Pattern Recognition 44 (2011) 901–916

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification

Isaac Triguero [a,*], Salvador García [b], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain

[b] Department of Computer Science, University of Jaén, 23071 Jaén, Spain

ARTICLE INFO

Article history:
Received 2 June 2010
Received in revised form
6 September 2010
Accepted 24 October 2010

Keywords:
Differential evolution
Prototype generation
Prototype selection
Evolutionary algorithms
Classification

ABSTRACT

Nearest neighbor classification is one of the most used and well known methods in data mining. Its simplest version has several drawbacks, such as low efficiency, high storage requirements and sensitivity to noise. Data reduction techniques have been used to alleviate these shortcomings. Among them, prototype selection and generation techniques have been shown to be very effective. Positioning adjustment of prototypes is a successful trend within the prototype generation methodology.

Evolutionary algorithms are adaptive methods based on natural evolution that may be used for searching and optimization. Positioning adjustment of prototypes can be viewed as an optimization problem, thus it can be solved using evolutionary algorithms. This paper proposes a differential evolution based approach for optimizing the positioning of prototypes. Specifically, we provide a complete study of the performance of four recent advances in differential evolution. Furthermore, we show the good synergy obtained by the combination of a prototype selection stage with an optimization of the positioning of prototypes previous to nearest neighbor classification. The results are contrasted with non-parametrical statistical tests and show that our proposals outperform previously proposed methods.

1. Introduction

The nearest neighbor (NN) algorithm [1] and its derivatives have been shown to perform well for classification problems in many domains [2,3]. These algorithms are also known as instance-based learning [4] and belong to the lazy learning family of methods [5]. The extended version of NN to k neighbors is considered one of the most influential data mining algorithms [6] and it has attracted much attention and research efforts in recent years [7–10]. The NN classifier requires that all of the data instances are stored and unseen cases classified by finding the class labels of the closest instances to them. In order to determine how close two instances are, several distances or similarity measures have been proposed [11–13] and this issue is continually under review [14,15]. Despite its simplicity and high classification accuracy, it suffers from several drawbacks such as high computational cost, high storage requirement and sensitivity to noise.

Data reduction processes are very useful in data mining to improve and simplify the models extracted by the algorithms [16]. In NN, apart from feature selection [17,18], two main data reduction techniques have been used with promising results: prototype selection (PS) and prototype generation (PG) [19,20]. The former is limited to selecting a subset of instances from the original training set. Typically, three types of PS methods are known: condensation [21], edition [22] and hybrid methods [23]. Condensation methods try to remove examples which are redundant or irrelevant, which it means that these examples do not offer any capabilities in the classification task. However, edition methods focus on removing noisy examples, which are those examples that induce classification errors. Finally, hybrid methods combine both approaches.

In the specialized literature, a wide number of PS techniques have been proposed. Since the first approaches for data condensation and edition, CNN [21] and ENN [22], many other proposals of PS have become well-known in this field. For example, IBL methods [4], DROP family methods [19] and ICF [23]. Recent approaches to PS are introduced in [24–26].

Regarding PG methods, also known as prototype abstraction methods [27], they are not only able to select data, but can also modify them, allowing interpolations, movements of instances and artificial generation of new data. Well known methods for PG are PNN [28], learning quantization vector (LVQ) [29], Chen's algorithm [30], ICPL [27], HYB [31] and MixtGauss [32]. A good study of PS and PG can be found in [33].

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: triguero@decsai.ugr.es (I. Triguero), sglopez@ujaen.es (S. García), herrera@decsai.ugr.es (F. Herrera).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.10.020

Evolutionary algorithms (EAs) [34] have been successfully used in different data mining problems [35,36]. Given that PS and PG problems could be seen as combinatorial and optimization problems, EAs have been used to solve them with excellent results [37–40]. PS can be expressed as a binary space search problem and, as far as we know, the best evolutionary model proposed for PS is based on memetic algorithms [41] and is called SSMA [38]. PG is expressed as a continuous space search problem. EAs for PG are based on the positioning adjustment of prototypes, which is a suitable method to optimize the position of prototypes, however, it usually depends upon an initial subset of prototypes extracted from the training set. Several proposals are presented on this topic, such as ENPC [42] or PSCSA [43].

Particle swarm optimization (PSO) [44,45] and differential evolution (DE) [46,47] are two effective evolutionary optimization techniques for continuous spaces. In fact, PSO has been satisfactorily used for prototype adjustment [39,40]. The first attempts at using DE for PG can be found in [48]. In that contribution, we did a preliminary study on the use of DE, concluding that the classic DE scheme offers competitive results compared to other PG approaches to small size data sets.

The first contribution of this paper is the use of the DE algorithm [47] for prototype adjustment. The specialized literature on DE collects several advanced schemes: SADE [49], OBDE [50], DEGL [51], JADE [52] and SFLSDE [53]. We will study the mentioned proposals for PG, except OBDE. This last one, as the authors state, may not be used for problems where basic knowledge is available. It constantly seeks the opposite solution to the one evaluated in the search process and, in PG, this behavior does not make sense. The remaining proposals will be compared with evolutionary and non-evolutionary PG algorithms and we will analyze the behavior of each DE algorithm in this problem.

It is common to use classical PS methods in pre- or late stages of a PG algorithm as mechanisms for removing noisy or redundant prototypes. For example, some PG methods implement ENN or DROP algorithms as early filtering processes [27,54] and, in [31], a hybridization method based on LVQ3 post-processing of conventional prototype reduction approaches is proposed.

The second contribution of this paper follows a similar idea to that presented in [31], but it is extended so that PS methods can be hybridized with any positioning adjustment of prototype method. Specifically, we study the use of LVQ3, PSO and DE algorithms for the optimization positioning of prototypes after a PS process. We will see that LVQ3 does not produce optimal positioning in most cases, whereas PSO and DE result in excellent accuracy rates in comparison with isolated PS methods and PG methods. We especially emphasize the use of SSMA in combination with one of the two mentioned optimization approaches to also achieve high reduction rates in the final set of prototypes obtained.

As we have stated before, the use of DE algorithms for the PG problem motivates the global purpose of this paper, which can be divided into three objectives:

- To make an empirical study for analyzing the DE algorithms for the PG problem in terms of accuracy and reduction capabilities. Our goal is to identify the best DE methods and stress the relevant properties of each one when they tackle the PG problem.
- To understand how positioning adjustment techniques can improve the classification accuracy of PS and PG methods with the use of hybridization models.
- To check the behavior and scaling-up capabilities of DE approaches and hybrid approaches for PG when tackling large size data sets.

The experimental study will include a statistical analysis based on non-parametric tests and we will conduct experiments involving a total of 56 small and large size data sets.

In order to organize this paper, Section 2 describes the background of PS, PG and DE. Section 3 explains the DE algorithm proposed for tackling the position adjustment problem. Section 4 presents the framework of the hybridization proposed. Section 5 discusses the experimental framework and Section 6 presents the analysis of results. Finally, in Section 7 we summarize our conclusions.

2. Background

This section covers the background information necessary to define and describe our proposals. Section 2.1 presents the background on PS and PG. Next, Section 2.2 shows the main characteristics of DE and the most recent advances proposed in the literature are presented in Section 2.3.

2.1. PS and PG algorithms

This section presents the definition and notation for both PS and PG problems.

A formal specification of the PS problem is the following: Let $\mathbf{x}_p$ be an example where $\mathbf{x}_p = (\mathbf{x}_{p1}, \mathbf{x}_{p2}, \ldots, \mathbf{x}_{pD}, \omega)$, with $\mathbf{x}_p$ belonging to a class ω given by $\mathbf{x}_{p\omega}$ and a D-dimensional space in which $\mathbf{x}_{pi}$ is the value of the i-th feature of the p-th sample. Then, let us assume that there is a training set TR which consists of $\mathbf{n}$ instances $\mathbf{x}_p$ and a test set TS composed of $\mathbf{t}$ instances $\mathbf{x}_q$, with ω unknown. Let $SS \subseteq TR$ be the subset of selected samples resulting from the execution of a PS algorithm, then we classify a new pattern $\mathbf{x}_q$ from TS by the NN rule acting over SS.

The purpose of PG is to obtain a prototype generated set GS, which consists of $\mathbf{r}$, $\mathbf{r} < \mathbf{n}$, prototypes, which are either selected or generated from the examples of TR. The prototypes of the generated set are determined to represent efficiently the distributions of the classes and to discriminate well when used to classify the training objects. Their cardinality should be sufficiently small to reduce both the storage and evaluation time spent by an NN classifier.

Both evolutionary and non-evolutionary approaches to PS and PG will be analyzed in the experimental study. A brief description of the methods compared will be detailed in Section 5.

2.2. Differential evolution

Differential evolution follows the general procedure of an EA. DE starts with a population of NP candidate solutions, so-called individuals. The initial population should cover the entire search space as much as possible. In some problems, this is achieved by uniformly randomizing individuals, but in other problems, such as that considered in this paper, basic knowledge of the problem is available and the use of other initialization mechanisms is more effective. The subsequent generations in DE are denoted by $G = 0, 1, \ldots, G_{max}$.

It is usual to denote each individual as a D-dimensional vector $X_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}$, called a "target vector".

2.2.1. Mutation operation

After initialization, DE applies the mutation operator to generate a mutant vector $V_{i,G}$, with respect to each individual $X_{i,G}$, in the current population. For each target $X_{i,G}$, at the generation G, its associated mutant vector $V_{i,G} = \{V^1_{i,G}, \ldots, V^D_{i,G}\}$. The method of creating this mutant vector is that which differentiates one DE

scheme from another. Six of the most frequently referenced strategies are listed below:

- "DE/Rand/1":
$$V_{i,G} = X_{r_1^i,G} + F \cdot (X_{r_2^i,G} - X_{r_3^i,G}) \quad (1)$$
- "DE/Best/1":
$$V_{i,G} = X_{best,G} + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) \quad (2)$$
- "DE/RandToBest/1":
$$V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) \quad (3)$$
- "DE/Best/2":
$$V_{i,G} = X_{best,G} + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) + F \cdot (X_{r_3^i,G} - X_{r_4^i,G}) \quad (4)$$
- "DE/Rand/2":
$$V_{i,G} = X_{r_1^i,G} + F \cdot (X_{r_2^i,G} - X_{r_3^i,G}) + F \cdot (X_{r_4^i,G} - X_{r_5^i,G}) \quad (5)$$
- "DE/RandToBest/2":
$$V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) + F \cdot (X_{r_3^i,G} - X_{r_4^i,G}) \quad (6)$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, r_5^i are mutually exclusive integers randomly generated within the range [1, *NP*], which are also different from the base index *i*. These indices are randomly generated once for each mutation. The scaling factor *F* is a positive control parameter for scaling the difference vectors. $X_{best,G}$ is the best individual of the population in terms of fitness.

2.2.2. Crossover operator

After the mutation phase, a crossover operation is applied to increase the potential diversity of the population. The DE algorithm can use three kinds of crossover schemes, known as "Binomial", "Exponential" and "Arithmetic" crossovers. This operator is applied to each pair of the target vector $X_{i,G}$ and its corresponding mutant vector $V_{i,G}$ to generate a new trial vector that we denote $U_{i,G}$. The mutant vector exchanges its components with the target vector $X_{i,G}$.

We will focus on the binomial crossover scheme, which is performed on each component whenever a randomly picked number between 0 and 1 is less than or equal to the crossover rate (*CR*), The *CR* is a user-specified constant within the range [0,1), which controls the fraction of parameter values copied from the mutant vector. This scheme may be outlined as

$$U_{i,G}^j = \begin{cases} V_{i,G}^j & \text{if rand}(0,1) <= CR \text{ or } j = j_{rand} \\ X_{i,G}^j & \text{Otherwise} \end{cases} \qquad j = 1,2,\ldots,D. \quad (7)$$

where rand[0,1) ∈ [0.1] is a uniformly distributed random number, and $j_{rand} \in \{1,2,\ldots,D\}$ is a randomly chosen index, which ensures that $U_{i,G}$ gets at least one component from $V_{i,G}$.

Finally, we describe the arithmetic crossover, which generates the trial vector $U_{i,G}$ like this,

$$U_{i,G} = X_{i,G} + K \cdot (V_{i,G} - X_{i,G}) \quad (8)$$

where *K* is the combination coefficient which is usually used in the interval [0, 1]. This strategy is known as "DE/CurrentToRand/1".

2.2.3. Selection operator

When the trial vector has been generated, we must decide which individual between X_{i_G} and $U_{i,G}$ should survive in the population of the next generation *G*+1. The selection operator is described as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \text{ is better than } f(X_{i,G}) \\ X_{i,G} & \text{Otherwise} \end{cases} \quad (9)$$

where *f*() is the fitness function to be minimized. If the new trial vector yields a solution equal to or better than the target vector, it replaces the corresponding target vector in the next generation; otherwise the target is retained in the population. Therefore, the population always gets better or retains the same fitness values, but never deteriorates. This one-to-one selection procedure is generally kept fixed in most of the DE algorithms.

2.3. Advanced proposals for DE

The success of DE in solving a specific problem crucially depends on choosing the appropriate mutation strategy and its associated control parameter values (*F* and *CR*) that determine the convergence speed. Hence, a fixed selection of these parameters can produce slow and/or premature convergence depending on the problem. Thus, researchers have investigated the parameter adaptation mechanisms to improve the performance of the basic DE algorithm.

Now, we describe four of the newest and best DE algorithms proposed in the literature.

2.3.1. Self-adaptive differential evolution (SADE)

SADE [49] was proposed by Qin et al. to alleviate the expensive trial-and-error search for the most adequate parameters and mutation strategy. They simultaneously implement four mutation strategies (Eqs. (1) (5), (6), (8)) that are called candidate pool.

For each target vector $X_{i,G}$ in the current population, we have to decide which strategy is selected. Initially, the probability with respect to each strategy is 1/*S*, where *S* is the number of strategies. SADE adapts the probability of generating offspring by either strategy based on their success ratios in the past *LP* generations. Specifically, they introduce success and failure memories to store the number of $U_{i,G}$ that enter the next generation, and the number of discarded $U_{i,G}$.

In SADE, the mutation factors F_i are independently generated at each generation according to a normal distribution *N*(0.5,0.3). The proper choice of *CR* can lead to successful optimization, so they consider gradually adjusting the range of *CR* values according to the previous values. It is adjusted by using a memory, to store the *CR* values with respect to an *S*-strategy, and a normal distribution.

2.3.2. Adaptive differential evolution with optional external archive (JADE)

JADE [52] is proposed by Zhang and Sanderson and it is based on a new mutation strategy and parameter adaptation. The new strategy is called DE/RandTo*p* best with an optional archive that is created to resolve the premature convergence of greedy strategies such as DE/RandToBest/k[1] and DE/Best/k. The authors call *p* the percentage (per unit) of individuals that are considered in the mutation strategy. A mutation vector with DE/RandTo*p* Best/1 with archive is generated as follows:

$$V_{i,G} = X_{i,G} + F_i \cdot (X_{best,G} - X_{i,G}) + F_i \cdot (X_{r1,G} - X'_{r2,G}) \quad (10)$$

where $X_{i,G}$, $X_{r1,G}$ and $X_{best,G}$ are selected from *P* (current population), while $X'_{r2,G}$ is randomly chosen from the union of $P \bigcup A$, where *A* is the archive of inferior solutions stored from recent explorations.

The archive is initially empty. Then, after each generation, the solutions that fail in the selection process are stored to the archive. When the archive size exceeds a certain threshold, some solutions are randomly removed from the archive *A*. Furthermore, this algorithm proposes a parameter adaptation where the mutation factor F_i of each individual is independently generated according to a Cauchy distribution [55,56] with location parameter μ_F and scale

[1] It is also known as DE/CurrentToBest/1.

parameter 0.1, and then it is truncated to be 1 if $F_i >= 1$ or regenerated if $F_i <= 0$. At each generation, the crossover rate CR_i is generated according to a normal distribution $N(\mu_{CR}, 0.1)$, and then truncated to [0,1].

2.3.3. Differential evolution using a neighborhood-based mutation operator (DEGL)

DEGL [51] is also motivated by DE/RandToBest/1. They propose a new mutation model based on neighborhoods. The authors make two kinds of neighborhood called "Local" and "Global" neighborhoods, so they propose two kinds of mutation operator. When they talk about the local neighborhood is not necessarily local in the sense of their geographical nearness or similar fitness values.

These mutation operators are combined in one, in the following manner. For each member of the population, a local trial vector is created by employing the best (fittest) vector in the neighborhood as

$$L_{i,G} = X_{i,G} + F \cdot (X_{Lbest_i,G} - X_{i,G}) + F \cdot (X_{p,G} - X_{q,G}) \quad (11)$$

where $Lbest_i$ is the best vector in the local neighborhood of $X_{i,G}$, and p, q are the indices of two random vectors extracted from the local neighborhood.

Similarly, the global trial vector is created as

$$g_{i,G} = X_{i,G} + F \cdot (X_{gbest_i,G} - X_{i,G}) + F \cdot (X_{r_1,G} - X_{r_2,G}) \quad (12)$$

where $gbest_i$ is the best vector in the current population, and r_1 and r_2 are randomized in the interval [1,NP].

To combine both operators, they use a new parameter, known as "scalar weight" $\omega \in (0,1)$, and they use the following expression:

$$V_{i,G} = \omega \cdot g_{i,G} + (1-\omega) \cdot L_{i,G} \quad (13)$$

As with other adaptive methods, they propose different schemes for adaptation. They introduce three kinds of performance: the adaptation of the new ω parameter, a deterministic linear or exponential increment, and a random value for each vector or a self-adaptive weight factor scheme. However, they do not present an adaptive control parameter for F and CR.

2.3.4. Scale factor local search in differential evolution (SFLSDE)

Scale factor local search in differential evolution was proposed by Neri and Tirronen [53]. This self-adaptive algorithm was inspired by memetic algorithms. In order to guarantee a high quality solution, SFLSDE uses two local search algorithms in the scale factor space to find the appropriate parameters for a given $X_{i,G}$. Specifically, they follow two different approaches: scale factor golden section search (SFGSS) and scale factor hill-climb (SFHC). Both are based on changing the scale factor value and calculate the fitness value of the trial vector $U_{i,G}$ after the mutation and crossover phases.

SFLSDE follows the typical DE scheme, but at each iteration, five random numbers are generated ($rand_1,\ldots,rand_5$) and they are used to determine the corresponding trial vector $U_{i,G}$. The values of the parameters are as follows:

$$F_i = \begin{cases} \text{SFGSS} & \text{if } rand_5 < \tau_3 \\ \text{SFHC} & \text{if } \tau_3 <= rand_5 < \tau_4 \\ \begin{cases} F_l + F_u \cdot rand_1 & \text{if } rand_2 < \tau_1 \\ F_i & \text{otherwise} \end{cases} & \text{if } rand_5 > \tau_4 \end{cases} \quad (14)$$

$$CR_i = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_i & \text{otherwise} \end{cases} \quad (15)$$

where τ_k, $k \in 1,2,3,4$ are constant threshold values. In [53], the authors only use the DE/rand/1/Bin mutation strategy. In the experimental study, we incorporate other mutation strategies.

3. Differential evolution for prototype generation

In this section we explain the proposal to apply the underlying idea of DE to the PG problem as a position adjusting of prototypes scheme. Fig. 1 shows the pseudo-code of the model proposed with the DE/Rand/1 mutation strategy and binomial crossover. In the following we describe the most significant instructions enumerated from 1 to 34.

First of all, it is necessary to define the solution codification. In the proposed DE algorithm, each individual $X_{i,G}$ in the population encodes a complete solution; that is, a reduced set of prototypes are encoded sequentially in each individual.

The number of prototypes encoded in each individual will define its *individual size* and it is denoted **r** as previously. A user parameter will set this value **r**. It is necessary to point out that this

```
1:  G = 0
2:  for i = 1 to NP do
3:      X_{i,G} = Initialization (TR,r)
4:      accuracy_{i,G} = 1NN-leave-one-out (X_{i,G}, TR)
5:  end for
6:  Determine the best individual from the population.
7:  while G < Number of Iterations do
8:      for i = 1 to NP do
9:          Select 3 individuals Xr_{1,G}, Xr_{2,G} and Xr_{3,G}
            {Mutation}
10:         for j = 1 to r do
11:             for z = 1to D do
12:             V_{i,G}[j][z] = Xr_{1,G}[j][z] + F·(Xr_{2,G}[j][z] − Xr_{3,G}[j][z])
13:             end for
14:         end for
15:         V_{i,G} = Applying Thresholds (V_{i,G});
            {Crossover}
16:         U_{i,G} = V_{i,G}
17:         for j = 1 to r do
18:             if R and (0,1) < CR then
19:                 U_{i,G}[j] = V_{i,G}[j]
20:             end if
21:         end for
22:         accuracyTrial[i]=1NN-leave-one-out(U_{i,G},TR);
            {Selection}
23:         if accuracy Trial [i] > accuracy_{i,G} then
24:             accuracy_{i,G} = accuracyTrial [i]
25:             X_{i,G} = U_{i,G}
26:         end if
27:         if accuracy_{i,G} > best Accuracy then
28:             best Accuracy = accuracy_{i,G}
29:             best = i
30:         end if
31:     end for
32:     G = G + 1
33: end while
34: return X_{best,G}
```

Fig. 1. DE algorithm basic structure.

parameter **r** is different to the parameter D explained in Section 2.2. The dimensionality D corresponds to the number of input attributes of the problem.

Following the notation used in Section 2.2, $X_{i,G}$ defines the target vector, but in our case, the target vector can be represented as a matrix. Table 1 describes the structure of an individual. Furthermore, each prototype p_j, $1 \leq j \leq \mathbf{r}$, of an individual $X_{i,G}$ has a class $x_{p\omega,j}$. This class value remains unchangeable by the DE operators throughout the evolutionary cycle, and it is fixed from the beginning of the process. The number of prototypes evolved for each class is assigned in the initialization stage.

3.1. Initialization

DE begins with a population of NP individuals $X_{i,G}$. Given that this problem provides some knowledge based on the initial arrangement of training samples, instruction 3 initializes each individual $X_{i,G}$ by choosing r random prototypes from the TR.

The initialization process ensures that every class has at least one representative prototype. Specifically, we use an initial random stratified selection of prototypes which guarantees that the number of prototypes encoded in each individual for each class is proportional to the number of them in TR. There must be at least one prototype for each class encoded in the individuals. Fig. 2 shows an example.

It is important to point out that every solution must have the same structure, thus they must have the same number of prototypes per class, and the classes must have the same arrangement in the matrix $X_{i,G}$. Following the example of Fig. 2, each individual $X_{i,0}$ should contain four prototypes, in the following order: three prototypes of *Class* 0, one of *Class* 1.

Table 1
Encoding of a set of prototypes in an individual $X_{i,G}$ for the DE algorithm.

	Attribute 1	**Attribute 2**	…	**Attribute *D***	**Class**
Prototype 1	$\mathbf{x}_{p1,1}$	$\mathbf{x}_{p2,1}$	…	$\mathbf{x}_{pD,1}$	$\mathbf{x}_{p\omega,1}$
Prototype 2	$\mathbf{x}_{p1,2}$	$\mathbf{x}_{p2,2}$	…	$\mathbf{x}_{pD,2}$	$\mathbf{x}_{p\omega,2}$
…					
Prototype r	$\mathbf{x}_{p1,r}$	$\mathbf{x}_{p2,r}$	…	$\mathbf{x}_{pD,r}$	$\mathbf{x}_{p\omega,r}$

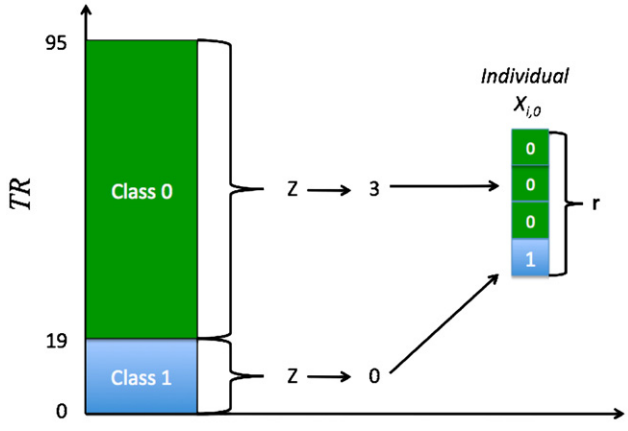

Fig. 2. Initialization process for an individual $X_{i,0}$ in *Appendicitis* data set. TR contains 95 examples. If we established the reduction rate (RR) at 0.95, let us assume that $Z = 1 - RR$, $r = Z \cdot 95$ examples $= 4$ prototypes (truncating this value). *Appendicitis* is composed of two classes, with 76 and 19 prototypes respectively. Hence, the individual $X_{i,0}$ should contain: $Z \cdot 76 = 3$ prototypes of *Class* 0, and $Z \cdot 19 = 0$ prototype of *Class* 1. We ensure that *Class* 1 has at least one prototype.

3.2. Mutation and crossover operators

The mutation and crossover strategies explained in Section 2 have been implemented. From instructions 9–14, the algorithm selects 3 or 5 random individuals, depending on the mutation strategy, and then, it generates the mutant matrix $V_{i,G}$ with respect to each individual $X_{i,G}$, in the current population. The operations of addition, subtraction and scalar product are carried out as typical matrices. This is the justification for the individuals having the same structure. In order for the mutation operator to make sense, the operators must act over the same attributes and over prototypes of the same class in all cases.

After applying this operator, it is necessary to check that the mutant matrix $V_{i,G}$ has been generated with correct values for all features of the prototypes, i.e. to check that the values are in the correct range. Instruction 15 normalizes all attributes of the data set to the [0, 1] range, so this procedure only needs to check if there have been values out of range of [0,1]. If a computed value is greater than 1, we truncate it to 1, and if is lower than 0, we establish it at 0.

Our previous work [48] indicates that the binomial crossover operator has more suitable behavior for the PG problem than the rest of the operators. The new trial matrix is generated by using Eq. (7) and the instructions 16–21 show this operation. In PG, instead of interchanging attributes values, the mutant matrix $V_{i,G}$ exchanges its prototypes with the target $X_{i,G}$ to generate a new trial matrix $U_{i,G}$.

3.3. Selection operator

This operator must decide which individual between $X_{i,G}$ and $U_{i,G}$ should survive in the population of the next generation $G+1$ (instructions 23–26). The NN rule, with $k=1$ (1NN), guides this operator. The instances in TR are classified with the prototypes encoded in $X_{i,G}$ or $U_{i,G}$ by the 1NN rule with a *leave-one-out* validation scheme, and their corresponding fitness values are measured as the *accuracy*$(\cdot)$ obtained, which represents the number of successful hits (correct classifications) relative to the total number of classifications. We try to maximize this value, so the selection operator can be viewed as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } accuracy(U_{i,G}) >= accuracy(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (16)$$

In case of a tie between the values of accuracy, we select the $U_{i,G}$ in order to give the mutated individual the opportunity to enter the population.

Finally, instructions 27–30 check if the selected individual obtains the best fitness in the population, and instruction 34 returns the best individual found during the evolutionary process.

4. Hybridizations of prototype selection and generation methods

This section presents the hybridization model that we propose. Section 4.1 enumerates the arguments that justify hybridization. Section 4.2 explains how to construct the hybrid model.

4.1. Motivation

As we stated before, PS and PG relate to different problems. The main drawback of PS methods is that they assume that the best representative examples can be obtained from a subset of the original data whereas PG methods generate new representative examples if needed. Specifically, positioning adjustment methods

aim to correct the position of a subset of prototypes from the initial set by using an optimization procedure.

However, the positioning adjustment methods are not free of different drawbacks.

- They relate to a more complex problem than PS, i.e. the search space can be more difficult to explore.
- As a result of the above, finding a promising solution by using positioning adjustment methods requires a higher cost than a PS method.
- Positioning adjustment methods usually initialize the generated set *GS* with a fixed number of random prototypes from *TR*, which will be modified in successive iterations. This characteristic is one of the weaknesses of these methods because this parameter can be very dependent on the specific problem. In principle, a practitioner must know the exact number of prototypes which will compose the final solution for each problem, but moreover, the proportion of prototypes between classes should be estimated in order to obtain good solutions. Thus, two schemes of initialization are commonly used:
  - The number of representative instances for each class is proportional to the number of them in the input data.
  - All the classes are represented with the same number of prototypes.

  As we have seen, the appropriate choice of the number of prototypes per class has not been addressed by positioning adjustment techniques.

4.2. Hybrid model

Random selection (stratified or not) of prototypes from the *TR* may not be the most adequate procedure to initialize the *GS*. Instead, we can use a PS algorithm prior to the adjustment process to initialize a subset of prototypes. Making use of this idea, we mitigate the first and second drawbacks stated before as most of the effort performed by positioning adjustment is made over a localized search area given by a PS solution. We also tackle the third weakness, because the heuristic of the PS methods is not forced to select a determinate number of prototypes of each class; it selects the most suitable number of prototypes per class. In addition to this, if the prototypes selected by a PS method can be tuned in the search space, the main drawback associated with PS is also overcome.

To hybridize PS and positioning adjustment methods, two different methods of initialization of the positioning adjustment algorithm will be used, depending on the type of codification of the solution:

- *Complete solution per individual*: This corresponds to the case where each individual of the population encodes a complete *GS* (i.e., that used by DE and PSO). The *SS* must be inserted once as one of the individuals of the population, initializing the rest of the individuals as the standard procedure does.
- *Others*: This is the case where the complete *GS* is optimized (i.e., the scheme used by LVQ3). The resulting SS of the PS methods is used by the positioning adjustment procedure as the initial set.

When each individual of the evolutionary algorithm encodes a complete *GS*, it helps to alleviate the complexity of the optimization procedure, because there is a promising initial individual in the population. Operators used by DE and PSO benefit from the presence of this individual. Furthermore, this type of codification tries to avoid getting stuck at a local optimum, initializing the rest of the individuals with random solutions extracted from the *TR*, keeping the same structure as the *S* selected by the PS method, as in the example given in Section 3.1.

Fig. 3 shows the two different hybrid models. Specifically, Fig. 3(a) presents the scheme to hybridize a PS method with DE and PSO, and Fig. 3(b) shows the hybridization process with LVQ3.

5. Experimental framework

In this section, we show the factors and issues related to the experimental study. We provide the measures employed to evaluate the performance of the algorithms (Section 5.1), details of the problems chosen for the experimentation (Section 5.2), an enumeration of the algorithms used for comparison with their respective parameters (Section 5.3) and finally, the statistical tests employed to contrast the results obtained are described (Section 5.4).

5.1. Performance measures for standard classification

In this work, we deal with multi-class data sets. In these domains, two measures are widely used for measuring the effectiveness of classifiers because of their simplicity and successful application. We refer to accuracy and Cohen's kappa rate. Furthermore, the reduction rate will be used as the classification efficiency measure. They are explained as follows:

- *Accuracy*: is the number of successful hits (correct classifications) relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [57,58].
- *Cohen's kappa* (*Kappa rate*): is an alternative measure to the *classification rate*, since it compensates for random hits [59]. In contrast to accuracy, kappa evaluates the portion of hits that can be attributed to the classifier itself (i.e., not to mere chance), relative to all the classifications that cannot be attributed to chance alone. Cohen's kappa ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement). For multi-class problems, kappa is a very useful, yet simple, meter for measuring a classifier's accuracy while compensating for random successes.
- *Reduction rate*: One of the main goals of the PG and PS methods is to reduce storage requirements. Another goal closely related to this is to speed up classification. A reduction in the number of stored instances will typically yield a corresponding reduction in the time it takes to search through these examples and classify a new input vector.

Note that *Accuracy* and *Kappa* measures are applied over the training data with a *leave-one-out* validation scheme.

5.2. Data sets

In the experimental study, we selected 56 data sets from the UCI repository [60] and the KEEL-dataset repository[2] [61]. Table 2 summarizes the properties of the selected data sets. It shows, for each data set, the number of examples (#Ex.), the number of attributes (#Atts.), and the number of classes (#Cl.). The data sets are grouped into two categories depending on the size they have. Small data sets have less than 2000 instances and large data sets have more than 2000 instances. The data sets considered are partitioned using the 10-fold cross-validation (10-fcv) [62,63] procedure.

[2] http://sci2s.ugr.es/keel/datasets

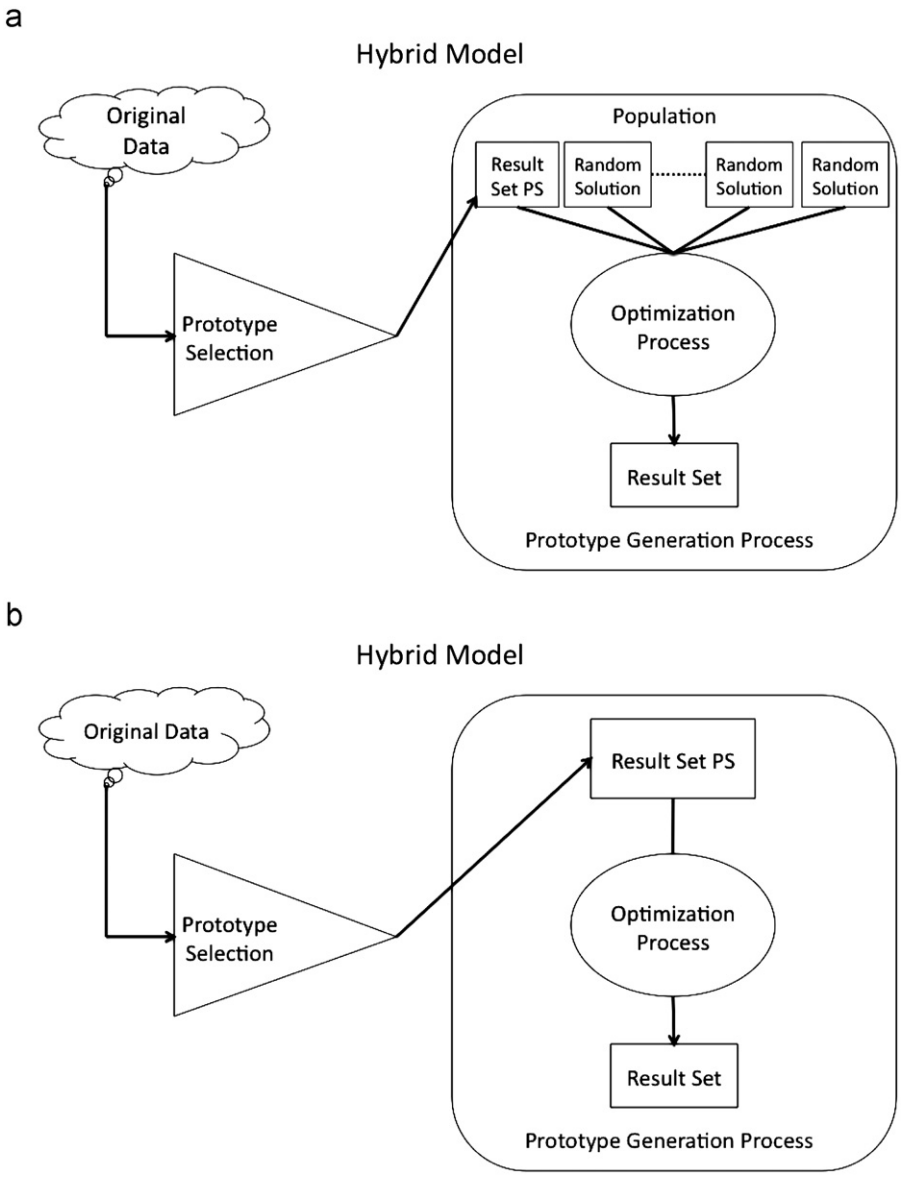

Fig. 3. Hybrid model. (a) PSO and DE approaches; (b) LVQ approach.

In K-fold cross-validation (K-fcv), the original sample is randomly partitioned into K subsamples. Of the K subsamples, a single subsample is retained as the validation data for testing the model, and the remaining K-1 subsamples are used as training data. The cross-validation process is then repeated K times (the folds), with each of the K subsamples used exactly once as the validation data. The K results from the folds will be averaged to produce a single estimation.

5.3. Comparison algorithms and parameters

Several methods, evolutionary and non-evolutionary, have been selected to perform an exhaustive study of the capabilities of our proposals. Those methods are as follows:

1*NN*: The 1NN rule is used as a baseline limit of performance.

Prototype selection methods:

- *DROP*3: This combines an edition stage with a decremental approach where the algorithm checks all the instances in order to find those instances which should be deleted from *GS* [19].
- *ICF*: This method follows an iterative procedure in which those instances susceptible to removal from *GS* based on reachability and coverage properties of the instance are determined [23].
- *SSMA*: This memetic algorithm makes use of a local search or meme specifically developed for the prototype selection problem. This interweaving of the global and local search phases allows the two to influence each other [38].

Prototype generation methods:

- *LVQ*3: Learning vector quantization can be understood as a special case of artificial neural network in which a neuron corresponds to a prototype and a competition weight based is carried out in order to locate each neuron in a concrete place of the *m*-dimensional space to increase the classification accuracy [29]. It will be used as an optimizer in the proposed hybrid models.
- *MixtGauss*: This is an adaptive PG method considered in the framework of mixture modeling by Gaussian distributions, while assuming a statistical independence of features. The prototypes are chosen as the mean vectors of the optimized

Table 2
Summary description for classification data sets.

Data set	#Ex.	#Atts.	#Cl.
Abalone	4174	8	28
Appendicitis	106	7	2
Australian	690	14	2
Autos	205	25	6
Balance	625	4	3
Banana	5300	2	2
Bands	539	19	2
Breast	286	9	2
Bupa	345	6	2
Car	1728	6	4
Chess	3196	36	2
Cleveland	297	13	5
Coil2000	9822	85	2
Contraceptive	1473	9	3
crx	125	15	2
Dermatology	366	33	6
Ecoli	336	7	8
Flare-solar	1066	9	2
German	1000	20	2
Glass	214	9	7
Haberman	306	3	2
Hayes-roth	133	4	3
Heart	270	13	2
Hepatitis	155	19	2
Housevotes	435	16	2
Iris	150	4	3
Led7digit	500	7	10
Lymphography	148	18	4
Magic	19020	10	2
Mammographic	961	5	2
Marketing	8993	13	9
Monks	432	6	2
Movement_libras	360	90	15
Newthyroid	215	5	3
Pageblocks	5472	10	5
Penbased	10992	16	10
Pima	768	8	2
Saheart	462	9	2
Satimage	6435	36	7
Segment	2310	19	7
Sonar	208	60	2
Spambase	4597	57	2
Spectheart	267	44	2
Splice	3190	60	3
Tae	151	5	3
Texture	5500	40	11
Thyroid	7200	21	3
Tic-tac-toe	958	9	2
Titanic	2201	3	2
Twonorm	7400	20	2
Vehicle	846	18	4
Vowel	990	13	11
Wine	178	13	3
Wisconsin	683	9	2
Yeast	1484	8	10
Zoo	101	16	7

Gaussians, whose mixtures are fit to model each of the classes [32].

- *HYB*: This constitutes a hybridization of several prototype reduction techniques. Concretely, HYB combines support vector machines with LVQ3 and executes a search in order to find the most appropriate parameters of LVQ3 [31].
- *RSP*3: This technique is based on Chen's algorithm [30]. The main difference between them is that in Chen's algorithm any subset containing a mixture of instances belonging to different classes can be chosen to be divided. By contrast, in RSP3 [54], the subset with the highest overlapping degree is the one picked to be split. This process tries to avoid drastic changes in the form of decision boundaries associated with *TR* which are the main shortcomings of Chen's algorithm.
- *ENPC*: This follows a genetic scheme with five operators, which focus their attention on defining regions in the search space [42].
- *PSO*: This adjusts the position of an initial set with the PSO rules, attempting to minimize the classification error [39]. We will use it as an optimizer in the proposed hybrid models.
- *PSCSA*: This is based on an artificial immune system [64], using the clonal selection algorithm to find the most appropriate position for a prototype set [43].

Many different configurations are established by the authors of each paper for the different techniques. We focus this experimentation on the recommended parameters proposed by their respective authors, assuming that the choice of the values of the parameters was optimally chosen. The configuration parameters, which are common for all problems, are shown in Table 3. Note that some methods have no parameters to be fixed, so they are not included in this table. In all of the techniques, Euclidean distance is used as a similarity function and those which are stochastic methods have been run three times per partition.

5.4. Statistical tools for analysis

In this paper, we use the hypothesis testing techniques to provide statistical support for the analysis of the results [65,66]. Specifically, we use non-parametric tests, due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility with these parametric tests. These tests are suggested in the studies presented in [67,68,65,69], where their use in the field of machine learning is highly recommended.

Throughout the study, we perform several non-parametric tests. The Wilcoxon test [67,68] will be used to perform a multiple pairwise comparison between the different schemes of our proposals. It will be adopted considering a level of significance of $\alpha = 0.1$.

Furthermore, in order to perform multiple comparisons between our proposals and the rest of the techniques considered, we will use the Friedman Aligned-Ranks test [70] to detect statistical differences among a group of results and the Holm post-hoc test [71], to find out which algorithms are distinctive among the $1^{*}n$ comparisons performed [69]. A complete description of these statistical tests can be found in Appendix A.

More information about these tests and other statistical procedures can be found at http://sci2s.ugr.es/sicidm/.

6. Analysis of results

In this section, we analyze the results obtained from different experimental studies. Specifically, our aims are:

- To compare the different DE schemes to each other and to several classical and recent prototype reduction techniques for 1NN based classification over small data sets (Section 6.1).
- To test the performance of our DE schemes when the size of the problems is increased (Section 6.2).
- To show the convergence process of basic and advanced DE algorithms (Section 6.3).
- To analyze the benefits of hybrid models over small data-sets (Section 6.4).
- To check if the performance of hybrid models is maintained with large data sets (Section 6.5).

6.1. Analysis and results of DE schemes over small size data sets

This study is divided into two parts. First, in Section 6.1.1 we compare the different schemes of DE and identify the best alternatives for the positioning adjustment of prototypes. The Wilcoxon test will be used to support this analysis [67]. Next, Section 6.1.2 shows a comparative study of the better DE methods with other classical and recent PG techniques. In this case, the Friedman Aligned Ranks test for multiple comparisons will be used in association with the Holm post-hoc test [69]. We have used a total of 40 small data sets of the general framework for both experiments.

6.1.1. Results of DE schemes over small data sets

We focus this experiment on comparing the differences in performance of the DE methods based on the experimental framework stated previously.

Table 4 shows the average results (and its standard deviations "+-") obtained over small data sets in training and test data by six different mutation strategies for the basic DE, two configurations for SADE parameters, one for JADE, four different schemes for DEGL and finally SFLSDE has been tested with two mutation strategies. The best case for each column is highlighted in bold. The Wilcoxon test is conducted to compute for each method, with a level of significance of $\alpha = 0.1$, the number of algorithms outperformed by it and the number of algorithms with no detected differences in performance. Specifically, the column denoted by "+" reflects the number of methods outperformed by the method in the row and the column "+=" shows the number of methods with similar or worse performance by the method in the row.

Observing Table 4, we can point out some interesting facts:

- The choice of an adequate mutation strategy seems to be an important factor that influences the results obtained. When the perturbation process is based on the selection of random individuals to generate a new solution, it may be affected by a lack of exploitation capability. However, when the best individual guides the search, exploration capabilities are reduced. RandToBest strategies have reported the best results because they perform a good balance between exploration (random individual) and exploitation (best individual).
- Advanced proposals such as JADE and DEGL, that are completely motivated by the RandToBest strategy, clearly outperform those basic DE techniques which are based on Rand and Best strategies. SADE probably loses accuracy in the iterations that only execute Rand and Best strategies. The DEGL algorithm, with an exponential increment of the parameter ω, has reported the best kappa test and the statistical test shows that it also

Table 3
Parameter specification for all the methods employed in the experimentation.

Algorithm	Parameters
SSMA	Population=30, Eval=10 000, Cross=0.5, Mutation=0.001
LVQ3	Iterations = 500, α = 0.1, WindowWidth=0.2, *epsilon* = 0.1, Reduction Rate = 0.95/0.99
HYB	Search_Iter = 200, Optimal_Iter = 1000, *alpha* = 0.1, I_*epsilon* = 0, F_*epsilon* = 0.5 Initial_Window = 0, Final_Window = 0.5, *delta* = 0.1, delta_Window = 0.1, Initial Selection = SVM
RSP3	Subset Choice = Diameter
ENPC	Iterations = 250
PSO	SwarmSize = 40, Iterations = 500 C1 = 1, C2 = 3, Vmax = 0.25, Wstart = 1.5, Wend = 0.5, Reduction Rate = 0.95/0.99
PSCSA	HyperMutation Rate = 2, Clonal Rate = 10, Mutation Rate = 0.01, Stim_Threshold = 0.89, α = 0.4
DE	PopulationSize = 40, Iterations = 500, F = 0.5, CR = 0.9, Reduction Rate = 0.95/0.99
SADE	PopulationSize = 40, Iterations = 500, Learning Period = 50 and 100, Reduction Rate = 0.95/0.99
JADE	PopulationSize = 40, Iterations = 500, p=0.05, c= 0.1, Reduction Rate = 0.95/0.99
DEGL	PopulationSize = 40, Iterations = 500, F = 0.8, CR =0.9, WeightFactor=0.0, WeightScheme= Exponential, Adaptive, Random and Linear, Reduction Rate = 0.95/0.99
SFLSDE	PopulationSize = 40, Iterations = 500, iterSFGSS =8, iterSFHC=20, Fl=0.1, Fu=0.9, Reduction Rate = 0.95/0.99

Note: The parameter reduction rate on fixed reduction algorithms has been established at 0.95 for small size data set, 0.99 for large.

Table 4
Results of different DE models over small data sets.

Algorithm	Accuracy		Kappa rate		Acc Tst		Kappa Tst	
	Training	Test	Training	Test	+	+=	+	+=
DE/Rand/1/Bin	0.7679 ±0.1536	0.7268 ±0.1670	0.5816 ±0.2216	0.4947 ±0.2579	0	13	0	10
DE/Best/1/Bin	0.8005 ±0.1275	0.7393 ±0.1464	0.6355 ±0.1986	0.5155 ±0.2468	0	10	0	10
DE/RandToBest/1/Bin	0.8279 ±0.1192	0.7524 ±0.1460	0.6859 ±0.1887	0.5384 ±0.2453	0	**14**	1	13
DE/Best/2/Bin	0.8285 ±0.1174	0.7434 ±0.1445	0.6854 ±0.1875	0.5212 ±0.2463	1	13	2	13
DE/Rand/2/Bin	0.7962 ±0.1456	0.7348 ±0.1563	0.6295 ±0.2187	0.5061 ±0.2484	0	12	0	7
DE/RandToBest/2/Bin	0.8231 ±0.1250	0.7567 ±0.1426	0.6735 ±0.2031	0.5406 ±0.2484	0	**14**	1	**14**
SADE LP 50	0.8195 ±0.1563	0.7513 ±0.1452	0.6708 ±0.1563	0.5335 ±0.2482	1	12	2	13
SADE LP 100	0.8243 ±0.1232	0.7502 ±0.1435	0.6776 ±0.1979	0.5324 ±0.2432	0	12	0	12
JADE	0.8209 ±0.1219	0.7541 ±0.1417	0.6708 ±0.1974	0.5417 ±0.2415	0	**14**	3	**14**
DEGL EXP	0.8144 ±0.1563	0.7597 ±0.1394	0.6728 ±0.1563	**0.5529 ±0.2328**	**6**	**14**	**4**	**14**
DEGL ADAP	0.8211 ±0.1563	0.7525 ±0.1401	0.6606 ±0.1563	0.5351 ±0.2436	1	**14**	1	**14**
DEGL RANDOM	0.8146 ±0.1563	0.7488 ±0.1392	0.6615 ±0.1936	0.5329 ±0.2335	0	13	2	12
DEGL LINEAR	0.8187 ±0.1563	0.7550 ±0.1404	0.6689 ±0.1930	0.5437 ±0.2371	1	13	2	13
SFLSDE/Rand/1/Bin	0.8347 ±0.1563	0.7582 ±0.1563	0.6960 ±0.1948	0.5461 ±0.1563	2	**14**	**4**	**14**
SFLSDE/RandToBest/1/Bin	**0.8411 ±0.1563**	**0.7619 ±0.1563**	**0.7079 ±0.1563**	0.5516 ±0.1563	2	**14**	2	13

overcomes more methods supported with a level of significance $\alpha = 0.1$ in terms of accuracy rate. The other DEGL's variants obtain similar results except for the random approach that is probably affected by a lack of convergence.
- SFLSDE with the RandToBest strategy achieves the best average results in accuracy. This technique involves the best mutation strategy and two local searches which allow it to find the most suitable parameters during the evolution process.

Looking at accuracy, kappa rate and the statistical test; three methods deserve particular mention: DEGL exponential, SFLSDE Rand and SFLSDE RandToBest. We will use these methods in the comparison with other PG methods.

6.1.2. Comparison with other PG techniques over small data sets

In this section, we perform a comparison between the best three DE models checked before (SFLSDE Rand, RandToBest and DEGL exponential) with respect to the other 7 PG methods. Table 5 shows the average results collected. In this case, we add the reduction rate, which is, an important measure to compare the methods.

Table 6 presents the rankings obtained by the Friedman Aligned (FA) procedure with the accuracy measure. In this table, algorithms are ordered from the best to the worst ranking. Furthermore, the third column shows the adjusted p-value with the Holm's test (*Holm APV*) [69]. Note that the SFLSDE RandToBest is established as the control algorithm because it has obtained the best FA ranking. Holm's test uses the same level of significance as Wilcoxon, $\alpha = 0.1$. Algorithms highlighted in bold are those which have been outperformed with this level of significance.

Observing both Tables 5 and 6, we want to make some interesting comments:

- DE methods significantly outperform the other PG techniques, except for PSO, in accuracy and kappa rate. PSO is clearly the most competitive PG algorithm for DE. PSO has the same type of solution codification as DE and a similar evolutionary scheme, but advanced proposals of the DE algorithm usually obtain better average results.
- We could also have stressed RSP3, HYB and ENPC algorithms as competitive algorithms for DE. But, as we can see in the table, they obtain a good performance over training results, but they do not report great test results, therefore they have a higher overfitting than the DE algorithms.
- In terms of reduction capabilities, DE has been fixed to 0.95. Only MixtGauss and PSCSA obtain better reduction rates, but they offer lower accuracy/kappa rates. DE outperforms the rest of the methods with similar or lower reduction rates.

Now, we focus our attention on the first statement. Holm's test has no reported significant differences between DE and PSO. PSO probably benefits from the multiple comparison test, because it significantly outperforms the rest of the PG techniques. For this reason, we want to check the comparison between PSO and DE with the Wilcoxon test. Table 7 shows the p-values obtained with the Wilcoxon test. As we can see, advanced DE proposals always outperform the PSO algorithm with a level of significance of $\alpha = 0.1$.

6.2. Analysis and results of DE schemes over large size data sets

This section presents the study and analysis of large size data sets. The goal of this study is to analyze the effect of scaling up the data in DE methods. Again, we divide this section into two different stages. First, in Section 6.2.1 we look for the best DE method over large data sets. Next, Section 6.2.2 compares the results with other PG methods.

6.2.1. Results of DE schemes over large data sets

In order to test the performance of DE methods we have established a high reduction rate 0.99. Table 8 presents

Table 6
Average rankings of the algorithms over small data sets (Friedman Aligned-Ranks + Holm's Test).

Algorithm	FA ranking	Holm APV
SFLSDE/RandToBest/1/Bin	131.4625	–
SFLSDE/Rand/1/Bin	138.3	1.0
DEGL EXP	139.4	1.0
PSO	158.275	1.0
RSP3	225.3999	**0.0044**
1NN	226.0	**0.0044**
HYB	258.3625	$\mathbf{4.85} \times 10^{-5}$
ENPC	268.15	$\mathbf{1.07} \times 10^{-5}$
Mixt_Gauss	269.6875	$\mathbf{9.33} \times 10^{-6}$
PSCSA	286.1875	$\mathbf{4.75} \times 10^{-7}$
LVQ3	324.275	$\mathbf{1.19} \times 10^{-10}$

Table 7
Results of the Wilcoxon test compared with PSO over small data sets.

Comparison	p-Value
DEGL EXP vs. PSO	**0.0106**
SFLSDE Rand vs. PSO	**0.0986**
SFLSDE RandToBest vs. PSO	**0.0408**

Table 5
Comparison between the three best DE models and other PG approaches over small data sets.

Algorithm	Accuracy		Kappa rate		Reduction
	Training	Test	Training	Test	
1NN	0.7369 ± 0.1654	0.7348 ± 0.1664	0.4985 ± 0.2910	0.4918 ± 0.2950	0.0000 ± 0.0000
MixtGauss	0.7138 ± 0.1545	0.6932 ± 0.1668	0.4888 ± 0.2473	0.4546 ± 0.2680	0.9552 ± 0.0084
LVQ3	0.6931 ± 0.1560	0.6763 ± 0.1662	0.4421 ± 0.2458	0.4114 ± 0.1563	0.9488 ± 0.0083
HYB	0.8309 ± 0.0154	0.7153 ± 0.1651	0.6988 ± 0.2573	0.4790 ± 0.1563	0.4278 ± 0.1563
RSP3	0.7924 ± 0.1373	0.7325 ± 0.1591	0.6112 ± 0.2420	0.5004 ± 0.2861	0.7329 ± 0.1185
ENPC	0.8247 ± 0.1477	0.7167 ± 0.1597	0.6800 ± 0.2532	0.4818 ± 0.2936	0.7220 ± 0.1447
PSO	0.8238 ± 0.1274	0.7501 ± 0.1409	0.6791 ± 0.1950	0.5332 ± 0.2402	0.9491 ± 0.0072
PSCSA	0.6787 ± 0.1835	0.6682 ± 0.1874	0.4461 ± 0.2466	0.4231 ± 0.2540	**0.9858 ± 0.1563**
DEGL EXP	0.8144 ± 0.1563	0.7597 ± 0.1394	0.6728 ± 0.1563	**0.5529 ± 0.2328**	0.9483 ± 0.1563
SFLSDE/Rand/1/Bin	0.8347 ± 0.1563	0.7582 ± 0.1563	0.6960 ± 0.1948	0.5461 ± 0.1563	0.9481 ± 0.1563
SFLSDE/RandToBest/1/Bin	**0.8411 ± 0.1563**	**0.7619 ± 0.1563**	**0.7079 ± 0.1563**	0.5516 ± 0.1563	0.9481 ± 0.1563

Table 8
Results of different DE models over large data sets.

Algorithms	Accuracy		Kappa rate		Acc Tst		Kappa Tst	
	Training	Test	Training	Test	+	+=	+	+=
DE/Rand/1/Bin	0.7831 ±0.0055	0.7798 ±0.2075	0.5709 ±0.2713	0.5639 ±0.2761	0	12	0	12
DE/Best/1/Bin	0.8025 ±0.0038	0.7881 ±0.2069	0.5883 ±0.2849	0.5605 ±0.2894	0	8	0	8
DE/RandToBest/1/Bin	0.8124 ±0.0032	0.7968 ±0.2087	0.6115 ±0.2845	0.5815 ±0.2888	2	10	2	10
DE/Best/2/Bin	0.8183 ±0.0045	0.7988 ±0.2086	0.6224 ±0.2859	0.5843 ±0.2897	1	11	1	11
DE/Rand/2/Bin	0.7888 ±0.0068	0.7838 ±0.2088	0.5803 ±0.2691	0.5686 ±0.2753	0	10	0	10
DE/RandToBest/2/Bin	0.8243 ±0.0045	0.8088 ±0.2113	0.6377 ±0.2920	0.6073 ±0.2928	6	**14**	6	**14**
SADE LP 50	0.8107 ±0.0036	0.7966 ±0.2063	0.6178 ±0.2722	0.5918 ±0.2748	2	13	2	13
SADE LP 100	0.8070 ±0.0032	0.7941 ±0.2070	0.6030 ±0.2793	0.5789 ±0.2829	0	12	0	12
JADE	0.8136 ±0.0110	0.8020 ±0.2058	0.6204 ±0.2803	0.5969 ±0.2830	6	13	6	13
DEGL EXP	0.8076 ±0.0032	0.7951 ±0.2058	0.6044 ±0.2783	0.5792 ±0.2829	0	11	0	11
DEGL ADAP	0.8069 ±0.0041	0.7946 ±0.2074	0.6027 ±0.2811	0.5761 ±0.2883	1	10	1	10
DEGL RANDOM	0.8069 ±0.0037	0.7938 ±0.2079	0.6025 ±0.2799	0.5772 ±0.2844	0	8	0	8
DEGL LINEAR	0.8088 ±0.0031	0.7961 ±0.2058	0.6052 ±0.2810	0.5811 ±0.2831	1	11	1	11
SFLSDE/Rand/1/Bin	0.8327 ±0.0046	**0.8181 ±0.2074**	0.6541 ±0.2840	**0.6243 ±0.2925**	9	**14**	9	**14**
SFLSDE/RandToBest/1/Bin	**0.8341 ±0.0030**	0.8154 ±0.2072	**0.6556 ±0.2879**	0.6184 ±0.2910	**11**	**14**	**11**	**14**

Table 9
Comparison between the two best DE models and other PG approaches over large data sets.

Algorithm	Accuracy		Kappa rate		Reduction
	Training	Test	Training	Test	
1NN	0.8197 ±0.0023	0.8072 ±0.0100	0.6195 ±0.0229	0.5948 ±0.0181	0.0000 ±0.0000
MixtGauss	0.7534 ±0.0141	0.7505 ±0.2315	0.4913 ±0.3251	0.4860 ±0.3255	0.9514 ±0.0001
LVQ3	0.6840 ±0.0057	0.6767 ±0.2680	0.4409 ±0.2926	0.4264 ±0.2962	0.9899 ±0.0011
HYB	0.7888 ±0.0234	0.7618 ±0.2168	0.5992 ±0.2790	0.5567 ±0.3153	0.5727 ±0.2903
RSP3	0.7922 ±0.2545	0.7556 ±0.2708	0.6299 ±0.3266	0.5597 ±0.3397	0.8100 ±0.1369
ENPC	**0.8809 ±0.1610**	0.7986 ±0.2188	**0.7613 ±0.2497**	0.6170 ±0.2949	0.8205 ±0.1919
PSO	0.8022 ±0.0055	0.8049 ±0.2136	0.6177 ±0.2887	0.5948 ±0.2880	0.9899 ±0.0011
PSCSA	0.6730 ±0.2190	0.6707 ±0.2205	0.3900 ±0.2376	0.3842 ±0.2824	**0.9988 ±0.0017**
SFLSDE/Rand/1/Bin	0.8388 ±0.0039	**0.8249 ±0.2205**	0.6570 ±0.3036	**0.6281 ±0.3131**	0.9901 ±0.0002
SFLSDE/RandToBest/1/Bin	0.8414 ±0.0028	0.8236 ±0.2199	0.6598 ±0.3079	0.6240 ±0.3131	0.9901 ±0.0002

the comparative study. Again, we use the Wilcoxon test to differentiate between the different proposals.

We can make several observations from these results:

- Some models present important differences when tackling large data sets. We can stress JADE as a good algorithm when the size of the data set is higher. With large data sets, this algorithm overcomes most of the advanced proposals; except for SFLSDE which remains the best advanced DE model.
- The number of difference vectors to be perturbed by the mutation operator does seem to be an important factor that influences the final result obtained when dealing with large data sets.
- SFLSDE/Rand/1/Bin has reported the best average results in accuracy and kappa rate. However, the statistical test shows that SFLSDE RandToBest has better behavior than the Rand strategy. As we can observe in Table 8, SFLSDE/Rand/1/Bin is able to overcome nine methods and SFLSDE RandToBest 11 methods. The rest of the proposals advanced are not able to overcome SFSLDE.
- When dealing with large data sets, the statistical test notes higher differences between the methods. Concretely, we can observe that SFLSDE RandToBest outperforms a total of 11 methods out of 14 with a level of significance of 0.1.

We select both SFLSDE algorithms for the next comparison as they have, in general, reported the best results.

Table 10
Average rankings of the algorithms over large data sets (Friedman Aligned-Ranks + Holm's Test).

Algorithm	FA ranking	Holm APV
SFLSDE/RandToBest/1/Bin	47.0313	–
SFLSDE/Rand/1/Bin	48.0938	0.9482
1NN	65.1875	0.7359
PSO	66.0625	0.7359
ENPC	71.875	0.5174
RSP3	86.9063	**0.0746**
HYB	92.6875	**0.0319**
Mixt_Gauss	94.375	**0.0269**
LVQ3	113.5313	$\mathbf{3.9323} \times 10^{-4}$
PSCSA	119.25	$\mathbf{9.3584} \times 10^{-5}$

6.2.2. Comparison with other PG techniques over large data sets

In this section, we perform a comparison between the two best DE models obtained for large data sets (SFLSDE models) with the same algorithms as in Section 6.1.2. Table 9 shows the average results obtained, and Table 10 displays the FA ranking and the adjusted p-value obtained with Holm's test.

Observing Tables 9 and 10, we can summarize that:

- Most of the PG methods present clear differences when dealing with large data sets. For instance, ENPC outperforms its ranking obtained over small data sets. Together with PSO, they are the most competitive PG techniques for the DE model and Holm's test supports this statement. However, SFLSDE usually obtains better average results.

- DE methods significantly overcome the rest of the PG techniques. Specifically, accuracy and the kappa rate demonstrate that SFLSDE Rand is able to overcome in 0.02 the average results obtained for the best PG technique (PSO).
- In order to improve the efficiency of the 1NN rule when tackling large data sets, the reduction rate becomes more important. FA ranking indicates that only SFLSDE models are able to outperform the 1NN with a high reduction rate (0.99), to a greater extent than the rest of the PG methods.

Again, we use the Wilcoxon test to check if the DE models are able to outperform the most competitive PG algorithms. Specifically, we carry out a study with ENPC and PSO. Table 11 presents the results.

The Wilcoxon test shows that ENPC is outperformed with $\alpha = 0.1$. However, this hypothesis is rejected with PSO, but the p-value is smaller than the adjusted p-value of Holm's test.

Table 11
Results of the Wilcoxon test compared with PSO and ENPC over large data sets.

Comparison	p-Value
SFLSDE Rand vs. PSO	0.1981
SFLSDE RandToBest vs. PSO	0.1928
SFLSDE Rand vs. ENPC	**0.0183**
SFLSDE RandToBest vs. ENPC	**0.0214**

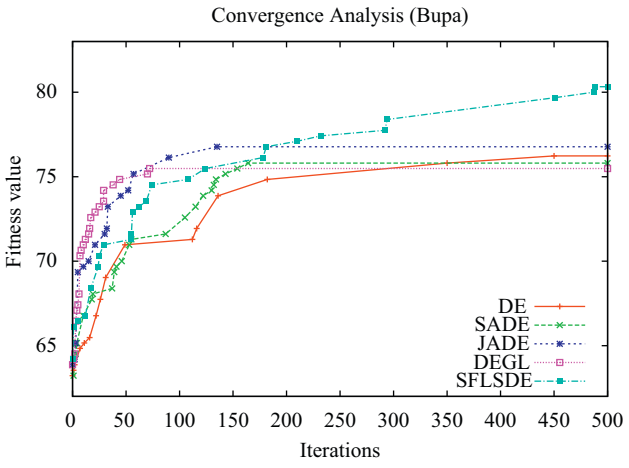

Fig. 4. Map of convergence: Bupa data set.

6.3. Analysis of convergence

One of the most important issues in the development of any EA is the analysis of the convergence of its population. If the EA does not evolve in time, it will not be able to obtain suitable solutions.

We show a graphical representation of the convergence capabilities of DE models, Fig. 4. Specifically, the best basic DE (DE/RandToBest/2/Bin), SADE (LP=50), JADE, DEGL exponential, and SFLSDE with RandToBest.

To perform this analysis we have selected the *Bupa* small data set. The graphics show a line representing the fitness value of the best individual of each population. The X-axis represents the number of iterations carried out, and the Y-axis represents the fitness value currently achieved.

As we can see in the graphic, SADE and DEGL quickly find promising solutions and they waste more than 300 iterations without an improvement. However, SFLSDE and DE are slower to converge, which usually allows them to find a better solution at the end of the process. They find a great balance between exploration and exploitation during the evolution.

6.4. Analysis and results of hybrid models over small size data sets

This section shows the average results obtained for our hybrid models when they are applied to small data sets. Table 12 collects the average results and the Wilcoxon test. The abilities of hybrid models are shown and their performance is compared with the basic components that take part in it.

The results achieved in this part of the study allow us to conclude the following:

- Hybrid models always outperform the basic algorithms upon which they are based. The good synergy between PG and PS methods is clearly demonstrated with the obtained results. We selected three PS methods with different reduction rates (ICF 0.7107, DROP3 0.8202 and SSMA 0.9553). A priori a lower

Table 13
Average runtime of the optimizer algorithms over small data sets.

Runtime		
SFLSDE	PSO	LVQ3
40.5483	42.3168	0.2316

Table 12
Hybrid models with small data sets.

Algorithm	Accuracy		Kappa rate		Reduction	Acc Tst		Kappa Tst	
	Training	Test	Training	Test		+	+=	+	+=
DROP3	0.7527 ±0.1240	0.7011 ±0.1497	0.5498 ±0.2139	0.4544 ±0.2651	0.8202 ±0.0148	0	5	1	6
ICF	0.7118 ±0.1343	0.6784 ±0.1505	0.4797 ±0.2364	0.4175 ±0.2644	0.7107 ±0.1369	0	4	0	4
SSMA	0.8207 ±0.1335	0.7581 ±0.1518	0.6685 ±0.2089	0.5455 ±0.2685	**0.9554 ±0.0343**	7	**15**	6	**15**
LVQ3	0.6931 ±0.1560	0.6763 ±0.1662	0.4421 ±0.2458	0.4114 ±0.1563	0.9488 ±0.0083	0	4	0	2
PSO	0.8238 ±0.1274	0.7501 ±0.1409	0.6791 ±0.1950	0.5332 ±0.2402	0.9491 ±0.0072	6	**15**	5	14
SFLSDE/RandToBest/1/Bin	0.8411 ±0.1563	0.7619 ±0.1563	0.7079 ±0.1563	0.5516 ±0.1563	0.9481 ±0.1563	9	14	7	13
DROP3+LVQ3	0.7666 ±0.1197	0.7027 ±0.1471	0.5705 ±0.2200	0.4553 ±0.2703	0.8202 ±0.0809	0	5	1	5
DROP3+PSO	0.8645 ±0.0970	0.7501 ±0.1349	0.7474 ±0.1688	0.5286 ±0.2588	0.8202 ±0.0809	5	9	6	13
DROP3+SFLSDE/RandToBest/1/Bin	0.8711 ±0.0958	0.7620 ±0.1348	0.7605 ±0.1687	0.5488 ±0.2572	0.8202 ±0.0809	6	13	6	14
ICF+LVQ3	0.7384 ±0.1189	0.6865 ±0.1413	0.5229 ±0.2191	0.4282 ±0.2590	0.7107 ±0.1369	0	5	0	5
ICF+PSO	0.8677 ±0.0980	0.7523 ±0.1398	0.7526 ±0.1678	0.5318 ±0.2593	0.7107 ±0.1369	5	9	3	8
ICF+SFLSDE/RandToBest/1/Bin	**0.8738 ±0.0991**	0.7618 ±0.1401	**0.7642 ±0.1730**	0.5462 ±0.2695	0.7107 ±0.1369	7	14	7	**15**
SSMA+LVQ3	0.8347 ±0.1100	0.7704 ±0.1267	0.6842 ±0.2087	0.5619 ±0.2657	**0.9554 ±0.0343**	9	14	6	**15**
SSMA+PSO	0.8617 ±0.1007	0.7770 ±0.1267	0.7376 ±0.1852	0.5727 ±0.2515	**0.9554 ±0.0343**	7	**15**	9	**15**
SSMA+SFLSDE/RandToBest/1/Bin	0.8651 ±0.1010	**0.7845 ±0.1256**	0.7407 ±0.1891	**0.5836 ±0.2524**	**0.9554 ±0.0343**	**10**	**15**	**10**	**15**
1NN	0.7369 ±0.1654	0.7348 ±0.1664	0.4985 ±0.2910	0.4918 ±0.2950	0.0000 ±0.0000	2	11	4	10

Table 14
Hybrid models with large data sets.

Algorithms	Accuracy		Kappa rate		Reduction	Acc Tst		Kappa Tst	
	Training	Test	Training	Test		+	+=	+	+=
DROP3	0.7744 ± 0.0232	0.7472 ± 0.2256	0.5592 ± 0.3079	0.5119 ± 0.3248	0.9061 ± 0.0569	1	5	1	5
ICF	0.6781 ± 0.1863	0.6621 ± 0.2016	0.4202 ± 0.3079	0.3940 ± 0.3143	0.8037 ± 0.1654	0	2	0	2
SSMA	0.8493 ± 0.0021	0.8196 ± 0.2220	0.6725 ± 0.3134	0.6221 ± 0.3177	0.9844 ± 0.0100	9	14	9	14
LVQ3	0.6840 ± 0.0057	0.6767 ± 0.2680	0.4409 ± 0.2926	0.4264 ± 0.2962	0.9899 ± 0.0011	0	4	0	4
PSO	0.8022 ± 0.0055	0.8049 ± 0.2136	0.6177 ± 0.2887	0.5948 ± 0.2880	0.9899 ± 0.0011	3	13	3	13
SFLSDE/RandToBest/1/Bin	0.8414 ± 0.0028	0.8236 ± 0.2199	0.6598 ± 0.3079	0.6240 ± 0.3131	**0.9901 ± 0.0002**	7	12	7	12
DROP3+LVQ3	0.7730 ± 0.2166	0.7412 ± 0.2382	0.5661 ± 0.3140	0.5106 ± 0.3308	0.9061 ± 0.0569	3	6	3	6
DROP3+PSO	0.8386 ± 0.1913	0.7974 ± 0.2236	0.6496 ± 0.2911	0.5768 ± 0.3130	0.9061 ± 0.0569	5	9	5	9
DROP3+SFLSDE/RandToBest/1/Bin	0.8538 ± 0.1868	0.8152 ± 0.2260	0.6843 ± 0.2875	0.6173 ± 0.3135	0.9061 ± 0.0569	6	14	6	14
ICF+LVQ3	0.6851 ± 0.1796	0.6641 ± 0.1997	0.4297 ± 0.3042	0.3960 ± 0.3133	0.8302 ± 0.1386	0	3	0	3
ICF+PSO	0.8397 ± 0.1950	0.8046 ± 0.2259	0.6444 ± 0.3050	0.5846 ± 0.3226	0.8302 ± 0.1386	5	12	5	12
ICF+SFLSDE/RandToBest/1/Bin	0.8367 ± 0.1924	0.8145 ± 0.2260	0.6434 ± 0.2933	0.6082 ± 0.3151	0.8302 ± 0.1386	6	13	6	13
SSMA+LVQ3	0.8534 ± 0.1998	0.8244 ± 0.2197	0.6793 ± 0.3120	0.6312 ± 0.3163	0.9847 ± 0.0099	6	**15**	6	**15**
SSMA+PSO	0.8576 ± 0.2007	0.8241 ± 0.2225	0.6970 ± 0.3030	0.6384 ± 0.3092	0.9847 ± 0.0099	8	**15**	8	**15**
SSMA+SFLSDE/RandToBest/1/Bin	**0.8635 ± 0.1979**	**0.8291 ± 0.2213**	**0.7056 ± 0.3005**	**0.6442 ± 0.3105**	0.9847 ± 0.0099	**11**	**15**	**11**	**15**
1NN	0.8197 ± 0.0023	0.8072 ± 0.0100	0.6195 ± 0.0229	0.5948 ± 0.0181	0.0000 ± 0.0000	3	**15**	3	**15**

reduction rate should allow better accuracy results to be obtained. In terms of accuracy/kappa rates, we can observe how the hybrid models, DROP3+SFSLDE and ICF+SFLSDE probably produce overfitting in training data, because they do not present good generalization capabilities, obtaining lower accuracy/kappa rates in test results.

- SFSLDE is the best performing method in comparison with PS and PG basic algorithms. Furthermore, when it is applied as an optimizer method in the hybrid models it achieves the best accuracy/kappa rates. As we stated before, it can sometimes produce overfitting. However, as we can see from the test results of SSMA+SFLSDE, when a high reduction PS method is applied, SFLSDE is very effective. We can extrapolate this statement for LVQ3 and PSO, which do not produce overfitting over the resulting set selected by SSMA.
- Although LVQ3 does not offer competitive results, when it is used to optimize a PS solution, LVQ3 is able to improve appropriately the position of the prototypes. For instance, as we can see with SSMA+LVQ3 in comparison with SSMA, LVQ3 works properly when it starts from a good solution. Although PSO and DE outperform LVQ3 as optimizers, an advantage of LVQ is that it is faster. Table 13 shows the average runtime[3] of the optimizers over small data sets. As we can see, the learning time of LVQ3 is clearly lower than PSO and DE which codify a complete solution per individual.
- With the same reduction rate, PSO outperforms LVQ3, and it is more effective when we use it in the hybrid models. Nevertheless, with the obtained results and in comparison with the DE algorithms, PSO is probably affected by a lack of convergence because of the absence of an adaptive process to improve its own parameters during the evolution.

6.5. Analysis and results of hybrid models over large size data sets

In this section we want to check if the performance of hybrid models is maintained when dealing with large data sets. Table 14 shows this experiment.

We briefly summary some interesting facts:

- When dealing with large data sets, reduction rate must be taking into consideration as one of the main parameters to improve the efficiency of the 1NN rule. The DE model has been fixed with a high reduction rate (0.99) and the advanced proposal SFLSDE outperforms the rest of the PG and PS basic techniques which are far from achieving this reduction rate.
- SSMA was proposed to cover a drawback of the conventional evolutionary PS methods: their lack of convergence when facing large problems. We can observe that it is the best PS method and its performance is improved when we hybridize with an optimization procedure. SSMA provides a promising solution which enables any optimization process, including LVQ3 which does not offer great results in combination with ICF and DROP3, to converge quickly.

7. Conclusions

In this work, we have presented differential evolution and its recent advanced proposals as a data reduction technique. Specifically, it was used to optimize the positioning of the prototypes for the nearest neighbor algorithm, acting as a prototype generation method.

The first aim of this paper is to determine which proposed DE algorithm works properly to tackle the PG problem. Specifically, we have studied the different mutation strategies recognized in the literature, and the recent approaches to adapt the parameters of this evolutionary algorithm in order to find a good balance between exploration and exploitation.

The second contribution of this paper shows the good relation between PS and PG in obtaining hybrid algorithms that allow us to find very promising solutions. Hybrid models are able to tackle several drawbacks of the isolated PS and PG methods. Concretely, we have analyzed the use of positioning adjustment algorithms as an optimization procedure after a previous PS stage. Our DE model is an appropriate optimizer which has reported the best results in terms of accuracy and reduction rate.

The wide experimental study performed has allowed us to justify the behavior of DE algorithms when dealing with small and large data sets. These results have been compared with several non-parametric statistical procedures, which have reinforced the conclusions.

Acknowledgement

Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01.

[3] These results have been obtained with an Intel(R) Core(TM) i7 CPU 920 at 2.67 GHz.

Appendix A. Friedman Aligned Ranks and adjusted *p*-values

The Friedman test is based on n sets of ranks, one set for each data set in our case; and the performances of the algorithms analyzed are ranked separately for each data set. Such a ranking scheme allows for intra-set comparisons only, since inter-set comparisons are not meaningful. When the number of algorithms for comparison is small, this may pose a disadvantage. In such cases, comparability among data sets is desirable and we can employ the method of aligned ranks [70].

In this technique, a value of location is computed as the average performance achieved by all algorithms in each data set. Then it calculates the difference between the performance obtained by an algorithm and the value of location. This step is repeated for algorithms and data sets. The resulting differences, called aligned observations, which keep their identities with respect to the data set and the combination of algorithms to which they belong, are then ranked from 1 to kn relative to each other. Then, the ranking scheme is the same as that employed by a multiple comparison procedure which employs independent samples; such as the Kruskal–Wallis test [72]. The ranks assigned to the aligned observations are called aligned ranks.

The Friedman aligned ranks test statistic can be written as

$$T=\frac{(k-1)\left[\sum_{j=1}^{k}\hat{R}_j^{\,2}-(kn^2/4)(kn+1)^2\right]}{\{[kn(kn+1)(2kn+1)]/6\}-(1/k)\sum_{i=1}^{n}\hat{R}_{i\cdot}^{\,2}} \tag{17}$$

where $\hat{R}_{i\cdot}$ is equal to the rank total of the i-th data set and $\hat{R}_{\cdot j}$ is the rank total of the j-th algorithm.

The test statistic T is compared for significance with a chi-square distribution for $k-1$ degrees of freedom. Critical values can be found at Table A3 in [66]. Furthermore, the p-value could be computed through normal approximations [73]. If the null-hypothesis is rejected, we can proceed with a post-hoc test. In this study, we use the Holm post-hoc procedure.

We focus on the comparison between a control method, which is usually the proposed method, and a set of algorithms used in the empirical study. This set of comparisons is associated with a set or family of hypotheses, all of which are related to the control method. Any of the post-hoc tests is suitable for application to non-parametric tests working over a family of hypotheses.

The test statistic for comparing the i-th algorithm and j-th algorithm depends on the main non-parametric procedure used. In this case, it depends on the Friedman Aligned Ranks test:

Since the set of related rankings is converted to absolute rankings, the expression for computing the test statistic in Friedman Aligned Ranks is the same as that used by the Kruskal–Wallis test [72,74]

$$z=(\hat{R}_i-\hat{R}_j)\Big/\sqrt{\frac{k(n+1)}{6}}, \tag{18}$$

where $\hat{R}_i, \hat{R}_j$ are the average rankings by Friedman Aligned Ranks of the algorithms compared.

In statistical hypothesis testing, the p-value is the probability of obtaining a result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. It is a useful and interesting datum for many consumers of statistical analysis. A p-value provides information about whether a statistical hypothesis test is significant or not, and it also indicates something about "how significant" the result is: the smaller the p-value, the stronger the evidence against the null hypothesis. Most importantly, it does this without committing to a particular level of significance.

When a p-value is considered in a multiple comparison, it reflects the probability error of a certain comparison, but it does not take into account the remaining comparisons belonging to the family. If one is comparing k algorithms and in each comparison the level of significance is α, then in a single comparison the probability of not making a Type I error is $(1-\alpha)$, then the probability of not making a Type I error in the $k-1$ comparison is $(1-\alpha)^{(k-1)}$. Then the probability of making one or more Type I error is $1-(1-\alpha)^{(k-1)}$. For instance, if $\alpha=0.05$ and $k=10$ this is 0.37, which is rather high.

One way to solve this problem is to report adjusted p-values (APVs) which take into account that multiple tests are conducted. An APV can be compared directly with any chosen significance level α. We recommend the use of APVs due to the fact that they provide more information in a statistical analysis.

The z value in all cases is used to find the corresponding probability (p-value) from the table of normal distribution $N(0,1)$, which is then compared with an appropriate level of significance α [66, Table A1]. The post-hoc tests differ in the way they adjust the value of α to compensate for multiple comparisons.

Next, we will define the Holm procedure and we will explain how to compute the APVs. The notation used in the computation of the APVs is as follows:

- Indexes i and j each correspond to a concrete comparison or hypothesis in the family of hypotheses, according to an incremental order of their p-values. Index i always refers to the hypothesis in question whose APV is being computed and index j refers to another hypothesis in the family.
- p_j is the p-value obtained for the j-th hypothesis.
- k is the number of classifiers being compared.

The Holm procedure adjusts the value of α in a step-down manner. Let $p_1, p_2,\ldots,p_{k-1}$ be the ordered p-values (smallest to largest), so that $p_1 \le p_2 \le \cdots \le p_{k-1}$, and $H_1, H_2,\ldots,H_{k-1}$ be the corresponding hypotheses. The Holm procedure rejects H_1 to H_{i-1} if i is the smallest integer so that $p_i > \alpha/(k-i)$. Holm's step-down procedure starts with the most significant p-value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we are allowed to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well.

Holm APV_i: $\min\{v;1\}$, where $v=\max\{(k-j)p_j : 1\le j\le i\}$.

References

[1] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.
[2] A.N. Papadopoulos, Y. Manolopoulos, Nearest Neighbor Search: A Database Perspective, Springer, 2004.
[3] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, 2007.
[4] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (1991) 37–66.
[5] E.K. Garcia, S. Feldman, M.R. Gupta, S. Srivastava, Completely lazy learning, IEEE Transactions on Knowledge and Data Engineering 22 (9) (2010) 1274–1285.
[6] X. Wu, V. Kumar (Eds.), The Top Ten Algorithms in Data Mining. Chapman & Hall/CRC Data Mining and Knowledge Discovery, 2009.
[7] B.M. Steele, Exact bootstrap k-nearest neighbor learners, Machine Learning 74 (3) (2009) 235–255.
[8] P. Chaudhuri, A.K. Ghosh, H. Oja, Classification based on hybridization of parametric and nonparametric classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (7) (2009) 1153–1164.
[9] Y.-C. Liaw, M.-L. Leou, C.-M. Wu, Fast exact k nearest neighbors search using an orthogonal search tree, Pattern Recognition 43 (6) (2010) 2351–2358.
[10] J. Derrac, S. García, F. Herrera, IFS-CoCo: instance and feature selection based on cooperative coevolution with nearest neighbor rule, Pattern Recognition 43 (6) (2010) 2082–2105.
[11] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 6 (1997) 1–34.
[12] R. Paredes, E. Vidal, Learning weighted metrics to minimize nearest-neighbor classification error, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (7) (2006) 1100–1110.

[13] M.Z. Jahromi, E. Parvinnia, R. John, A method of learning weighted similarity function to improve the performance of nearest neighbor, Information Sciences 179 (17) (2009) 2964–2973.
[14] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research 10 (2009) 207–244.
[15] Y. Chen, E.K. Garcia, M.R. Gupta, A. Rahimi, L. Cazzanti, Similarity-based classification: concepts and algorithms, Journal of Machine Learning Research 10 (2009) 747–776.
[16] D. Pyle, Data Preparation for Data Mining, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, 1999.
[17] H. Liu, H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers, 2001.
[18] Y. Li, B.-L. Lu, Feature selection based on loss-margin of nearest neighbor classification, Pattern Recognition 42 (9) (2009) 1914–1921.
[19] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (3) (2000) 257–286.
[20] H.A. Fayed, S.R. Hashem, A.F. Atiya, Self-generating prototypes for pattern classification, Pattern Recognition 40 (5) (2007) 1498–1509.
[21] P.E. Hart, The condensed nearest neighbor rule, IEEE Transactions on Information Theory 18 (1968) 515–516.
[22] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on System, Man and Cybernetics 2 (3) (1972) 408–421.
[23] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining and Knowledge Discovery 6 (2) (2002) 153–172.
[24] E. Marchiori, Hit miss networks with applications to instance selection, Journal of Machine Learning Research 9 (2008) 997–1017.
[25] H.A. Fayed, A.F. Atiya, A novel template reduction approach for the k-nearest neighbor method, IEEE Transactions on Neural Networks 20 (5) (2009) 890–896.
[26] E. Marchiori, Class conditional nearest neighbor for large margin instance selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2) (2010) 364–370.
[27] W. Lam, C.K. Keung, D. Liu, Discovering useful concept prototypes for classification based on filtering and abstraction, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (8) (2002) 1075–1090.
[28] C.-L. Chang, Finding prototypes for nearest neighbor classifiers, IEEE Transactions on Computers 23 (11) (1974) 1179–1184.
[29] T. Kohonen, The self organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.
[30] C.H. Chen, A. Jóźwik, A sample set condensation algorithm for the class sensitive artificial neural network, Pattern Recognition Letters 17 (8) (1996) 819–823.
[31] S.W. Kim, J. Oomenn, A brief taxonomy and ranking of creative prototype reduction schemes, Pattern Analysis and Applications 6 (2003) 232–244.
[32] 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.
[33] J.C. Bezdek, L.I. Kuncheva, Nearest prototype classifier designs: an experimental study, International Journal of Intelligent Systems 16 (2001) 1445–1473.
[34] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer-Verlag, Berlin, 2003.
[35] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer-Verlag, Berlin, 2002.
[36] G.L. Pappa, A.A. Freitas, Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach, Natural Computing, Springer, 2009.
[37] J.-R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Transactions on Evolutionary Computation 7 (6) (2003) 561–575.
[38] S. García, J.R. Cano, F. Herrera, A memetic algorithm for evolutionary prototype selection: a scaling up approach, Pattern Recognition 41 (8) (2008) 2693–2709.
[39] L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing 72 (4–6) (2008) 1092–1097.
[40] A. Cervantes, I.M. Galván, P. Isasi, AMPSO: a new particle swarm method for nearest neighborhood classification, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 39 (5) (2009) 1082–1091.
[41] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE Transactions on Evolutionary Computation 9 (5) (2005) 474–488.
[42] F. Fernández, P. Isasi, Evolutionary design of nearest prototype classifiers, Journal of Heuristics 10 (4) (2004) 431–454.
[43] U. Garain, Prototype reduction using an artificial immune model, Pattern Analysis and Applications 11 (3–4) (2008) 353–363.
[44] J. Kennedy, R. Eberhart, Learning representative exemplars of concepts: an initial case study, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[45] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (1) (2007) 33–57.
[46] R. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (10) (1997) 341–359.
[47] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution A Practical Approach to Global Optimization, Natural Computing Series, 2005.
[48] I. Triguero, S. García, F. Herrera, A preliminary study on the use of differential evolution for adjusting the position of examples in nearest neighbor classification, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2010, pp. 630–637.
[49] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[50] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition-based differential evolution, IEEE Transaction on Evolutionary Computation 12 (1) (2008) 64–79.
[51] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.
[52] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.
[53] F. Neri, V. Tirronen, Scale factor local search in differential evolution, Memetic Computing 1 (2) (2009) 153–171.
[54] J.S. Sánchez, High training set size reduction by space partitioning and prototype abstraction, Pattern Recognition 37 (7) (2004) 1561–1564.
[55] H.A. David, H.N. Nagaraja, Order Statistics, third ed., Wiley, 2003.
[56] T.J. Rothenberg, F.M. Fisher, C.B. Tilanus, A note on estimation from a cauchy sample, Journal of the American Statistical Association 59 (306) (1966) 460–463.
[57] E. Alpaydin, Introduction to Machine Learning, second ed., MIT Press, Cambridge, MA, 2010.
[58] I.H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, second ed., Morgan Kaufmann, San Francisco, 2005.
[59] A. Ben-David, A lot of randomness is hiding in accuracy, Engineering Applications of Artificial Intelligence 20 (2007) 875–885.
[60] A. Asuncion, D. Newman, UCI machine learning repository, 2007, URL: ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.
[61] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318.
[62] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.
[63] R. Nisbet, J. Elder, G. Miner, Handbook of Statistical Analysis and Data Mining Applications, Elsevier, 2009.
[64] L.N. de Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer, 2002.
[65] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics—based machine learning: accuracy and interpretability, Soft Computing 13 (10) (2009) 959–977.
[66] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, fourth ed., Chapman & Hall/CRC, 2006.
[67] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[68] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[69] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (2010) 2044–2064.
[70] J. Hodges, E. Lehmann, Ranks methods for combination of independent experiments in analysis of variance, Annals of Mathematical Statistics 33 (1962) 482–497.
[71] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.
[72] W.H. Kruskal, W.A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American Statistical Association 47 (1952) 583–621.
[73] M. Abramowitz, Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables, Dover Publications, 1974.
[74] W.W. Daniel, Applied Nonparametric Statistics, Duxbury Thomson Learning, 1990.

Isaac Triguero Velázquez received the M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2009.
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 and evolutionary algorithms.

Salvador García López received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively.
He is currently an Assistant Professor in the Department of Computer Science, University of Jaén, Jaén, Spain. His research interests include data mining, data reduction, data complexity, imbalanced learning, statistical inference and evolutionary algorithms.

Francisco Herrera Triguero 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 150 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 five international books and co-edited 20 special issues in international journals on different Soft Computing topics. He acts as associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Mathware and Soft Computing, Advances in Fuzzy Systems, Advances in Computational Sciences and Technology, and International Journal of Applied Metaheuristics Computing. 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 several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation.

His current research interests include computing with words and decision making, data mining, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.