Genetic Algorithm to Compute Fuzzy FS-Testors

FRANCISCO MARTINEZ[1], GUILLERMO SANCHEZ[2], BONIFACIO RUGERIO[3]
[1]Departamento de Ciencias Computacionales
Instituto Nacional de Astrofísica, Optica y Electrónica

[2]Centro de Investigación en Tecnologías de Información y Sistemas
Universidad Autónoma del Estado de Hidalgo

[3]Centro de Investigación en Computación
Instituto Politécnico Nacional

MEXICO

Abstract: - In this paper, a genetic algorithm to compute sets of attributes, which maintain similarity between objects in a same class, and at same time, these sets of attributes distinguish objects in a class of objects in the other classes (FS-testors), is presented. Since the computational complexity to compute fuzzy FS-testors is exponential, our algorithm appear as an alternative to solve practical problems of classification or feature selection. The application of the algorithm to a set of objects in a standard public database is shown.

Key-Words: - FS-Testors, Testors, Genetic Algorithm, Feature Selection

1 Introduction

The testor concept grew out related to the fault functions in electric circuits; these functions describe the behavior of a circuit with a certain fault [1]. The first formulation of testor concept, connected to Pattern Recognition to solve classification problems in Geology was introduced by Dmitriev et. al 1966 [2]. A testor in its more simple form is a set of attributes, which does not confuse objects belonging to different classes [3, 4]. In this definition and its proposed extensions the implicit similarity functions to compare objects are Boolean. Alternatively, in those cases where the function is not Boolean, it is assumed that the image of the similarity function can be divided in two disjoint sets. The set of values where the objects are similar and the set where them are dissimilar. Recently, the FS-testor concept for any similarity function was introduced [5,6]. This concept has as characteristic that the comparison function between objects can be anyone. In this new concept only is assumed that the comparison between r-tuples of membership to the classes is computed with a Boolean function, i.e., given a pair of objects, concerning FS-testor definition, both objects are in a same class or in different classes.

Therefore, a set of attributes is a FS-testor if the objects in different classes are low similar (the set of attributes does not confuse objects belonging to different classes), and at same time, objects into the same class are very similar. In this paper we are interested in compute those sets of attributes, which satisfy the definition of FS-testor in certain degree (fuzzy FS-testors).

On the other hand, the computational complexity to compute fuzzy FS-testors in a training sample (grouped in classes) depends exponentially of the amount of attributes used to describe the objects. If the amount of attributes is not very great, (more than 30 attributes, for example) the time required to compute the fuzzy FS-testors may be considerable. This situation diminishes the applicability of fuzzy FS-testors to solve practical problems of classification or feature selection.

The genetic algorithms are optimization and random search techniques guided by genetic and natural evolution principles, which can help us to solve the problem of compute fuzzy FS-testors in less time. These algorithms are processes of efficient search, robust, which obtain solutions near to the optimum [7,8].

In this paper, a genetic algorithm to compute fuzzy FS-testors will be presented. This algorithm is the first proposed to compute fuzzy FS-testors.

2 Similarity Functions

Let U be a finite universe of objects.

Let $R=\{x_1,...,x_n\}$ the set of attributes which describe the elements of U, i.e., for each element $O_j \in U$ we can assign a description as the n-tuple $I(O_j)=(x_1(O_j),...,x_n(O_j))$. Let M_p the set of admissible values for x_p, $p=1,...,n$, i.e., $x_p(O_s)\in M_p$, $s=1,...,m$.

Definition Let $T=\{x_{p1} \mid m_T(x_{p1}),...,x_{ps} \mid m_T(x_{ps})\}$ a fuzzy subset of R. A subdescription or partial description of O_j in terms of the attributes in T is the s-tuple

$$I_{/T}(Oj)=((x_{p1}(O_j), m_T(x_{p1})),...,(x_{ps}(O_j), m_T(x_{ps}))).$$

Definition Let Γ be a similarity function that assigns to each pair of subdescriptions (in terms of attributes in the support of the fuzzy set T) the value $\Gamma(I_{/T}(O_i), I_{/T}(O_j))$. This value represents the similarity in terms of the membership degree to T, i.e., $\Gamma : \bigcup_{\substack{T\subseteq R \\ T\neq\varnothing}} [A_{p1}(T)\times...\times A_{ps}(T)]^2 \to V$, where $A_{ph}(T)=(M_{ph}\times\{m_T(x_{ph})\})$, $h=1,...,s$. Also Γ satisfy the following:

1.-Symmetry: $\Gamma(I_{/T}(O_i), I_{/T}(O_j))=\Gamma(I_{/T}(O_j), I_{/T}(O_i))$.

2.-Concordance with partial evaluations: Let $T_1,T_2,...,T_s$, fuzzy subsets of R (non empty subsets) such that

if $\forall h, l=1,2,...,s, l\neq h, sopT_h \cap sopT_1=\varnothing$

$\Gamma(I_{/T_1}(O_i), I_{/T_1}(O_j))\leq\Gamma(I_{/T_1}(O_g), I_{/T1}(O_h))\ldots$

$\Gamma(I_{/T_s}(O_i), I_{/T_s}(O_j))\leq\Gamma(I_{/T_s}(O_g), I_{/T_s}(O_h))$

then, if $T=\bigcup_{u=1}^{s} T_u$ we have that $\Gamma(I_{/T}(O_i), I_{/T}(O_j))\leq\Gamma(I_{/T}(O_g), I_{/T}(O_h))$; where $\leq$ denotes "less or equal to" in the order defined for V.

3.-Maximum similarity:

a) $\forall O_i \in U$ and $\forall T\subseteq R$ $\max_{j=1,...,m}\Gamma(I_{/T}(O_i), I_{/T}(O_j))=\Gamma(I_{/T}(O_i), I_{/T}(O_i))$

b) $\forall T_1, T_2\subseteq R, \forall O_i \in U$ $\Gamma(I_{/T_1}(O_i), I_{/T_1}(O_i)) = \Gamma(I_{/T_2}(O_i), I_{/T_2}(O_i))$

c) $\forall T\subseteq R$ and $\forall O_i, O_j \in U$ $\Gamma(I_{/T}(O_i), I_{/T}(O_i)) = \Gamma(I_{/T}(O_j), I_{/T}(O_j))$

There are many cases where the similarity function is defined using a comparison criteria for each attribute.

Definition A comparison criterion for the attribute $x_i\in R$ is a function $C_i:M_i\times M_i\to V_i$, where $\forall i$ V_i is a totally ordered set, this function give us the similarity degree between a pair of admissible values for x_i.

Here has been introduced the general formulation for Γ, however in those cases where the function is defined for crisp subsets of attributes, the membership degree may be ignored in the analytic expressions.

3 Fuzzy FS-Testors

Let a supervised classification problem with m objects $O_1,...,O_m$ described through n attributes $x_1,...,x_n$ and distributed in q classes $K_1,...,K_q$. This information is represented in the learning matrix LM (see table 1), where $\bar{a}(O_j)=(a_1(O_j),...,a_q(O_j))$ is denominated q-tuple of membership to the classes and $a_h(O_j)$ is the membership degree of O_j to K_h.

Let $R^*\subseteq R$; v a function to compare q-tuples of membership to the classes, $v:[0,1]\times[0,1]\to V'$; and D' the subset of V' where the q-tuples of membership are considered similar.

Definition $T\subseteq R$, is a FS-differentiater set of attributes for LM with respect to v, D', R^* and Γ iff $\forall O_i, O_j \in LM$ $[v(\bar{a}(O_i), \bar{a}(O_j))\notin D'] \Rightarrow [\Gamma(I_{/T}(O_i), I_{/T}(O_j))\leq\Gamma(I_{/R^*}(O_i), I_{/R^*}(O_j))]$ (*differentiating condition*).

	x_1	...	x_n	a_1	...	a_q
O_1	$x_1(O_1)$	...	$x_n(O_1)$	$a_1(O_1)$	...	$a_q(O_1)$
.		.			.	
.		.			.	
.		.			.	
O_m	$x_1(O_m)$	...	$x_n(O_m)$	$a_1(O_m)$	...	$a_q(O_m)$

Table 1. Learning Matrix *LM*.

This definition may be interpreted as follows: T is a FS-differentiater set of attributes for LM with respect to v, D', R^* and Γ, if T distinguishes to each pair of objects with different q-tuples of membership (in different classes) non-worse than R^*. In other words, the similarity considering only the attributes in T for any pair of objects with different q-tuples of membership is less or equal than the similarity considering the attributes in R^*.

The FS-differentiater set of attributes definition preserves the essence of the classical testor definition, i.e., the set of attributes keeps or improves the capability of distinguishing objects against a given reference set. In the case of classical testors, the reference set is R, because the used similarity function (matching function) has the property that when we reduce the amount of attributes, the similarity is preserved or it increases but never decreases. However there are similarity functions, which do not have this property so theoretically, is very important to consider other reference sets.

In the analysis of the informational properties for a set of attributes is very important not only take account its ability to distinguish objects belonging to different classes but also its ability to keep a good similarity between the objects belonging to a same class.

When a classical testor is computed the comparisons between objects belonging to a same class are not taken account. Note that the matching function (used in classical testors) is such that there are not any subset of *R* with worse similarity than *R* in the objects belonging to the same class, i.e., If two objects are similar, comparing them through *R*, the objects will be similar if they are compared through any subset of *R*. However, the new FS-testor concept is applied for any function, and it not necessarily must satisfies this property.

Definition $T \subseteq R$, is a set FS-characterizer of attributes for *LM* with respect to *v*, *D'*, R^* and Γ iff $\forall O_i, O_j \in LM$ $[v(\overline{a}(O_i), \overline{a}(O_j)) \in D'] \Rightarrow [\Gamma(I_{/R^*}(O_i), I_{/R^*}(O_j)) \leq \Gamma(I_{/T}(O_i), I_{/T}(O_j))]$ (*characterizing condition*).

Then a set $T \subseteq R$ of attributes will be a FS-testor if its ability to distinguish and characterize is not worse than the reference set.

Definition $T \subseteq R$ is a FS-testor for *LM* with respect to *v*, *D'*, R^* and Γ iff *T* is at the same time a FS-differentiater and a FS-characterizer set for *LM* with the same parameters.

Given as subset of *R* and a reference set, we can evaluate if *R* is or not a FS testor against the reference set, i.e., we can know if the subset of *R* satisfies or not, the property established by the FS-testor definition. As result of applying this criterion, a crisp family of FS-testors is generated.

The fuzzy family of FS testors is defined as follows: All the subsets satisfying the FS-testor definition belong with degree 1 to the fuzzy FS-testors family. The remaining subsets belong with a degree in the interval [0,1). The membership degree associated to a subset increases if this subset is near of satisfying the properties expressed in the FS-testor definition.

Let *T* and *R** subsets of *R*. Also, let $OD = \{(O_i, O_j)/n(\overline{a}(O_i), \overline{a}(O_j) \notin D')\}$ the set of pairs of objects with different tuples of membership and $C(OD) = \{(O_i, O_j)/n(\overline{a}(O_i), \overline{a}(O_j) \in D')\}$ the set of pairs of objects with similar tuples of membership. Then we define $S^{R^*}(T)$ and $D^{R^*}(T)$ as the following sets:

$$S^{R^*}(T) = \{(O_i, O_j) \in OD, \Gamma(I_{/T}(O_i), I_{/T}(O_j)) > \Gamma(I_{/R^*}(O_i), I_{/R^*}(O_j))\}$$

$$D^{R^*}(T) = \{(O_i, O_j) \in C(OD), \Gamma(I_{/T}(O_i), I_{/T}(O_j)) < \Gamma(I_{/R^*}(O_i), I_{/R^*}(O_j))\}$$

$S^{R^*}(T)$ contains the pairs of objects, belonging to different classes not satisfying the differentiating property. $D^{R^*}(T)$contains the pairs of objects, belonging to the same class not satisfying the characterizing property.

Definition The fuzzy family of FS-testors for *LM* with respect to *v*, *D'*, R^* and Γ is defined as the fuzzy set $\xi = \{T_p \mid m_x(T_p) : T_p \subseteq R\}$, where $m_x(T_p) = 1 - \frac{|S^{R^*}(T_p) \cup D^{R^*}(T_p)|}{|OD \cup C(OD)|}$.

Defining the fuzzy family of FS-testors we can distinguish all the subsets of attributes that are not FS-testors. In addition, the above definition establishes a magnitude that informs us how many near satisfying FS-testor definition a subset is. The membership degree evaluates the capability of a set of attributes (against the reference set) to distinguish objects in different classes and characterize objects into a same class.

4 Genetic Algorithm

The individuals handled by the genetic algorithm are represented as *n*-tuples composed by *0* and *1* values (genes), these values represent respectively absence or presence of an attribute and *n* is the total number of attributes.

We denote these n-tuples as T_i, $i=1,...,m$, where *m* is the number of individuals in a population. In addition, we denote the attribute subset of each individual as $I_i=\{x_{i1},...,x_{ik}\}$, where $k \leq n$. For example, if $T_1=[0,0,1,0,1,0,0,1,1]$ then this represents the subset of attributes $I_1=\{x_3,x_5,x_8,x_9\}$.

The proposed genetic algorithm uses the following operators:

Fitness function. This function is used to determine how many pairs of objects do not satisfy the characterizing and differentiating properties, when objects are compared in a subset of attributes.

The fitness function is the following

$$M(I) = 1 - \frac{|S^{R^*}(I) \cup D^{R^*}(I)|}{|OD \cup C(OD)|}$$

Where *M(I)* is the fitness for the *I* subset, or the membership degree of *I* to the fuzzy FS-testors set. Note when *S(I)* (the pairs of objects, belonging to different classes not satisfying the differentiating property) and *D(I)* (the pairs of objects, belonging to the same class not satisfying the characterizing property) increase *M(I)* tends to 0, and vice versa.

Crossover. To apply the crossover operator, the population is ordered in descendent way, with respect to the fitness function. We take two individuals and apply uniform crossover, using a mask randomly created, to generate two new individuals. The first and the last individuals in the population are crossing over. This procedure with the remaining objects is repeated (in the defined order).

Mutation. We apply the generative mutation operator in the algorithm. This operator takes randomly one point of the individual (gene), and changes its value, if the point has value 0, the new value will be 1, and vice versa.

In general, the genetic algorithm proceeds as follow: First, it generates the initial individual population randomly. The size (i.e. number of genes) of each individual will be the number of attributes. Each point of an individual will be 0 or 1 value. Second, all individuals are evaluated in order to determine their fitness (the algorithm verifies if each individual is characterizing and differentiating). The individuals of the population are crossed between them, to generate another new. In this procedure, the attributes of the individuals with great fitness are preserved (i.e. the individuals that were more characterizing and differentiating). The crossover operation generates a new population that can replace the previous population, or this new population can be mixed with the old population in order to get a population with better fitness.

The proposed algorithm in this paper is as follows:

Genetic Algorithm to compute fuzzy FS-testors.

Input: LM (Learning Matrix), Size (population size), and Num_iter (number of iterations).

Output: FS (Fuzzy FS-testors subset)

```
1.  FS ← ∅
    Generate_initial_population(current_population)
    Evaluate_population(current_population, FS)
2.  Repeat for i=1 until i=Num_iter
        Crossover(current_population,
                  new_population)
        Evaluate_population(new_population, FS)
        Actualize_population(current_population,
                             new_population)
        Mutation(current_population)
        Evaluate_population(current_population,
                            FS)
3.      Print(FS)
```

The function *Generate_initial_population(population)* randomly creates the initial *population*. *Evaluate_population(population, FS)* evaluates *population* and keeps the fuzzy FS-testors found in the *FS* list. The *Crossover(source_population, target_population)* procedure applies the crossover operator to *source_population*, and generate a new population, which will be keep on *target_population*.

Actualize_population(target_population, source_population) copy the individuals from *source_population* to *target_population*. Finally, the *Mutation(population)* procedure applies the mutation operator to *population*.

5 Experimentation

An example of application using a public standard database is shown. We have chosen the zoo database from http://www1.ics.uci.edu/pub/machine-learning-databases/. This data is a simple database containing 16 Boolean-valued attributes (from x_1 to x_{16}). The attributes used to describe animals are: hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic, and catsize. All of them are Boolean except the attribute legs which is numeric. The animals are distributed in 7 classes with 41, 20, 5, 13, 4, 8, and 10 animals respectively.

In this example, we defined as comparison criteria for all attributes, the Boolean function that assigns one if two values for an attribute are equal and zero in otherwise, see (1).

$$C\left(x(O_i), x(O_j)\right) = \begin{cases} 1 & if \quad x(O_i) = x(O_j) \\ 0 & otherwise \end{cases} \qquad (1)$$

The animals in zoo database are organized in seven classes then a tuple of membership for O_i has seven entries, one for each class. The function to compare *7*-tuples of membership to the classes is defined in (2). Therefore we consider that *D'*=1, i.e., two *7*-tuples will be similar if th function *v* take vale 1.

$$n\left(\overline{a}(O_i), \overline{a}(O_j)\right) = \begin{cases} 1 & if \quad \overline{a}(O_i) = \overline{a}(O_j) \\ 0 & otherwise \end{cases} \qquad (2)$$

Here R^* (the reference set) is the set *R*, the whole set of attributes, we will compare subsets of *R* against *R*.

The similarity function used in this example is such that allow us compare two subdescriptions $I_{/T}(O_i)$ and $I_{/T}(O_j)$ in terms of the attributes in the set *T*. This function depends of the comparison criteria for the attributes (see (3)).

$$\Gamma\left(I_{/T}(O_i), I_{/T}(O_j)\right) = \frac{1}{|T|} \sum_{x \in T} C\left(x(O_i), x(O_j)\right) \quad (3)$$

Remind that the similarity function, in this approach, is a parameter so if the data set changes we must define an appropriate similarity function according to the practical problem.

In the table 2, we show the fuzzy FS-testors with respect to *v*, *D'*, R^* and Γ after apply our genetic algorithm. Here we present three proofs in a PC Pentium II, 400Mhz. In each column appear the fuzzy FS testors for each class. In all cases, we define population size as 15 and the number of iterations as 10. The first result was accomplished in 5 minutes and 5 seconds. The second and third results were achieved in 5minutes and 37 seconds and 6 minutes and 2 seconds respectively.

Fuzzy FS-testors, following the definition, are such that if we compare any pair of objects into the same class through a FS-testor (with similar tuple of membership) then we will get high similarity. In addition, if we compare objects of the class against objects out of class through a FS-testor then we will get low similarity. We can see above property using the class 3 and 6 shown in the table 3. In his table appear the subdescription of animals in terms of attributes belonging to the fuzzy FS-testor ($\{x_1, x_6, x_8, x_{11}, x_{14}, x_{16}\}$,0.67). If we evaluate the similarity between pitviper and seasnake using (3) then we obtain a similarity of 0.833, while if we compare pitviper (in class 3) with the first frog (in class 6) then we get a similarity of 0.5. In general, the similarity between animals in the same class is higher than similarity between animals of different classes.

Animal	x_1	x_6	x_8	x_{11}	x_{14}	x_{16}
pitviper	0	0	1	1	1	0
seasnake	0	1	1	1	1	0
slowworm	0	0	1	0	1	0
tortoise	0	0	0	0	1	1
tuatara	0	0	1	0	1	0
------------						
frog	0	1	1	0	0	0
frog	0	1	1	1	0	0
newt	0	1	1	0	1	0
toad	0	1	1	0	0	0

Table 3. Descriptions of animals for class 3 and 6 in terms of attributes in the fuzzy FS-testor ($\{x_1, x_6, x_8, x_{11}, x_{14}, x_{16}\}$, 0.67).

In this example we show how computing FS testor we can find sets of attributes that help us to characterize objects in a class, and at same time, distinguish them of objects in other classes.

6 Conclusions

The proposed algorithm allows handling any similarity function to compare two objects of the data set. Therefore, it eliminates the restriction presented by other techniques based on genetic algorithms and evolutionary strategies, which calculate testors handling Boolean similarity functions [9,10]. In addition, the proposed algorithm can process data sets containing a large number of attributes and objects.

Here only we present the genetic algorithm and some results on public standard databases. In a second stage, we pretend to include this algorithm to solve supervised classification problems. We will use the algorithms based in partial precedence as *voting algorithms*, *Kora-W* and *representative sets* [11]. These algorithms allow us in natural way consider FS-testors and their membership degree, in order to classify new objects. The idea of these algorithms is to classify objects according to their description and their similitude with the objects already classified in the K_i classes. The similarity is evaluated analyzing sub-descriptions of objects instead of whole descriptions. The FS-testors define the set of features that should be compared through a convenient similarity function. Therefore, It would be possible to apply any of these algorithms to *LM* in order to classify new objects, obtaining the decision criteria for the class it belongs to.

On the other hand, there are applications that can utilize a subset of fuzzy FS-Testors to improve their outcome. For example: the LC conceptual clustering algorithm, which can use fuzzy FS-Testors to characterize clusters. In addition, we can mention the k-d algorithm [13], which can generates a decision tree as a fuzzy FS-Testor. The fuzzy FS-testors can be used to solve problems of dimensionality reduction as [14] but applying fuzzy FS-testors.

References:

[1] Cheguis, I. and Yablonskii, S., *About testors for electrical outlines.* Usp. Mat. Nauk 4, 66, 1955, pp. 182-184 (in Russian).

[2] Dimitriev, A., Zhuravliev, Y. and Krendeleiev, F., *On the mathematical principles of pattern and phenomena classification.* Diskretnyi Anal 7, 1966, pp. 3-15 (in Russian).

[3] Lazo-Cortes M., Ruiz Shulcloper J., and Cabrera-Alba E., *An overview of the evolution of the evolution of the concept of testor.* Pattern Recognition, 34, 2001, pp. 753-762.

[4] Martínez-Trinidad F. and Guzmán-Arenas A., *The logical combinatorial approach to pattern recognition an overview through selected works.* Pattern Recognition, 34, 4, 2001, pp. 1-11.

[5] Alba-Cabrera E. *Nuevas extensiones del concepto de testor para diferentes tipos de funciones de semejanza.* Tesis Doctoral Universidad de la Habana, Cuba, 1997. (In Spanish).

[6] Alba E. and Lazo M. *Una solución global para la utilización de los testores en problemas de reconocimiento de patrones.* Proc. III Taller Iberoamericano de Reconocimiento de Patrones, Mexico, 1998, pp.209-218 (In spanish).

[7] Michalewicz Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer 1996, USA.

[8] Mitchell Tom. *Machine Learning.* McGraw Hill, 1997, USA.

[9] Sánchez G., Lazo M., and Fuentes O. *Algoritmo genético para calcular testores típicos de costo mínimo.* Proc. IV Simposium Iberoamericano de Reconocimiento de Patrones, La Habana, Cuba, 1999, pp. 207-213 (In Spanish).

[10] Alba E., Santana R., Ochoa A., and Lazo M. *Finding typical testors by using an evolutionary strategy.* Proc. 5th Iberoamerican Symposium on Pattern Recognition, Lisbon, Portugal, 2000, pp. 267-278.

[11] J. Ruiz-Shulcloper and M. Lazo-Cortés, *Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence, Mathematical and Computer Modelling* 29, 1999, pp. 111-119.

[12] Martínez-Trinidad F., Ruiz J. *Algoritmo LC conceptual para el agrupamiento de objetos.* Proc. Simp. Int. de Computación CIC-IPN, Mexico, 1997, pp. 411-418, (In Spanish).

[13] García A., Martínez G., Nuñez E., Guzmán A. *Clasificación supervisada, inducción de arboles de decisión, algoritmo k-d.* Proc. Simp. Int. de Comp. CIC´98, CIC-IPN, México, 1997, pp. 602-614 (In Spanish).

[14] Lazo M., Douglas de la Peña and M., Quintana T. *Testores por clase. Una aplicación al reconocimiento de caracteres.* Proc. III taller Iberoamericano de Reconocimiento de Patrones, México, 1998, pp. 229-236 (In Spanish).

Class	Fuzzy FS-testors	Fuzzy FS-testors	Fuzzy FS-testors
1	$(\{x_1, x_2, x_7, x_{11}\},0.90)$	$(\{x_1, x_3, x_4, x_7, x_{10}, x_{12}, x_{13}\},0.79)$ $(\{x_3, x_4, x_5, x_7, x_8, x_{12}\},0.76)$ $(\{x_1, x_4, x_7, x_9, x_{13}, x_{14}\},0.75)$	$(\{x_1, x_3, x_6, x_8, x_{13}, x_{14}, x_{15}\},0.77)$
2	$(\{x_1, x_2, x_3, x_5, x_6, x_8, x_9, x_{13}\},0.97)$ $(\{x_2, x_3\},0.93)$	$(\{x_2, x_3, x_8, x_9, x_{10}, x_{13}, x_{16}\},0.88)$ $(\{x_1, x_2, x_5, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{16}\},0.85)$	$(\{x_1, x_2, x_3, x_8, x_{10}, x_{11}, x_{13}, x_{14}, x_{15}\},0.67)$ $(\{x_4, x_5, x_7, x_{12}, x_{13}, x_{15}\},0.64)$
3	$(\{x_1, x_6, x_8, x_{11}, x_{14}, x_{16}\},0.67)$	$(\{x_1, x_8, x_{10}, x_{13}\},0.83)$ $(\{x_6, x_{13}\},0.81)$	$(\{x_1, x_4, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{16}\},0.73)$
4	$(\{x_7\ x_8\ x_{10}\ x_{13}\},0.89)$	$(\{x_1, x_8, x_{12}, x_{13}, x_{15}\},0.89)$	$(\{x_2, x_4, x_6, x_{10}, x_{12}, x_{13}, x_{14}, x_{16}\},0.95)$ $(\{x_1, x_3, x_8, x_{10}, x_{12}, x_{13}, x_{16}\},0.93)$
5	$(\{x_2, x_3, x_7, x_8, x_{10}, x_{11}, x_{13}, x_{16}\},0.62)$ $(\{x_5, x_6, x_7, x_8, x_9, x_{11}, x_{13}, x_{14}\},0.61)$ $(\{x_3, x_7, x_9, x_{16}\},0.58)$ $(\{x_3, x_4, x_6, x_{10}, x_{11}, x_{13}, x_{15}\},0.57)$	$(\{x_1, x_2, x_3, x_4, x_5, x_6, x_{10}, x_{13}, x_{14}\},0.76)$	$(\{x_5, x_6, x_{12}, x_{14}, x_{16}\},0.73)$ $(\{x_1, x_4, x_{11}, x_{13}, x_{14}, x_{16}\},0.71)$
6	$(\{x_1, x_4, x_5, x_6, x_{13}, x_{14}\},0.91)$ $(\{x_2, x_4, x_5, x_6, x_8, x_9, x_{10}, x_{13}\},0.86)$	$(\{x_4, x_6, x_7, x_8, x_{11}, x_{13}, x_{14}, x_{15}\},0.71)$ $(\{x_2, x_5, x_7, x_8, x_{13}, x_{15}, x_{16}\},0.66)$	$(\{x_1, x_4, x_6, x_{13}, x_{14}, x_{16}\},0.84)$
7	$(\{x_1, x_{10}, x_{13}\},0.76)$	$(\{x_4, x_6, x_7, x_9, x_{11}, x_{12}, x_{13}, x_{14}\},0.84)$	$(\{x_1, x_7, x_{10}, x_{13}\},0.76)$

Table 2. FS-Testors for zoo database.