lex}$ be the lexicographic order. Then,

$$T_1 \preceq_{BR} T_2 \text{ if } \begin{cases} first(T_1) \preceq first(T_2), \\ first(T_1) = first(T_2) \text{ but } |T_1| \leq |T_2|, \\ first(T_1) = first(T_2) \text{ and } |T_1| = |T_2| \\ \qquad\qquad \text{but } T_1 \preceq_{lex} T_2. \end{cases}$$

Starting with the first non-empty subset in that order, each currently analyzed subset T_i is tested. If it is an irreducible testor, then it is added to the final result and all its supersets are excluded from subsequent testing.

Otherwise (e.g. T_i is not an irreducible testor), all subsets that can be constructed as $T_i \cup \{x_j\}$ are analyzed. Those subsets are called T_i's descendants, and they can fall into one of two possible groups:

The first group contains those features that do not reduce the number of zero rows in the submatrix defined by $T_i \cup \{x_j\}$, or that ruin the typicality of any feature in T_i (see discussion after Definition 2). These features are labeled as *exclusive*.

The second group contains all remaining features that neither form an irreducible testor, nor ruin the typicality of any feature in T_i. Those features are labeled as *candidate*. Only features into this last group are considered as suitable candidates to become an irreducible testor and they are recursively analyzed in the same fashion. For a more detailed explanation of the operation of this algorithm, the reader should refer to [19].

Let *start*(T) be a function that returns a set with all elements in T, except the last one, and let *last*(T) be the function that returns precisely that last element in T. Also, let *TypicalTestor*(T) be a predicate function that returns true if T is an irreducible testor in the basic matrix B and false otherwise. Finally, let *Candidate*(T) be another predicate function that returns true if *last*(T) is non-exclusive with respect to *start*(T) and false otherwise. The *BR* implementation, whose pseudocode is shown in Algorithm 2, works with an auxiliary pseudo-stack structure managed with the classic function *Pop*() for extracting the last element in the structure, but with insertion of multiple elements in reverse order and at diverse positions. Given a set T, all its supersets following the lexicographic order are called its descendants.

The Binary-Recursive algorithm and its associated auxiliary function Analyze are outlined in Algorithm 2 and Function 2 respectively.

Algorithm 2 Binary-Recursive Algorithm

Input: A basic matrix B.
Output: The set $\psi^*(B)$ of all irreducible testors in B.
Reorder B placing the row with least and left-most 1s at the top
Let $\psi^*(B) = \emptyset$ and *Stack* empty
Reverse push into *Stack* all features with a value 1 on the first row of B
$\psi^*(B)$ =Analyze(*Stack*)

Function 2 Analyze (*List*)

Input: A list with feature sets of B.
Output: The set $\psi^*(B)$.
If *Empty*(*List*) then return $\emptyset$
else
  T = *Pop*(*List*)
  If *TypicalTestor*(T) then
    remove all its super-sets from *List* and return $T \cup$Analyze(*List*)
  else
    If *Candidate*(T) then reverse push into *List* all its descendants,
    starting from the position prior to the next element with just one
    feature
    else
      remove all its super-sets from *List*
    endIf
    return Analyze(*List*)
  endIf
endIf

TABLE 1. Binary matrix used for illustrating the strategies followed by the *KS* and *BR* algorithms.

a	b	c	d	e	f
1	0	0	0	1	0
1	1	0	0	0	1
0	0	1	0	0	1
1	0	0	1	0	1

C. COMPARING STRATEGIES

The goal of this subsection is to illustrate the similarities and differences between the strategies for computing minimal transversals / irreducible testors followed by the algorithms previously described. For this purpose, we will use the same binary matrix for both algorithms. This binary matrix is interpreted as the incidence matrix (transposed) of a hypergraph by the *KS* algorithm, and as a basic matrix by the *BR* algorithm.

As stated previously, both analyzed algorithms were redesigned to use a list as data structure for representing the set of vertices or features pending from revision. Since the *KS* algorithm was defined as a depth-first traversal in the search tree, its list is strictly used as a LIFO (Last In - First Out) structure. On each iteration, all the descendants of the currently analyzed set are generated and they are pushed in reverse order into the LIFO structure (called Stack), thus resulting in the expected depth-first transversal.

On the other hand, the searching order followed by the *BR* algorithm does not exactly fit neither a depth-first nor a breadth-first traversal, so its list allows direct access insertions, but always extracts the first element of the list.

The binary matrix selected for illustrating the strategy each algorithm follows for computing irreducible testors or minimal transversals is shown in Table 1.

In order to help the reader to follow the example, each column of Table 2 shows only the contents of the main data structure of each algorithm, as well as the element extracted from it in each iteration. From Table 2, it can observed that both algorithms follow completely different search strategies, however they yield the exact same answer.

As previously mentioned the strategy in the *KS* algorithm is to construct minimal transversals by testing only the entries in the input matrix. The depth-first search strategy used by this algorithm outperforms the breadth-first strategy previously proposed by Berge [10] since it does not need to wait until the last input matrix row is processed to know the set of minimal transversals found.

On the other hand, regarding *BR*, as the pseudocode in Algorithm 2 states, the reordering of the input matrix places the row with fewer ones on the top. Rows 1 and 3 have both only two 1s, but since row 1 has its leftmost 1 on the first column, then it is selected. Then, the 1s on the selected row must be placed in contiguous columns, placing column 'e' in second place. Following this ordering the final position for the input matrix columns is 'a, e, b, c, d, f'. This final position is used as the base for subsequent *lexicographic ordering* of column subsets, and that is why subset $\{a, e\}$ is analyzed before subsets $\{a, b\}$, $\{a, c\}$ and $\{a, d\}$ (see row 3 on Table 2).

Once the matrix has been reordered, and the columns with a 1 on the first row have been pushed into the main list (subsets $\{a\}$ and $\{e\}$), only descendants of candidate subsets will be pushed into that same list. Since those descendants are placed before the last element with just one feature (see the first else block in pseudocode on Function 2), the *BR*-order is preserved, effectively creating a depth-first traversal of a tree composed of all subsets extracted from the main list.

TABLE 2. Contents of the main data structures used by the *KS* and *BR* algorithms.

KS **Algorithm**	*BR* **Algorithm**
Stack=$\langle[\{(ae)\},1]\rangle$	List= $\langle\{a\},\{e\}\rangle$
Pop $[\{(ae)\},1]$	Get $\{a\}$ candidate
KSDesc $=\{\{(a)\},\{(e),(bf)\}\}$	List $=\langle\{a,e\},\{a,b\},\{a,c\},\{a,d\},\{a,f\},\{e\}\rangle$
Stack=$\langle[\{(a)\},2],[\{(e),(bf)\},2]\rangle$	Get $\{a,e\}$ exclusive
Pop $[\{(a)\},2]$	Get $\{a,b\}$ exclusive
KSDesc $=\{[\{(a),(c)\},3]\}$	Get $\{a,c\}$ irreducible testor
Stack=$\langle[\{(a),(c)\},3],[\{(a),(f)\},3],[\{(e),(bf)\},2]\rangle$	Get $\{a,d\}$ exclusive
Pop $[\{(a),(c)\},3]$	Get $\{a,f\}$ irreducible testor
KSDesc $=\{[\{(a),(c)\},4]\}$	Get $\{e\}$ candidate
Tr $=\{\{a,c\}\}$	List $=\langle\{e,b\},\{e,c\},\{e,d\},\{e,f\}\rangle$
Stack=$\langle[\{(a),(f)\},3],[\{(e),(bf)\},2]\rangle$	Get $\{e,b\}$ candidate
Pop $[\{(a),(f)\},3]$	List $=\langle\{e,c\},\{e,d\},\{e,f\},\{e,b,c\},\{e,b,d\},\{e,b,f\}\rangle$
KSDesc $=\{[\{(a),(f)\},4]\}$	Get $\{e,c\}$ candidate
Tr $=\{\{a,c\},\{a,f\}\}$	List $=\langle\{e,d\},\{e,f\},\{e,b,c\},\{e,b,d\},\{e,b,f\},\{e,c,d\},\{e,c,f\}\rangle$
Stack=$\langle[\{(e),(bf)\},2]\rangle$	Get $\{e,d\}$ exclusive
Pop $[\{(e),(bf)\},2]$	List $=\langle\{e,f\},\{e,b,c\},\{e,b,f\},\{e,c,f\}\rangle$
KSDesc $=\{[\{(e),(f)\},3],[\{(e),(b),(c)\},3]\}$	Get $\{e,f\}$ irreducible testor
Stack=$\langle[\{(e),(f)\},3],[\{(e),(b),(c)\},3]\rangle$	List $=\langle\{e,b,c\}\rangle$
Pop $[\{(e),(f)\},3]$	Get $\{e,b,c\}$ candidate
KSDesc $=\{[\{(e),(f)\},4]\}$	List $=\langle\{e,b,c,d\},\{e,b,c,f\}\rangle$
Tr $=\{\{a,c\},\{a,f\},\{e,f\}\}$	Get $\{e,b,c,d\}$ irreducible testor
Stack=$\langle[\{(e),(b),(c)\},3]\rangle$	Get $=\{e,b,c,f\}$ exclusive
Pop $[\{(e),(b),(c)\},3]$	List $=\oslash$
KSDesc $=\{[\{(e),(b),(c),(d)\},4]\}$	Irreducible Testors=$\{\{a,c\},\{a,f\},\{e,f\},\{e,b,c,d\}\}$
Tr $=\{\{a,c\},\{a,f\},\{e,f\},\{e,b,c,d\}\}$	

Each time an irreducible testor or an exclusive subset is extracted from the list, all its supersets are removed from the list to avoid unnecessary further tests.

VI. CONCLUDING REMARKS

In this paper, the relation between the concepts of irreducible testor and minimal transversal was identified. Although these concepts have different semantics and applications in each theory, and derived from the equivalence between a basic matrix and the incidence matrix of a simple hypergraph, they both share the same mathematical properties. As a consequence, an expanded perspective and a wider opportunity scenario is available to both fields, giving rise to diverse and potentially useful joint research lines.

As it was shown, the first and obvious benefit from the relation between the concepts of irreducible testor and minimal transversal lies in the possibility of choosing algorithms designed within one of the two fields and seamlessly use it to solve problems stated within the other. Each field has plenty of different algorithms based on diverse approaches; some of them have not been explored on one field or the other. Although covering all the possible future studies is out of the scope of this paper, we do want to at least sketch some of the immediate and rather promising ones.

Given differences in the strategies for computing irreducible testors and minimal transversals, it seems reasonable to assume the convenience of finding a unified, maybe hybrid, strategy, which takes the best from both fields for computing irreducible testors or minimal transversals. Although in both fields prevails some theoretical clarity about the impossibility of finding the absolute best algorithm for any problem instance (a No-Free-Lunch region), it would be very useful to unify the evaluation criteria used for current algorithms and building a generalized benchmark for all currently published algorithms. An interesting possibility for doing so is a somewhat novel strategy which has recently emerged from the testor theory arena, that postulates the convenience of creating synthetic basic matrices with controlled parameters for benchmarking purposes [17].

Testor Theory can be benefited from the possibility of exploiting the duality property for the design and validation of new testor-finding algorithms. On the other hand, graph theory can be benefited from several extensions made to the original concept of irreducible testor, that widen the spectrum of practical applications allowing a more versatile modeling process. For example, it would seem quite useful to explore the relationship between the concept of fuzzy hypergraph [21] and the concept of irreducible fuzzy testor [6].

Finally, it should be noted that the theoretical concepts of minimal transversal and irreducible testor have both been used for solving a rather disjoint set of practical problems. Irreducible testors, for instance, are generally associated with feature selection, as well as supervised classification; while transversal hypergraphs have been used mainly for data mining and associative network modeling. A unified approach that merges the fundamental strategies used by both fields would greatly improve the possibilities for tackling new practical problems efficiently.

REFERENCES

[1] M. Ortíz-Posadas, F. Martínez-Trinidad, and J. Ruíz-Shulcloper, "A new approach to differential diagnosis of diseases" *Int. J. Bio-med. Comput.*, vol. 40, no. 3, pp. 179–185, 2001.

[2] A. Pons-Porrata, R. Gil-García, and R. Berlanga-Llavori, "Using typical testors for feature selection in text categorization," in *Progress in Pattern Recognition, Image Analysis and Applications*. Berlin, Germany: Springer, 2007, pp. 643–652.

[3] A. Pons-Porrata, J. Ruiz-Shulcloper, and R. Berlanga-Llavori, "A method for the automatic summarization of topic-based clusters of documents," in *Progress in Pattern Recognition, Speech and Image Analysis*. Berlin, Germany: Springer, 2003, pp. 596–603.

[4] 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.

[5] A. N. Dmitriev, Y. I. Zhuravlev, and F. P. Krendeliev, "About mathematical principles and phenomena classification," (in Russian), *Diskretnyi Analiz*, no. 7, pp. 3–15, 1966.

[6] M. Lazo-Cortés, J. Ruiz-Shulcloper, "Determining the feature relevance for non-classically described objects and a new algorithm to compute typical fuzzy testors," *Pattern Recognit. Lett.*, vol. 16, no. 12, pp. 1259–1265, 1995.

[7] M. Lazo-Cortés, J. Ruiz-Shulcloper, E. Alba-Cabrera, "An overview of the evolution of the concept of testor," *Pattern Recognit.*, vol. 34, no. 4, pp. 753–762, 2001.

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

[9] J. Santos, A. Carrasco, and J. F. Martínez, "Feature selection using typical testors applied to estimation of stellar parameters," *Computación y Sistemas*, vol. 8, no. 1, pp. 15–23, 2004.

[10] C. Berge, *Hypergraphs: Combinatorics of Finite Sets*, vol. 45. Amsterdam, The Netherlands: Elsevier, 1989.

[11] T. Eiter and G. Gottlob, *Hypergraph Transversal Computation and Related Problems in Logic and AI*. Berlin, Germany: Springer, 2002, pp. 549–564.

[12] D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen, "Data mining, hypergraph transversals, and machine learning," in *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst.*, 1997, pp. 209–216.

[13] M. Hagen, "Lower bounds for three algorithms for transversal hypergraph generation," *Discrete Appl. Math.*, vol. 157, no. 7, pp. 1460–1469, 2009.

[14] E. Khaled, M. Hagen, and I. Rauf, "A Lower Bound for the HBC Transversal Hypergraph Generation," *Fundamenta Informaticae*, vol. 130, no. 4, pp. 409–414, 2014.

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

[16] D. J. Kavvadias and E. C. Stavropoulos, "An efficient algorithm for the transversal hypergraph generation," *J. Graph Algorithms Appl.*, vol. 9, no. 2, pp. 239–264, 2005.

[17] E. Alba-Cabrera, J. Ibarra-Fiallo, and S. Godoy-Calderon, "A theoretical and practical framework for assessing the computational behavior of typical testor-finding algorithms," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (Lecture Notes in Computer Science), vol. 8258. New York, NY, USA: Springer, 2013, pp. 351–358.

[18] E. Alba-Cabrera, J. Ibarra-Fiallo, S. Godoy-Calderon, and F. Cervantes-Alonso, "YYC: A fast performance incremental algorithm for finding typical testors," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (Lecture Notes in Computer Science), vol. 8827. New York, NY, USA: Springer, 2014, pp. 416–423.

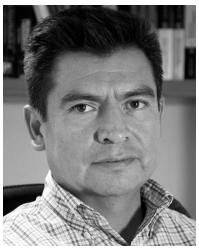
[19] A. Lias-Rodríguez and A. Pons-Porrata, "BR: A new method for computing all typical testors," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (Lecture Notes in Computer Science), vol. 5856. New York, NY, USA: Springer, 2009, pp. 433–440.

[20] L. Khachiyan, E. Boros, K. Elbassioni, and V. Gurvich, "A new algorithm for the hypergraph transversal problem," *Computing and Combinatorics*. Berlin, Germany: Springer, 2005, pp. 767–776.
[21] M. Akram and W. A. Dudek, "Intuitionistic fuzzy hypergraphs with applications," *Inf. Sci.*, vol. 218, pp. 182–193, Jan. 2013.

EDUARDO ALBA-CABRERA received the M.Sc. degree in mathematics from Kharkiv National University, Ukraine, and the Ph.D. degree in mathematics from the Instituto de Cibernética, Matemática y Física de La Habana, Cuba. Since 2001, he has been a Professor and a Researcher with the Universidad San Francisco de Quito, Ecuador, where he has been an Associate Dean of the Science and Engineering School, since 2015. His current research interests include pattern recognition, discrete algorithms, and testor theory.

SALVADOR GODOY-CALDERON was born in Mexico, Mexico, in 1968. He received the bachelor's degree in computer engineering from ITAM, Mexico, in 1992, the M.Sc. degree from CINVESTAV, in 1994, and the Ph.D. degree in computer science from the Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN), Mexico, in 2006, where he is currently the Head of the Artificial Intelligence Laboratory. His research interests include pattern recognition, testor theory, symbolic formal systems, and automated reasoning.

MANUEL S. LAZO-CORTÉS received the Ph.D. degree in mathematics from the Universidad Central de Las Villas, Cuba, in 1994. He has lectured and belongs to the teaching corpus of several postgraduate programs in universities in Cuba, Mexico, and other countries. He is currently a full-time Professor with the Graduate Division, Instituto Tecnológico de Tlalnepantla, which belongs to the Tecnológico Nacional de México. His current research interests include pattern recognition, data mining, testor theory, feature selection, and clustering.



J. FCO. MARTÍNEZ-TRINIDAD received the B.S. degree in computer science from the Physics and Mathematics School, Autonomous University of Puebla (BUAP), Mexico, in 1995, the M.Sc. degree in computer science from the Faculty of Computer Science, Autonomous University of Puebla, Mexico, in 1997, and the Ph.D. degree from the Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN), Mexico, in 2000. He is currently a member of the Computer Science Department, National Institute for Astro-Physics, Optics and Electronics (INAOE), Mexico. He has edited or authored 13 books and around 100 and 50 journals and conference papers on subjects related to pattern recognition.

JESÚS A. CARRASCO-OCHOA received the Ph.D. degree in computer science from the Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN), Mexico, in 2001. He is currently a part of the Staff of the Department of Computer Science, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Mexico. He has published more than 150 papers on topics related to pattern recognition and data mining, and co-edited 19 books. He has been part of the organizing committee of several international conferences and has served as part of the program committee for many international conferences and journals. He has directed and codirected more than 30 postgraduate theses. His current research interests include logical-combinatorial pattern recognition, data mining, testor theory, feature and prototype selection, document analysis, and clustering.