2014-05-01

An evolutionary algorithm with acceleration operator to generate a subset of typical testors

Sánchez-Díaz, Guillermo; Díaz-Sánchez, Germán; Mora-González, Miguel; Aguirre-Salado, Carlos A.; Huerta-Cuéllar, Guillermo; Piza-Dávila, Hugo I.; Reyes-Cárdenas, Óscar; Cárdenas-Tristán, Abraham

Sanchez-Diaz, G.; Diaz-Sanchez, G.; Mora-Gonzalez, M; Piza-Davila, H.I.; Aguirre-Salado, C.A.; Huerta-Cuellar, G; Reyes-Cardenas, O.; Cardenas-Tristan, A. (2014). "An evolutionary algorithm with acceleration operator to generate a subset of typical testors". Pattern Recognition Letters. Volume 41, 1 May, pp.34-42.

Enlace directo al documento: http://www.sciencedirect.com/science/article/pii/S0167865513004297

An evolutionary algorithm with acceleration operator to generate a subset of typical testors

Guillermo Sanchez-Diaz[a,1,*], German Diaz-Sanchez[b], Miguel Mora-Gonzalez[c], Ivan Piza-Davila[d], Carlos Aguirre-Salado[a], Guillermo Huerta-Cuellar[e], Oscar Reyes-Cardenas[a], Abraham Cardenas-Tristan[a]

[a] *Universidad Autonoma de San Luis Potosi, Dr. Manuel Nava 8, SLP, Mexico*
[b] *Centro de Investigacion y de Estudios Avanzados del I.P.N., Zapopan, Jal. Mexico*
[c] *Universidad de Guadalajara, Enrique Diaz de Leon 1144, Lagos de Moreno, Jal. Mexico*
[d] *Instituto Tecnologico y de Estudios Superiores de Occidente, Tlaquepaque, Jal. Mexico*
[e] *Diseño y Desarrollo Optomecatronico de Mexico, Capri 107, Leon, Gto., Mexico*

Abstract

This paper is focused on introducing a hill-climbing algorithm as a way to solve the problem of generating typical testors -or non-reducible descriptors- from a training matrix. All the algorithms reported in the state-of-the-art have exponential complexity. However, there are problems for which there is no need to generate the whole set of typical testors, but it suffices to find only a subset of them. For this reason, we introduce a hill-climbing algorithm that incorporates an acceleration operation at the mutation step, providing a more efficient exploration of the search space. The experiments have shown that, under the same circumstances, the proposed algorithm performs better than other related algorithms reported so far.

Keywords: Hill climbers, feature selection, typical testors, pattern recognition

*Corresponding author
Email address: guillermo.sanchez@uaslp.mx (Guillermo Sanchez-Diaz)
[1]Tel. (+52) 444-8262330 ex. 6010

1. Introduction

Data dimensionality reduction has become very important in machine learning over the past few decades. Many problems related to image processing, text mining and bioinformatics -among other disciplines- involve handling large datasets which instances can be described as a set of features.

A number of dimension-reduction techniques have emerged as a pre-processing step in tasks dealing with large datasets, such as: data analysis and supervised classification. Some of these techniques are about feature subset selection. The main difference between these techniques and other reduction techniques (like projection and compression) is that the first ones do not transform the input features, but they select a subset of them [17].

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 meta-heuristics [11, 13, 30], multi-objective point of view [19], etc. Nevertheless, results at this time are not conclusive.

Zhuravlev [9] introduced the concept of test to pattern recognition problems. He defined a test as a subset of features that allows differentiating objects from different classes. This concept has been extended and generalized in several ways [14, 43].

In Logical Combinatorial Pattern Recognition approach [18, 25], feature selection is addressed using Testor Theory [14].

In the eighties, Ruiz-Shulcloper introduced a typical testor characterization for computing all the typical testors of a training matrix, with object descriptions defined in terms of any kind of features, not only booleans [4, 26]. The first algorithms to generate the entire set of typical testors of a training matrix were then developed [27, 2, 3].

The concept of testor and typical testor have also been used by V. Valev, under the names of descriptor and non-reducible descriptor, respectively [44].

Typical testors have been widely used in voting algorithms for object classification, based on partial-precedence determination [28].

Besides, they have been used for evaluating the relevance of features on differential diagnosis of diseases [21], and for estimating stellar parameters with remotely sensed data [36]. In addition, typical testors have been employed for: feature selection on natural-disaster texts classifications [5], dimensionality reduction on image databases [20], text categorization [23], and automatic summarization of documents [22].

There are some real world problems which do not require the entire set of typical testors, but only a subset. Some examples include:

- Determination of risk factors associated to pregnant Mexican women [40]. In this work, a problem of finding the most relevant features concerning neonatal morbidity on pregnant women is introduced. A genetic algorithm to find typical testors was used. Some of the features considered in this problem include: mother's age and weight, number of pregnancies, number of deliveries, bled, Apgar test within the first minute of the baby's life, and gestational age. The matrix employed to generate the typical testors has 32,768 rows and 29 columns.

- Determination of factors associated with Transfusion Related Acute Lung Injury (TRALI) [39]. This paper describes the determination of informational weight of features related to TRALI, using a hybrid genetic algorithm for the identification of risk factors and the establishment of an assesment to each variable. In this problem, each typical testor denotes a set of features that best differentiates patients who will present TRALI from those who will not. The matrix used to generate the typical testors has 174 rows and 31 columns.

- Medical electrodiagnostic using pattern recognition tools [16]. This work introduces a medical diagnosis problem using neuroconduction studies, electromyography, signs and symptoms. The objects are assigned one of the following classes: lumbosacral radiculopathy, neuropathies, Guillain-Barre, myopathies, traumatic injuries of sciatic and Charcot-Marie-Tooth. This work used typical testors as support sets system, in the second step of a voting classification algorithm. The matrix used to generate the typical testors has 1,215 rows and 105 columns.

The number of rows of the matrix employed in the first example is too large. An algorithm capable to generate the whole set of typical testors takes several days.

The second example introduces a cut-off criterion for calculating the informational weight of features obtained from the generated typical testors. This criterion can be automatically calculated.

In the last example presented, the entire set of typical testors has not been found yet. The authors divided the matrix in three parts to find other typical testors, but without taking into account all features described in the problem. This fact affects the accuracy of the classification.

The computation of the entire set of typical testors requires exponential time [41]. In general, two approaches have been developed to address this problem: a) algorithms that generate the entire set (LEX (Lexicographic Order Algorithm)[35], CT_EXT (Complete elements extended)[31], BR (binary operations)[15], and Fast-CT_EXT (Fast-Complete elements extended)[34]); and b) algorithms that find only a subset of typical testors (GA (Simple Genetic Algorithm)[32], UMDA (Evolutionary Strategy)[1] and AGHPIA (Genetic algorithm with evolutionary mechanisms)[38]).

Nevertheless, these global-search heuristics become too slow as the number of features grows significantly. One reason is because the goal of this techniques is to reach the global maximum which, in this case, refers to the entire set of typical testors. However, each typical testor can be considered a local maximum for this particular problem.

This paper introduces a local-search heuristic based on the Hill-Climbing algorithm, that incorporates an acceleration operation, useful to find a subset of the entire set of typical testors. The goal of this Hill Climbing technique is to generate a single typical testor, iteratively, across the space search.

Preliminary results of this algorithm were presented in [7], but this work explains in detail typical-testor concepts, and shows experimentally the stability of the proposed algorithm when different values of its parameters are handled, using different basic matrices.

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

2. Background

Let $TM = \{O_1, O_2, \cdots, O_m\}$ be a training matrix containing m objects, each belonging to a class $K_i \in \{K_1, K_2, \cdots, K_c\}$, described in terms of n features $R = \{x_1, x_2, \cdots, x_n\}$. Each feature $x_i \in R$ takes values in a set M_i, $i = 1, \cdots, n$. A comparison criterion of dissimilarity $D_i : M_i \times M_i \to \{0, 1\}$ is associated to each x_i (0=similar, 1=dissimilar) [8, 29].

An example of training matrix which was taken from [43] is the following:

Example

A medical doctor can tell whether a patient suffers from a step throat or from a flu by the presence or absence of the following symptoms: sore throat, cough, cold and fever.

In this example, patients are the objects $(O_1, O_2, \cdots, O_7)$, symptoms are the features (x_1, x_2, x_3, x_4), and diseases are the classes (K_1, K_2).

The training matrix (shown in table 1) stores the information of seven patients; the first two suffers from strep throat (class K_1), and the last five suffers from a flu (class K_2).

Each row in the training matrix denotes the presence (1) and absence (0) of every symptom on a patient.

Definition 1. *If a feature subset $T \subseteq R$ allows to distinguish objects belonging to different classes, then T is called a testor (or descriptor) [9].*

Table 1: Training matrix of patients

Objects	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

Definition 2. *If a given testor T , does not allow to distinguish objects belonging to different classes after removing any attribute $x_i \subset R$, then T is called typical testor (or non-reducible descriptor), and it is denoted by TT [9].*

In the training matrix of patients, the set of features $\{x_1, x_2, x_4\}$ is a testor. Also, the set $\{x_1, x_4\}$ is a typical testor of this training matrix.

In addition, a comparison criterion of dissimilarity $D : M_i \times M_i \to \{0, 1\}$ is associated to each x_i (0=similar, 1=dissimilar), where M_i is the admissible values set of x_i.

Definition 3. *The dissimilarity matrix (denoted as DM) for the objects $O_i \in TM$, is a Boolean matrix, where the rows are obtained by feature* *comparison between every pair of objects, using a dissimilarity comparison criteria [8].*

The DM corresponding to the training matrix of patients was obtained for all the features using he comparison criteria D_s shown in (2). Such DM is the following:

$$DM = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \tag{1}$$

The first row of the DM above was obtained from comparing O_1 and O_3. In the same way, the second row was obtained comparing O_1 and O_4, the third row by the comparison of O_1 and O_5, and so on. Finally, the last row was obtained from comparing O_2 and O_7.

$$D_s(x_s(O_i), x_s(O_j)) = \begin{cases} 1 & if\ x_s(O_i) \neq x_s(O_j) \\ 0 & otherwise \end{cases} \tag{2}$$

[10] shows additional comparison criteria useful to create a DM.

Remark 1. Computationally, it is faster to work with the DM instead of their belonging TM. Because, for creating the DM, the comparison between two arbitrary objects of TM is performed only once, and the DM is a Boolean matrix.

Definition 4. *We say that* p *is a subrow of* q *if:* $\forall_j[q_j = 0 \Rightarrow p_j = 0]$ *and* $\exists_i[p_i = 0 \Rightarrow q_i = 1]$ *[29].*

Definition 5. *A row* p *of* DM *is called basic if no row in* DM *is a subrow of* p *[29].*

Definition 6. *The submatrix obtained of* DM *containing all its basic rows (without repetitions), is called a basic matrix (denoted by* BM*) [29].*

The BM obtained of the DM (1) is the following [33]:

$$BM = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

Only rows 7^{th} and 8^{th} of DM (1) are basic; thus, BM (3) is comprised of these rows.

Remark 2. The typical testor set of a TM may be obtained using DM or BM. A theorem introduced in [14] proves that the set of all typical testors generated using DM is the same as that using BM. This theorem is shown below:

Let $\tau(DM)$ be the set of all the typical testors of a training matrix TM making use of its belonging dissimilarity matrix DM. Let $\tau(BM)$ be the set of all the typical testors of TM making use of its corresponding basic matrix BM.

Theorem 1. *$\tau(DM) = \tau(BM)$*

Commonly, algorithms used for computing typical testors make use of BM instead of DM, due to the substantial reduction of rows (see remark 1). Now, the characterization of a typical testor working with the basic matrix is presented.

Definition 7. *Columns $j_1, j_2, \cdots, j_d$ of an arbitrary matrix $A = a[i,j]$; $i = 1, \cdots, s$, $j = 1, \cdots, n$ form a covering if there is no row $p = 1, \cdots, s$ from matrix A such that $a_{p,j_q} = 0$, for each $q = 1, \cdots, d$ [42].*

Definition 7 means that a subset of columns of a matrix forms a covering if there are no rows containing only zeros in this subset of columns.

Let E be a matrix created from a subset of columns of the basic matrix BM, generated from TM.

Theorem 2. *If the columns $j_1, \cdots, j_d$ of the matrix E form a covering of BM, then the set $T = \{x_{j_1}, \cdots, x_{j_d}\}$ is a testor of TM. [42].*

Theorem 2 means that a testor is a subset of features $T = \{x_{i_1}, \cdots, x_{i_s}\}$ of TM for which a full row of zeros does not appear in the remaining submatrix of BM, after eliminating all the columns corresponding to the features in $R \backslash T$ [42].

Definition 8. *Two elements $a[i_1, j_1]$ and $a[i_2, j_2]$ belonging to the basic matrix BM are called compatible elements, if:*

1. *$a[i_1, j_1] = a[i_2, j_2]$ =1, for $i_1 \neq i_2$ and $j_1 \neq j_2$,*
2. *$a[i_1, j_2] = a[i_2, j_1]$ = 0.*

[8].

Definition 9. *Elements $a[i_1, j_1], a[i_2, j_2], \cdots, a[i_d, j_d]$ are called a sequence of compatible elements (SCE), if:*

1. *for $d = 1,\ a[i_1, j_1] = 1$,*
2. *for $d > 1$, each pair of elements is a pair of compatible elements.*

[8].

Definition 9 means that the every row $i_1, \cdots, i_d$ and every column $j_1, \cdots, j_d$ from the matrix E are comprised by $d - 1$ zeros and a one [8].

Definition 10. *The number of compatible elements d of a SCE is called a rank of this SCE and it is denoted by SCE^d [42].*

The matrix E formed by the rows 1 and 2, and columns 1 and 4 belonging to BM (3), which form a SCE^2 is the following:

$$E = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{4}$$

Theorem 3. *If the set $TT = \{x_{j_1}, \cdots, x_{j_d}\}$ is a testor of TM (generated by columns $j_1, \cdots, j_d$ of matrix Z, which form a covering of BM), and rows $i_1, \cdots, i_d$ of E, whose elements $a[i_1, j_1], \cdots, a[i_d, j_d]$ form a SCE^d, then the set $TT = \{x_{j_1}, \cdots, x_{j_d}\}$ is a typical testor of TM. [42].*

Theorem 3 means that TT is a typical testor if there is no proper subset of any subset of features T that meets the testor condition. Thus, each typical testor is of minimal length. Therefore, each typical testor can no longer be reduced [42].

2.1. Hill Climbing algorithm

The Hill-Climbing algorithm [12, 37] is a local-search stochastic method which, in general, uses a bit string to represent either a set of prototypes or, in some experiments, a collection of features.

Hill-Climbing can be considered as an evolutionary strategy with one individual which was intended to solve complex optimization problems arising from engineering design problems [6].

Consider the set

$$P(R) = \{\emptyset, \{x_1\}, \cdots, \{x_n\}, \{x_1, x_2\}, \cdots, \{x_{n-1}, x_n\}, \cdots, \{x_1, ..., x_n\}\} \quad (5)$$

where $P(R)$ is the power set of feature set R, and n is the cardinal of R. Now, consider the follow set

$$SS(R) = P(R) \setminus \{\emptyset\} \quad (6)$$

where $SS(R)$ is the entire search space of the set R. Then, $SS(R)$ contains all possible combinations of features that can form in the set R.

Let BM be the Basic Matrix obtained from a Training Matrix TM, and m_{BM} be the number of rows of BM. Let $Z = \{x_{i_1}, \cdots, x_{i_s}\}$, $Z \subseteq R$ and $Z \in SS(R)$.

We want to obtain a set Z that minimizes the absolute value of the performance index.

$$J(Z) = 1 - (\sum_{p=1}^{m_{BM}} zr_p + \frac{1}{(\sum_{q=i_1}^{i_s} or_q) + 1}) \quad (7)$$

zr_p refers those rows having only zeros at columns $i_1, \cdots, i_s$ such that they do not allow to form a covering of BM; or_q refers to those columns from $i_1, \cdots, i_s$ not having compatible elements and not allowing to form a sequence of compatible elements (SCE).

Remark 3. Notice that for any feature subset Z, $v \leq J(Z) < 1$, $v \leq 0$. If the performance index $J(Z)$ reaches the value 0, then Z is a typical testor (Z meets theorem 2 and theorem 3). If $J(Z)$ is a positive value, then Z just a testor, but it is not typical testor (Z only meets theorem 2). Otherwise, if $J(Z)$ is negative, Z is not a testor (Z does not meets theorem 2).

Considering this problem of feature selection as a problem of location of zeros, the hill climbing algorithm is designed to obtain the feature subsets Z, such that the performance index $J(Z)$ proposed in this paper reaches a zero (i.e. to find a feature subset $Z \subseteq R$ and $Z \in SS(R)$, such that $J(Z) = 0$).

3. The proposed Hill Climbing algorithm for generated typical testors

3.1. The acceleration operation

The proposed Hill-Climbing algorithm incorporates an acceleration operator at the mutation step. This operator improves the exploration capability of the mutation, being able to find a feature subset $Z = \{x_{i_1}, \cdots, x_{i_s}\}$ which meets the typical testor property, with a lower number of computations.

The accelerator operator is independent to the mutation operator because the latter can be performed without the accelerator operator proposed, as is done in simple Hill Climbing algorithm.

This acceleration operator is applied differently. It depends on the performance index found and based on the behavior of the combination of feature subset, according to the following rules:

Rule 1. If $J(Z) = 0$ (Z is a typical testor), then:

a) one feature x_j is removed from Z, such that $j = i_1, \cdots, i_s$

b) one feature x_p is added to Z, such that $p \neq j$, $p = i_1, \cdots, i_s$

Rule 2. If $J(Z) > 0$ (Z is a testor), then k_t-features x_j, $j = i_1, \cdots, i_s$, $0 < k_t < i_s$ are removed of Z.

Rule 3. If $J(Z) < 0$ (Z is not a testor), then k_{nt}-features x_p, $p = i_1, \cdots, i_s$ $0 < k_{nt} < i_s$ are added to Z.

Remark 4. According to different experiments with several algorithms, we could observe that, in most cases, two different typical testors, could be equated to perform a permutation of two features x_i, x_j, $i \neq j$ as follows: if $x_i = 1$ and $x_j = 0$ then set $x_i = 0$ and $x_j = 1$. This reasoning is applied to Rule 1.

Remark 5. If a feature subset Z is a testor, but it is not a typical testor, then Z does not satisfy theorem 3. This means that Z can be reduced, and some features can be removed from Z. In Rule 2, this reasoning is applied.

Remark 6. Finally, if a feature subset Z is not a testor, then Z does not satisfy theorem 2. Thus, Z needs more features to satisfy theorem 2, and some features can be added to Z. This reasoning is applied to Rule 3

The Hill-Climbing algorithm includes two parameters to calculate the number of features to either add or remove to Z, namely, the mutation probability for non-testors and the mutation probability for testors, respectively.

Such parameters can be fixed or calculated based on the value of the performance index $J(Z)$. In the latter case, the number of features to add o remove to/from Z would be proportional to the absolute value of $J(Z)$, i.e., if $J(Z)$ is large, a considerable amount of attributes would then be added or removed to/from Z; otherwise, this amount would be small.

Besides, the proposed algorithm allows to find typical testors:

a) of minimum length or weight [32],

b) with a specified length (e.g. length = 3), or

c) without any of the restrictions mentioned above.

Step 4 of the algorithm shown below verifies such restrictions.

The algorithm will stop if, either the maximum number of iterations is reached, or the expected number of typical testors is found. The algorithm is designed as follows:

Input : BM (basic matrix); Iter (number of iterations); NumTT (number of typical testors to find); p_t (mutation probability for a testor); p_{nt} (mutation probability for a non testor); CondTT (condition about what type of typical testor should be found)

Ouput: TT (list of typical testor subset found)

1. *Prototypes representation and initialization.* A feature combination Z is encoded in an n-dimensional binary array as: $A = [a_1, \cdots, a_n]$, where each $a_j = 1$ means that feature x_j is present in Z. Otherwise, if $a_j = 0$ indicates the absence of feature x_j in Z.
   The performance index $J(Z)$ will be handled as the fitness value $F(A)$. Start from an empty list of typical testors TT; $Iter \leftarrow 1$.
2. *Array initialization.* Each component a_j of array A, is generated randomly. Call this array best-evaluated and calculate the fitness value $F(A)$ (i.e. the belonging performance index $J(Z)$ is obtained). If $F(A) = 0$ then, add A to the list TT.
3. *Mutation.* First, the values of mutated array are assigned as $A_{mut}(a_i) = A(a_i)$, $i = 1, \cdots, n$. Second, the value of some components of the mutated array are randomly mutates using a Uniform random variable, according to the rules defined below in the acceleration operator, using a procedure as follows: $Mutate(A_{mut}, F(A), p_t, p_{nt})$. If probabilities p_t, p_{nt} are not fixed, then these will be calculated regarding the value of $F(A)$.
4. *Fitness calculation.* Compute the Fitness of the mutated array A_{mut}, as $F(A_{mut})$. If $F(A) = 0$, verify whether A is already in the list TT; if not, verify if CondTT holds for add it to the list.
5. *Compare the fitness obtained.* If $abs(F(A_{mut})) < abs(F(A))$, where $abs(F)$ indicates the absolute value of F, or if $F(A_{mut}) = 0$, then set the mutated array as best-evaluated ($A(a_i) = A_{mut}(a_i)$, $i = 1, \cdots, n$).
6. *Stop condition.* If the maximum number of iterations has been reached ($Iter > MaxIter$), or the expected number of typical testors has been

found, then return the list of typical testors TT. Otherwise, go to step 3.

4. Experiments

The first experiment consists on a performance comparison between four different algorithms: 1) Genetic Algorithm [32], 2) Univariate Marginal Distribution Algorithm [1], 3) Hill-Climbing algorithm without the acceleration operator, and 4) the method proposed in this paper. These algorithms are denoted hereafter as GA, UMDA, HC and HCTT, respectively. The performance is measured as the number of evaluations required to find a given number of typical testors. All the experiments were conducted in a PC, with a Pentium IV 2Ghz processor, and 1 Gbyte of RAM.

Remark 7. This experiment is intended to compare the number of evaluations required by each algorithm to find a fixed amount of typical testors, as carried out in [32] and [1]. An evaluation involves all the required steps to determine whether a feature combination satisfies the property to be testor, typical testor or none of the above. The execution time of the algorithms is not included due to hardware variations.

Please note that we do not make comparisons with the GA published in [38], because the authors did not provide the algorithm to make comparisons with the proposal Hill Climbing algorithm.

The experiments were carried out with four basic matrices described in [32] and [1]. In this case, the parameters were: $p_t = 0.2$ and $p_{nt} = 0.01$, which were selected after performing a number of experiments with different values from them. The results are shown in table 2. In this table, EV represents the number of evaluations carried out by the algorithm. The dimensions of the matrices are expressed as $rows \times columns$. The goal number of typical testors to find by the compared algorithms is denoted as TTF.

Table 2: Number of evaluations required by: GA, UMDA, simple HC and the HCTT algorithms

Matrices	**TTF**	EV-GA	EV-UMDA	EV-HC	EV-HCTT
1215x105	**105**	22 500 000	336 700	718 356	8 933
269x42	**318**	5 000 000	89 800	138 564	11 036
40x42	**655**	1 400 000	142 500	210 879	30 813
209x47	**1967**	5 000 000	706 900	558 530	80 066

In the same table, (+) denotes that HCTT performed only 400,000 iterations to find such fixed number of typical testors...

Table 3 shows a comparison between HCTT and the deterministic algorithm fast-CT_EXT [34]. We employed six basic matrices described in [32]. For this case, a collection of six matrices described in [32] and [1]. Besides, two new basic matrices with a considerable number of features were tested. For the first five matrices, the number of all typical testor found is known, because fast-CT_EXT calculates this set in a relatively short time. For the remaining three matrices, the entire set still remains unknown. In table 3, (*) denotes that fast-CT_EXT algorithm was added a condition that stops the execution when a fixed number of typical testors has been found. In the same table, (+) denotes that HCTT performed only 400,000 iterations to find such fixed number of typical testors. TIME denotes the run time execution of the algorithm in seconds. TTF and EV are handled in the same way as in Table 2.

We carried out 1 000 000 and 10 000 000 iterations respectively, to verify the computational complexity growth factor, as well as the proportion of typical testors found, when the number of iterations carried out by the algorithm is increased.

Table 3: Run time required and number of typical testors found by fast-CT_EXT and HCTT algorithms

	fast CT_EXT		EV-HCTT = 1 000 000		EV-HCTT = 10 000 000	
Matrices	TTF	TIME	TTF	TIME	TTF	TIME
40x42	8 963	0	2 991	106	5 387	1 147
80x42	32 277	2	5 669	117	11 035	1 024
110x42	65 299	6	8 200	127	19 849	1 286
269x42	302 066	120	11 335	174	38 407	1 837
209x47	184 920	72	7 820	149	20 658	1 620
1215x105	11 166 (*)	15	11 166	809		
	79 467 (*)	348			79 467	9 252
500x160	25 817 (*)	4 246			25 817 (+)	350
	10 000 (*)	1 624	10 077	140		
300x300	0	259 200	3	552	54	5 575

4.1. Discussion

In the first experiment, the execution time of HCTT ranged from 2 to 13 seconds. In all cases, the number of evaluations required by the proposed algorithm (which can be considered as a constant-time process) is significantly lower than that from the compared algorithms.

Table 3 shows that deterministic algorithms are not suitable when dealing with matrices with a large number of feature (for example, hyperspectral images consisting of 256 bands). Unlike them, the proposed hill climbing algorithm was developed to process data sets with a great number of features in training matrix (with 100 features or more).

As the matrix dimension grows, the runtime required to find a fixed number of typical testors by the proposed algorithm becomes considerably less than that of the fast-CT_EXT algorithm.

On the other hand, the typical testors obtained after stopping a deterministic algorithm at a certain moment have no properties in general, because these algorithms are intended to find the entire set of typical testors, but not to find only minimal typical testors, or to find only those where some features appear in most of them, to determine informational weights or the relevance in a specific problem. In this sense, the subset of typical testors obtained by the proposed hill climbing algorithm, provides an equivalent way to calculate the informational weight or relevance of features.

4.2. Stability of the algorithm

We introduce the stability of the proposed algorithm experimentally; in particular, when modifications are made to the parameters of the acceleration operator: p_t and p_{nt}, at the mutation step.

We used two of the basic matrices listed in Table 3 3: BM_{40x42} and BM_{209x47}, varying the value of p_t or p_{nt} and the number of iterations of the algorithm.

Using BM_{40x42}, Figure 1(a) shows the number of typical testors found with $p_t = 0.1$, varying the value of p_{nt} at $0.01, 0.03, 0.05, 0.07, 0.09$, performance 1000000, 3000000, 5000000, 8000000 and 10000000 iterations.

Just as figure 1(a), figures 1(b), 1(c), 1(d) and 1(e) show the number of typical testors found varying the values of p_t at 0.3, 0.5, 0.7, 0.9, and p_{nt} at $0.01, 0.03, 0.05, 0.07, 0.09$, performing the same number of iterations.

Likewise, we use second basic matrix BM_{209x47}. In figures 2(a), 2(b), 2(c), 2(d) and 2(e) the number of typical testors found varying the values of p_t and p_{nt} is shown, performance 1000000 and 10000000 iterations.

As shown in figures 1 and 2, the diference among the number of iterations required and the runtime of the algorithm is small. In all cases, the best results were obtained with $p_{nt} = 0.01$ and $p_t = 0.9$ (considering a balance among the number of typical testors found, number of iterations required and run time excecution of the algorithm). Besides, as the number of iterations grows, also increases the number of typical testors found.

The runtime spent on finding a subset of typical testors is similar. As the number of iterations grows, the run time of the algorithm increases too. In table 4, the maximum and minimum values of the run time required for BM_{40x42} are shown. NI denote the minimum and maximum values, respectively, of runtime spent by the Hill-Climbing algorithm. Likewise, in Table 5, the maximum and minimum values of the runtime required for BM_{209x47} are shown. NI, MN and MX are used in the same way as in table 4.

Table 4: Execution time in seconds for BM_{40x42}

	$p_t = 0.1$		$p_t = 0.3$		$p_t = 0.5$		$p_t = 0.7$		$p_t = 0.9$	
NI	MN	MX	MN	MX	MN	MX	MN	MX	MN	MX
1000000	30	36	22	30	20	28	18	30	18	27
3000000	90	112	71	91	61	86	56	94	56	80
5000000	153	188	121	151	108	144	95	157	97	140
8000000	252	303	198	244	171	234	156	261	157	217
10000000	313	384	233	314	218	300	201	336	196	270

Table 5: Execution time in seconds for BM_{209x47}

	$p_t = 0.1$		$p_t = 0.3$		$p_t = 0.5$		$p_t = 0.7$		$p_t = 0.9$	
NI	MN	MX	MN	MX	MN	MX	MN	MX	MN	MX
1000000	113	204	113	187	149	299	151	249	153	270
10000000	2100	3810	1961	3985	2021	3743	1841	5105	1860	3692

5. Conclusions

A new Hill Climbing algorithm that incorporates an acceleration operation for generating typical testor from a training matrix was introduced.

This acceleration operator had a powerful effect on reducing the number of computations required to find a given number of typical testors.

The superior performance of the proposed algorithm over: a) the Genetic Algorithm reported in [32], b) the UMDA publised in [1], and c) a simple Hill-Climbing was shown in this paper and experimentally demonstrated.

The Hill Climbing algorithm with the acceleration operator generates the same number of typical testors as the reported heuristics, but with a fewer number of evaluations and with significantly less time.

If the number of features is not big, it is convenient to choose a deterministic algorithm -such as fast-CT_EXT- and go for the entire set of typical testors. As this number gets bigger, say over one hundred, the execution time required by a deterministic algorithm grows exponentially because of the combinatorial explosion, and there is a chance that not a single typical testor could be found. In such a case, the proposed hill-climbing algorithm will be useful; naturally, if the number of features is bigger, this algoritm will run more iterations to find a fixed number of typical testors, but the execution time grows polynomially.

Future work includes the implementation of the hill-climbing algorithm on hardware devices, such as Field Programmable Gate Arrays and Graphics Processing Units, in order to accelerate the calculation of typical testors.

References

[1] Alba, E., Santana, R., Ochoa, A., Lazo, M.: Finding Typical Testors By Using an Evolutionary Strategy. Proc. of V Iberoamerican Workshop on Pattern Recognition, Lisbon, Portugal, 267-278 (2000)

[2] Aguila-Feroz, L., Ruiz-Shulcloper, J.: Algorithm MB for the elaboration of k-valued information in pattern recognition problems. Revista Ciencias Matematicas, Cuba (in Spanish), V, 3, 89-101 (1984)

[3] Bravo-Martinez, A.: CT algorithm for the calculation of typical testors

of a k-valued matrix. Revista Ciencias Matematicas, Cuba (in Spanish), IV, 2, 123-144 (1983) (in Spanish)

[4] Bravo, A., Ruiz-Shulcloper, J.: A natural algorithm for elaborating the information in recognition problems. Revista Ciencias Matematicas, Cuba (in Spanish), III, 3, 155-163 (1982)

[5] Carrasco-Ochoa, J.A., and Martnez-Trinidad, J.F.: Feature selection for natural disaster texts classification using testors. Proc. 5^{th} Int. Conf. on Intelligent Data Engineering and Automated Learning, LCNS, vol. 3177, Springer, 424–429 (2004)

[6] De Jong, K.: Evolutionary computation. John Wiley & Sons, Inc., 1, 52-56 (2009)

[7] 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. Proc. 12th International Conference on Intelligent Data Engineering and Automated Learning, LNCS, vol.6936, Springer, 58-65 (2011)

[8] Diukova, E.V.: About an algorithm for constructing test. Sbornik, rabot po matematicheskoi, Kibernetiki (in Russian), 1, 167–185 (1976)

[9] Dmitriev, A., Zhuravlev, I., Krendeliev, F.: About mathematical principles and phenomena classification. Diskretni Analiz (in Russian), 7, 3-15 (1966)

[10] Hernandez-Rodriguez, S., Carrasco-Ochoa, J., Martnez-Trinidad, J.:

Fast k Most Similar Neighbor Classier for Mixed Data Based on Approximating and Eliminating. Proc. 12th Pacic-Asia Conference on Knowledge Discovery and Data Mining, LNAI, vol. 5012 Springer, 697-704 (2008).

[11] Inza, I., Larrañaga, P., Etxeberria, R., Sierra, B.: Feature Subset Selection by Bayesian networks based optimization. Artificial Intelligence, no. 123, 1, 157-184 (1999)

[12] Jensen, M. : Helper-objetives: Using multi-objetive evolutionary Algorithms for single-objetive optimization. Journal of Mathematical Modeling and Algorithms, 4, 323-347, (2004).

[13] Kohavi, R., Jhon, G.: Wrappers for Feature Subset Selection. Artificial Intelligence, 97, 273-324, (1997)

[14] Lazo-Cortes, M., Ruiz-Shulcloper, J., Alba-Cabrera, E.: An Overview of the evolution of the concept of testor. Pattern Recognition, no. 34, 4, 753-762 (2001)

[15] Lias-Rodriguez, A., Pons-Porrata, A.: BR: A new method for computing all typical testors. Proc. XIV Iberoamerican Conference on Pattern Recognition, LNCS, vol. 5856, Springer, 443-440 (2009)

[16] Lopez-Perez, S., Lazo-Cortes, M., Estrada-Garcia, H.: Medical electrodiagnostic using pattern recognition tools. Proc. of the Iberoamerican Workshop on Pattern Recognition (TIARP 97), 237-244 (1997)

[17] Lozano J., Larrañaga P., Inza I., Bengoetxea I.: Towards a new evolutionary computation: advances in the estimation of distribution algorithms. Springer, 55-82 (2006)

[18] Martinez-Trinidad, J., Guzman-Arenas, A.: The Logical Combinatorial approach for pattern recognition. An overview through selected Works. Pattern Recognition, no. 34, 4, 741-751 (2001)

[19] Mierswa, I., Michael, W.: Information Preserving Multi-Objective Feature Selection for Unsupervised Learning. Proc. of the Genetic and Evolutionary Computation Conference, ACM Press, 1545-1552 (2006)

[20] Ochoa, J., Valdes, M., Moctezuma, I:, Ayala, C.: Dimension Reduction in Image Databases using the Logical Combinatorial Approach. Innovations and advances techniques in systems, computing sciences and software engineering. Springer, 260–265 (2008)

[21] Ortiz-Posadas, M., Martinez-Trinidad, M., Ruiz-Shulcloper, J.: A new approach to diferential diagnosis of diseases. International Journal of Biomedical Computing, no. 40, 3, 179-185 (2001)

[22] Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavori, R.: A Method for the Automatic Summarization of Topic-Based Clusters of Documents. Proc. VIII Iberoamerican Conference on Pattern Recognition, LNCS, vol. 2905, Springer, 596-603 (2003)

[23] Pons-Porrata, A., Gil-Garcia, R., Berlanga-Llavori, R.: Using Typical Testors for Feature Selection in Text Categorization. Proc. XII

Iberoamerican Conference on Pattern Recognition, LNCS, vol. 4756, Springer, 643-652 (2007)

[24] Ruiz-Shulcloper, J.: Pattern Recognition with Mixed and Incomplete Data. Pattern Recognition and Image Analysis, vol. 18, 4, 563-576 (2008).

[25] Ruiz-Shulcloper, J., Abidi, M.: Logical Combinatorial Pattern Recognition: A Review. In: Pandalai, S., (Ed) Recent Research Developments in Pattern Recognition, Transword Research Networks, Kerala, India, 133-176 (2002)

[26] Ruiz-Shulcloper, J. Aguila-Feros, L., Bravo-Martinez, A.: Algorithms foe the automatic processing of information related to the description and classification of objects and phenomena. Revista Ciencias Matematicas, Cuba (in Spanish), IV, 2, 139-149 (1983)

[27] Ruiz-Shulcloper, J., Bravo-Martinez, A., Aguila-Feros, L.: BT and TB algorithms for calculation of all typical testors. Revista Ciencias Matematicas, Cuba (in Spanish), VI, 2, 11-18 (1985)

[28] Ruiz-Shulcloper, J., Lazo-Cortes, M.: Mathematical algorithms for the supervised classification based on fuzzy partial precedence. Mathematical and Computer Modelling, vol. 29, 111-119 (1999)

[29] Ruiz-Shulcloper, J., Soto, A., Fuentes, A.: A characterization of the typical testor concept in terms of a notable set of columns. Revista Ciencias Matematicas, Cuba (in Spanish), I, 2-3, 123-134 (1980)

[30] Saeys, Y., Degroeve, S., Van de Peer, Y.: Digging into Acceptor Splice Site Prediction: An Iterative Feature Selection Approach. Proc. of PKDD, 386-397 (2004)

[31] Sanchez-Diaz, G., Lazo-Cortes, M.: CT_EXT: an external escale algorithm for generated typical testors. Proc. XII Iberoamerican Conference on Pattern Recognition, LNCS, vol. 4756, Springer, 506-514 (2007)

[32] Sanchez-Diaz, G., Lazo-Cortes, M., Fuentes-Chavez, O.: Genetic algorithm for calculating typical testors of minimal cost. Proc. of the Iberoamerican Symposium on Pattern Recognition (SIARP 99), 207-213 (1999)

[33] 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, vol. 5, 6, 1025-1039 (2012).

[34] Sanchez-Diaz, G., Piza-Davila, I., Lazo-Cortes, M., Mora-Gonzalez, M., Salinas-Luna, J.: A Fast Implementation of the CT_EXT Algorithm for the Testor Property Identication. Proc. of 9th Mexican International Conference on Artificial Intelligence, LNAI, vol. 6438, Springer, 92-103 (2010)

[35] Santiesteban-Alganza, Y., Pons-Porrata, A.: LEX: A new algorithm for calculating typical testors. Revista Ciencias Matematicas (Cuba), no. 21, 1, 85-95 (2003)

[36] Santos, J.A., Carrasco, A., Martinez, J.F.: 2004. Feature selection using typical testors applied to estimation to stellar parameters. Computacion y Sistemas, vol. 8, 1, 15–23 (2004)

[37] Schuetze, O., Lara, A., Coello, C.: Evolutionary continuation methods for optimization problems. Proc. of the genetic and evolutionary computation conference, 651-658 (2009)

[38] Torres, D., Ponce-de-Leon, E., Torres, A., Ochoa, A., Diaz, E.: Hybridization of evolutionary mechanisms for feature subset selection in unsupervised learning. Proc. 8th Mexican International Conference on Artificial Intelligence, LNAI, vol. 5845, Springer, 610-621 (2009)

[39] Torres, D., Torres, A., Cuellar, F., Torres M., Ponce-de-Leon, E., Pinales, F.: Evolutionary computation in the identification of Risk Factors, Case of TRALI. To appear in Expert Systems with Applications.

[40] Torres, D., Torres, A., Ponce-de-Leon, E.: Genetic algorithm and typical testors in feature subset selection problem. Proc. 6th Iberoamerican Conf. on Systemics, Cybernetics and Informatics, 1–5 (2006)

[41] Valev, V., Asaithambi, A.: On computational complexity of non-reducible descriptors. Proc. of the IEEE Int. Conf. on Information Reuse and Integration, 208-211 (2003)

[42] Valev, V. Radeva, P.: A method of solving pattern or image recognition problem by learning boolean formulas, Proc. of 11th International Conference on Pattern Recognition, Vol. II, IEEE Computer Society Press, 359362 (1992)

[43] Valev, V., Sankur, B.: Generalized non-reducible descriptors. Pattern Recognition, no. 37, 9, 1809-1815 (2004)

[44] Valev, V., Zhuravlev, Y.: Integer-valued problems of transforming the training tables in k-valued code in pattern recognition problems. Pattern Recognition, no. 24, 4, 283-288 (1991)

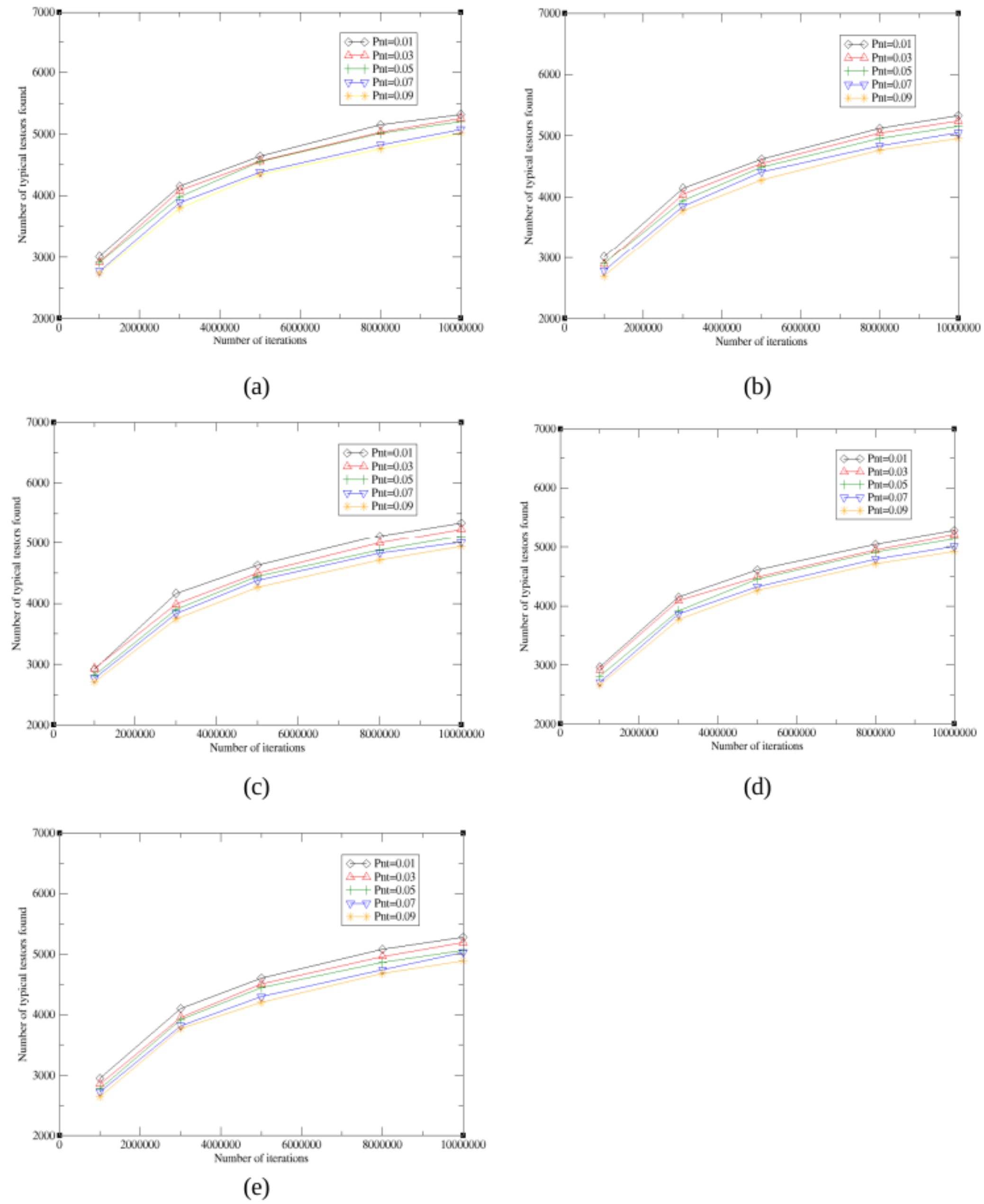

Figure 1: Variation in the number of testors found with different values of p_{nt} and p_t, as follows: (a) $p_t = 0.1$; (b) $p_t = 0.3$; (c) $p_t = 0.5$; (d) $p_t = 0.7$; (e) $p_t = 0.9$

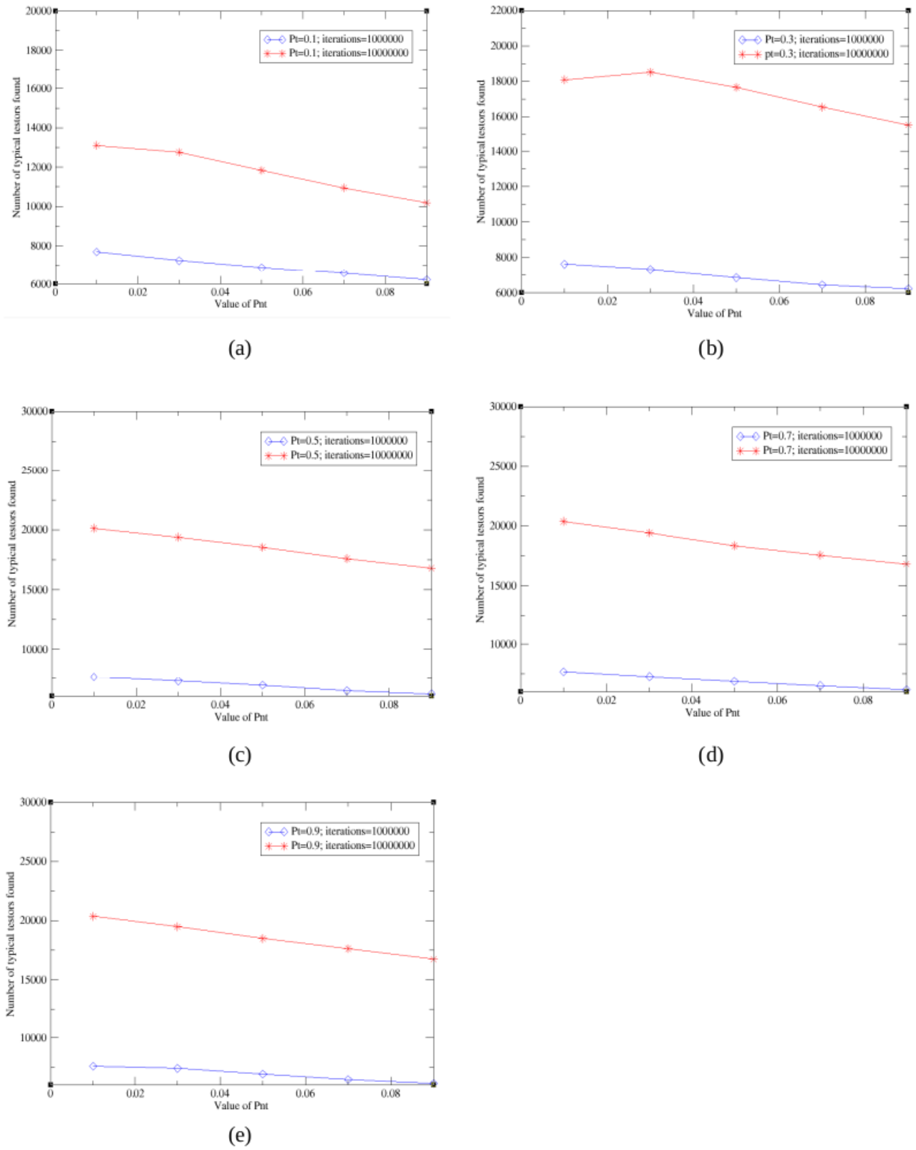

Figure 2: Variation in the number of testors found performing 1000000 and 10000000 iterations, with different values of p_{nt} and p_t, as follows: (a) $p_t = 0.1$; (b) $p_t = 0.3$; (c) $p_t = 0.5$; (d) $p_t = 0.7$; (e) $p_t = 0.9$