ISSN 2007-9737

Adding Learning Capabilities to the LEX Algorithm for Computing Minimal Transversals

Ingrid Guevara[1], Salvador Godoy-Calderon[2], Eduardo Alba[3]

[1] Escuela Politécnica Nacional, Quito, Ecuador

[2] Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico City, Mexico

[3] Universidad San Francisco de Quito, Colegio de Ciencias e Ingenierías, Quito, Ecuador

ingrid.guevara@epn.edu.ec , sgodoyc@cic.ipn.mx, ealba@usfq.edu.ec

Abstract. Despite being little known and poorly documented, LEX is part of the family of typical testors-finding algorithms that generally has better performance than other much more divulged similar algorithms. The recently published relationship between typical testors and minimal hitting sets, potentially extends the usefulness and applicability of this algorithm to the hypergraphs and data mining fields. Unfortunately, the high time-complexity of both typical testors and minimal hitting sets algorithms still remains a major obstacle. Therefore, alternatives that can help overcome difficult problems are constantly being researched. In this paper we propose the inclusion of a symbolic learning behavior into the implementation of the LEX algorithm. The incorporated symbolic learning is a general strategy for optimizing the search process, and thus improves the efficiency of minimal transversals and typical testors algorithms. In addition, the performance of the resulting algorithm is assessed by using carefully designed benchmark test matrices.

Keywords. LEX algorithm, learning strategy, minimal transversals, hypergraph, irreducible testor.

1 Introduction

The publication of the theoretical convergence between Typical Testors and Minimal Transversal concepts [2] opened new possibilities for the study, development, and application within the hypergraph and testor theories. Particularly, algorithms for computing the set of all typical testors, as well as those for computing minimal transversals, can now be applied interchangeably in any of these areas.

Graph theory is one of the most relevant fields of discrete mathematics because of its ability to model a wide range of problems.

The concept of minimal transversal, also known as minimal hitting sets [5], has been applied in relevant areas such as artificial intelligence, reliability theory, database theory, integer programming, and learning theory [8, 12].

Whereas in Pattern Recognition, testor theory is a useful tool for feature selection and evaluation solving various practical problems like medical diagnosis [17, 23, 24, 10], text categorization [6], document summarization [13], document clustering [14], stellar structure [17], dimension reduction in image databases [19], reduction of neural network models [25], number recognition [18], etc.

However, the high time complexity that minimal transversal finding algorithms have [13, 9] limits the possibility of applying these concepts in situations that require handling large amounts of

Algorithm 1. LEX (original)

Input: The incidence matrix B of a simple hypergraph $\mathcal{H}$ with n vertices.
Output: the set of all minimal transversals of B.

1: Sort B: Find the row with minimum amount of 1's, if there is more than one, choose any of them. Put it as first row in B. Sort columns of B putting as first the ones that have 1 in the first row.
2: Initialization: $L = [\,]$, $v = v_1$ (v first candidate of L).
3: Candidate Evaluation
   a. If $L = [\,]$ and the column corresponding to v has zero in the first row then END.
   b. If v is exclusive with L ((1) and (2)) go to 4.
   c. If $r(i, L + [v]) > 0$ for every row i of B, then save $L + [v]$ go to step 4.
   d. If $v = v_n$ go to step 4 (–).
   e. Do $L = L + [v]$. Update $r(i, L)$ for all rows of B and $F(v_t, L)$ for all v_t elements in L.
4: Selection of new candidate
   a. If $v \neq v_n$ then let j be the index of v in B, do $v = v_{j+1}$ and go to step 3.
   b. If $L = [\,]$ then END.
   c. If $L+[v]$ was a minimal transversal or v was not exclusive with L find v_p gap of $L + [v]$. Do $v = v_{p+1}$. Remove from L all items from v_p to the last vertex of L (–). Update $r(i, L)$ for all rows of B and $F(v_t, L)$ for each v_t in L.
   d. Else there is no v_p, END (–).
5: Do $v = v_q$, where v_q is the last vertex of L and $L = L \setminus [v_q]$. Update $r(i, L)$ for all rows of B and $F(v_t, L)$ for each v_t of L.
6: Return to step 4.

data, cases that would paradoxically be the most interesting and useful ones. The above limitation has encouraged the search for alternatives that allow an increase in the performance of these algorithms, so that the lowest possible computational cost can be reached.

The LEX algorithm [22] is a particular case of these algorithms: it imposes a lexicographic order over the power set of all vertices/features and uses that order for generating combinations of columns from its input Boolean matrix, which represents an incidence matrix of a hypergraph.

In order to deal with its exponential complexity and facilitate LEX application in a wider range of problems, improving its efficiency becomes a matter of major significance.

In that same context, the recent initiative of embedding learning strategies into minimal transversal computation algorithms [11] also becomes relevant.

This technique can be applied to a wide range of algorithms for computing typical testors and minimal transversals.

The symbolic model learned, as the host algorithm progresses, guides the search process for these algorithms by adding rules that allow a faster traversal of the search space, thus offering the possibility of a more efficient exploitation of the background knowledge, as well as of the newly acquired knowledge.

Since its publication, this technique has been tested only on two algorithms, BR [15] and YYC [4], in the same article where the technique was originally proposed.

However, in that article there is no description of the methodology for using it with other algorithms, nor are the modifications to the original algorithms made explicit.

Consequently, a more detailed explanation of the integration between a learning strategy and a host algorithm can help pave the way for using this technique on other algorithms.

On the other hand, since the BR and YYC algorithms use a very different search strategy, it is completely unknown whether a learning strategy can benefit the LEX algorithm in the same way.

Thus, in this article we incorporate a learning strategy into the LEX algorithm and test the result to assess its performance improvement over the original version of the same algorithm.

First, we determine the stages of the original algorithm in which the learning strategy can be used, then we proceed to integrate it and we show the code of the new algorithm, which we have called LEX*.

For testing purposes, we designed a set of benchmark matrices [1] varying the number of rows and columns. Among the test matrices there are some with high density of minimal transversals, that is a high number of minimal transversals with respect to the cardinality of the power set of features.

Also there are others with large dimensions but few minimal transversals. Also, the comparison criteria for assessing the performance of LEX* is clearly defined and uniformly used in all experiments. Finally, results from the experiments are reported and conclusions are drawn.

ISSN 2007-9737

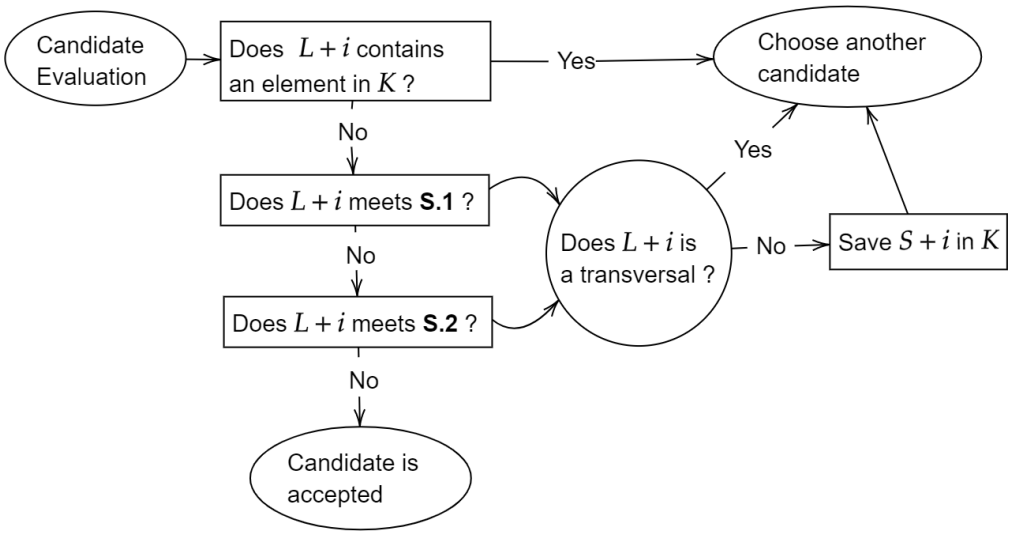


Fig. 1. Evaluation candidate i in LEX using the table of accumulated knowledge K

2 Conceptual and Theoretical Background

In graph theory, an hypergraph is a generalization of the graph concept, i.e. an ordered pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of objects and $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ a covering of $\mathcal{V}$ ($\mathcal{E}_i \neq \emptyset, i = 1, \ldots, m$ and $\cup_{i=1}^{m} \mathcal{E}_i = \mathcal{V}$), where each $\mathcal{E}_i$ is called an hyperedge. $\mathcal{H}$ is simple if for every pair $(\mathcal{E}_i, \mathcal{E}_j)$, $\mathcal{E}_j \subseteq \mathcal{E}_i \Rightarrow j = i$.

A transversal or hitting set of a simple hypergraph $\mathcal{H}$ is a subset of vertex that intersects all edges, i.e. a subset $\tau \subseteq \mathcal{V}$ such that $\forall \mathcal{E}_i \in \mathcal{E}, \tau \cap \mathcal{E}_i \neq \emptyset$. τ is minimal if no proper subset of τ is a transversal of $\mathcal{H}$.

The incidence matrix $A = [a_{ij}]_{m \times n}$ of $\mathcal{H}$ is a matrix whose rows and columns correspond to the vertices and 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.

2.1 Relation Between Minimal Transversals and Typical Testors

The terms irreducible testor and typical testor are used indistinctly in the following references. In [2], a theoretical convergence between the concepts of minimal transversals and typical testors was presented.

As in most research work, in testor theory a Boolean pairwise comparison matrix $A = [a_{ij}]_{m \times n}$ is considered, which holds the information of all objects contained in a partition set U of k disjoint classes described by a set of n features.

When an element $a_{ij} = 1$, it means that objects within pair i have different values in feature j, and $a_{ij} = 0$ as objects within pair i have similar values in feature j. The matrix is called a difference matrix.

Let R_A be the set of rows and $\mathcal{F}_A$ be the set of columns in A respectively, the matrix $A_{|\tau}$ is a submatrix containing all columns in the subset $\tau \subseteq \mathcal{F}_A$.

A subset of columns $\tau \subseteq \mathcal{F}_A$ is called a testor if $A_{|\tau}$ does not have any row composed exclusively by zeros, and is a typical testor if no proper subset of τ is a testor.

Generally, a difference matrix is commonly reduced for applications to a matrix called basic matrix. By analyzing the definitions of basic matrix, incidence matrix and simple hypergraph, the following theorems are stated and proved in [2]:

Theorem 1. A transposition over the incidence matrix of a simple hypergraph, results in a matrix that fulfills all required properties to be a basic matrix.

Theorem 2. Let $\psi^*(B) = \{\tau_1, \ldots, \tau_s\}$ be the complete family of typical 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 $B^\intercal$, then $\psi^*(B) = Tr(\mathcal{H})$.

Through this equivalence, computing the minimal transversals of a simple hypergraph $\mathcal{H}$ and computing the irreducible testors (also called typical testors) from the transposed incidence matrix of $\mathcal{H}$ can be done with either an algorithm for computing typical testors or an algorithm for computing minimal transversals.

2.2 Definitions and Properties of the LEX Algorithm

The strategy of LEX is to construct vertex lists that are minimal, i.e. lists where none if its subsets intersects the same amount of edges, and check whether this set is a transversal (originally presented as properties of typicality and testor respectively). LEX imposes a lexicographic order over the power set of all vertices, which is the reason of its name.

The order is denoted with the symbol $\ll$. For example, if we consider the set of vertices $V = \{v_1, v_2, v_3\}$ and the lists $[v_1, v_3]$, $[v_2, v_3]$, $[v_1, v_2, v_3]$ then $[v_1, v_2] \ll [v_1, v_2, v_3] \ll [v_2, v_3]$.

Let L be a vertex list and v_t a vertex of an incidence matrix A, we use the following notation:

- $F(v_t, L)$: the set of hyperedges that contain v_t but do not contain any other vertex of L.
- $r(i, L)$: the total of vertices v in L such that v belongs to $\mathcal{E}_i$.
- Gap of L: the maximum of the indexes in L such that the index of the next element is not consecutive.

For example, consider the following matrix:

$$\begin{array}{c} \\ \mathcal{E}_1 \\ \mathcal{E}_2 \\ \mathcal{E}_3 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}. \quad (1)$$

Algorithm 2. LEX*

Input: An incidence matrix B of a simple hypergraph $\mathcal{H}$ with n vertices.

Output: the set of all minimal transversals of B.

1: Sort B: Find the row with minimum amount of 1's, if there is more than one, choose any of them. Put it as first row in B. Sort columns of B putting as first the ones that have a 1 in the first row.
2: Initialization: $L = [\,]$, $v = v_1$ (v first candidate of L).
3: Candidate Evaluation
  a. If $L = [\,]$ and the column corresponding to v has zero in the first row then END.
  b. If $L + [v] \supseteq k$, where $k \in K$, then go to 4.
  c. If v is exclusive with L ((1) y (2)).

    If for all rows i, $r(i, L + [v]) > 0$ go to step 4

    Else: If (1) is met, let S be the smallest sub list of L such that $v_{j_k} \in s$ and $F(v, S) \subseteq U$. If (2) is met, let S be the smallest sublist of L such that $F(v, S)$ is empty. Add $S + [v]$ to K. Go to step 4.

  d. If $r(i, L + [v]) > 0$ for every row i of B, then save $L + [v]$ is a minimal transversal, go to step 4.
  e. If $v = v_n$ go to step 4 (–).
  f. Do $L = L + [v]$, the candidate is accepted. Update the values of $r(i, L)$ for all rows of B and $F(v_t, L)$ for all v_t elements in L.
4: Selection of new candidate
  a. If $v \neq v_n$ then let j be the index of v in B, do $v = v_{j+1}$.
  b. Go to step 3.
  c. If $L = [\,]$ then END.
  d. If $L+[v]$ was a minimal transversal or v was not exclusive with L find v_p gap of $L + [v]$. Do $v = v_{p+1}$. Remove from L all items from v_p up to the end (–). Update $r(i, L)$ for all rows of B and $F(v_t, L)$ for each v_t in L.
  e. Else there is no v_p, END (–).
  f. Do $v = v_q$, where v_q is the last feature of L and $L = L \setminus [v_k]$. Update $r(i, L)$ for all rows of B and $F(v_t, L)$ for each v_t of L.
  g. Return to step 4.

If $L = [v_1, v_3]$ then $F(v_3, L) = \{\mathcal{E}_1\}$, $r(1, L) = 1$, $r(3, L) = 2$ and L has gap 1. In [22], it is proven that the following statements are true:

Let be L and v_t a vertex not belonging to L.

(1) Let U be the set of hyperedges that contain v_t. If there is $v_k \in L$ such that $F(L, v_k) \subseteq U$, then v_t will not be part of any minimal transversal together with all the elements of L.

(2) If $F(v_t, L)$ is empty then v_t will not be part of any minimal transversal together with all the elements of L.

Table 1. Design of the benchmark matrices set

Matrix	Operator	N	Columns	Rows	Minimal Transversals
A	θ	1 to 5	$4N$	5^N	$4N$
	φ	1 to 5	$4N$	5	$4N^3$
	γ	1 to 10	$4N$	$5N$	4^N
B	θ	1 to 10	$10N$	2^N	$4N$
	φ	1 to 10	$10N$	2	$4N^2$
	γ	1 to 10	$10N$	$2N$	4^N
C	θ	1 to 4	$9N$	12^N	$8N$
	φ	1 to 5	$9N$	12	$5N^2 + 2N + N^3$
	γ	1 to 5	$9N$	$12N$	8^N

If either of the above cases is met, v_t is said to be exclusive with respect to L. Let L be a vertex list that contains the vertex of the last column in the incidence matrix. In all of the following the symbol ¡¡ is used with the semantics of not including ends.

– If L is a minimal transversal and has a gap p. Let L' be a vertex list obtained removing from L all the vertices from v_p up to the end and adding the vertex v_{p+1}. A list λ such that $L \ll \lambda \ll L'$ is a subsets of L and thus, λ is not a minimal transversal.
– If L is a minimal transversal that has no gaps. A list λ such that $L \ll \lambda$ is a subset of L and is not a minimal transversal.
– If L is not a transversal and p is a gap of L. Let L' be a vertex list obtained removing from L all the vertices from v_p up to the end and adding the vertex v_{p+1}, then a list λ such that $L \ll \lambda \ll L'$ is a subset of $L + [v]$ and it is not a transversal.

The original LEX algorithm is outlined in Algorithm 1.

3 Experimental Assesment

3.1 Implementing the Learning Strategy in LEX

The proposed learning strategy takes advantage of both the knowledge acquired during the search, and some prior knowledge about the nature of the objects sought (minimal hitting sets), identifying incompatible vertex sets.

In order to identify incompatible vertex combinations inside an incidence matrix A of a simple hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the strategy analyzes its structure.

If $A_{|\tau}$ is the submatrix obtained by removing all vertices from A that are not in τ with $\tau \subseteq \mathcal{V}$, it is possible to characterize transversals and minimal sets in structural (symbolic) terms: τ is a transversal iff the submatrix $A_{|\tau}$ does not contain any row composed exclusively by zeros.

τ is minimal iff $A_{|\tau}$ contains all the rows of an identity matrix. If τ satisfy both, it is a minimal transversal. If neither of them is satisfied, the set is undetermined.

Considering these concepts, the following relation between vertices can be defined:

Definition 1 (Domination)**.** Vertex v_1 dominates vertex v_2 iff $\forall \mathcal{E}_i \in \mathcal{E}, v_2 \in \mathcal{E}_i \implies v_1 \in \mathcal{E}_i$. That is, if in any row of the incidence matrix A, where v_2 has a one, v_1 also has it (single domination). This relation can also happen as a subset of vertices dominating another vertex (multiple domination).

Therefore, the following rules are defined:

– When finding a transversal not minimal, all its supersets are excluded from the rest of the search process since they cannot be minimal.

Table 2. Results for matrices generated from A

Matrices	N	Rows	Columns	Minimal Transversals	Percentage tests reduced	Accumulated knowledge
$\theta^N(A)$	1	4	5	4	0.00%	2
	2	16	10	8	23.24%	6
	3	64	15	12	27.83%	9
	4	256	20	16	29.05%	12
	5	1024	25	20	29.61%	15
$\gamma^N(A)$	1	4	5	4	0.00%	2
	2	8	10	16	31.02%	7
	3	12	15	64	36.13%	11
	4	16	20	256	36.87%	15
	5	20	25	1024	37.00%	19
$\varphi^N(A)$	1	4	5	4	0.00%	2
	2	4	10	32	16.00%	13
	3	4	15	108	25.00%	33
	4	4	20	256	30.32%	62
	5	4	25	500	33.80%	100
	6	4	30	864	36.24%	147
	7	4	35	1372	38.05%	203
	8	4	40	2048	39.44%	268
	9	4	45	2916	40.54%	342
	10	4	50	4000	41.43%	425

– When finding a minimal set, all of its subsets are also excluded because they cannot be part of a transversal.
– When finding a minimal transversal, all of its subsets and supersets are excluded, as they cannot be minimal transversals.
– Any other set found (undetermined) contains at least one pair of incompatible vertices and that pair must also be excluded from the rest of the search process.

Whereas rules 1 to 3 are embeded into the search strategies of all minimal transversals and irreducible testors finding algorithms, rule number 4 is commonly not considered and consequently it is of main interest for the implementation.

Moreover, it does not represent a potential risk to performance of the algorithm. Incompatibilities can be found as the host algorithm searches for minimal or trasnversal sets.

Notice that in LEX the algorithm excludes candidates if it finds that the current list and the current vertex candidate are exclusive, i.e. both meet rules 1 or 2. Thus, the steps where these conditions are checked, are essential to include rule number 4.

Considering all of the above, it seems clear that the crucial stages to incorporate the strategy are: partitioning the search space, identifying the order the algorithm follows and identifying the contribution the strategy can reach.

Table 3. Results for matrices generated from B

Matrices	N	Rows	Columns	Minimal Transversals	Percentage tests reduced	Accumulated knowledge
$\theta^N(B)$	1	2	10	4	31.58%	7
	2	4	20	8	68.63%	16
	3	8	30	12	77.27%	24
	4	16	40	16	79.67%	32
	5	32	50	20	80.50%	40
	6	64	60	24	80.68%	48
	7	128	70	28	80.98%	56
	8	256	80	32	81.11%	64
	9	512	90	36	81.17%	72
	10	1024	100	40	81.20%	80
$\gamma^N(B)$	1	2	10	4	31.58%	7
	2	4	20	16	71.2%	16
	3	6	30	64	79.2%	24
	4	8	40	256	81.0%	32
	5	10	50	1024	81.5%	40
	6	12	60	4096	81.7%	48
	7	14	70	16384	81.8%	56
	8	16	80	65536	81.8%	64
	9	18	90	262144	81.8%	72
	10	20	100	1048576	81.8%	80
$\varphi^N(B)$	1	2	10	4	31.58%	7
	2	2	20	16	48.65%	18
	3	2	30	36	54.55%	33
	4	2	40	64	57.53%	52
	5	2	50	100	59.34%	75
	6	2	60	144	60.55%	102
	7	2	70	196	61.42%	133
	8	2	80	256	62.07%	168
	9	2	90	324	62.58%	207
	10	2	100	400	62.98%	250

First, the candidate evaluation steps are going to define the search space.

3.2 Partition of the Search Space in LEX

In LEX, the candidate evaluation (step 3 in algorithm 1.) is the key to the classification of vertex sets in an incidence matrix through proposition (1) and (2). The current list and vertex are saved as indicated in rule 4, in a table of accumulated knowledge named K, for later use of the algorithm.

The combination between the background knowledge and the learned knowledge makes possible a smarter selection for the next vertex to evaluate.

When an exclusive vertex exists in LEX, it means the combination evaluated is a transversal not minimal, or an incompatible set. To fill K, the second case is of major interest.

– When a vertex set meets (1), it means the current candidate removes the minimal property from the analyzed vertex list. In other words, let i and L be the vertex and the list being reviewed in LEX respectively, where $L + i$ meet (1), then there is a subset $S \subseteq L$ that dominates i (definition 1).

– Another incompatible pair can be found by applying a similar reasoning to case (2). When the candidate to evaluate fits in this case means

Table 4. Results for matrices generated from C

Matrices	N	Rows	Columns	Minimal Transversals	Percentage tests reduced	Accumulated knowledge
$\theta^N(C)$	1	12	9	8	6.12%	1
	2	144	18	16	10.09%	2
	3	1728	27	24	10.99%	3
	4	20736	36	32	11.23%	4
$\gamma^N(C)$	1	12	9	8	6.12%	1
	2	24	18	64	50.27%	78
	3	36	27	512	57.05%	117
	4	48	36	4096	57.78%	156
	5	60	45	32768	57.91%	195
$\varphi^N(C)$	1	12	9	8	6.12%	1
	2	12	9	32	20.52%	11
	3	12	9	78	27.53%	30
	4	12	9	152	31.64%	58
	5	12	9	260	34.32%	95

it has 1's where one or more vertex of L already have unit values. Therefore, there is a subset $S \subseteq L$ such that $S + i$ dominates some vertex j in L, with j not in S.

Briefly, the candidate evaluation process for any vertex i and list L is explained in figure 1.

3.3 The New Algorithm

Finally, we show the LEX algorithm with the learning strategy using the established partition of its search space explained above. The pseudo code is presented in algorithm 2.. We call the new algorithm LEX*.

The host algorithm selects a vertex using its predefined lexicographic search order. With the acquired knowledge stored in table K, the tested subset is classified. If it includes a pair of exclusive vertex and sublist ((1) and (2)) the pair is classified as incompatible and K is updated.

Since the new steps reduce the number of elements tested by the algorithm, its performance is enhanced.

3.4 Experiments

Since we are improving an algorithm that is known for its exponential complexity [14], it seems necessary to perform a more detailed analysis of all experimental results.

The learning strategy incorporated into the original LEX algorithm aims at predicting all possible incompatibilities, that the imput matrix contain, and that the host algorithm will test.

Therefore, the performance of the proposed algorithm (LEX*) will be assessed with two relevant criteria: the number of vertex sets that the algorithm tests and the overall execution time.

We focus the assessment on the total number of vertex subsets and the proportion of omitted sets for obtaining conclusions because these criteria does not depend on the hardware characteristics of the equipment or on the programming language used.

Moreover, as the main reason for the use of the strategy is reducing the number of tested sets, this analysis seems more appropiate.

Table 5. Results for matrices from real-world data

Dataset	Rows	Columns	Transversals	Percentage test reduced	Accumulated knowledge
Sponge	81	44	6177	82.28%	26329
QSAR-biodeg	276	41	2881	66.89%	10180
Flag	195	29	1243	52.53%	5696
Cylinder	163	69	844	34.37%	1171

3.4.1 Description of the Benchmark Matrices Used in the Performed Experiments

In order to avoid testing an algorithm with a biased set of problems, we referred to [3, 1] where a set of matrix operators $(\theta, \gamma, \varphi)$ is defined in order to design test matrices.

These test matrices are usefull because the total number of minimal transversals they contain is known in advance. Starting from considerable small matrices, the operators allow the generation of new matrices with various characteristics.

Our experiments include matrices with high density of minimal transversals, as well as matrices with large dimensions but very few minimal transversals.

The operators allow the construction of matrices with a growing number of rows and columns in different ways, so we get three methods for generating our matrices by applying several times the same operator: growing the number of rows (using operator θ), growing the number of columns (using operator φ), and growing the number of rows and columns (using operator γ).

When an operator is applied N consecutive times over an incidence matrix M, we use $\theta^N(M), \gamma^N(B)$ or $\varphi^N(M)$ to represent the resulting matrices. Consider the following matrices:

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \tag{2}$$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \tag{3}$$

$$M_3 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \tag{4}$$

$$M_4 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{5}$$

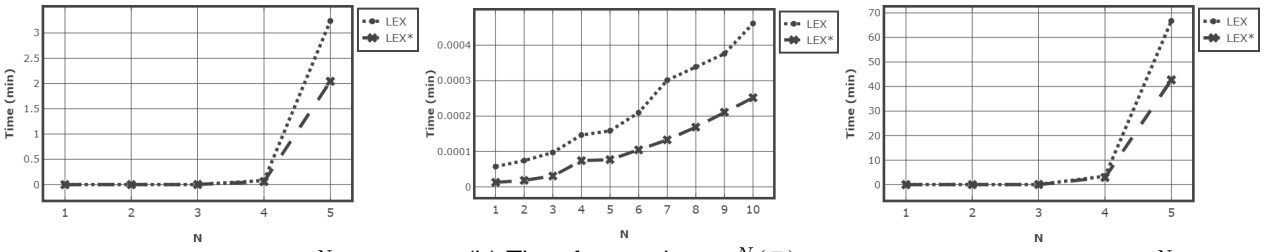
The initial incidence matrices used in these experiments possess the following structures:

– A matrix whose number of rows is similar to its number of columns, that is, it resembles a square matrix. We took the 4×5 matrix M_1 and we renamed it A.

– A matrix whose number of columns is greater than the number of rows. We applied operator φ twice to the 2×5 matrix M_4 obtaining a new matrix of dimension 2×10. We named it B.

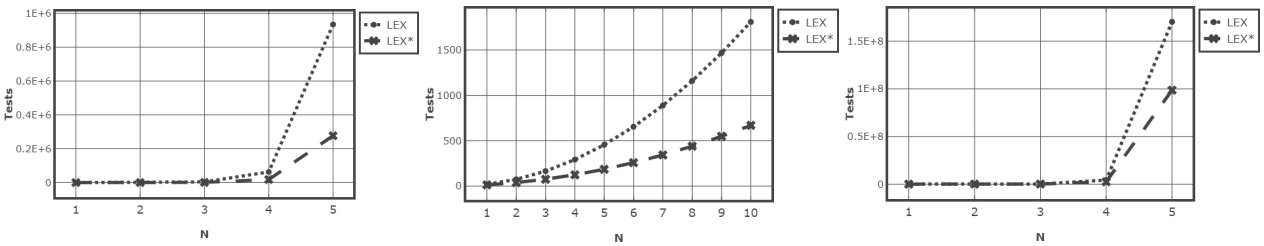
– Finally, a matrix such that the number of rows is greater than the number of columns. We took matrices M_2 and M_3 and in that order we applied operator θ obtaining a matrix of dimension 12×9 named C.

Starting from these three matrices, a benchmark matrix set is generated by consecutively applying the operators to each matrix for different values of N as shown in table 1. Notice that the θ operator generates the desired matrices with large dimensions but few minimal transversals, while γ



(a) Time for matrices $\theta^N(A)$ (b) Time for matrices $\varphi^N(B)$ (c) Time for matrices $\gamma^N(C)$

Fig. 2. Run time comparison between LEX and LEX*



(a) Tests for matrices $\theta^N(A)$ (b) Tests for matrices $\varphi^N(B)$ (c) Tests for matrices $\gamma^N(C)$

Fig. 3. Tests comparison between LEX and LEX*

generates matrices with high density of minimal transversals; matrices with these properties are very common in practice.

On the other hand, the θ operator allows the creation of matrices with a large number of rows, while the φ operator yields matrices with large number of columns. It is possible to reach some relevant conclusions based on these parameters.

Lastly, we use real-world datasets from true study cases to construct 4 incidence matrices: Sponge, Flags, QSAR-biodeg and Cylinder.

The datasets are available in the UC Irvine Machine Learning Repository [7] and are used on different works by the machine learning community.

4 Discussion

In order to properly assess the experimental results, we present tables containing the number of rows and columns, the number of minimal transversals, the number of registered incompatibilities (knowledge), and the proportion of omitted sets for each operator and each matrix.

In addition, two graphs are included, the first one corresponds to the run time of the LEX algorithm versus the time of LEX*, the other graph is the total number of tests in LEX and LEX*.

In both graphics, the dotted line with circular marker corresponds to LEX, and the dashed line with cross marker corresponds to LEX*.

As it was expected, the gain resulting from using the learning strategy increases when the dimensions of the input matrix increase. Here is a brief description of all experimental results tables, starting with matrix A.

In table 2, we observe that for operator θ from $N = 2$ between 20% and 30% of tests were reduced. For operator γ applied to matrix A there is up to 37% of omitted sets.

We see that there is more accumulated knowledge than in operator θ. The results for operator φ unveil that the number of sets in accumulated knowledge increases.

Also, up to $N = 10$ there is more than 40% of omitted tests. If we compare with previous operators up to $N = 5$ we see that more tests are dismissed than in θ but less than γ.

The results for matrix B are presented in table 3. Notably, it can be seen that, thanks to the learning strategy, the percentage of tests omitted exceeds 80%.

Although the same amount of knowledge has been accumulated in θ and γ matrices, the proportion of reduced test in γ is slightly higher.

Again, for φ there are more sets in accumulated knowledge. Fewer combinations are excluded than in the previous matrices generated from B, but we still get a good percentage of dismissed sets.

The difference between the performance of both algorithms for φ, is appreciated in Figures 2b and 3b. In table 4 is also shown that for the matrices obtained from basic matrix C the learning strategy reduce the number of reviewed candidates to minimal transversals in any of the three operators.

As in A and B scenarios, the highest percentage of reduced tests is obtained for the γ operator. Lastly, Figures 2c and 3c evidence the benefit of the strategy in matrices constructed from C for operator γ.

Finally, LEX* results for the real-world datasets matrices are in table 5. As in the synthetic matrices, there is a percentage of reduced tests for all matrices, which is higher for the matrices with more transversals.

5 Conclusions and Future Research

Results showed in the above section suggest how the supplementary knowledge about the problem search space supplied by the learning strategy enhances LEX's general efficiency.

The latter is measured through the number of tests performed and the run-time achieved using a set of designed benchmark matrices. However, we have not limited ourselves to presenting evidence of it, but also to studying how the strategy acts within LEX through these results. In all experiments the learning strategy identified incompatibilities which make possible that the host algorithm reduce further unnecessary set testing.

Although there is no procedure that allows to establish which algorithm is better overall, we can consider certain parameters that lead us to determine certain interesting behaviors.

In our experiments, we have focused on matrices with a large quantity of minimal transversals and matrices with large dimensions and a small number of minimal transversals.

Additionally, we have varied the number of rows and columns of the base matrices to obtain incidence matrices with diverse structures.

The knowledge table is consulted by the host algorithm in accordance with its search method, in the case of the LEX algorithm, the lexicographic order enables the number of learned sets to be small compared to the number of avoided tests.

An interesting observation is that all matrices to which the γ operator is applied report a higher percentage of dismissed sets.

It would not be appropriate to conclude that matrices from a specific operation are handled more efficiently by LEX*, as in practice not all matrices come from any of the matrix operations used in the experiments.

Nevertheless, the γ operator produces matrices with considerable amount of testors. Thus, according to the results obtained in this paper, the incorporated learning strategy in LEX reduces in greater proportion the number of tests within matrices with large numbers of minimal transversals in comparison with those with large dimensions and few minimal transversals which were obtained by using the operator θ. We also notice a similar behavior in the real-world data matrices from table 5.

Conversely, in θ^N matrices the number of incompatible sets in the knowledge table is less than or equal to the number of sets registered in the remaining operators. This may be due to their large size, or more specifically to the large number of rows that characterize these matrices. In most cases of φ^N matrices, whose distinctiveness is a higher number of columns than rows, LEX* has learned more than in θ^N and γ^N matrices.

In addition, matrices coming from B, which contains a small number of rows and several columns, show to be the ones where the strategy is the most advantageous.

The number of revisions has decreased more respect to the matrices coming from other base matrices. Matrices generated from A, whose structure is similar to a square matrix have a

smaller percentage of reduced tests in contrast with matrices coming from B. Lastly, those created from matrix C, which contain more rows than columns, are the ones with the lowest percentage.

Probably LEX* can reduce its search space in matrices which number of columns is higher than its number of rows, but also saves more incompatibilities in the knowledge table.

The results reached in these experiments can serve as a guide for future research work that contribute to the theoretical development in the testors field and, due to their theoretical convergence between concepts, also in the field of hypergraphs. Some of the following options may be considered:

– Contrast the behavior of other algorithms, for example these published in recent works [20, 21, 16], with symbolic learning integrated. Hence, find relationships between the matrix structure and the strategy's ability to decrease the number of tests if it is feasible.

– Likewise, check if there are certain search methods within algorithms that are more efficient than others when they are provided with the learning strategy support.

– Besides varying the dimensions of the matrix, it is possible to study the performance of LEX* using other parameters such as the ratio of inputs inside a basic matrix with unit values or zeros. Moreover, analyze their behavior in row echelon form matrices known to be difficult to handle because of their exorbitant number of minimal transversals.

– Comparing the number of learned incompatibilities that are stored in different types of algorithms, as well as how useful those are, can also be an interesting point.

Finally, besides providing a more detailed description of the functioning of the learning strategy, and a demonstration of its impact, this work motivated to contribute to the theoretical development of the fields involved, makes available the LEX* algorithm for applications in various problems capable of being modeled through minimal transversals or typical testors when needed.

Acknowledgments

Second author thanks the financial support for personal research work given by COFAA, SIP-IPN and CONACYT through grant SIP 20211424 and SIP 20221068. First and third authors received no financial support for this research.

References

1. **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 Recognition Letters, Vol. 80, pp. 46–51. DOI: 10.1016/j.patrec.2016.04.020.
2. **Alba-Cabrera, E., Godoy-Calderon, S., Lazo-Cortés, M. S., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A. (2019).** On the relation between the concepts of irreducible testor and minimal transversal. IEEE Access, Vol. 7, pp. 82809–82816. DOI: 10.1109/ACCESS.2019.2922231.
3. **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. Iberoamerican Congress on Pattern Recognition, DOI: 10.1007/978-3-642-41822-8_44.
4. **Alba-Cabrera, E., Ibarra-Fiallo, J., Godoy-Calderon, S., Cervantes-Alonso, F. (2014).** YYC: A fast performance incremental algorithm for finding typical testors. Iberoamerican Congress on Pattern Recognition, pp. 416–423. DOI: 10.1007/978-3-319-12568-8_51.
5. **Berge, C. (1984).** Hypergraphs: Combinatorics of finite sets. North-Holland: Mathematical Library, Vol. 45.
6. **Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F. (2004).** Feature selection for natural disaster texts classification using testors. Yang, Z. R., Yin, H., Everson, R. M. (eds) Intelligent Data Engineering and Automated

Learning–IDEAL 2004: 5th International Conference, Vol. 3177, pp. 424–429. DOI: 10.1007/978-3-540-28651-6_62.

7. **Dua, D., Graff, C. (2017).** UCI machine learning repository. [http://archive.ics.uci.edu/ml]. irvine, ca: University of California, School of Information and Computer Science.

8. **Eiter, T., Gottlob, G. (2002).** Hypergraph transversal computation and related problems in logic and ai. European Workshop on Logics in Artificial Intelligence, pp. 549–564. DOI: 10.1007/3-540-45757-7_53.

9. **Elbassioni, K., Hagen, M., Rauf, I. (2014).** A lower bound for the HBC transversal hypergraph generation. Fundamenta Informaticae, Vol. 130, No. 4, pp. 409–414. DOI: 10.3233/FI-2014-997.

10. **Gallegos, A., Torres, D., Álvarez, F., Soto, A. T. (2016).** Feature subset selection and typical testors applied to breast cancer cells. Research in Computing Science, Vol. 121, No. 1, pp. 151–163.

11. **González-Guevara, V. I., Godoy-Calderon, S., Alba-Cabrera, E., Calvo, H. (2019).** Symbolic learning for improving the performance of transversal-computation algorithms. IEEE Access, Vol. 7, pp. 19752–19761. DOI: 10.1109/ACCESS.2019.2895296.

12. **Gunopulos, D., Mannila, H., Khardon, R., Toivonen, H. (1997).** Data mining, hypergraph transversals, and machine learning. Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, pp. 209–216. DOI: 10.1145/263661.263684.

13. **Hagen, M. (2009).** Lower bounds for three algorithms for transversal hypergraph generation. Discrete Applied Mathematics, Vol. 157, No. 7, pp. 1460–1469. DOI: 10.1016/j.dam.2008.10.004.

14. **Kavvadias, D. J., Stavropoulos, E. C. (2005).** An efficient algorithm for the transversal hypergraph generation. Journal of Graph Algorithms and Applications, Vol. 9, No. 2, pp. 239–264.

15. **Lias-Rodríguez, A., Pons-Porrata, A. (2009).** Br: A new method for computing all typical testors. Iberoamerican Congress on Pattern Recognition, pp. 433–440.

16. **Lias-Rodriguez, A., Sanchez-Diaz, G. (2013).** An algorithm for computing typical testors based on elimination of gaps and reduction of columns. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 27, No. 08, pp. 1350022. DOI: 10.1142/S0218001413500225.

17. **Lopez-Perez, S., Lazo-Cortes, M., Estrada-Garcia, H. (1997).** Medical electro-diagnostic using pattern recognition tools. Proceedings of the iberoamerican workshop on pattern recognition (TIARP 97), pp. 237–244.

18. **Muenala, K., Ibarra-Fiallo, J., Intriago-Pazmiño, M. (2019).** Study of the number recognition algorithms efficiency after a reduction of the characteristic space using typical testors. New Knowledge in Information Systems and Technologies, Vol. 1, pp. 875–885. DOI: 10.1007/978-3-030-16181-1_82.

19. **Ochoa Somuano, J., Valdés Marrero, M. A., Moctezuma Cantorán, I., Ayala Esquivel, C. (2008).** Dimension reduction in image databases using the logical combinatorial approach. Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, pp. 260–265. DOI: 10.1007/978-1-4020-8735-6_49.

20. **Pino Gomez, J., Hernandez Montero, F. E., Charles Sotelo, J., Gomez Mancilla, J. C., Villuendas Rey, Y. (2021).** RoPM: An algorithm for computing typical testors based on recursive reductions of the basic matrix. Vol. 9, pp. 128220–128232. DOI: 10.1109/ACCESS.2021.3112385.

21. **Piza-Davila, I., Sanchez-Diaz, G., Lazo-Cortes, M. S., Noyola-Medrano, C. (2018).** Enhancing the performance of YYC algorithm useful to generate irreducible testors. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 32, No. 1, pp. 1860001. DOI: 10.1142/S0218001418600017.
22. **Santiesteban-Alganza, Y., Pons-Porrata, A. (2003).** LEX: A new algorithm for calculating typical testors. Revista Ciencias Matematicas, Vol. 21, No. 1, pp. 85–95.
23. **Torres, D., Torres, A., Ponce-de Leon, E. (2006).** Genetic algorithm and typical testors in feature subset selection problem. Proceedings of 6th iberoamerican conference on systemics, cybernetics and informatics, pp. 1–5.
24. **Torres, M. D., Torres, A., Cuellar, F., de la Luz Torres, M., de León, E. P., Pinales, F. (2014).** Evolutionary computation in the identification of risk factors. Case of Trali. Expert systems with applications, Vol. 41, No. 3, pp. 831–840. DOI: 10.1016/j.eswa.2013.08.013.
25. **Vázquez, R., Godoy-Calderón, S. (2007).** Using testor theory to reduce the dimension of neural network models. Special Issue in Neural Networks and Associative Memories, Vol. 28, pp. 93–103.

Article received on 19/05/2022; accepted on 01/03/2023. Corresponding author is Salvador Godoy-Calderon.