A Theoretical and Practical Framework for Assessing the Computational Behavior of Typical Testor-Finding Algorithms*

Eduardo Alba-Cabrera[1], Julio Ibarra-Fiallo[1], and Salvador Godoy-Calderon[2]

[1] Universidad San Francisco de Quito, Colegio de Ciencias e Ingeniería, Diego de Robles y Vía Interoceánica, Quito, Ecuador
[2] Instituto Politécnico Nacional-Centro de Investigación en Computación (CIC) Av. Juan de Dios Bátiz, Esq. Miguel Othón de Mendizábal. Col. Nueva Industrial Vallejo. D.F., México
{ealba,jibarra}@usfq.edu.ec, sgodoyc@cic.ipn.mx
http://www.usfq.edu.ec,
http://www.cic.ipn.mx

Abstract. Although the general relevance of Testor Theory as the theoretical ground for useful feature selection procedures is well known, there are no practical means, nor any standard methodologies, for assessing the behavior of a testor-finding algorithm when faced with specific circumstances. In this work, we present a practical framework, with proven theoretical foundation, for assessing the behavior of both deterministic and meta-heuristic testor-finding algorithms when faced with specific phenomena.

Keywords: Feature selection, testor theory, typical testor algorithms.

1 Introduction

A matrix A is called Boolean if all its entries are 0 or 1. Let $\mathcal{R}_A = \{a_1, ..., a_m\}$ and $\mathcal{C}_A = \{x_1, ...x_n\}$ be the set of rows and the set of columns of A, respectively. Two rows of $\mathcal{R}_A$, a_p and a_q, are incomparable if $(\exists i)\,[a_{pi} \geq a_{qi}] \wedge (\exists j)\,[a_{qj} \geq a_{pj}]$.

A row is called a basic row if it is incomparable with any other row from $\mathcal{R}_A$. A Boolean matrix is called a basic matrix if it is composed exclusively of basic rows. $T \subseteq \mathcal{C}_A$ is a testor in A if the submatrix $A|_T$, obtained by eliminating from A all columns not in the subset T, doesn't have any zero rows. Also, T is a typical testor if no subset of T can be found that is also a testor in A.

Typical testors play an important role in solving some feature selection [4] [14], diagnosis of diseases [8], text categorization [10], document summarization [9] and document clustering [6]. The concept of typical testor has been extended and generalized in several ways [5]. The problem of finding the set $\Psi^*(A)$ of all typical

* This research was supported by Collaboration Grants from Universidad San Francisco de Quito, Ecuador. The third author is also grateful for the finantial support by Instituto Politécnico Nacional and CONACyT, México, particularly through project SIP-20130932

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 351–358, 2013.
© Springer-Verlag Berlin Heidelberg 2013

testors in a basic matrix A is an old problem that has had an important development in the last ten years. To support this statement, consider the number of published papers with new algorithms related to this problem [3] [13] [7] [12].

There are two classes of typical testor-finding algorithms (TTAs): deterministic and meta-heuristic. Deterministic algorithms guarantee that they will find all typical testors at the expense of an exponential complexity. On the other hand, meta-heuristic algorithms have no guarantee to find all the typical testors in a given problem, but they are feasible to be used on extremely large search spaces [1] [3]. The complexity of deterministic TTAs has not been sufficiently studied. This lack of sufficient study can be regarded as the cause of why most published works about TTAs fail, in the opinion of the authors of this paper, to properly justify their selection of basic matrices for comparative performance experimentation between different algorithms. On one hand, since the number of matrices selected for experimentation is considerably low, the obtained results lack statistical significance. On the other hand, by not using a specific criterion for selecting test matrices and testing any new algorithm with the same matrices the characteristic behavior of each algorithm in the presence of certain stereotypical phenomena is not captured.

However, a formal and convenient strategy for selecting matrices for algorithm testing is certainly viable. In [2], a feasible strategy for studying the behavior of the TTAs was presented for the first time. That strategy was based on the construction of test matrices (TM) which are basic matrices whose sets of typical testors can be determined in advance. This property allows the assessment of the behavior of the computational implementation of any deterministic TTA, as well as the validation of the answer completeness of any meta-heuristic TTA. Since both the amount of typical testors and their length can be preset, TMs can be generated for studying the behavior of an algorithm varying only one parameter at a time. For example, we can consider the exponential increase in the number of matrix rows with only a linear increase in the number of typical testors, or the opposite phenomenon, a linear increase in the number of matrix rows, resulting in an exponential growth of the number of typical testors.

In this paper, we worked along two main directions. First, we significantly extended the theoretical framework of the TM strategy to allow the generation of a whole new set of TMs that are more flexible and versatile. Second, we show how TMs can be used to study the behavior of a TTA in the presence of specific phenomena. We also selected three previously published TTAs, tested them against specific TMs, and reported and discussed the obtained results.

2 Theoretical Background

Let $A = [a_{ij}]_{m \times n}$ and $B = [b_{ij}]_{m' \times n'}$ be two basic matrices. In [2] the operators ϕ and θ were defined on pairs of matrices A and B. The result of a φ operation is a new Boolean matrix obtained by concatenating two basic matrices with the same number of rows. The resulting matrix has exactly the same number of rows of A and B, but it has $n + n'$ columns (the sum of the number of columns from

A and B). On the other hand, the θ operator produces a new matrix having $m \times m'$ rows (the product of the number of rows in A and B), and also having $n + n'$ columns.

One important property of the φ and θ operators is that, when applied to basic matrices, the resulting matrix is also basic, since it preserves the portion of the matrix that guarantees incomparability of the rows. Moreover, it can be demonstrated that if A, B and C are basic matrices, the φ operator is associative. As a consequence, we will write $\varphi^N(A)$ to represent the resulting matrix of applying the φ operator to the matrix A N times. Likewise, the θ operator is also associative, so we will write $\theta^N(A)$ to represent the result of applying the θ operator consecutively N times.

Now, let $\mathcal{C}_A = \{x_1, ...x_n\}$ be the set of columns in a basic matrix A, and let $x_j \in \mathcal{C}_A$. We will write $[x_j]_N$ to denote the class of all columns in A exactly equal to x_j in $\varphi^N(A)$. In other words, $[x_j]_N = \{x_j, x_{j+n}, ..., x_{j+(N-1)n}\}$.

Given $S \subseteq \mathcal{C}_A$ and $S = \{x_{j_1}, ..., x_{j_s}\}$, $[S]_N$ will denote the set of all subsets of columns from $\varphi^N(A)$ that can be obtained by replacing one or more columns in S with any other column in the same class, that is,$[S]_N = [x_{j_1}]_N \times ... \times [x_{j_s}]_N$. Then it is easy to verify that $|[S]_N| = N^s = N^{|S|}$.

Therefore, if A and B are basic matrices such that the sets $\Psi^*(A)$ and $\Psi^*(B)$ of all typical testors are known, then the next two propositions establish how the sets $\Psi^*\left(\varphi^N(A)\right)$ and $\Psi^*(\theta(A,B))$ can be obtained from them.

Proposition 1. $\Psi^*\left(\varphi^N(A)\right) = \{[T]_N \mid T \in \Psi^*(A)\}$.

Proposition 2. $\Psi^*(\theta(A,B)) = \Psi^*(A) \cup \Psi^*(B)$.

Proposition 1 is proved by observing that, with the exception of the order of the columns, the submatrices in $\varphi^N(A)$ that form the elements of $[T]_N$ are always identical to $A|_T$. This is, because a column in T can only be replaced by another from the same class, and therefore all elements in $[T]_N$ are, by definition, typical testors. Now, let's make the assumption that some typical testor $S = \{x_{j_1}, ..., x_{j_s}\}$ exists outside $\{[T]_N \mid T \in \Psi^*(A)\}$. There must be at least one column of S that is not part of A. Lets replace all columns in S with the column from A within their same equivalence class. Then we would have a contradiction, because the resulting submatrix must determine a typical testor, which must be in A, and therefore S must be in $[T]_N$.

In order to prove Proposition 2, it is enough to observe that if we identify in $\theta(A,B)$ the columns from A and from B, and we eliminate repeated rows in each set, we end up with the original matrices A and B. Therefore, the set of typical testors in A and B is preserved in $\theta(A,B)$. Also, we cannot find any testor in $\theta(A,B)$ with columns of both matrices, because if the selected columns were testors in A or in B, then we would be constructing supersets of testors, and they would not be typical any more. On the other hand, if the selected columns were not testors, the following reasoning applies:

Let S_A and S_B be sets of non-testor columns from A and B respectively. Let also a be the row from A with zeros in the columns of S_A , and let b be the row from B with zeros in the columns of S_B. Since the row $[ab]$ is in $\theta(A,B)$ the

Table 1. Test Matrices $\varphi^N(B)$ and $\theta^N(\theta(A,B))$

N	Rows	Cols	$\lvert\Psi^*\left(\varphi^N(B)\right)\rvert$	N	Rows	Cols	$\lvert\Psi^*\left(\theta^N(\theta(A,B))\right)\rvert$
1	4	5	4	1	16	10	8
2	4	10	18	2	256	20	16
...	...	...	...	...	...	...	...
N	4	$2N$	$N+2N^2+N^3$	N	16^N	$2N$	$8N$

submatrix of $\theta(A,B)$, formed by $S_A \cup S_B$ must also have a row with zeros, and therefore it cannot be a testor.

So, we have that $|\{[T]_N \mid T \in \Psi^*(A)\}| = \sum_{T\in\Psi^*(A)} [T]_N = \sum_{T\in\Psi^*(A)} N^{|T|}$, and also that within $\theta(A,B)$, $\Psi^*(A) \cap \Psi^*(B) =$. These two properties allow us to state the following corollaries from Propositions 1 and 2.

Corollary 1. $\left|\Psi^*\left(\varphi^N(A)\right)\right| = \sum_{T\in\Psi^*(A)} N^{|T|}$

Corollary 2. $|\Psi^*(\theta(A,B))| = |\Psi^*(A)| + |\Psi^*(B)|$

Example 1. Let $A = \begin{array}{c} x_1\ x_2\ x_3\ x_4\ x_5 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$ and $B = \begin{array}{c} x_6\ x_7\ x_8\ x_9\ x_{10} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$

We can verify that $\Psi^*(A) = \{\{x_1,x_2,x_3\},\{x_1,x_2,x_5\},\{x_1,x_3,x_4\},\{x_1,x_3,x_5\}\}$, and $\Psi^*(B) = \{\{x_6\},\{x_8,x_{10}\},\{x_7,x_9,x_{10}\},\{x_7,x_8\}\}$.Note that A has 4 typical testors of length 3. By using Corollary 1 above, we know that $\left|\Psi^*\left(\varphi^3(A)\right)\right| = 4 * 3^3 = 108$. Likewise, since B has one typical testor of length 1, two of length 2, and one of length 3, we can establish using the same corollary that $\left|\Psi^*\left(\varphi^3(B)\right)\right| = 3^1 + 2 * 3^2 + 3^3 = 48$. We can also determine how many typical testors to expect in $\theta(A,B)$ using corollary 2: $|\Psi^*(\theta(A,B))| = 8$.

Test matrices allow us to control particular aspects that we wish to study regarding the performance of a TTA. In Table 1, the effect of appying the operator φ^N to matrix B from example 1 is shown.As can be seen, the generated matrices preserve the same number of rows, while the number of columns increase linearly; but the resulting number of typical testors grows according to a cubic polynomial.

If one wishes to assess the effect of an exponential growth in the number of rows while sustaining a linear behavior of both the number of columns and the number of typical testors, then one would only need to use the test matrix $\theta^N(\theta(A,B))$ with A and B being exactly those from example 1. The resulting behavior of such test is shown in Table 1.

3 Taxonomy and Nature of TTAs

Deterministic TTAs can be classified in two sets: external and internal. External TTAs always set an order to test the power set of columns in the basic matrix.

Following that order, each subset of columns is tested to determine if it is a typical testor or not. However, the test process is not an exhaustive search over the power set. Some properties of each tested subset allow the TTA to infer which other successive subsets, following the established order, cannot be typical testors, and therefore it is not worthwhile to test them. The act of bypassing the test of some subsets of columns is commonly referred to as *jumping*. In general, the selected order for traversing the power set of columns, along with the magnitude of the jumps (i.e. the number of subsets not tested), and the specific procedure applied to a subset for testing if it is a typical testor or not, determine the behavior of an external TTA.

On the other hand, internal TTAs do not test the power set of columns in a basic matrix. Their strategy lies in iteratively selecting the entries in the basic matrix and using them to construct typical testor candidates. An extended version of this work, which includes analysis and comparison of experimental results, both on internal and external TTAs as [7], is being prepared. In order to meet the space requirements, this work will exclusively show results obtained with experiments on external TTAS.

Three external TTAs were selected for experimentation with representative TMs. These algorithms are *BT* [11], *LEX* [13], and *FastCTExt* [12]. In the next section, all the above mentioned algorithms are tested against specifically selected families of TMs, and the obtained results discussed.

4 Experimental Results

As an example of how the TMs framework can be used for TTA testing purposes, we designed specific experiments to asses the TTA's performance when facing different phenomena. For each experiment, a custom-designed TM was created with a specific combination of the θ and φ operators, applied to the A and B matrices, as well as some identity matrices. Identity matrices are denoted by I_N, where N is the dimension of the identity matrix.

All experiments were run on an Intel i7 processor, with 4GB in RAM. However, since the ultimate goal of this work is to promote the usefulness of the proposed framework, and not to rigorously test each algorithm, absolute execution times as well as the hardware platform, are not relevant. For the intended goal, relative execution times are sufficient.

The first set of experiments was designed with successive powers of the φ operator, applied to the identity matrix I_5. Each one of those operations generates a basic matrix with 5 rows, but with a linearly increasing number of columns, and with a number of typical testors equal to the selected power of φ raised to the power of the dimension of the matrix (5) . Algorithms LEX and $FastCTExt$ were tested against each matrix, and in order to asses the resulting performance in each case, their execution times are recorded relative to the time needed for solving the first experiment (the one described in the first row of the table). Table 2 summarizes all the experiments performed.

As in the reports from several research works, Table 2 shows that the performance of the LEX and BT algorithms are clearly below that of the $FastCTExt$

Table 2. Relative execution times of the LEX, $FastCTExt$, and BT algorithms when facing powers of the φ operator, applied to the identity matrix I_5.

#	Test Matrix	Rows	Cols	$\lvert\Psi^*\rvert$	LEX	FastCTExt	BT
1	$\varphi^1(I_5)$	5	5	1	x	y	z
2	$\varphi^2(I_5)$	5	10	32	$4x$	y	$2z$
3	$\varphi^3(I_5)$	5	15	243	$22x$	$9y$	$87z$
4	$\varphi^4(I_5)$	5	20	1024	$86x$	$31y$	$2,113z$
5	$\varphi^5(I_5)$	5	25	3,125	$246x$	$90y$	$37,246z$

algorithm. However, the fundamental premise of this work is that without using a practical test framework capable of generating a wide diversity of phenomena, it is not possible to identify performance bottlenecks and special cases where a different processing technique is needed. As concrete evidence of this premise, the reader should consider the next set of experiments, where different combinations of the θ and φ operators applied to the reference matrices are used to induce specific test phenomena for the same algorithms. Table 3, summarizes the basic matrix phenomena that both algorithms were confronted with. Like in the previous set of experiments, all execution times are expressed relative to the time recorded in the first experiment.

As recorded in Table 3, the LEX algorithm turns out to be far more sensible to an increase in the number of columns of the basic matrix than to an increase in its number of rows. When the number of rows is kept constant and the number of columns and typical testors only slightly increase (rows 2, 3, 5, 6 & 7), the execution time also stays approximately the same. However, when the number of rows is doubled, and the number of columns (and typical testors) is kept constant (rows 4, 7 & 10), LEX's execution times almost doubles.

The $FastCTExt$ algorithm roughly follows the same behavior pattern, although experiments show it to be even more sensitive to the same phenomenon than the LEX algorithm. Rows 9 and 10, in Table 3, show the exact quantification for this phenomenon. By observing rows 9 and 10, it seems obvious that when doubling the number of columns and typical testors (from 46 to 96

Table 3. Relative execution times of the LEX and $FastCTExt$ algorithms when facing alternate powers of θ and φ operators

#	Test Matrix	Rows	Cols	$\lvert\Psi^*\rvert$	LEX	FastCTExt
1	$S_1 = \theta(A, B)$	16	10	8	x	y
2	$S_2 = \varphi^2(S_1)$	16	20	50	$17x$	$22y$
3	$S_3 = \theta(S_2, \theta^2(I_1))$	16	22	52	$17x$	$23y$
4	$S_4 = \theta(S_2, \theta^{20}(I_1))$	16	40	70	$18x$	$40y$
5	$S_5 = \theta(S_2, I_2)$	32	22	51	$54x$	$65y$
6	$S_6 = \varphi^2(S_5)$	32	44	360	$2,009x$	$4,245y$
7	$S_7 = \theta(S_6, \theta^2(I_1))$	32	46	362	$2,021x$	$4,321y$
8	$S_8 = \theta(S_6, \theta^{44}(I_1))$	32	88	404	$2,044x$	$5,113y$
9	$S_9 = \theta(S_6, I_2)$	64	46	361	$5,466x$	$11,671y$
10	$S_{10} = \varphi^2(S_9)$	64	92	2716	$486,052x$	$2,147,293y$

Table 4. Relative execution times of the LEX, $FastCTExt$, and BT algorithms using identity matrices of different sizes

#	Test Matrix	Rows	Cols	$\|\Psi^*\|$	LEX	$FastCTExt$	BT
1	I_5	5	5	1	x	y	z
2	I_{10}	10	10	1	$25x$	$7y$	$0.25z$
3	I_{15}	15	15	1	$885x$	$231y$	$0.5z$
4	I_{20}	20	20	1	$35,574x$	$7,758y$	$0.5z$
5	I_{25}	25	25	1	$1428772x$	$259,945y$	$0.75z$

columns), LEX's execution times increases by a factor 88, while $FastCTExt$'s time increases to 296 times that of the previous experiment. The same behavior, in a lesser magnitude, can also be observed in rows 3 and 4 of the same table.

For the last set of experiments, a comparison of the three studied algorithms is performed, using simple identity matrices. The results are summarized in Table 4. Clearly, an identity matrix has only one typical testor, regardless of its dimension. Surprisingly enough, the BT algorithm turns out to have the best performance, far beyond those of the LEX and $FastCTExt$ algorithms. This particular behavior could not be observed without the proposed framework.

5 Conclusions and Recommendations

Deterministic TTAs, as previously discussed, are guaranteed to find the complete set of typical testors in a basic matrix. The order in which a TTA traverses the search space, as well as the particular pre-search procedures it applies to the basic matrix, ultimately determine its behavior and general performance. However, the TTA's sensitivity to specific phenomena, such as the growth in rows, columns, or typical testors on the basic matrix is, in general, not sufficiently assessed.

We have presented a theoretical and practical framework that allows a researcher to test a TTA against specific pre-designed phenomena. The combination of θ and φ operators, in conjunction with sufficiently studied basic matrices, turns out to be a versatile tool for generating almost any conceivable phenomenon a researcher could wish a TTA to confront. The behavior observed during those tests can potentially yield enough information for identifying performance bottle-necks, and help design the appropriate fine-tuning procedures.

Meta-heuristic TTA's, on the other hand, are pseudo-random search procedures that, by nature, don't offer enough guarantees about the completeness of the resulting typical testors set. In order to validate these kind of TTA's, a wise course of action is to test the TTA against sufficiently studied basic matrices (those for which the total number of typical testors is known in advance). Matrices that satisfy such requirements are generally small ones, not suited for serious testing purposes. The TM framework herein proposed allows the generation, based on one original matrix, of increasingly larger matrices, with any desired dimensions, for which the total number of typical testors is always known.

In conclusion, TTA testing, under realistic conditions, appears to be a valuable asset for advancing the general state-of-the-art in pattern recognition.

By providing practical means for testing both deterministic and meta-heuristic TTAs, the TM framework seems to fullfill the current gap between theoretical developments and practical implementations.

References

1. Alba-Cabrera, E., Santana, R., Ochoa-Rodriguez, A., Lazo-Cortes, M.: Finding typical testors by using an evolutionary strategy. In: Proceedings of the V Ibero American Symposium on Pattern Recognition, pp. 267–278 (2000)
2. Alba, E., Guilcapi, D., Ibarra, J.: New strategies for evaluating the performance of typical testor algorithms. In: Alvarez, L., Mejail, M., Gomez, L., Jacobo, J. (eds.) CIARP 2012. LNCS, vol. 7441, pp. 813–820. Springer, Heidelberg (2012)
3. Diaz-Sanchez, G., Piza-Davila, I., Sanchez-Diaz, G., Mora-Gonzalez, M., Reyes-Cardenas, O., Cardenas-Tristan, A., Aguirre-Salado, C.: Typical Testors Generation Based on an Evolutionary Algorithm. In: Alagar, V.S., Nivat, M. (eds.) AMAST 1995. LNCS, vol. 936, pp. 58–65. Springer, Heidelberg (1995)
4. Lazo-Cortes, M., Ruiz-Shulcloper, J.: Determining the feature relevance for non classically described objects and a new algorithm to compute typical fuzzy testors. Pattern Recognition Letters 16, 1259–1265 (1995)
5. Lazo-Cortes, M., Ruiz-Shulcloper, J., Alba-Cabrera, E.: An overview of the evolution of the concept of testor. Pattern Recognition 34(4), 753–762 (2001)
6. Li, F., Zhu, Q.: Document clustering in research literature based on NMF and testor theory. Journal of Software 6(1), 78–82 (2011)
7. Lias-Rodríguez, A., Pons-Porrata, A.: BR: A New Method for Computing All Typical Testors. In: Bayro-Corrochano, E., Eklundh, J.-O. (eds.) CIARP 2009. LNCS, vol. 5856, pp. 433–440. Springer, Heidelberg (2009)
8. Ortiz-Posadas, M., Martinez-Trinidad, F., Ruiz-Shulcloper, J.: A new approach to diferential diagnosis of diseases. International Journal of Biomedical Computing 40(3), 179–185 (2001)
9. Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavori, R.: A method for the automatic summarization of topic-based clusters of documents. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 596–603. Springer, Heidelberg (2003)
10. Pons-Porrata, A., Gil-García, R., Berlanga-Llavori, R.: Using Typical Testors for Feature Selection in Text Categorization. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 643–652. Springer, Heidelberg (2007)
11. Ruiz-Shulcloper, J., Bravo, M., Aguila, F.: Algoritmos BT y TB para el cálculo de todos los tests típicos. Revista Ciencias Matemáticas 6(2) (1982)
12. Sanchez-Diaz, G., Lazo-Cortes, M., Piza-Davila, I.: A fast implementation for the typical testor property identication based on an accumulative binary tuple. International Journal of Computational Intelligence Systems 5(6) (2012)
13. Santiesteban-Alganza, Y., Pons-Porrata, A.: LEX: A new algorithm for calculating typical testors. Revista Ciencias Matematicas 21(1), 85–95 (2003)
14. Vázquez, R., Godoy-Calderon, S.: Using testor theory to reduce the dimension of neural network models. Special Issue in Neural Networks and Associative Memories 28, 93–103 (2007)