Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 41 (2008) 2693–2709
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

A memetic algorithm for evolutionary prototype selection: A scaling up approach☆

Salvador García[a,*], José Ramón Cano[b], Francisco Herrera[a]

[a]Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain
[b]Department of Computer Science, University of Jaén, 23700 Linares, Jaén, Spain

Received 8 September 2007; received in revised form 18 January 2008; accepted 14 February 2008

Abstract

Prototype selection problem consists of reducing the size of databases by removing samples that are considered noisy or not influential on nearest neighbour classification tasks. Evolutionary algorithms have been used recently for prototype selection showing good results. However, due to the complexity of this problem when the size of the databases increases, the behaviour of evolutionary algorithms could deteriorate considerably because of a lack of convergence. This additional problem is known as the scaling up problem.

Memetic algorithms are approaches for heuristic searches in optimization problems that combine a population-based algorithm with a local search. In this paper, we propose a model of memetic algorithm that incorporates an ad hoc local search specifically designed for optimizing the properties of prototype selection problem with the aim of tackling the scaling up problem. In order to check its performance, we have carried out an empirical study including a comparison between our proposal and previous evolutionary and non-evolutionary approaches studied in the literature.

The results have been contrasted with the use of non-parametric statistical procedures and show that our approach outperforms previously studied methods, especially when the database scales up.
© 2008 Elsevier Ltd. All rights reserved.

Keywords: Data reduction; Evolutionary algorithms; Memetic algorithms; Prototype selection; Scaling up; Nearest neighbour rule; Data mining

1. Introduction

Considering supervised classification problems, we usually have a training set of samples in which each example is labelled according to a given class. Inside the family of supervised classifiers, we can find the nearest neighbour (NN) rule method [1,2] that predicts the class of a new prototype by computing a similarity [3,4] measure between it and all prototypes from the training set, called the k-nearest neighbours (k-NN) classifier. Recent studies show that k-NN classifier could be improved by employing numerous procedures. Among them, we could cite proposals on instance reduction [5,6], for incorporating weights for improving classification [7], and for accelerating classification task [8], etc.

Prototype selection (PS) is an instance reduction process consisting of maintaining those instances that are more relevant in the classification task of the k-NN algorithm and removing the redundant ones. This attempts to reduce the number of rows in data set with no loss of classification accuracy and obtain an improvement in the classifier. Various approaches of PS algorithms were proposed in the literature, see Refs. [6,9] for review. Another process used for reducing the number of instances in training data is the prototype generation, which consists of building new examples by combining or computing several metrics among original data and including them into the subset of training data [10].

Evolutionary algorithms (EAs) have been successfully used in different data mining problems (see Refs. [11–13]). Given that PS problem could be seen as a combinatorial problem, EAs [14] have been used to solve it with promising results [15], which we have termed evolutionary prototype selection (EPS).

☆ This work was supported by Projects TIN2005-08386-C05-01 and TIN2005-08386-C05-03. S. García holds a FPU scholarship from Spanish Ministry of Education and Science.
* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: salvagl@decsai.ugr.es (S. García), jrcano@ujaen.es (J.R. Cano), herrera@decsai.ugr.es (F. Herrera).

0031-3203/$30.00 © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.02.006

The increase of the database's size is a staple problem in PS (which is known as scaling up problem). This problem produces excessive storage requirement, increases time complexity and affects generalization accuracy. These drawbacks are presented in EPS because they result in an increment in chromosome size and time execution and also involve a decrease in convergence capabilities of the EA. Traditional EPS approaches generally suffer from excessively slow convergence between solutions because of their failure to exploit local information. This often limits the practicality of EAs on many large-scale problems where computational time is a crucial consideration. A first rapprochement about the use of EAs when this problem scales up can be found in Ref. [16].

The combination of EAs with local search (LS) was named "memetic algorithm" (MA) in Ref. [17]. Formally, a MA is defined as an EA that includes one or more LS phases within its evolutionary cycle [18]. The choice of name is inspired by concept of a *meme*, which represents a unit of cultural evolution that can show local refinement [19]. MAs have been shown to be more efficient (i.e., needing fewer evaluations to find optima) and more effective (identifying higher quality solutions) than traditional EAs for some problem domains. In the literature, we can find a lot of applications of MAs for different problems; see Ref. [20] for an understanding of MA issues and examples of MAs applied to different domain problems.

The aim of this paper is to present a proposal of MA for EPS for dealing with the scaling up problem. The process of designing effective and efficient MAs currently remains fairly ad hoc. It is frequently hidden behind problem-specific details. In our case, the meme used is ad hoc designed for the PS problem, taking advantage of its divisible nature and simplicity of hybridization within the EA itself, and allowing us good convergence with increase of the problem size. We will compare it with other EPS and non-EPS algorithms already studied in the literature, paying special attention to the scaling up problem, analysing its behaviour when we increase the problem size.

This paper is organized in the following manner. Section 2 presents the PS problem formally and enumerates some PS methods. A review of EPS is given in Section 3. In Section 4 we explain our MA approach and meme procedure. Details of empirical experiments and results obtained are reported in Section 5. Section 6 contains a brief summary of the work and the conclusions reached.

2. Preliminaries: PS

PS methods are instance selection methods [5] which expect to find training sets offering best classification accuracy by using the nearest neighbour rule (1-NN).

A formal specification of the problem is the following: Let $\overrightarrow{x_p}$ an example where $\overrightarrow{x_p} = (x_{p1}, x_{p2}, \ldots, x_{pm}, x_{pl})$, with $\overrightarrow{x_p}$ belonging to a class c given by x_{pl} and a m-dimensional space in which x_{pi} is the value of the ith feature of the pth sample. Then, let us assume that there is a training set TR which consists of n instances $\overrightarrow{x_p}$ and a test set TS composed by t instances $\overrightarrow{x_p}$. Let $S \subseteq TR$ be the subset of selected samples resulted for the execution of a PS algorithm, then we classify a new pattern from TS by the 1-NN rule acting over S.

Wilson and Martinez in Ref. [6] suggest that the determination of the k value in the k-NN classifier may depend according to the proposal of the PS algorithm. In k-NN, setting k greater than 1, decreases the sensitivity of the algorithm to noise and tends to smooth the decision boundaries. In some PS algorithms, a value $k > 1$ may be convenient, when its interest lies in protecting the classification task of noisy instances. In any case, Wilson and Martinez state that it may be appropriate to find a value of k to use during the reduction process, and then redetermine the best value of k in the classification task. In EPS we have used the value $k = 1$, given that EAs need to have the greatest possible sensitivity to noise during the reduction process. In this manner, an EPS algorithm could better detect the noisy instances and the redundant ones in order to find a good subset of instances perfectly adapted to the simplest method of NNs. By considering only an instance during the evolutionary process, the reduction–accuracy trade-off is more balanced and the efficiency is improved. The implication of this fact is the use of $k = 1$ in the classification, as Wilson and Martinez point out.

In the next subsection, we will describe the algorithms used in this study but not the EAs (which will be described in Section 3).

2.1. PS methods

Algorithms for PS may be classified according to the heuristic followed in the selection. We have selected the most representative and well-known methods belonging to the non-evolutionary family and the algorithms that offer the best performance for the PS problem.

- Enn [21]. Edited NN edits out noisy instances, as well as close border cases, leaving smoother decision boundaries. It also retains internal points. It works by editing out those instances in which class does not agree with the majority of classes of its k NNs.
- Allknn [22]. Allknn is an extension of Enn. The algorithm, for $i = 1$ to k flags as bad any instance not correctly classified by its i NNs. When the loop is completed k times, it removes the instances flagged as bad.
- Pop [23]. This algorithm consists of eliminating the samples that are not within the limits of the decision boundaries. This means that its behaviour is in opposite direction from that of Enn and Allknn.
- Rnn [24]. The reduced NN rule searches a minimal and consistent subset which correctly classifies all the learning instances.
- Drop3 [6]. An associate of $\overrightarrow{x_p}$ is that sample $\overrightarrow{x_i}$ which has $\overrightarrow{x_p}$ as NN. This method removes $\overrightarrow{x_p}$ if at least as many of its associates in TR would be classified correctly without $\overrightarrow{x_p}$. Prior to this process, it applies a noise reduction filter (Enn).
- Ib3 [25]. It introduces the *acceptable* concept, based on the statistical confidence of inserting a certain instance in the subset, to carry out the selection.

- Cpruner [26]. C-Pruner is a sophisticated algorithm constructed by extending concepts and procedures taken from algorithms Icf [27] and Drop3.
- Explore [28]. Cameron-Jones used an *encoding length heuristic* to determine how good the subset S is in describing TR. Explore is the most complete method belonging to this group and it includes three tasks:
  - It starts from the empty set S and adds instances if only the cost function is minimized.
  - After this, it tries to remove instances if this helps to minimize the cost function.
  - Additionally, it performs 1000 mutations to try to improve the classification accuracy.
- Rmhc [29]. First, it randomly selects a subset S from TR which contains a fixed number of instances s ($s = \%|TR|$). In each iteration, the algorithm interchanges an instance from S with another from $TR - S$. The change is maintained if it offers better accuracy.
- Rng [30]. It builds a graph associated with TR in which a relation of neighbourhood among instances is reflected. Misclassified instances by using this graph are discarded following a specific criterion.

2.2. The scaling up problem

Any algorithm is affected when the size of the problem which it is applied increases. This is the scaling up problem, characterized for producing:

- Excessive storage requirements.
- Increment of time complexity.
- Decrement of generalization capacity, introducing noise and over-fitting.

A way of avoiding the drawbacks of this problem was proposed in Ref. [16], where a stratified strategy divides the initial data set into disjoint strata with equal class distribution. The number of strata chosen will determine their size, depending on the size of the data set. Using the proper number of strata we can significantly reduce the training set and we could avoid the drawbacks mentioned above.

Following the stratified strategy, initial data set D is divided into t disjoint sets D_j, strata of equal size, $D_1, D_2, \ldots, D_t$ maintaining class distribution within each subset. Then, PS algorithms will be applied to each D_j obtaining a selected subset DS_j. Stratified prototype subset selected (SPSS) is defined as

$$SPSS = \bigcup_{j \in J} DS_j, \quad J \subset \{1, 2, \ldots, t\}. \tag{1}$$

3. EPS: a review

In this section, we will review the main contributions that have included or proposed an EPS model.

The first appearance of application of an EA to PS problem can be found in Ref. [31]. Kuncheva applied a genetic algorithm (GA) to select a reference set for the k-NN rule. Her GA maps the TR set onto a chromosome structure composed by genes, each one with two possible states (binary representation). The computed fitness function measures the error rate by application of the k-NN rule. This GA was improved in Refs. [32,33].

At this point, all EPS algorithms reviewed above correspond to adapting a classical GA model to PS problem. Later, a development of more conditioned EPS algorithms to the problem is made. The first example of this can be found in Ref. [34]. In this paper, an estimation of distribution algorithm (EDA) is used.

A GA design for obtaining an optimal NN classifier is proposed in Ref. [35]. Ho et al. propose an intelligent genetic algorithm (IGA) based on orthogonal experimental design used for PS and feature selection. IGA is a GGA that incorporates an intelligent crossover (IC) operator. IC builds an orthogonal array (OA) (see Ref. [35]) from two parents of chromosomes and searches within the OA for the two best individuals according to the fitness function. It takes about $2^{\lceil \log_2(\gamma+1) \rceil}$ fitness evaluations to perform an IC operation, where γ is the number of bits that differ between both parents. Note that only an application of IC on large-size chromosomes (resulting chromosomes from large-size data sets) could consume a high number of evaluations.

The technical term EPS has been adopted by Cano et al. in Ref. [15], in which they analyse the behaviour of different EAs, steady-state GAs (SSGAs), GGAs, the CHC model [36] and PBIL [37] (which can be considered as one of the basic EDAs). The representation of solutions as chromosomes follows the guidelines in Ref. [31], but fitness function used in these models combines two values: classification rate (*clas_rat*) by using 1-NN classifier and percentage reduction of prototypes of S with regards to TR (*perc_red*):

$$Fitness(S) = \alpha \cdot clas\_rat + (1 - \alpha) \cdot perc\_red. \tag{2}$$

Finally, as a multi-objective approach, we can find an EPS algorithm in Ref. [38].

In our empirical study, the four models developed in Refs. [15,36,37] together with the IGA proposal [35] will be used to compare with the MA-EPS proposal. In order to prepare IGA to be applied only as PS method, we ignore feature selection functionality. We must point out that the GGA described in Ref. [15] is really an improved model of Kuncheva's and Ishibuchi's GGAs.

4. A MA for EPS

In this section, we introduce our proposal of MA for EPS. It is a steady-state MA (SSMA) that makes use of a LS or meme specifically developed for this purpose. In Section 4.1 we introduce the foundations of the SSMAs. In Section 4.2 we explain the details of the proposed algorithm. Finally, in Section 4.3 we clarify the application of the ad hoc meme in the algorithm.

4.1. Steady-state MAs

In SSGA usually one or two offspring are produced in each generation. Parents are selected to produce offspring and then

1. Select two parents from the population.
2. Create one/two offspring using crossover and mutation.
3. Evaluate the offspring with the fitness function.
4. Select one/two individuals in the population, which may be replaced by the offspring.
5. Decide if this/these individuals will be replaced.

Fig. 1. Pseudocode algorithm for the SSGA model.

a decision is made as to which individuals in the population to select for deletion in order to make room for the new offspring. The basic algorithm steps of SSGA are shown at Fig. 1.

In step 4, one can choose the *replacement strategy* (e.g., replacement of the worst, the oldest, or a randomly chosen individual), and step 5, the *replacement condition* (e.g., replacement if the new individual is better, or unconditional replacement). A widely used combination is to replace the worst individual only if the new individual is better. We will call this strategy the *standard replacement strategy*. In Ref. [39] it was suggested that the deletion of the worst individuals induced a high selective pressure, even when the parents were selected randomly.

Although SSGAs are less common than GGAs, different authors [40,41] recommend the use of SSGAs for the design of MAs because they allow the results of LS to be kept in the population from one generation to the next.

SSMAs integrate global and local searches more tightly than generational MAs. This interweaving of the global and local search phases allows the two to influence each other, e.g., the SSGA chooses good starting points, and LS provides an accurate representation of that region of the domain. In contrast, generational MAs proceed in alternating stages of global and local searches. First, the GGA produces a new population, then the meme procedure is performed. The specific state of meme is generally not kept from one generation to the next, though meme results do influence the selection of individuals.

4.2. SSMA model for PS problem

The main characteristics of our proposed MA are:

- *Population initialization*: The first step that the algorithm makes consists of the initialization of the population, which is carried out by generating a population of random chromosomes.
- *Representation*: Let us assume a data set denoted *TR* with n instances. The search space associated with the instance selection of *TR* is constituted by all the subsets of *TR*. Therefore, the chromosomes should represent subsets of *TR*. This is achieved by using a binary representation. A chromosome consists of n genes (one for each instance in *TR*) with two possible states: 0 and 1. If the gene is 1, then its associated instance is included in the subset of *TR* represented by the chromosome. If it is 0, then this does not occur.
- *Fitness function*: Let S be a subset of instances of *TR* to evaluate and be coded by a chromosome. We define a fitness function considering the number of instances correctly classified using the 1-NN classifier and the percentage of reduction achieved with regard to the original size of the training data. The evaluation of S is carried out by considering all the training set *TR*. For each object y in S, the NN is searched for among those in the set $S\backslash\{y\}$

$$Fitness(S) = \alpha \cdot clas\_rat + (1 - \alpha) \cdot perc\_red.$$

  The objective of the MA is to maximize the fitness function as defined. Note that the fitness function is the same as that used by EPS models previously proposed.
- *Parent selection mechanism*: In order to select two parents for applying the evolutionary operators, binary tournament selection is employed.
- *Genetic operators*: We use a crossover operator that randomly replaces half of first parent's bits with second parent's bits and vice versa. The mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed to its other state.
- *Replacement strategy*: This will use a standard replacement strategy, which was defined above.
- *Mechanism of LS application*: It is necessary to control the operation of the LS over the total visited solutions. This is because the additional function evaluations required for total search can be very expensive and the MA in question could become a multi-restart LS and not take advantage of the qualities of the EAs. In order to do this, we have included in the algorithm the *Adaptive P_{LS} Mechanism*, which is an adaptive fitness-based method that is very simple but it offers good results in Ref. [41]. Indeed, this scheme assigns a LS probability value to each chromosome generated by crossover and mutation, c_{new}:

$$P_{LS} = \begin{cases} 1 & \text{if } f(c_{new}) \text{ is better than } f(C_{worst}), \\ 0.0625 & \text{otherwise,} \end{cases} \quad (3)$$

  where f is the fitness function and C_{worst} is the current worst element in the population. As was observed by Hart [42], applying LS to as little of 5% of each population results in faster convergence to good solutions.
- *Termination condition*: The MA continues carrying out iterations until a specified number of evaluations is reached.

Fig. 2 shows the SSMA pseudocode. After the initialization of the population is done, each generation of the algorithm is composed by a selection of two parents (step 3), together with the application of the genetic operators: crossover to create two offspring (step 4) and mutation applied to each one (with the corresponding probability associated per bit at step 5). At this point, the two individuals generated are evaluated, followed by computation of the value of P_{LS} mechanism for each offspring. A LS optimization is performed here only if the mechanism decides so. The computation of the value of P_{LS} is done in step 8 and in the next step, the result of adaptive mechanism is determined with an uniform distribution $u(0, 1)$. After step

1. Initialize population.
2. While (*not termination-condition*) do
   3.Use Binary Tournament to select two parents
   4.Apply crossover operator to create offspring (Off_1, Off_2)
   5. Apply mutation to Off_1 and Off_2
   6. Evaluate Off_1 and Off_2
   7.Foreach Off_i
      8. InvokeAdaptive-P_{LS}-mechanism to obtain P_{LS_i} for Off_i
      9. If u(0,1) < P_{LS_i} then
         10. Perform meme optimization for Off_i
      11. End if
   12. End for
   13. Employ standard replacement for Off_1 and Off_2
14. End while
15. Return the best chromosome

Fig. 2. Pseudocode algorithm for the proposed MA.

number 12, the replacement strategy can be carried out. The algorithm returns the best chromosome of the population once the stop criterion is satisfied.

4.3. Ad hoc meme procedure

In this subsection, we explain the procedure of optimization via LS performed within the evolutionary cycle described in Fig. 2 (step 10) according to the following structure: firstly we present the evaluation mechanism with total and partial evaluations; secondly, we present the pseudocode describing the whole procedure; thirdly, we will explain the two strategies used associated to fitness improvement, improving accuracy stage and avoiding premature convergence stage with loss of the local objective; and finally, an illustrative example is shown.

4.3.1. Evaluation mechanisms

During the operation of the SSMA, a fixed number of evaluations must take place in order to determine the quality of the chromosomes. We can distinguish between *total evaluation* and *partial evaluation*.

- *Total evaluation* consists of a standard evaluation of performance of a chromosome in EPS, computing the NN of each instance belonging to the selected subset and counts the correctly classified instances. Total evaluations always take place outside the optimization procedure, that is, within the evolutionary cycle.
- *Partial evaluation* can be carried out on a neighbour solution of a current instance that has already been evaluated and differs only in one bit position changed from value 1 to 0. If a total evaluation counts as one evaluation in terms of taking account of number of evaluations for the stop condition, a partial evaluation counts as

$$PE = \frac{N_{nu}}{n}, \tag{4}$$

where N_{nu} is the number of neighbours updated when a determined instance is removed by meme procedure and $n = |TR|$ is the size of the original set of instances (also the size of the chromosome). Partial evaluations always take place inside the local optimization procedure.

The SSMA computes total evaluations; when we consider a partial evaluation we add to the counter evaluation variable the respective partial value PE (expression (4)). Therefore, a certain number of partial evaluations (depending on the PE values) will be considered as a total evaluation.

4.3.2. Description of the optimization procedure

The meme optimization procedure used (step 10 in Fig. 2) in this method is an iterative process that aims to improve individuals of a population by reducing the number of prototypes selected and by enhancing classification accuracy.

The pseudocode described in Fig. 3 corresponds to the procedure in question. It can be described as follows: To achieve the double objective (to improve the number of classified patterns while reducing the subset size) the procedure considers neighbourhood solutions with $m-1$ instances selected, where m is equal to the number of instances selected in a current chromosome (positions with value 1 in the chromosome). In other words, a neighbour is obtained by changing 1 to 0 in a gene. In this way, the number of samples represented in a chromosome after optimization has been carried out will always be less than or equal to the initial number of samples selected in the chromosome. The algorithm in Fig. 3 receives as inputs the chromosome to optimize and its list of associated neighbours, called U (lines 1–3). The R list will include the identifiers of the instances that have already been removed without having obtained a sufficient gain according to the threshold value. The U list contains the identifiers of the instances considered the NNs of each instance. U has room for n identifiers of instances. It links the instance identified by a number stored in the ith cell as the NN of the instance identified by the number i. In this way, the search of the NN of each instance is only needed when the instance is removed from the subset selected. Note that the U list could be upgraded in order to contain more of one neighbour per instance; the case explained here is the easiest.

A partial evaluation can take advantage of U and of the divisible nature of the PS problem when instances are removed.

Next, we explain the concepts necessary to understand the procedure. Two functions are very useful in this procedure:

- $Nearest\_Neighbour_S(\cdot)$: This function returns the NN of an instance considering only those instances selected by the chromosome S. It requires as input an integer that will be the identifier of an instance within the training set TR, which is composed of n instances. The output will also be an integer that identifies the nearest instance of the input belonging to the S subset (or selected instances in the chromosome).

1. Let $S = \{s_1, s_2, ..., s_n\}$ be a chromosome
for optimizing
2. Let $R = \emptyset$
3. Let $U = \{u_1, u_2, ..., u_n\}$ be an associated neighbours'
list where $u_i = Nearest\_Neighbour_S(i)/i = 1, 2, ..., n$
4. While $(\exists s_i \in S/s_i = 1$ and $i \notin R)$ do
5. Choose j randomly from S where
$s_j = 1$ and $j \notin R$
6. $gain = 0$
7. $s_j = 0$
8. Copy U to U'
9. For each $u_i \in U/u_i = j$ do
10. $u_i = Nearest\_Neighbour_S(i)$
11. If $(cls(i) = cls(u'_i)$ and $cls(i) \neq cls(u_i))$ then
12. $gain = gain - 1$
13. End if
14. If $(cls(i) \neq cls(u'_i)$ and $cls(i) = cls(u_i))$ then
15. $gain = gain + 1$
16. End if
17. End for
18. If $(gain >= threshold)$ then
19. $fitness_S = fitness_S + fitness_{gain}$
20. $R = \emptyset$
21. Else
22. Recover U from U'
23. $s_j = 1$
24. $R = R \bigcup j$
25. End if
26. End while
27. Return S

Fig. 3. Pseudocode of meme optimization.

- $cls(\cdot)$: This function returns the class that the instance belongs to.

From step 4 to 14 in Fig. 3, the procedure tries to find instances for removing from the subset selected (which are randomly chosen). Once a choice of removal is done, the procedure attempts to compute the gain of this choice, making a backup copy of the U structure at step 8 (the choice may not be good). In order to do this, it locates the instances which have the choice of removal as NN and updates the new NN (step 10) by using the remainder of instances belonging to the subset. Meanwhile, it simultaneously computes the gain, checking if the new neighbours produce a success or a fail in the classification of the instance (steps from 11 to 16). This gain, computed in a relative and efficient way, allows the algorithm to decide if the choice of removal is maintained or discarded. The gain only refers to classification accuracy, since the reduction profit is implicit by the fact that the choice is always a removal.

Looking at Fig. 3, the variable called *gain* maintains an account of LS contributions carried out when a move on the meme procedure is performed. It may be negative or positive, depending on whether or not the NN of an instance changes the class label of the previous NN. A negative contribution, which subtracts 1 from the local objective, is caused when the new neighbour misclassifies a correctly classified instance. A positive contribution, which adds 1 to the local objective, is caused when the new neighbour classifies correctly a badly classified instance. A null contribution is caused when an instance maintains the same state of classification, which remains misclassified or correctly classified. This process is carried out over all instances whose NN has changed by using the identifiers of NNs stored in structure U, appropriately updated after a move of meme procedure.

The acceptance of a choice of removal depends upon the gain in accuracy that the algorithm is looking for, which is defined according to the current LS stage.

4.3.3. LS stages

Two stages can be distinguished within the optimization procedure. Each stage has a different objective and its application depends on the progress of the actual search process: the first one is an exclusive improvement of fitness and the second stage is a strategy for dealing with the problem of premature convergence.

- *Improving accuracy stage*: This starts from the initial assignment (a recently generated offspring) and iteratively tries to improve the current assignment by local changes. If, in the neighbourhood of the current assignment, a better assignment is found, it replaces the current assignment and it continues from the new one. The selection of a neighbour is randomly made without repetition from among all the solutions that belong to the neighbourhood. In order to consider an assignment as better than the current one the classification accuracy must be greater than or equal to the previous one, but in this last case, the number of selected instances will be lower than previously, so the fitness function value is always increased.
- *Avoiding premature convergence stage*: When the search process has advanced, a tendency of the population to premature convergence toward a certain area of the search space takes place. A local optimization promotes this behaviour when it considers solutions with better classification accuracy. In order to prevent this, the meme optimization procedure proposed will accept worse solutions in the neighbourhood, in terms of accuracy of classification. Here, the fitness function value cannot be increased; it may be decreased or maintained.

The parameter *threshold* is used in order to determine the way the algorithm operates depending on the current stage. When *threshold* has a value greater or equal to 0, then the stage in progress is the *improving accuracy stage* because a new

Class	Instances		
A	{1, 2, 3, 4, 5, 6, 7}		
B	{8, 9, 10, 11, 12, 13}		
	Current Solution	→	Neighbour Solution
Representation	0110110100010	→	0100110100010
U structure	{3, 5, 8, 8, 3, 2, 6, 2, 8, 8, 3, 2, 3}	→	{**12**, 5, 8, 8, **2**, 2, 6, 2, 8, 8, **8**, 2, **8**}
Gain	{1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0}	→	{-1,·,·,·,0,·,·,·,·,·,+1,·,+1}
Correct classified patterns	7	→	7 − 1+1+1
Partial evaluation account: PE = $\frac{N_{nu}}{n} = \frac{4}{13}$			
Number of correct classified patterns: 8			

Fig. 4. Example of a move in meme procedure and a partial evaluation.

generated chromosome via meme optimization is accepted when its fitness contribution is not negative ($gain \geqslant 0$). Note that $gain = 0$ implies that the new chromosome is as good as the original considering accuracy objective, but it will have fewer instances (improving reduction objective). On the other hand, if *threshold* value is less than 0, then the stage in progress is the *avoiding premature convergence stage* because the new chromosome is accepted when its fitness contribution is not positive. This process is described in Fig. 3 in step 18.

Initially, the algorithm starts with a *threshold* value equal to 0. The parameter is affected by three possible conditions:

- After a certain number of generations, the classification accuracy of the best chromosome belonging to the population has not been improved: $threshold = threshold + 1$.
- After a certain number of generations, the reduction of the subset selected with respect to the original set of the best chromosome belonging to the population has not been improved: $threshold = threshold - 1$.
- After a certain number of generations, neither accuracy nor reduction objectives have been improved: *threshold* value does not change.

The exact number of generations was tested by an empirical way, and at the end we checked that a value of 10 generations worked appropriately.

Once the change is accepted due to the fact that the gain equals or exceeds the current threshold, in step 19 of Fig. 3, the new fitness value for the optimized solution is computed as

$$fitness_{gain} = \left(\frac{gain}{n} \cdot 100 + \frac{1}{n} \cdot 100\right) \Big/ 2.$$

That is, the gain obtained is changed in terms of percentage of classification accuracy and this value is added to the percentage of reduction profit, which is always $1/n$ given that a LS movement is always a removal of an instance.

4.3.4. Example of ad hoc meme

An example is illustrated at Fig. 4, where a chromosome of 13 instances is considered. Meme procedure removes the instance number 3. Once removed, the instance number 3 cannot appear in the *U* structure as NN of another instance. *U* must be updated at this moment obtaining the new NNs for the instances that had instance number 3 as NN. Then a relative objective with respect to the original chromosome fitness is calculated (the gain of instances 1, 5, 11 and 13). The result obtained is a new chromosome with a higher number of correctly classified patterns (8 instead of 7) that, as well, takes up only 4 of the 13 evaluations, a number which will count towards the total of evaluations carried out.

Table 1
Small data sets characteristics

Name	N. instances	N. features	N. classes
Bupa	345	7	2
Cleveland	297	13	5
Glass	294	9	7
Iris	150	4	3
Led7Digit	500	7	10
Lymphography	148	18	4
Monks	432	6	2
Pima	768	8	2
Wine	178	13	3
Wisconsin	683	9	2

5. Experimental study

This section presents the framework used in the experimental study carried out together with results. To scale appropriately the problem we have used three sizes of problems: small, medium and large. We intend to study the behaviour of the algorithms when the size of the problem increases. When considering large data sets, stratification process [16] is used obtaining strata of medium size. The small-size data sets are summarized in Table 1 and medium and large data sets can be seen in Table 2. The data sets have been taken from the UCI Machine Learning Database Repository [43]. *Ringnorm* data set comes from DELVE project.[1]

[1] URL: http://www.cs.toronto.edu/~delve/.

Table 2
Medium and large data sets characteristics

Name	N. instances	N. features.	N. classes
Nursery	12 960	8	5
Page-Blocks	5476	10	5
Pen-Based	10 992	16	10
Ringnorm	7400	20	2
Satimage	6435	36	7
Spambase	4597	57	2
Splice	3190	60	3
Thyroid	7200	21	3
Adult (large)	45 222	14	2

Table 3
Parameters used in PS algorithms

Algorithm	Parameters
CHC	$Pop = 50$, $Eval = 10\,000$, $\alpha = 0.5$
IGA	$Pop = 10$, $Eval = 10\,000$ $p_m = 0.01$, $\alpha = 0.5$
GGA	$P_m = 0.001$, $P_c = 0.6$, $Pop = 50$ $Eval = 10\,000$, $\alpha = 0.5$
PBIL	$LR = 0.1$, $Mut_{shift} = 0.05$, $p_m = 0.02$, $Pop = 50$ $Negative_{LR} = 0.075$, $Eval = 10\,000$
SSGA	$P_m = 0.001$, $P_c = 1$, $Pop = 50$ $Eval = 10\,000$, $\alpha = 0.5$
Ib3	$Acept\ level = 0.9$, $Drop\ level = 0.7$
Rmhc	$S = 90\%$, $Eval = 10\,000$
Rng	$Order = 1$
SSMA	$Pop = 30$, $Eval = 10\,000$, $p_m = 0.001$, $p_c = 1$ $\alpha = 0.5$

We will distinguish two models of partitions used in this work:

- *Tenfold cross-validation classic* (*Tfcv classic*): where TR_i, $i = 1, \ldots, 10$ is a 90% of D and TS_i is its complementary 10% of D. It is obtained as the following equations indicate:

$$TR_i = \bigcup_{j \in J} D_j, \tag{5}$$

$$J = \{j / 1 \leqslant j \leqslant (i-1) \text{ and } (i+1) \leqslant j \leqslant 10\},$$

$$TS_i = D \backslash TR_i. \tag{6}$$

- *Tenfold cross-validation strat* (*Tfcv strat*): where $SPSS_i$ is generated using the DS_j instead of D_j (see Section 2.2).

$$SPSS_i = \bigcup_{j \in J} DS_j,$$

$$J = \{j / 1 \leqslant j \leqslant b \cdot (i-1) \text{ and } (i \cdot b) + 1 \leqslant j \leqslant t\}. \tag{7}$$

The data sets considered are partitioned using the *tfcv classic* (see expressions (5) and (6)) except for the adult data set, which is partitioned using the *tfcv strat* procedure with $t = 10$ and $b = 1$. (see expression (7)). Deterministic algorithms have been run once over these partitions, whereas probabilistic algorithms (including *SSMA*) run 3 trials over each partition and we show the average results over these trials.

Whether small, medium or large data sets are evaluated, the parameters used are the same, as specified in Table 3. They are specified by following the indications given for their respective authors. With respect to the standard EAs employed in the study, *GGA* and *SSGA*, the selection strategy is the binary tournament. The mutation operator is the same one used in our model of *SSMA* while *SSGA* uses standard replacement strategy. The crossover operator used by both algorithms defines two cut points and interchanges substrings of bits.

To compare results we propose the use of non-parametric tests, according to the recommendations made in Ref. [44]. They are safer than parametric tests since they do not assume normal distribution or homogeneity of variance. As such, these non-parametric tests can be applied to classification accuracies, error ratios or any other measure for evaluation of classifiers, even including model sizes and computation times. Empirical results suggest that they are also stronger than the parametric test. Demšar recommends a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers. We will use two tests with different purposes, the first of which is the Iman and Davenport test [45], a non-parametric test, derived from the Friedman test, and equivalent to the repeated-measures ANOVA. Under the null-hypothesis, which states that all the algorithms are equivalent, a rejection of this hypothesis implies the existence of differences of performance among all the algorithms studied. The second is the Wilcoxon signed-ranks test [46]. This is analogous to the paired t-test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences in the behaviour of two algorithms.

We will present four types of tables according to the subsequent structure:

(1) *Complete results table*: This shows the average of the results obtained by each algorithm in all data sets evaluated (small or medium group of data sets). These tables are grouped in columns by algorithms. For each one it shows the average reduction, accuracy in training and accuracy in test data with their respective standard deviations (SDs) (see for example Table 4). The two last rows compute the average and SD over the average results obtained on each data set, respectively.
(2) *Summary results table*: This shows the average of the results obtained by each algorithm in the data sets evaluated (small or medium group of data sets). The columns show the following information (see for example Table 9):
   - The first column shows the name of the algorithm.
   - The second and third columns contain the average execution time and the SD associated to each algorithm of the run of 10-fcv. They have been run in a HP Proliant DL360 G4p, Intel Xeon 3.0 GHz, 1 GB RAM.
   - The fourth and fifth columns show the average reduction percentage and associated SD from the initial training sets.

Table 4
Average results for EPS algorithms over small data sets

Data set	Measure	*CHC*			*GGA*			*IGA*			*PBIL*			*SSGA*			*SSMA*		
		Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.
Bupa	Mean	97.13	73.24	58.76	92.27	78.58	59.57	81.61	83.09	63.69	91.3	78.39	64.61	90.82	79.19	62.25	95.01	76.55	63.99
	SD	0.81	1.49	6.75	1.48	1.1	7.39	1.93	1.47	9.27	0.91	1.05	4.64	1.87	1.67	7.18	0.72	1.41	4.11
Cleveland	Mean	98.35	63.48	58.75	95.45	64.76	54.8	86.32	68.72	51.49	94.94	64.98	55.77	94.02	65.16	52.52	97.84	63.51	57.47
	SD	0.3	0.92	5.91	0.87	1	5.45	2.15	1.03	7.16	0.83	0.8	4.94	1.09	1.45	6.29	0.72	0.92	6.51
Glass	Mean	94.34	74.04	65.39	90.91	76.44	65.87	80.74	82.55	68.89	91.06	76.18	64.58	90.34	75.86	65.83	92.58	76.12	66.07
	SD	0.86	1.51	9.97	0.99	1.91	13.28	1.82	1.65	10.7	1.66	2.36	9.49	1.69	1.94	12.37	1.32	1.74	10.51
Iris	Mean	96.81	97.41	96.67	95.56	97.63	94	93.33	98.81	94	96.07	98.07	96	95.41	97.41	95.33	96.07	97.93	95.33
	SD	0.47	0.76	3.33	0.33	0.55	4.67	0.66	0.49	3.59	0.58	0.76	4.42	0.93	0.68	4.27	0.67	1.4	5.21
Led7Digit	Mean	96.58	68.71	64	95.64	68.64	63.8	95.16	65.89	67.6	93.91	68.67	66	95.71	66.44	64.8	96.71	54.11	75.4
	SD	0.18	2.88	7.32	0.18	3.27	4.85	0.33	2.21	4.18	0.78	3.09	2.53	0.37	3.18	5.15	0.45	8.78	4.29
Lymphography	Mean	96.55	54.36	39.49	91.96	57.21	35.26	86.27	60.59	40.57	94.75	54.88	41.71	92.33	55.57	43.62	94.67	55.78	42.88
	SD	0.68	1.42	9.99	1.5	1.52	9.83	2.2	2.35	11.1	1.31	1.92	13.85	2.32	3.11	9.94	1.52	2.51	12.12
Monks	Mean	99.05	96.86	97.27	93.98	94.62	92.3	84.83	98.15	85.75	92.34	94.57	89.17	92.31	94.88	93.39	97.66	97.22	96.58
	SD	0.23	0.42	2.65	0.73	0.99	3.71	1.61	0.81	6.98	1.13	1.31	7.15	1.08	1.27	3.72	1.27	1.31	3.26
Pima	Mean	98.78	80.14	75.53	95.11	81.57	70.73	86	86.81	70.84	91.9	82.03	72.27	94.23	83.02	73.32	97.38	82.15	73.21
	SD	0.28	0.87	3.11	0.75	0.7	4.87	0.71	0.78	2.68	0.49	0.57	2.96	0.64	0.97	4.53	0.68	0.67	5.5
Wine	Mean	96.94	98.69	94.93	95.69	98.69	93.82	93.76	99.88	94.97	96.32	98.63	94.41	94.69	98.69	97.19	96.44	98.69	93.82
	SD	0.51	0.81	4.62	0.71	0.9	4.61	1.01	0.25	6.78	0.59	0.73	5.56	1.29	0.81	3.75	0.68	0.34	6.31
Wisconsin	Mean	99.44	97.57	96.56	99.08	97.71	96	98.22	98.04	95.28	98.54	97.66	96.99	99.06	97.76	96.57	99.38	97.65	96.57
	SD	0.08	0.28	2.42	0.2	0.18	2.46	0.42	0.23	3.33	0.28	0.23	1.86	0.26	0.25	3.08	0.09	0.19	2.32
GLOBAL	Mean	**97.40**	80.45	74.74	94.57	81.59	72.62	88.62	**84.25**	73.31	94.11	81.41	74.15	93.89	81.40	74.48	96.38	79.97	**76.13**
	SD	1.53	16.27	20.63	2.37	15.13	20.69	6.03	14.86	18.95	2.47	15.58	19.08	2.58	15.63	19.86	1.92	17.76	18.94

- The last four columns include the accuracy mean and SD when 1-NN is applied using S; the first two show the training accuracy when classifying TR with S, while the last two show the test accuracy when classifying TS with S.

In the fourth, sixth and eighth columns, the best result per column is shown in bold.

(3) *Wilcoxon's test tables for $n \times n$ comparisons*: Given that the evaluation of only the mean classification accuracy over all the data sets would hide important information and that each data set represents a different classification problem with different degrees of difficulty, we have included a second type of table that shows a statistical comparison of methods over multiple data sets. As we have mentioned, Demšar [44] recommends a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers, one of which is the Wilcoxon signed-ranks test [46]. This is analogous to the paired t-test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences in the behaviour of two algorithms. In our study, we always consider a level of significance of $\alpha < 0.05$.

Let d_i be the difference between the performance scores of the two classifiers on ith out of N data sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the data sets in which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 10$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i), \tag{8}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i). \tag{9}$$

Let T be the smaller of the sums, $T = \min(R^+, R^-)$.

The Wilcoxon signed-ranks test is more sensitive than the t-test. It assumes commensurability of differences, but only qualitatively: greater differences count still more, which is probably to be desired, but absolute magnitudes are ignored. From the statistical point of view, the test is safer since it does not assume normal distributions. Also outliers (exceptionally good/bad performances on a few data sets) have less effect on Wilcoxon than on the t-test. Wilcoxon's test assumes continuous differences d_i, which therefore should not be rounded to, say, one or two decimals since this would decrease the power of the test due to a high number of ties.

A Wilcoxon table, in this paper, is divided into two parts: In the first part, we carry out a Wilcoxon test using as the performance measure the accuracy classification in the test set, while in the second part, a balance of reduction and classification accuracy is used as the performance measure. This balance corresponds to $0.5 \cdot clas\_rat + 0.5 \cdot perc\_red$. The structure of the tables presents $N_{alg} \times (N_{alg} + 2)$ cells to compare all the algorithms in them. In each of the $N_{alg} \times N_{alg}$ cells three symbols can appear: $+$, $-$ or $=$. They show that the algorithm situated in that row is better ($+$), worse ($-$) or equal ($=$) in behaviour (accuracy or balance accuracy–reduction) to the algorithm that appears in the column. The penultimate column represents the number of algorithms with worse or equal behaviour to the one that appears in the row (without considering the algorithm itself) and the last column represents the number of algorithms with worse behaviour than the one that appears in the row.

(4) *Wilcoxon's test tables to contrast results for a control algorithm*: These tables are made up of three columns: In the first one, the name of the algorithm is indicated, in the second and third columns, symbols $+$, $-$ or $=$ show the existence or absence of significant differences between the control algorithm and the algorithm specified in the row, according to accuracy performance and accuracy–reduction balance performance, respectively. Note that in this type of tables, a symbol $+$ indicates that the control method is better than the algorithm in the row. In the previous type of tables, the meaning of the symbols is just the opposite; that is, for example, a symbol $+$ indicates that the algorithm in the row is better than the algorithm located at the column.

(5) *Computation of Iman and Davenport statistic tables*: We follow the indications given in Ref. [44] incarrying out Iman and Davenport test. It ranks the algorithms for each data set separately, starting by assigning the rank of 1 to the best performing algorithm. Let r_i^j be the rank of the jth of k algorithms on the ith of N data sets. The Iman and Davenport statistic is defined as

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2}, \tag{10}$$

in which χ_F^2 is the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j \left(\frac{1}{N}\sum_i r_i^j\right)^2 - \frac{k(k+1)^2}{4}\right]. \tag{11}$$

F_F is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

These tables are made up of four columns. In the first and second, information about the conditions of the experiment is indicated: the type of result that is measured and the scale of the data sets, respectively. In the third, the computed value of F_F is shown and, in the last column, the corresponding critical value of the F-distribution table with $\alpha = 0.05$ is indicated. If the value F_F is higher than its associated critical value, then the null-hypothesis is rejected (this implies a significant difference of results among all methods considered).

We divide the experimental study into two groups: Comparison among EPS algorithms and comparison of our proposal with other non-EPS methods. The large-size data sets will be separately studied with the objective of ascertaining if the behaviour of the new proposal when the stratification process

Table 5
Average results for EPS algorithms over medium data sets

Data set	Measure	*CHC*			*GGA*			*IGA*			*PBIL*			*SSGA*			*SSMA*		
		Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.	Red.	Tra.	Tst.
Nursery	Mean	93.6	74.98	74.07	88.62	82.78	80.55	69.54	87.78	82.88	79.99	86.43	83.46	90.16	84.96	80.56	94.07	88.07	85.28
	SD	0.82	0.88	1.98	0.24	0.32	0.94	0.19	0.17	1.1	0.23	0.21	0.92	0.17	0.27	0.79	0.89	0.24	0.81
Page-Blocks	Mean	99.66	94.84	94.32	93.15	95.7	94.99	73.34	96.5	95.39	85.75	96.04	95.41	96.77	96.11	95.12	99.18	96.03	95.05
	SD	0.04	0.24	0.79	0.21	0.17	0.83	0.28	0.13	0.8	0.29	0.14	0.51	0.16	0.19	0.75	0.09	0.12	1.03
Pen-Based	Mean	98.91	95.87	95.87	91.65	98.7	98.27	73.95	99.32	98.94	83.32	99.08	98.74	95.27	98.95	98.37	98.56	98.74	98.04
	SD	0.12	0.29	0.62	0.2	0.11	0.32	0.55	0.08	0.28	0.15	0.1	0.35	0.07	0.12	0.29	0.28	0.16	0.6
Ring	Mean	99.59	88.08	87.7	90.14	80.11	77.45	67.11	82.66	77.07	81.08	81.64	76.65	93.31	87.71	84.36	98.88	95.53	92.66
	SD	0.05	1.07	1.51	0.28	0.44	1.66	0.17	0.29	0.82	0.2	0.39	1.17	0.22	0.44	0.94	0.13	0.38	0.89
Satimage	Mean	99.51	87.63	86.59	91.58	90.64	88.7	71.14	92.14	89.25	83.88	91.34	88.97	94.84	91.77	88.95	98.36	91.98	89.39
	SD	0.1	0.32	0.9	0.24	0.21	1.01	0.52	0.25	1.3	0.32	0.27	1.16	0.21	0.23	1.31	0.23	0.35	0.89
Spambase	Mean	99.53	87.63	86.84	91.71	89.39	87.3	68.7	91.08	87.82	84.76	90.53	87.77	94.79	90.94	87.64	98.12	91.23	88.3
	SD	0.05	0.9	1.64	0.29	0.52	1.68	0.46	0.36	1.46	0.43	0.44	0.9	0.24	0.42	2.18	0.22	0.55	1.49
Splice	Mean	99.14	75.41	71.57	91.13	78.96	72.51	65.28	80.95	73.73	83.9	80.92	74.11	93.07	83.16	72.92	97.04	83.6	74.7
	SD	0.16	0.76	1.66	0.53	0.83	2.7	0.62	0.37	1.52	0.5	0.44	1.57	0.49	0.5	1.84	0.48	0.38	2.67
Thyroid	Mean	99.9	92.72	92.53	92.45	93.64	93.32	71.71	93.16	91.94	84.38	93.51	93.1	96.65	93.78	93.58	99.83	94.32	93.82
	SD	0.03	2.51	2.45	0.18	0.3	0.5	0.27	0.18	0.71	0.29	0.16	0.81	0.16	0.2	0.5	0.07	0.14	0.54
GLOBAL	Mean	**98.73**	87.15	86.19	91.30	88.74	86.64	70.10	90.45	87.13	83.38	89.94	87.28	94.36	90.92	87.69	98.01	**92.44**	**89.66**
	SD	2.10	8.05	8.97	1.40	7.38	9.07	3.01	6.37	8.74	1.92	6.53	8.75	2.16	5.43	8.33	1.79	4.85	7.28

Table 6
Iman and Davenport statistic for EPS algorithms

Performance	Scale	F_F	Critical value
Accuracy	Small	2.568	2.422
Accuracy	Medium	5.645	2.485
Accuracy–reduction	Small	14.936	2.422
Accuracy–reduction	Medium	179.667	2.485

is employed follows the tendency shown for EAs [16]. A final subsection will be included to illustrate a time complexity analysis considering EPS algorithms over all the medium size data sets considered and to study how the execution time of *SSMA* is given out among evolutionary and optimization stages.

5.1. Part I: comparing SSMA with EAs

In this section, we carry out a comparison that includes all EPS algorithms described in this paper.

Tables 4 and 5 show the average results for EPS algorithms run over small and medium data sets, respectively.

Iman and Davenport test's result is presented in Table 6. Tables 7 and 8 present the statistical differences by using Wilcoxon's test among EPS algorithms, considering accuracy performance and accuracy–reduction balance performance, respectively.

The following conclusions from examination of Tables 4–8 can be pointed out:

- In Table 4, *SSMA* achieves the best test accuracy rate. EPS algorithms are prone to present over-fitting obtaining a good accuracy in training data but not in test data. The *SSMA* proposal does not stress this behaviour in a noticeable way.
- When the problem scales up, in Table 5, *SSMA* presents the best reduction and accuracy in training and test data rates.
- The Iman–Davenport statistic (presented in Table 6) indicates the existence of significant differences of results among all EPS approaches studied.
- Considering only the performance in classification over test data in Table 7, all algorithms are very competitive. Statistically, *SSMA* always obtains subsets of prototypes with equal performance to the remaining of the EPS methods, improving *GGA* with the use of small databases and *GGA* and *CHC* when the problem scales-up to a medium size.
- When the reduction objective is included in the measure of quality, Table 8, our proposal obtains the best result. Only *CHC* presents the same behaviour in small data sets. When the problem scales up, *SSMA* again outperforms *CHC*.

Finally, we provide a map of convergence of *SSMA*, in Fig. 5, in order to illustrate the optimization process carried out on the satimage data set. In it, the two goals, reduction and train accuracy, are shown. We can see that the two goals are opposite, but the trend of the two lines of convergence usually rises.

Table 7
Wilcoxon table for EPS algorithms considering accuracy performance

	6	5	4	3	2	1	⩾	>
Small data sets								
CHC (1)	=	=	=	=	+		5	1
GGA (2)	−	−	=	=		−	2	0
IGA (3)	=	=	=		=	=	5	0
PBIL (4)	=	=		=	=	=	5	0
SSGA (5)	=		=	=	+	=	5	1
SSMA (6)		=	=	=	+	=	5	1
Medium data sets								
CHC (1)	−	=	=	=	=		4	0
GGA (2)	−	−	=	=		=	3	0
IGA (3)	=	=	=		=	=	5	0
PBIL (4)	=	=		=	=	=	5	0
SSGA (5)	=		=	=	+	=	5	1
SSMA (6)		=	=	=	+	+	5	2

Table 8
Wilcoxon table for EPS algorithms considering reduction–accuracy balance performance

	6	5	4	3	2	1	⩾	>
Small data sets								
CHC (1)	=	+	+	+	+		5	4
GGA (2)	−	=	=	+		−	3	1
IGA (3)	−	−	−		−	−	0	0
PBIL (4)	−	=		+	=	−	3	1
SSGA (5)	−		=	+	=	−	3	1
SSMA (6)		+	+	+	+	=	5	4
Medium data sets								
CHC (1)	−	=	+	+	+		4	3
GGA (2)	−	−	+	+		−	2	2
IGA (3)	−	−	−		−	−	0	0
PBIL (4)	−	−		+	−	−	1	1
SSGA (5)	−		+	+	+	=	4	3
SSMA (6)		+	+	+	+	+	5	5

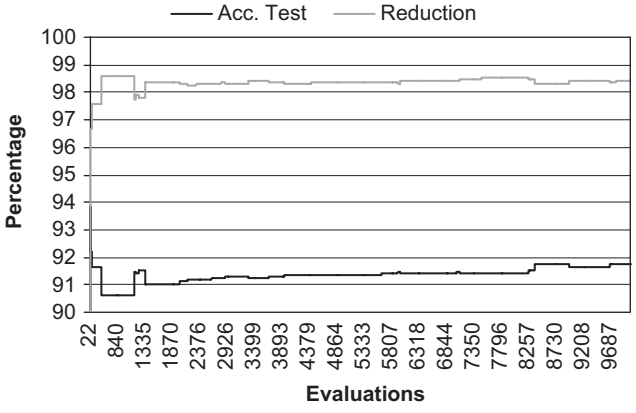

Fig. 5. Map of convergence of the *SSMA* algorithm on satimage data set.

5.2. Part II: comparing SSMA with non-evolutionary algorithms

Tables 9 and 10 show the average results for non-EPS algorithms together with the *SSMA* proposal run over small and medium data sets, respectively. This table summarizes the mean and SD of the results carried out over all data sets.

The Iman–Davenport test result is presented in Table 11. Table 12 summarizes the existence or absence of statistical difference by using Wilcoxon's test between the *SSMA* proposal and remaining non-EPS methods (as we mentioned before, every cell presents the comparison of *SSMA* with the respective algorithm in the row. The symbol + indicates better behaviour of *SSMA* than that of the corresponding one in the row).

We can show the following:

- In Table 9, the *SSMA* proposal clearly outperforms the other algorithms taking account of accuracy performance in test data.
- When the problem scales up, *SSMA* presents the best accuracy rate and the second best rate of reduction (results shown in Table 10).
- Non-EPS algorithms usually present a lower run-time than that of the EPS algorithm. However, those non-EPS algorithms that present best behaviour in both objectives are also algorithms with high running times.

Table 9
Average results for non-EPS algorithms over small data sets

Algorithm	Time (s)	SD time (s)	% Red.	SD red.	% Ac. trn.	SD ac. trn	% Ac. test	SD ac. test
1-*NN*	0.2	0.00	–	–	72.89	18.63	72.22	19.15
Allknn	0.18	0.03	37.00	23.97	77.51	16.29	72.60	18.40
Cpruner	0.26	0.02	92.59	5.09	66.38	24.16	65.31	23.22
Drop3	0.71	0.02	83.43	8.85	73.43	15.17	67.69	19.06
Enn	0.13	0.01	25.50	19.56	77.47	15.16	73.21	18.38
Explore	0.83	0.03	**97.66**	1.16	77.59	16.78	74.42	19.60
Ib3	0.17	0.01	68.34	19.85	64.16	22.95	69.96	20.18
Pop	0.05	0.01	14.56	15.17	70.49	20.19	72.10	19.60
Rmhc	11.82	0.05	90.18	0.17	**83.68**	14.25	73.93	19.44
Rng	1.35	0.03	25.07	20.45	78.24	15.31	73.85	18.32
Rnn	3.18	0.05	92.40	4.56	74.41	17.57	73.11	17.70
SSMA	6.38	0.16	96.37	1.92	79.97	17.76	**76.13**	18.94

Table 10
Average results for non-EPS algorithms over medium data sets

Algorithm	Time (s)	SD time (s)	% Red.	SD red.	% Ac. trn.	SD ac. trn	% Ac. test	SD ac. test
1-*NN*	72.93	0.15	–	–	89.11	7.75	87.79	8.94
Allknn	34.39	0.31	18.68	13.49	88.46	11.33	85.21	12.94
Cpruner	23.83	0.07	89.08	3.56	81.31	15.51	80.52	16.09
Drop3	99.48	0.82	88.17	8.35	81.99	9.88	78.84	13.35
Enn	13.48	0.03	12.22	10.21	87.85	10.66	85.98	12.14
Explore	1525.52	92.27	**98.74**	1.03	91.44	4.96	88.99	7.21
Ib3	2.70	0.05	77.42	17.83	84.91	10.80	86.36	9.49
Pop	1.20	0.04	23.17	32.94	86.09	13.55	84.97	13.78
Rmhc	6572.53	172.08	90.00	0.01	**94.30**	3.57	89.53	7.61
Rng	3149.19	26.62	7.20	4.90	89.92	7.74	87.99	9.14
Rnn	26 428.38	271.56	93.65	4.25	88.85	7.45	86.04	9.57
SSMA	3775.75	184.13	98.01	1.79	92.44	4.85	**89.66**	7.28

- As we can see in Table 12, Wilcoxon's test considers the existence of competitive algorithms in terms of classification accuracy. None of the non-EPS algorithms outperforms our proposal in both considerations of evaluation. The unique algorithm that equals the result of *SSMA* when the reduction and accuracy are combined is *Explore*. Note that this last algorithm obtains a greater reduction rate than *SSMA*, but our approach improves upon it when we consider the classification performance. This fact is interesting given that, although a good trade-off between reduction–accuracy is required, accuracy in terms of classification is more difficult to improve upon than the reduction, so the solutions contributed by *SSMA* may be considered of higher quality.
- There are algorithms, for example *Allknn*, *Rng* or *Pop*, that have a similar performance in classification accuracy but that do not achieve a high reduction rate. This could be critical when large data sets need to be processed. A small reduction rate implies a small decrease in classification resources (storage and time); therefore, an application of these algorithms on large data sets lacks interest.

In relation to the superiority of the proposed algorithm, we are able show difference in the results obtained by *SSMA* and the ones obtained by *CHC* and *Explore* by means of graphical representations of the difference in reduction and test accuracy between *SSMA* and the two mentioned algorithms over all data sets.

Table 11
Iman and Davenport statistic for non-EPS algorithms

Performance	Scale	F_F	Critical value
Accuracy	Small	6.009	1.938
Accuracy	Medium	5.817	1.969
Accuracy–reduction	Small	56.303	1.938
Accuracy–reduction	Medium	68.583	1.969

Fig. 6 displays these representations, where the positive bars indicate superiority of *SSMA*. The bias of *SSMA* is to obtain less reduction than its two strong competitors, but it achieves a better accuracy than them when the problem scales up, more notably when compared with *CHC*. Note that *SSMA* outperforms them in accuracy on 12 data sets to the two main competitors.

5.3. Part III: a large-size case study

We have seen before the promising results offered by *SSMA* in small and medium sized databases. However, a size limit of data sets exists which makes the execution of an EPS

algorithm over them impossible. This limit depends on the algorithms employed, the properties of data treated and easiness of reduction of instances in the data set. We could argue that, surely, it may not be possible to handle a size of a data set above 20,000 instances with EPS due to a time complexity of the evaluations of $O(n^2 \cdot m)$. *SSMA* proposal may get closer this limit.

Table 12
Wilcoxon table for comparing SSMA with Non-EPS algorithms

	Accuracy	$Acc. \cdot 0.5 + red. \cdot 0.5$
Small data sets		
Allknn	=	+
Cpruner	+	+
Drop3	+	+
Enn	=	+
Explore	=	=
Ib3	+	+
Pop	=	+
Rmhc	=	+
Rng	=	+
Rnn	+	+
Medium data sets		
Allknn	=	+
Cpruner	+	+
Drop3	+	+
Enn	=	+
Explore	+	=
Ib3	+	+
Pop	=	+
Rmhc	=	+
Rng	=	+
Rnn	+	+

In this case, authors in Ref. [16] recommend the use of *Stratification*, which could be combined with a cross-validation procedure as can be seen in expression (7). By using the *Tfcv-Strat*, we have run the algorithms considered in this study over the data set *adult* with $t = 10$ and $b = 1$. We chose these parameters to obtain subsets whose size is not too large as well as to show the effectiveness of the combination of an EPS and the stratification procedure.

Fig. 7 shows a representation of an opposition between the two objectives: reduction and test accuracy. Each algorithm located inside the graphic gets its position from the average values of each measure evaluated (exact position corresponding to the beginning of the name of the algorithm). Across the graphic, there is a line that represents the threshold of test accuracy achieved by the 1-NN algorithm without preprocessing.

As we can see, all EPS methods are above the 1-NN horizontal line. The graphic clearly emphasizes three methods as best with their position in the graphic at top-right. In addition to this, we can remark that the *SSMA* algorithm achieves the highest accuracy rate, and it is only surpasses in the reduction objective by the *CHC* model.

Fig. 8 presents the opposition reduction–accuracy for non-EPS algorithms. *Ib3* and *Pop* algorithms have been removed

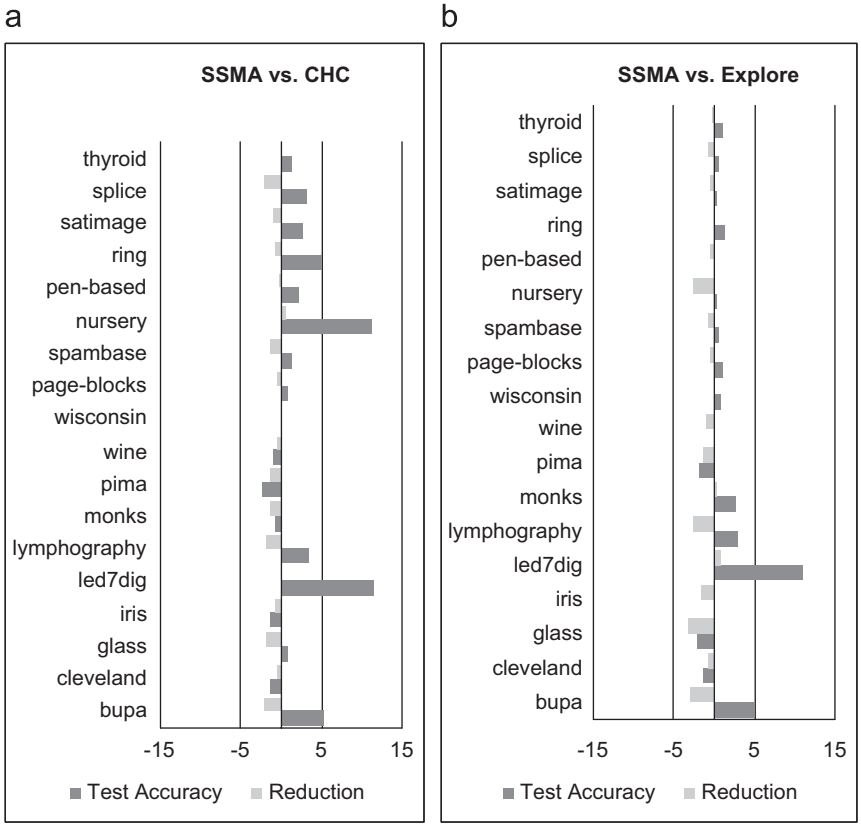

Fig. 6. Difference of results between *SSMA* and *CHC* and *SSMA* and *Explore*. (a) *SSMA* vs. *CHC*. (b) *SSMA* vs. *Explore*.

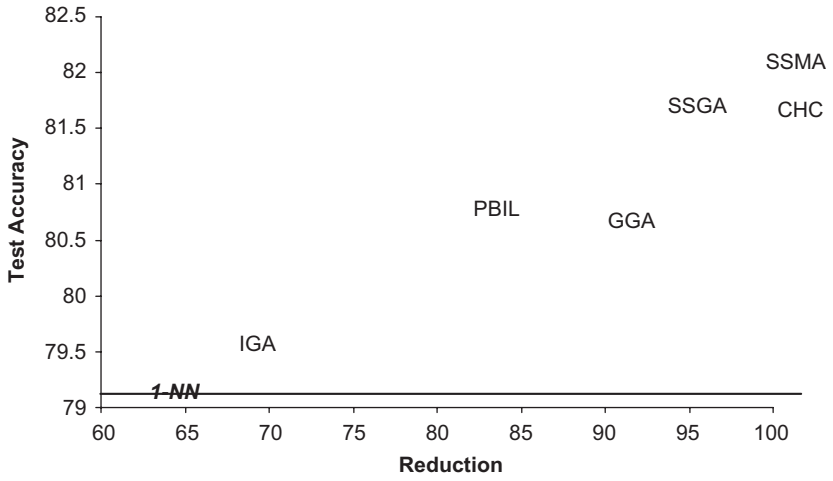

Fig. 7. Accuracy in test vs. reduction in adult data set for EPS algorithms.

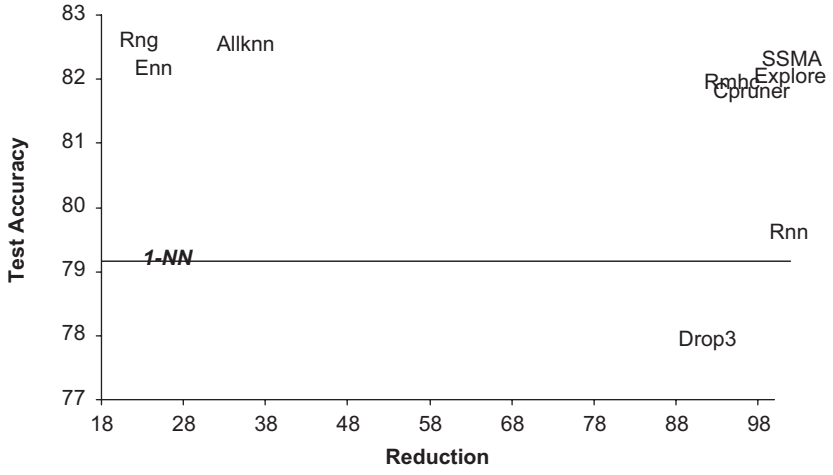

Fig. 8. Accuracy in test vs. reduction in adult data set for non-EPS algorithms.

from the graphic because of their poor accuracy and reduction, respectively.

The results offered by *SSMA* situate the algorithm at the top of the remaining methods that are nearest in the *x*-axis. This fact points to its superiority over the nearest methods. However, others methods further from it in *x*-axis are above *SSMA* in *y*-axis. The case studied practically shows the same behaviour as in previous cases, so it may be specified a generalization capacity of the results independently of the size of data.

5.4. Part IV: time complexity analysis for EPS

A way of estimating the efficiency of EPS algorithms could be by an empirical study. This implies a study of the execution time of each method by using different sizes of training data. We have showed the results obtained by using a graphical representation in Fig. 9 for medium size data sets.

As it can be seen in Fig. 9, the most efficient EPS algorithms are *SSMA* and *CHC*. Time complexity of an algorithm basically depends on two factors:

- *Reduction capacity*: When an EPS algorithm can quickly reduce the subset of instances selected in the chromosome, an evaluation will have to compute NNs over less data in order to calculate the fitness value. *CHC* and *SSMA* are inclined towards a tendency of removing instances and then improving classification accuracy, thus obtaining a good reduction rate. Remaining EPS algorithms try to improve both objectives simultaneously.
- *Operational cost*: In this case, any algorithm has an operational cost of time in all its procedures. For example, a *GGA* algorithm must sort and apply crossover and mutation operators to a part of a population, whereas *PBIL* must generate new populations from a single probability vector that receives all possible modifications. This explains why *GGA* takes more time than *PBIL* even if the former has a higher reduction rate than the latter. *SSMA* takes advantage of the partial evaluation mechanism, which is an efficient procedure to evaluate chromosomes.

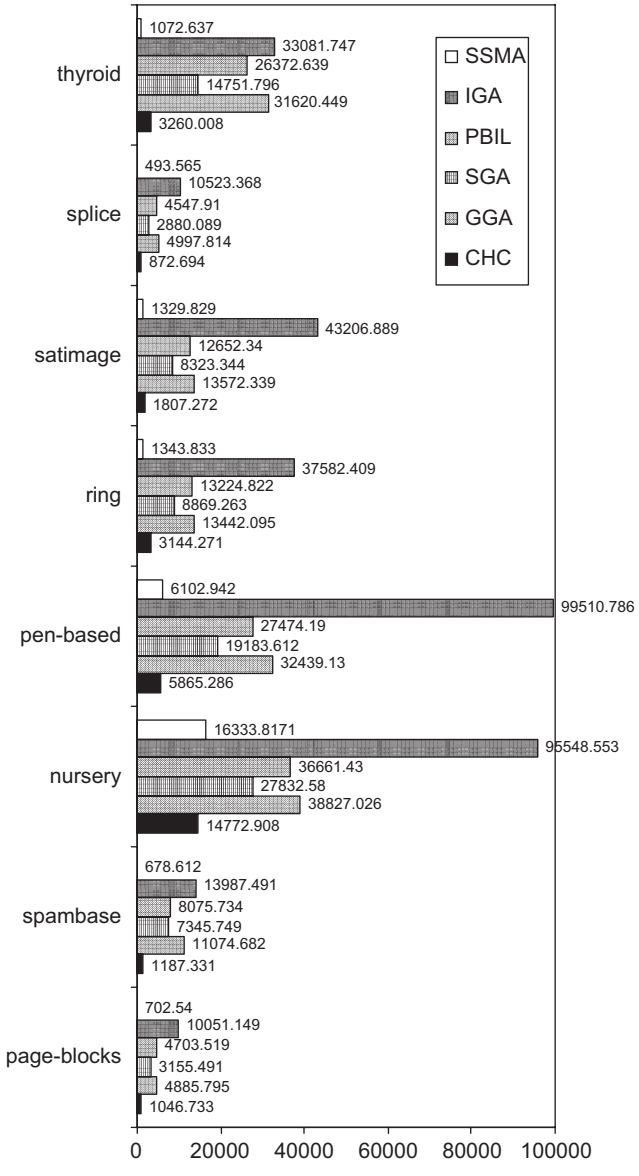

Fig. 9. Run-time in seconds for EPS over medium size data sets.

6. Conclusion

This paper presents a memetic algorithm for evolutionary prototype selection and its application over different sizes of data sets, paying special attention to the scaling up problem.

An experimental study was carried out to establish a comparison between this proposal and previous evolutionary and non-evolutionary approaches studied in the literature. The main conclusions reached are as follows:

- Our proposal of *SSMA* presents a good reduction rate and computational time with respect to other EPS schemes.
- *SSMA* outperforms the classical PS algorithms, irrespective of the scale of data set. Those algorithms that could be competitive with it in classification accuracy are not so when the reduction rate is considered.
- When the PS problem scales up, the tendency on the part of classical EPS is to inappropriately converge to two objectives (accuracy and reduction) at the same time.
- Considering all types of data sets, *SSMA* outperforms or equalizes all methods when both objectives have the same importance. Furthermore, it usually outperforms in test accuracy other methods that obtain good rates of reduction.

Finally, we would like to point out in conclusion that our SSMA proposal allows EPS to be competitive with other models for PS when the size of the databases increases, tackling the scaling up problem with excellent results.

An open problem, beyond of the scope of this paper, consists of hybridizing feature selection and weighting with the *SSMA* proposal in order to obtain a better behaviour in low complexity classifiers (like 1-NN) [47]. Future research will be focus on this together with tackling the prototype generation process with real-coded evolutionary algorithms.

References

[1] A.N. Papadopoulos, Y. Manolopoulos, Nearest Neighbor Search: A Database Perspective, Springer-Verlag Telos, 2004.
[2] G. Shakhnarovich, T. Darrel, P. Indyk (Eds.), Nearest-Neighbor Methods in Learning and Vision: Theory and Practice, MIT Press, Cambridge, MA, 2006.
[3] E. Pekalska, R.P.W. Duin, P. Paclík, Prototype selection for dissimilarity-based classifiers, Pattern Recognition 39 (2) (2006) 189–208.
[4] S.-W. Kim, B.J. Oommen, On using prototype reduction schemes to optimize dissimilarity-based classification, Pattern Recognition 40 (11) (2007) 2946–2957.
[5] H. Liu, H. Motoda, On issues of instance selection, Data Mining Knowl. Discovery 6 (2) (2002) 115–130.
[6] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Mach. Learn. 38 (2000) 257–286.
[7] R. Paredes, E. Vidal, Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization, Pattern Recognition 39 (2) (2006) 180–188.
[8] E. Gómez-Ballester, L. Micó, J. Oncina, Some approaches to improve tree-based nearest neighbour search algorithms, Pattern Recognition 39 (2) (2006) 171–179.
[9] M. Grochowski, N. Jankowski, Comparison of instance selection algorithms II. Results and comments, in: Proceedings of the International Conference on Artificial Intelligence and Soft Computing (ICAISC'04), Lecture Notes in Computer Science, vol. 3070, Springer, Berlin, 2004, pp. 580–585.
[10] M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, E. Pekalska, R.P.W. Duin, Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces, Pattern Recognition 39 (10) (2006) 1827–1838.
[11] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer, New York, 2002.
[12] A. Ghosh, L.C. Jain (Eds.), Evolutionary Computation in Data Mining, Springer, Berlin, 2005.
[13] N. García-Pedrajas, D. Ortiz-Boyer, A cooperative constructive method for neural networks for pattern recognition, Pattern Recognition 40 (1) (2007) 80–98.
[14] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, Berlin, 2003.
[15] J.R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Trans. Evol. Comput. 7 (6) (2003) 561–575.
[16] J.R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Lett. 26 (7) (2005) 953–963.

[17] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, Technical Report C3P 826, Pasadena, CA, 1989.
[18] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE Trans. Evol. Comput. 9 (5) (2005) 474–488.
[19] R. Dawkins, The Selfish Gene, Oxford University Press, Oxford, 1976.
[20] W.E. Hart, N. Krasnogor, J. Smith (Eds.), Recent Advances in Memetic Algorithms, Springer, Berlin, 2005.
[21] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[22] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. 6 (6) (1976) 448–452.
[23] J.C. Riquelme, J.S. Aguilar-Ruiz, M. Toro, Finding representative patterns with ordered projections, Pattern Recognition 36 (2003) 1009–1018.
[24] G.W. Gates, The reduced nearest neighbour rule, IEEE Trans. Inf. Theory 18 (3) (1972) 431–433.
[25] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[26] K.P. Zhao, S.G. Zhou, J.H. Guan, A.Y. Zhou, C-pruner: an improved instance prunning algorithm, in: Second International Conference on Machine Learning and Cybernetics (ICMLC'03), 2003, pp. 94–99.
[27] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining Knowl. Discovery 6 (2002) 153–172.
[28] R.M. Cameron-Jones, Instance selection by encoding length heuristic with random mutation hill climbing, in: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence, 1995, pp. 99–106.
[29] D.B. Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms, in: Proceedings of the Eleventh International Conference on Machine Learning (ML'94), Morgan Kaufmann, Los Altos, CA, 1994, pp. 293–301.
[30] J.S. Sánchez, F. Pla, F.J. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recognition Lett. 18 (1997) 507–513.
[31] L.I. Kuncheva, Editing for the k-nearest neighbors rule by a genetic algorithm, Pattern Recognition Lett. 16 (1995) 809–814.
[32] L.I. Kuncheva, J.C. Bezdek, Nearest prototype classification: clustering genetic, algorithms, or random search?, IEEE Trans. Syst. Man Cybern. 28 (1) (1998) 160–164.
[33] H. Ishibuchi, T. Nakashima, Evolution of reference sets in nearest neighbor classification, Second Asia-Pacific Conference on Simulated Evolution and Learning on Simulated Evolution and Learning (SEAL'98),f Lecture Notes in Computer Science, vol. 1585, Springer, Berlin, 1999, pp. 82–89.
[34] B. Sierra, E. Lazkano, I. Inza, M. Merino, P. Larrañaga, J. Quiroga, Prototype selection and feature subset selection by estimation of distribution algorithms. A case study in the survival of cirrhotic patients treated with tips, in: Proceedings of the Eighth Conference on AI in Medicine in Europe (AIME'01), Springer, London, UK, 2001, pp. 20–29.
[35] S.-Y. Ho, C.-C. Liu, S. Liu, Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm, Pattern Recognition Lett. 23 (13) (2002) 1495–1503.
[36] L.J. Eshelman, The CHC adaptative search algorithm: how to have safe search when engaging in nontraditional genetic recombination, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, 1991, pp. 265–283.
[37] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report CMU-CS-94-163, Pittsburgh, PA, 1994.
[38] J.-H. Chen, H.-M. Chen, S.-Y. Ho, Design of nearest neighbor classifiers: multi-objective approach, Int. J. Approx. Reasoning 40 (1–2) (2005) 3–22.
[39] D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, 1991, pp. 69–93.
[40] M.W.S. Land, Evolutionary algorithms with local search for combinatorial optimization, Ph.D. Thesis, University of California, San Diego, 1998.
[41] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, Real-coded memetic algorithms with crossover hill-climbing, Evol. Comput. 12 (3) (2004) 273–302.
[42] W.E. Hart, Adaptive global optimization with local search, Ph.D. Thesis, University of California, San Diego, 1994.
[43] A. Asuncion, D.J. Newman, UCI repository of machine learning databases, 2007. URL: ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[44] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[45] R.L. Iman, J.M. Davenport, Approximations of the critical region of the Friedman statistic, Commun. Stat. (1980) 571–595.
[46] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, Boca Raton, FL, 2003.
[47] D. Wettschereck, D.W. Aha, Weighting features, in: First International Conference of Case-Based Reasoning, Research and Development, 1995, pp. 347–358.

About the Author—SALVADOR GARCÍA received the M.Sc. degree in Computer Science form the University of Granada, Granada, Spain, in 2004. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. His research interests include data mining, data reduction, evolutionary algorithms, learning from imbalanced data and statistical analysis.

About the Author—JOSÉ RAMÓN CANO received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 1999 and 2004, respectively. He is currently an Associate Professor in the Department of Computer Science, University of Jaén, Jaén, Spain. His research interests include data mining, data reduction, interpretability-accuracy trade off, and evolutionary algorithms.

About the Author—FRANCISCO HERRERA received the M.Sc. degree in Mathematics in 1988 and the Ph.D. degree in Mathematics in 1991, both from the University of Granada, Spain.
He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has published more than 100 papers in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001). As edited activities, he has co-edited four international books and co-edited 16 special issues in international journals on different Soft Computing topics. He currently serves as area editor of the Journal Soft Computing (area of genetic algorithms and genetic fuzzy systems), and he serves as member of the editorial board of the journals: Fuzzy Sets and Systems, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, International Journal of Computational Intelligence Research, Mediterranean Journal of Artificial Intelligence, International Journal of Information Technology and Intelligent and Computing. He acts as associated editor of the journals: Mathware and Soft Computing, Advances in Fuzzy Systems, and Advances in Computational Sciences and Technology. His current research interests include computing with words and decision making, data mining, data preparation, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.