International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 32, No. 1 (2018) 1860001 (18 pages)

DOI: 10.1142/S0218001418600017

World Scientific
www.worldscientific.com

Enhancing the Performance of YYC Algorithm Useful to Generate Irreducible Testors

Ivan Piza-Davila[*,§], Guillermo Sanchez-Diaz[†,¶],
Manuel S. Lazo-Cortes[‡,‖] and Cristina Noyola-Medrano[†,**]

[*]*Instituto Tecnologico y de Estudios Superiores de Occidente*
Periferico Sur Manuel Gomez Morin 8585, Tlaquepaque, Jalisco 45604, Mexico
[†]*Facultad de Ingenieria, Universidad Autonoma de San Luis Potosi*
Dr. Manuel Nava 8, San Luis Potosi, SLP 78290, Mexico
[‡]*Instituto Nacional de Astrofisica, Optica y Electronica*
Luis Enrique Erro No. 1, Tonantzintla, Puebla 72840, Mexico
[§]*hpiza@iteso.mx*
[¶]*guillermo.sanchez@uaslp.mx*
[‖]*mlazo@inaoep.mx*
[**]*cristina.noyola@uaslp.mx*

Received 17 November 2016
Accepted 20 April 2017
Published 6 July 2017

In pattern recognition, irreducible testors have been used for feature selection. A number of exhaustive algorithms that find irreducible testors have been reported in the literature. One of the latest and more efficient algorithms reported is YYC, an incremental algorithm that finds all the irreducible testors from a training matrix. Its efficiency relies on building a smaller number of feature combinations by finding compatible sets from the top of the matrix to the current row. Nevertheless, as the number of sets currently found grows, YYC execution becomes too slow. This work proposes two improvements of YYC algorithm, incorporated in a pre-processing phase; additionally, a parallel version is implemented. The paper presents some experimental results using synthetic and real data.

Keywords: Irreducible testors; feature selection; pattern recognition; supervised classification.

1. Introduction

Feature selection is a significant task in supervised classification and other pattern recognition areas. It identifies those features that provide relevant information for the classification process. The problem of feature subset selection has been treated using multi-objective point of view,[10] metaheuristics,[6,21] etc. Nevertheless, results at this time are not conclusive.

¶Corresponding author.

The concept of testor (initially named test) in pattern recognition problems was introduced by Zhuravlev.[5] He defined a testor as a subset of features that allows differentiating objects from different classes. A testor is called irreducible, or typical, if it is not a superset of another testor. This concept has been extended and generalized in several ways.[7,25] In addition, other testor variations like irreducible descriptors have been proposed.[26] The testor theory[7] has been focused on feature selection tasks under the denominated Logical Combinatorial Pattern Recognition approach.[9,13] Computing the whole set of irreducible testors is NP-complete, with respect to the number of features.[24]

Irreducible testors have been used to solve several problems related to data mining and pattern recognition, including the following: medical diagnosis in terms of patients descriptions,[23] identification of risk factors in transfusion related acute lung injury,[22] dimension reduction in image databases,[11] topic detection in documents,[12] among others. Several exhaustive algorithms to generate the whole set of irreducible testors have been developed; some of the algorithms reported in the state-of-the-art include the following[2,4,8,14,18–20]:

One of the last reported exhaustive algorithm to generate irreducible testors is YYC.[1] It is an incremental algorithm that generates at row r the set of irreducible testors corresponding to the submatrix constituted by the first r rows of the input matrix. This entire set is used as input to build the next set of irreducible testors at row $r+1$. The main contribution of YYC with respect to other state-of-the-art algorithms,[4,18,17] is that it builds a smaller number of feature combinations by finding compatible sets from the top of the matrix to the current row. Clearly, both time and space complexities of YYC rely on the number of compatible sets currently found. Thus, as the number of sets currently found grows, the execution becomes slower, and many of these sets will not even lead us to an irreducible testor at the last row.

With the aim of enhancing the performance of YYC algorithm, this work incorporates to it a pre-processing over the rows of the data matrix used to generate the irreducible testors, and the incorporation of (pre, post)-processing over repeated columns at the data matrix. In addition, a parallel version of YYC-algorithm was implemented using threads.

The classic concept of testor, in which classes are assumed to be both crisp and disjointed, is used. The comparison criteria used for all features are Boolean, regardless of the feature type (qualitative or quantitative). These concepts are formalized in the following section.

2. Background

Let $\mathrm{TM} = \{O_1, O_2, \ldots, O_m\}$ be a training set containing m objects, described in terms of n features $R = \{x_1, x_2, \ldots, x_n\}$, and distributed in c classes $K_i, i = 1, \ldots, c$

Each feature $x_i \in R$ takes a qualitative or quantitative value for each object. Besides, a dissimilarity comparison criterion D_i between objects belonging to

different classes in the same attribute X_i, returns 0 or 1, if the compared objects are similar or dissimilar, respectively.

Let DM be a dissimilarity matrix, where their rows are obtained from feature-by-feature comparison between every pair of objects belonging to different classes, using a criterion D_i.[15]

Definition 1. Let p and q be two rows of DM. We say that p is a sub-row of q, if: $\forall\, j[q_j = 0 \Rightarrow p_j = 0]$ and $\exists_i[p_i = 0 \text{ AND } q_i = 1]$, $j \in \{1, \cdots, n\}$.[15]

Definition 2. A row p of DM is called basic if no row in DM is a sub-row of p.[15]

Definition 3. The submatrix obtained of DM containing all its basic rows (without repetitions), is called a basic matrix (denoted by BM).[15]

Remark 1. Commonly, the algorithms used to find irreducible testors use BM instead of DM, due to the substantial reduction of rows. A theorem[7] proves that the set of all typical testors generated using DM is the same as that using BM.

Furthermore, let $T \subseteq R$ be a subset of features. BM^{R-T} denotes the matrix obtained from BM after eliminating all columns belonging to the features in $R - T$. $|R - T|$ denotes the cardinality of the set $R - T$.

Definition 4. Let p be a row of BM^{R-T}, p is a zero row if $\forall\, j \in \{l_1, \cdots, l_{|R-T|}\}$, $p[j] = 0$, being $p[j]$ the element in row p corresponding to the column j.

Definition 5. A set T is a testor of TM, if there is not a zero row p of BM^{R-T}.[4]

Definition 6. Let $x_i \in T$. x_i is called a non-removable feature of T if there is a row p of BM^{R-T}, such as, $\exists\, s \in \{l_1, \cdots, l_{|R-T|}\}$, where $p[s] = 1$ and $\forall\, j \neq s$, $j \in \{l_1, \ldots, l_{|R-T|}\}$, $p[j] = 0$.

Definition 7. A set T is denominated an irreducible testor of TM, if T is a testor of TM and each feature $x_i \in T$ is a non-removable feature of T.[4]

It means that T is an irreducible testor if it is minimal for the inclusion relation, thus no subset of T is a testor.

3. Improvements to YYC Algorithm

This section describes in detail the improvements performed to YYC algorithm in order to enhance its throughput. Some of these improvements are incorporated as data preprocessing and/or post-processing steps, whilst others are embedded in the algorithm. The following sections describe each separately.

3.1. *Pre-processing over the rows of basic matrix*

The performance of YYC algorithm relies on the number of calls to the function that tests whether a compatible set is found or not given a new feature combination. Thus, the more combinations are constructed, the slower the algorithm.

The preprocessing of rows consists in sorting the rows of the basic matrix in ascending order according to: the number of ones on each row as a first criterion, and the Hamming distance to the current row as a second criterion. The main purpose of this sorting is to reduce the number of feature combinations built, or hits.[1]

The sorting performed is based on the Selection sort algorithm. A Hamming distance is calculated between the last row of the sorted submatrix and each non-sorted row r such that the number of 1s in r is equals to the current minimum.

The first step of YYC algorithm builds a unitary irreducible testor for each 1 found at the first row of the basic matrix. The more irreducible testors are found at row r, the more feature combinations are likely to be constructed at row $r+1$. In a basic matrix with this sorting, the first row produces no more hits than the original matrix.

Consider an irreducible testor $\mathrm{IT} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_s}\}$ found at row k. If any of the columns $i_1, i_2, \ldots, i_s$ contains 1 at row $k+1$, then IT remains as irreducible testor and YYC produces no hits. Otherwise, for each column c containing 1 at row $k+1$, a feature combination $\mathrm{IT} \cup \{c\}$ is built, and verified if a compatible set can be found. If the matrix is sorted using the criterion described above, the probability of finding 1 at any of such columns grows because row $k+1$ either contains more 1s, or contains 1s at common columns.

3.1.1. *Illustrative experiment*

With the aim of showing the impact of sorting the basic matrix as proposed, an illustrative experiment is presented next. A basic matrix was built using four different row orderings as can be seen in Fig. 1: $\mathrm{BM}_{\mathrm{orig}}$, is the original basic matrix with no kind of sorting; $\mathrm{BM}_{\mathrm{asc}}$ was sorted in ascending order according to the number of ones on each row; $\mathrm{BM}_{\mathrm{Hamm}}$ was sorted by the proposed algorithm, i.e. it uses the Hamming distance as a second criterion; and $\mathrm{BM}_{\mathrm{desc}}$ was sorted in descending order according to the number of ones on each row.

Table 1. Groups of basic matrices utilized.

Type	r × c	IT	% of 1s
Small	209 × 32	6,405	53.72
	209 × 47	184,920	45.80
	269 × 42	302,066	41.67
Real	Promoters (100 × 57)	77,467	73.96
	Promoters (250 × 57)	257,189	74.54
	Promoters (500 × 57)	726,700	75.01
	100 × 1230	400,387	96.70
Medium	150 × 70	4,299,547	60.60
	150 × 90	17,597,374	60.97
	120 × 105	8,213,928	37.90

$$BM_{orig} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0\,1\,0 \end{pmatrix} \quad BM_{Hamm} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \end{pmatrix}$$

$$BM_{asc} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \end{pmatrix} \quad BM_{desc} = \begin{pmatrix} 0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \\ 1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0\,1\,0 \\ 1\,0\,0\,0\,1\,1\,0\,0\,0\,0 \end{pmatrix}$$

Fig. 1. A basic matrix with four different row-orderings.

Then, we ran YYC algorithm with no pre-processing using each of these matrices. Clearly, the four arrangements of the basic matrix produces the same set of irreducible testors.

Table 2 shows the results obtained from this experiment, with the following notation: IT = number of irreducible testors currently found, LIT = number of lost irreducible testors, i.e. at the current row they lost the irreducible-testor property, and H = number of hits.[1]

We can observe from Table 2 that sorting the rows in ascending order of number of 1s, and using the Hamming distance as a second criterion produces the least number of hits, i.e. the total number of feature combinations built is the minimum. We can notice also that, with the proposed matrix sorting, the least amount of feature combinations are lost.

Table 2. Results of the illustrative experiment.

	BM_{orig}			BM_{Hamm}			BM_{asc}			BM_{desc}		
RP	IT	LIT	H	IT	LIT	H	IT	LIT	H	IT	LIT	H
1	3	0	0	3	0	0	3	0	0	5	0	0
2	7	2	8	3	1	3	3	1	3	20	5	20
3	11	1	5	5	1	4	12	3	12	20	8	32
4	10	1	4	10	2	8	15	3	12	16	4	16
5	14	5	20	16	5	20	16	4	16	14	6	18
6	16	4	12	16	0	0	16	0	0	16	4	12
Total		13	49		**9**	**35**		11	43		27	98

This happens when at row s, a feature combination that fulfilled the irreducible testor property at row $s-1$, does no longer fulfill such property because it was unable to find a new feature compatible with such combination.

For instance, on the 3rd row of $\mathrm{BM}_{\mathrm{orig}}$, the following irreducible testors are found: $\{x_1, x_5\}, \{x_1, x_6\}, \{x_1, x_8\}, \ldots$. However, on the 4th row the combination $\{x_1, x_8\}$ is no longer an irreducible testor because of the presence of 0s at both features.

Clearly, applying an inverse sorting on the basic matrix can be considered as a worst case of YYC. The number of hits produced is the highest, which leads the algorithm to eventually discard many feature combinations.

Remark 2. Through this experiment, we can conclude that the overall number of feature combinations built by YYC from a basic matrix with the proposed sorting, will be less than or equals to that from a basic matrix with descending sorting according to the number of ones per row, or with no sorting at all.

The following example intends to illustrate the impact of the Hamming distance on the number of hits.

$\mathrm{BM}_{\mathrm{asc}}$ (shown below) represents the first 4 rows of a basic matrix that was sorted in ascending order of number of ones per row as the only criterion. Rows 2, 3 and 4 have the same number of ones (3). The order in which these rows are presented corresponds to their original order considering that the sorting algorithm employed is stable.

$$\mathrm{BM}_{\mathrm{asc}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ & & \cdots & & & \end{pmatrix}.$$

The irreducible testors generated at first are $\{x_1\}, \{x_2\}$. When the second row is processed (see $\mathrm{BM}_{\mathrm{asc1}}$), both irreducible testors are lost, and the following ones are found: $\{x_1, x_2\}, \{x_1, x_5\}, \{x_1, x_6\}, \{x_2, x_4\}, \{x_4, x_5\}, \{x_4, x_6\}$. The distance Hamming between these two rows is 5.

$$\mathrm{BM}_{\mathrm{asc1}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ & & \cdots & & & \end{pmatrix}.$$

Now, let us assume that the sorting algorithm swaps rows 2 and 3. When the second row is processed (see $\mathrm{MB}_{\mathrm{asc2}}$), the combination $\{x4\}$ loses its irreducible-testor property, $\{x1\}$ preserves and the following ones are found: $\{x3, x4\}, \{x4, x6\}$. The distance Hamming between these two rows is 3.

$$\mathrm{BM}_{\mathrm{asc2}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ & & \cdots & & & \end{pmatrix}.$$

If the sorting algorithm incorporates the Hamming distance as explained above, row 4 will be selected as the minimum (see $\mathrm{MB}_{\mathrm{asc3}}$). In this case, both irreducible testors found at first preserve on row 2. Therefore, no feature combinations are built. The Hamming distance between these two rows is 1.

$$\mathrm{BM}_{\mathrm{asc3}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ & & \cdots & & & \end{pmatrix}.$$

3.1.2. *Testing the performance with several basic matrix*

In order to test the improvement of performance achieved by this preprocessing, some experiments were carried out. Table 1 describes the basic matrices characteristics that were utilized in all the experiments presented in this article.

Some of these basic matrices have been reported.[16] The real matrices were obtained,[3] except for the basic matrix 100×1230 which was generated from chemical and spectral information of 36 points randomly selected on the whole surface of Chapala Lake, in Jalisco, Mexico. At every point, physical-chemical elements and data of reflectivity in percentage were obtained. Each item has a different range of variability. Physical-chemical information is obtained in situ using a Hydrolab *DS*5 probe and in the laboratory of geochemistry from the Institute of Geology at the Autonomous University of San Luis Potosi. Furthermore, the spectral signature of each point is recorded. A training matrix storing this data was then built. It consists of thirty six objects, one for each point, and 1230 features, denoting specific chemical information and data of reflectivity in percentage. Besides, eight classes were identified using multiple-link clustering.

In Table 1, the density of each basic matrix is calculated as the percentage of 1 values. IT denotes the number of irreducible testors that each basic matrix contains. The dimensions of a matrix is expressed in all the tables as $r \times c$, denoting rows per columns.

Table 3 shows a comparison of hits produced by YYC using the original basic matrix, denoted by $\mathrm{BM}_{\mathrm{ori}}$, and the sorted basic matrix, denoted as $\mathrm{BM}_{\mathrm{sor}}$. *HRR* stands for the hits-reduction ratio achieved. Besides, the difference between the hits generated by $\mathrm{BM}_{\mathrm{ori}}$ and $\mathrm{BM}_{\mathrm{sor}}$ is denoted by DIF_{o-s}.

3.2. *Pre and post processing over repeated columns*

When building a basic matrix from a given training matrix, the dissimilarity criteria employed might result on many columns with the same values. The pre-processing of

Table 3. Comparison of hits performed by YYC using original BM and sorted BM.

	Hits Calculated by YYC			
$r \times c$	BM_{ori}	BM_{sor}	DIF_{o-s}	HRR
209×32	105,098	51,035	54,063	0.51
209×47	2,899,036	1,835,732	1,063,304	0.36
269×42	3,995,690	2,331,641	1,664,049	0.41
100×57	934,286	717,965	216,321	0.23
250×57	3,840,625	3,114,056	726,569	0.18
500×57	10,229,636	8,104,964	2,124,672	0.20
100×1230	46,617,887	10,592,603	36,025,284	0.77
150×70	64,849,246	54,409,509	10,439,737	0.16
150×90	248,883,037	221,804,136	27,078,901	0.10
120×105	159,886,787	96,195,454	63,691,333	0.39

columns consists in eliminating duplicate columns but keeping track of the original columns positions. With this, a reduction on the search space from 2^n to 2^{n-k}, where k is the number of eliminated columns, is achieved.

This reduction procedure was reported.[8] However, it is described in terms of exclusionary features, feature lists and acceptance masks, which are handled internally by the reported algorithm. In this work, it is generalized so that it can be used in any algorithm that generates irreducible testors. Algorithm 1 builds a new basic matrix BM′ without duplicate columns, and a data structure that maps column indices from BM′ to one or more column indices from the original basic matrix. Once YYC algorithm finds all the irreducible testors from BM′, algorithm 2 builds, for each irreducible testor found, the corresponding irreducible testors considering the columns removed.
These algorithms are shown in Appendix A.

Both algorithms are supported by the following propositions. The first proposition is used in the pre-processing step to remove duplicate columns from the basic matrix. The second proposition is utilized in the post-processing step, to generate the irreducible testors that correspond to the columns removed.

Let $T = \{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\}$ be an irreducible testor of TM, $x_p \in R$, $x_p \notin T$. Besides, let $l_{i_1}, l_{i_2}, \ldots, l_{i_q}$ be the columns of BM corresponding to T, and l_p the column of BM corresponding to the feature x_p.

Proposition 1. *If* $\exists\, l_{i_j}, j \in \{1, 2, \ldots, q\}$ *such that* $l_{i_j} = l_p$, *then* $T' = T \cup \{x_p\}$ *is not an irreducible testor of TM.*

Proof. Since $l_{i_j} = l_p$ then by definition 6, x_{i_j} is not a non-removeable feature of T'. Thus, by definition 7, T' is not an irreducible testor of TM. □

Proposition 2. *If* $\exists\, l_{i_j}, j \in \{1, 2, \ldots, q\}$ *such that* $l_{i_j} = l_p$, *then* $T'' = T - \{x_{i_j}\} \cup \{x_p\}$ *is an irreducible testor of TM.*

Table 4. Comparison of hits performed by YYC using original BM and BM without duplicate columns.

	Hits Calculated by YYC				
$r \times c$	BM_{ori}	BM_{rdc}	DIF_{o-r}	HRR	$Columns_{rdc}$
209×32	105,098	102,243	2,855	0.02	31
209×47	2,899,036	2,707,941	191,095	0.06	46
269×42	3,995,690	3,995,690	0	0.00	41
100×57	934,286	934,286	0	0.00	57
250×57	3,840,625	3,840,625	0	0.00	57
500×57	10,229,636	10,229,636	0	0.00	57
100×1230	46,617,887	1,140,191	45,477,696	0.97	144
150×70	64,849,246	64,849,246	0	0.00	70
150×90	248,883,037	85,364,362	163,518,675	0.65	74
120×105	159,886,787	56,276,252	103,610,535	0.64	87

Proof. Being $l_{i_j} = l_q$, obviously if we replace x_{i_j} by x_q we obtain a new irreducible testor. □

Table 4 shows a comparison of hits produced by YYC using the original basic matrix and using a basic matrix after removing duplicate columns, denoted as BM_{rdc}. Besides, the difference between the hits generated by BM_{ori} and BM_{rdc} is denoted as DIF_{o-r}. Finally, the number of columns of BM_{rdc} is denoted as $Columns_{rdc}$.

In addition, Table 5 shows a comparison of hits produced by YYC using the original basic matrix and using the sorted basic matrix without duplicate columns, denoted as $BM_{sor,rdc}$. The difference between the hits generated by BM_{ori} and $BM_{sor,rdc}$ is denoted as $DIF_{o-s,r}$.

Table 5. Comparison of hits performed using original BM and Sorted BM without duplicate columns.

	Hits Calculated by YYC			
$r \times c$	BM_{ori}	$BM_{sor,rdc}$	$DIF_{o-s,r}$	HRR
209×32	105,098	50,828	54,270	0.51
209×47	2,899,036	1,729,798	1,169,238	0.40
269×42	3,995,690	2,331,641	1,664,049	0.41
100×57	934,286	717,965	216,321	0.23
250×57	3,840,625	3,114,056	726,569	0.18
500×57	10,229,636	8,104,964	2,124,672	0.20
100×1230	46,617,887	727,965	45,889,922	0.98
150×70	64,849,246	54,409,509	10,439,737	0.16
150×90	248,883,037	75,863,731	173,019,306	0.69
120×105	159,886,787	27,787,307	132,099,480	0.82

3.3. *Parallel version of YYC*

The parallel version of YYC was implemented using Java threads. The number of threads to create is equal to the number of available processors P. The parallelization is carried out over the set of irreducible testors found at the first row.

Since YYC algorithm does not produce duplicate irreducible testors, and the initial sets of irreducible testors at each thread are disjoint, it is guaranteed that no two threads find the same irreducible testor at the end.

The first step of this version is to find the first row from the pre-processed BM such that the count of 1s is not less than P. This row is set as the first row r_0 of the basic matrix.

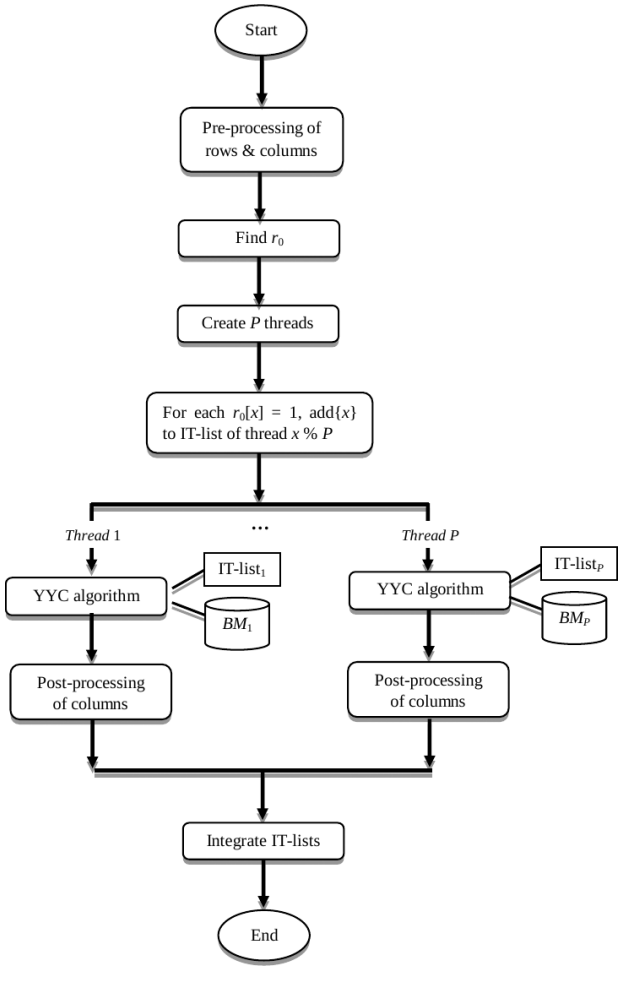

Fig. 2. Parallel version of YYC.

Next, for each $r_0[x] = 1$, add $\{x\}$ to the initial list of irreducible testors of thread $x \bmod P$. Thus, if the number of 1s at r_0 is greater than P, one or more threads will store more than one irreducible testor, initially.

Each x such that $r_0[x] = 1$, belongs to an irreducible testor T, whereby, x is a non-removable feature of T (see definition 6); thus, no feature combination that excludes x will be able to eventually produce an irreducible testor (according to definition 7). Hence, the parallelization process guarantees to keep all the irreducible testors.

After that, each thread runs YYC algorithm in parallel using its initial list of irreducible testors and a copy of the basic matrix.

The main process waits until the last thread has finished its execution. Finally, the main process integrates all the irreducible testors. The post-processing of columns is performed on each thread. This can be appreciated in Fig. 2.

4. Experiments

This section presents the results of the experiments carried out with both sequential and parallel versions of YYC algorithm on each of the basic matrices presented in Table 1. For each matrix, all the irreducible testors were found using: the original basic matrix (MB_{ori}), the basic matrix after row sorting (MB_{sor}), the basic matrix after column reduction (MB_{rdc}), the basic matrix with inverse sort (MB_{inv}), the basic matrix that combines descending row sorting and column reduction ($\text{MB}_{\text{des,rdc}}$) and the basic matrix that combines row sorting and column reduction ($\text{MB}_{\text{sor,rdc}}$).

The experiments ran under PCLinuxOS, in a workstation AMD Opteron with 12 cores running at 2.33 GHz. The original YYC algorithm and the improvements proposed in this work (row sorting, column reduction, parallelization) were all implemented using Java SE 1.8 and the IDE Eclipse.

Table 6 shows the execution time in seconds and the rate of irreducible testors found per second, denoted as RT, achieved by the sequential YYC algorithm using

Table 6. Runtime in seconds and rate performed by YYC using original BM, sorted BM, and sorted BM without duplicate columns.

	Runtime Required by YYC				Rate Performed by YYC			
$r \times c$	BM_{ori}	BM_{sor}	BM_{rdc}	$\text{BM}_{\text{sor,rdc}}$	RT_{ori}	RT_{sor}	RT_{rdc}	$\text{RT}_{\text{sor,rdc}}$
209×32	1.09	0.44	0.95	0.37	5,876.15	14,556.82	6,742.1	17,310.81
209×47	42.05	22.85	40.05	22.41	4,397.62	8,092.78	4,617.23	8,251.67
269×42	80.56	40.33	79.36	39.39	3,749.58	7,489.86	3,806.27	7,668.59
100×57	6.09	4.08	6.08	4.01	12,720.36	18,987.01	12,741.28	19,318.45
250×57	54.69	35.76	46.86	35.67	4,702.67	7,192.09	5,488.45	7,210.23
500×57	263.11	161.21	260.49	156.07	2,761.96	4,507.78	2,789.74	4,656.24
100×1230	5427.85	315.63	27.89	17.54	73.77	1,268.53	14,355.93	22,827.08
150×70	809.68	606.56	811.92	600.89	5,310.18	7,088.41	5,295.53	7,155.30
150×90	3582.67	2824.72	1171.68	937.78	4,911.80	6,229.78	15,018.92	18,764.93
120×105	3303.76	1733.54	995.76	461.02	2,486.24	4,738.24	8,248.90	17,816.86

Table 7. Runtime in seconds performed by parallel YYC using original BM, sorted BM, inverse sorted BM and sorted BM without duplicate columns.

	Runtime Required by Parallel YYC					
$r \times c$	BM_{ori}	$BM_{ori,rdc}$	BM_{des}	$BM_{des,rdc}$	BM_{sor}	$BM_{sor,rdc}$
209×32	0.35	0.35	0.97	0.83	0.11	0.1
209×47	12.46	11.94	73.36	58.68	7.35	6.82
269×42	33.08	32.43	110.11	107.91	8.87	8.14
100×57	1.10	1.09	1.48	1.49	0.76	0.7
250×57	9.59	9.69	18.07	17.82	6.61	6.11
500×57	44.43	43.76	80.01	80.16	30.89	28.86
100×1230	701.07	2.9	999.09	3.49	58.02	0.94
150×70	135.64	138.32	171.04	172.10	104.07	96.53
150×90	547.91	192.44	895.96	257.03	525.42	134.85
120×105	1,161.60	466.69	8965.37	2491.5	538.53	181.11

the different basic matrices described above. Table 7 shows the execution time in seconds and the rate of irreducible testors found per second achieved by the parallel version of YYC algorithm.

Finally, Table 8 shows the speedup achieved on the rate of the original YYC algorithm, after applying each of the different enhancements proposed in this work, including the parallel version.

4.1. *Discussion*

After carrying out the experiments, we can observe the following from the sequential version of YYC algorithm:

(1) The average speedup achieved after applying only row sorting is 3.27, and the trimmed mean is 1.78. The trimmed mean is calculated as the average

Table 8. Speed-up obtained by YYC and parallel YYC using original BM, sorted BM, and sorted BM without duplicate columns.

	Speed-Up Obtained by YYC			Speed-Up Obtained by Parallel YYC			
$r \times c$	BM_{sor}	BM_{rdc}	$BM_{sor,rdc}$	BM_{ori}	BM_{sor}	BM_{rdc}	$BM_{sor,rdc}$
209×32	2.48	1.15	2.95	3.11	9.91	3.11	10.90
209×47	1.84	1.05	1.88	3.37	5.72	3.52	6.17
269×42	2.00	1.02	2.05	2.44	9.08	2.48	9.90
100×57	1.49	1.00	1.52	5.54	8.01	5.59	8.70
250×57	1.53	1.17	1.53	5.70	8.27	5.64	8.95
500×57	1.63	1.01	1.69	5.92	8.52	6.01	9.12
100×1230	17.20	194.62	309.46	7.74	93.55	1871.67	5755.94
150×70	1.33	1.00	1.35	5.97	7.78	5.85	8.39
150×90	1.27	3.06	3.82	6.54	6.82	18.62	26.57
120×105	1.91	3.32	7.17	2.84	6.13	7.08	18.24
Average	3.27	20.84	33.34	4.92	16.38	192.96	586.29
Trim Mean	1.78	1.60	2.82	4.88	8.07	6.93	12.60

discarding the highest and lowest speedups. With this pre-processing, we have reduced the execution time by more than fifty percent in most of the basic matrices.

(2) The average speedup achieved after applying only column reduction is 20.84, and the corresponding trimmed mean is 1.60. When the basic matrix contains duplicate columns, this pre-processing results in a reduction on the execution time proportional to the number of columns removed. If the basic matrix contains very few or none duplicate columns, the runtime is no better than that without column reduction. This is because the time saved by working with a slightly thinner basic matrix is compensated by the time consumed by pre- and post- tasks of column reduction.

(3) The average speedup achieved after applying both row sorting and column reduction is 33.34 and the trimmed mean is 2.82. In the specific case of the matrix with 1,230 columns, this reduction was significant. The importance of this pre-processing relies on the existence of real-life high-dimensional classification problems that requires feature selection.

In such cases, training matrices typically include hundreds of features with very similar data as is the case of this reflectance matrix, as well as many other matrices that can be found at UCI Machine Learning Repository.[3]

Likewise, the following can be observed after running the parallel version of YYC algorithm:

(1) The average speedup achieved with no pre-processing is 4.92, and the corresponding trimmed mean is 4.88.

(2) The average speedup achieved after applying only row sorting is 16.38, and the trimmed mean is 8.07.

(3) The average speedup achieved after applying only column reduction is 192.96, and the trimmed mean is 6.93.

(4) The average speedup achieved after applying both row sorting and column reduction is 586.29, and the corresponding trimmed mean is 12.60.

(5) Sorting the rows of basic matrix in descending order according to the number of 1s results in very long executions of YYC.

When the pre-processing is applied, the basic matrix 100×1230 achieves the highest speedup, followed by the basic matrices 150×90 and 120×105. The benefits of pre-processing the basic matrix and parallelizing the algorithms are higher as the number of columns grows.

It can be observed that the parallel version of YYC using twelve physical cores is more than twelve times faster than the original YYC algorithm. Nevertheless, this speedup is achieved only if the pre-processing of rows and columns are both applied. The parallelization by itself speeds up the serial execution time by no more than seven times.

The combination of preprocessing and parallelization of YYC algorithm represents a more efficient alternative for finding all the irreducible testors from a basic matrix.

5. Conclusions

In this article, we have proposed three enhancements on YYC intended to improve its performance, that we have measured as the rate of irreducible testors found per time unit.

Two of these enhancements involve a pre-processing on the input basic matrix as well as a post-processing of the irreducible testors found. The third enhancement is a parallel version of the algorithm capable to run on any multicore CPU.

It was demonstrated experimentally that YYC algorithm improved its performance after applying the proposed pre-processing. On basic matrices with many duplicate columns, the speedup achieved was significant.

The average speedup achieved with the parallelization was around sixty percent of the number of available cores.

Appendix A

Algorithm 1. Pre-processing of columns

Input: a basic matrix BM with k rows and n columns
Output: a pair $< BM', Map >$ where BM' is the corresponding basic matrix without duplicate columns, and Map is an indexed mapping from BM' columns to BM columns

1: **Initialize** $Map = \emptyset$
2: **For each** non-visited column index p $(1 \leq p \leq n)$ **do**
3: **Initialize** $S = \{p\}$
4: **Add** to S every $p' > p$ such that: $BM[q,\ p] = BM[q,\ p']$, $1 \leq q \leq k$
5: **Mark** every p' as visited
6: $Map = Map \cup S$
7: **Let** BM' be a new basic matrix with dimensions $k \times |Map|$
8: **For each** $s \in Map$ **do**
9: **Let** i be the index of s, and p_0 the first value at s
10: $BM'[q,\ i] \leftarrow BM[q,\ p_0]$, $1 \leq q \leq k$
11: **Return** pair $< BM', Map >$

Algorithm 2. Post-processing of columns

Input: an irreducible testor T from a duplicate-column free Basic Matrix
Output: a list of irreducible testors related to T of the original basic matrix

Let $L = \emptyset$ be an empty list of irreducible testors
Let x be the first feature from T
For each $p \in Map[x]$ **do**
  Let $T' \leftarrow \{p\}$
  $Add - IT(x, T', L)$
Return the list L

$Add - IT(Featurex, IrreducibletestorT, ListL)$
  If x is the last feature from T **then**
    $L \leftarrow L \cup \{T\}$
  Else
    Let x' be the next feature from T
    For each $p \in Map[x']$ **do**
      Let $T' \leftarrow T \cup \{p\}$
      $Add - IT(x', T', L)$

Appendix B

The following is a description of the problem and their corresponding training matrix.[23]

Patients are characterized as suffering from strep-throat or flu depending on the presence or absence of a combination of the following symptoms: sore throat (x_1), cough (x_2), cold (x_3), and fever (x_4).

Let K_1 denote the class of patients suffering from strep-throat, and K_2, the class of patients suffering from flu. The training matrix consists of information pertaining to seven patients, the first two in K_1, and the last five in K_2. A 1 in a particular column represents the presence of the corresponding symptom, and a 0 represents the absence of that symptom. The information pertaining to each patient is represented as a row in the follow training matrix (Table B.1).

Table B.1. TM. Patient description.

Object	x_1	x_2	x_3	x_4	Class
O_1	1	1	0	0	K_1
O_2	1	0	1	0	K_1
O_3	0	0	1	1	K_2
O_4	1	0	1	1	K_2
O_5	0	0	1	0	K_2
O_6	0	1	1	0	K_2
O_7	0	1	1	1	K_2

Besides, the dissimilarity matrix (DM) obtained of TM and their corresponding basic matrix (BM) are shown below. The sorted applied to BM do not generated changes.

$$\mathrm{DM} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \mathrm{BM} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

The second and third columns are not removed of BM, because only contains 0s. The irreducible testor $\mathrm{IT}_1\{x_1\}$ is generated when the first row of BM is processing. But when the second row is processed, this irreducible testor if eliminated, computing one hit and generated now the irreducible testor $IT_2 = \{x_1, x_4\}$.

After processing the rows of BM, the only irreducible testor generated was $IT = \{x_1, x_4\}$, which belonging to sore throat and fever patient symptoms in original definition problem.

References

1. E. Alba-Cabrera, J. Ibarra-Fiallo, S. Godoy-Calderon and F. Cervantes-Alonso, YYC: A fast performance incremental algorithm for finding typical testors, *LNCS 8827* (Springer, 2014), pp. 416–423.
2. A. Asaithambi and V. Valev, Construction of all non-reducible descriptors, *Pattern Recognit.* **37** (2004) 1817–1823.
3. K. Bache and M. Lichman, UCI Machine Learning Repository [http://archive.ics.uci.-edu/ml], In: School of Information and Computer. University of California Science, Irvine, 2013.
4. A. Bravo-Martinez, CT algorithm for the calculation of typical testors of a k-valued matrix, *Revista Ciencias Matematicas, Cuba (in Spanish)* **IV** (1983) 123–144.
5. A. Dmitriev, I. Zhuravlev and F. Krendeliev, About mathematical principles and phenomena classification, *Diskretni Analiz (in Russian)* **7** (1966) 3–15.
6. R. Kohavi and G. Jhon, Wrappers for Feature Subset Selection, *Artif. Intell.* **97** (1997) 273–324.
7. M. Lazo-Cortes, J. Ruiz-Shulcloper and E. Alba-Cabrera, An Overview of the evolution of the concept of testor, *Pattern Recognit.* **34** (2001) 753–762.
8. A. Lias-Rodriguez and G. Sanchez-Diaz, An algorithm for computing typical testors based on elimination of gaps and reduction of columns, *Int. J. Pattern Recognit. Artif. Intell.* **27**(8) (2013) 1350022.

9. J. Martinez-Trinidad and A. Guzman-Arenas, The logical combinatorial approach for pattern recognition. An overview through selected works, *Pattern Recognit.* **34** (2001) 741–751.
10. I. Mierswa and W. Michael, Information preserving multi-objective feature selection for unsupervised learning, *Proc. Genetic and Evolutionary Computation Conf.* (ACM Press, 2006), pp. 1545–1552.
11. J. Ochoa, M. Valdes, I. Moctezuma and C. Ayala, Dimension reduction in image databases using the logical combinatorial approach, *Innovations and Advances Techniques in Systems, Computing Sciences and Software Engineering* (Springer, 2008), pp. 260–265.
12. A. Pons-Porrata, R. Berlanga-Llavori and J. Ruiz-Shulcloper, Topic discovery based on text mining techniques, *Inf. Process. Manage.* **43** (2007) 752–768.
13. J. Ruiz-Shulcloper and M. Abidi, Logical combinatorial pattern recognition: A review, *Recent Research Developments in Pattern Recognition* (Transword Research Networks, Kerala, India, 2002), pp. 133–176.
14. J. Ruiz-Shulcloper, A. Bravo-Martinez and L. Aguila-Feros, BT and TB algorithms for calculation of all typical testors, *Revista Ciencias Matematicas, Cuba (in Spanish)* **VI** (1985) 11–18.
15. J. Ruiz-Shulcloper, A. Soto and A. Fuentes, Characterization of the typical testor concept in terms of a notable set of columns, *Revista Ciencias Matematicas (in Spanish)* **I(2-3)** (1980) 123–134.
16. I. Piza-Davila, G. Sanchez-Diaz, C. Aguirre-Salado and M. Lazo-Cortes, A parallel hill-climbing algorithm to generate a subset of irreducible testors, *Appl. Intell.* **42** (2015) 622–641.
17. G. Sanchez-Diaz and M. Lazo-Cortes, Modifications to BT algorithm for improving its run time execution, *Revista Ciencias Matematicas, Cuba (in Spanish)*, **20** (2002) 129–136.
18. G. Sanchez-Diaz and M. Lazo-Cortes, CT_EXT: An external escale algorithm for generated typical testors, *LNCS 4756* (Springer, 2007), pp. 506–514.
19. G. Sanchez-Diaz, M. Lazo-Cortes and I. Piza-Davila, A fast implementation for the testor property identiffcation based in an accumulative binary tuple, *Int. J. Comput. Intell. Syst.* **5** (2012) 1025–1039.
20. Y. Santiesteban-Alganza and A. Pons-Porrata, LEX: A new algorithm for calculating typical testors, *Revista Ciencias Matematicas, Cuba (in Spanish)* **21** (2003) 85–95.
21. Y. Saeys, S. Degroeve and Y. Van de Peer, Digging into acceptor splice site prediction: An iterative feature selection approach, *Proc. PKDD*, 2004, pp. 386–397.
22. M. Torres, A. Torres, F. Cuellar, L. Torres, E. Ponce and F. Pinales, Evolutionary computation in the identification of risk factors. Case of TRALI, *Expert Syst. Appl.* **41** (2014) 831–840.
23. V. Valev, From binary features to non-reducible descriptors in supervised pattern recognition problems, *Pattern Recognit. Lett.* **45** (2014) 106–114.
24. V. Valev and A. Asaithambi, On computational complexity of non-reducible descriptors, *Proc. IEEE Int. Conf. Information Reuse and Integration*, 2003, pp. 208–211.
25. V. Valev and B. Sankur, Generalized non-reducible descriptors, *Pattern Recognit.* **37** (2004) 1809–1815.
26. V. Valev and Y. Zhuravlev, Integer-valued problems of transforming the training tables in k-valued code in pattern recognition problems, *Pattern Recognit.* **24** (1991) 283–288.

Ivan Piza-Davila received his B.S. degree in Computer Systems in 2000 from the Instituto Tecnologico de Colima, Colima, Mexico, and his M. S. degree and Ph.D. in Computer Science in 2002 and 2006, from Centro de Investigacion y de Estudios Avanzados del IPN, Mexico. He is currently a full time professor-researcher in the Instituto Tecnologico y de Estudios Superiores de Occidente, Guadalajara, Mexico. His research interests include pattern recognition, machine learning and parallel algorithms. He has published several papers in journals and conference proceedings.

Guillermo Sanchez-Diaz received his B.S. degree and his M.S. degree in Computer Science in 1995 and 1997, respectively, from Autonomous University of Puebla, Mexico, and his Ph.D. in Computer Science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. He is currently a full time professor-researcher in the Autonomous University of San Luis Potosi of Mexico. His research interests include pattern recognition, machine learning and data mining. He has published several papers in journals and conference proceedings. He has served regularly in program committees of international conferences and has also been a reviewer for some international journals in his fields.

Manuel S. Lazo-Cortes received his B.S. degree in Mathematics in 1979 and his Ph.D. in Mathematical Sciences in 1994 from University de las Villas, Cuba. He is currently a guest professor in the Instituto Nacional de Astrofisica, Optica y Electronica, Puebla, Mexico. His research interests include pattern recognition and machine learning. He has published several papers in journals, chapter books and conference proceedings. He has served regularly in program committees of international conferences and has also been a reviewer for some international journals in his fields.

Cristina Noyola Medrano Ph.D. is from the University of Paris Denis Diderot-Paris 7. She is a Professor-Researcher at the Faculty of Engineering at the Autonomous University of San Luis Potosi. She is a specialist in the treatment of satellite imagery and data analysis of spectro-radiometers for evaluation of natural resources such as continental water bodies, change of land use, desertification and other effects of global climate change.