EMOPG+FS: Evolutionary Multi-Objective Prototype Generation and Feature Selection

Alejandro Rosales-Pérez, Jesus A. Gonzalez, Carlos A. Coello Coello, Carlos A. Reyes-Garcia, and Hugo Jair Escalante

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Tonantzintla, Puebla, Mexico
[2] CINVESTAV-IPN, Computer Science Department
Mexico City, Mexico

Abstract. k-NN is one of the most popular and effective classifiers nowadays. However, it has some limitations that overcome its applicability in large scale scenarios: basically, it requires storing the whole training set, and it computes distances of a test sample with the training data set. These limitations have been traditionally alleviated with data reduction techniques. This paper introduces a multi-objective evolutionary approach for data reduction. Our method simultaneously generates prototypes and selects features for k-NN classifiers. Contrary to most of the existing approaches, our method treats the problem with multi-objective evolutionary optimizers. We show the effectiveness of our proposal in benchmark data and compare its performance with state of the art techniques.

1 Introduction

Classification is a very common task in pattern recognition. The goal is to build a classification model that learns to map instances to a set of predefined labels. Nowadays, there is a large number of pattern classification methods, being the k-nearest neighbor (k-NN) classifier one of the most well-known [19]. Its popularity relies on its simplicity and good performance. k-NN belongs to the lazy learning family, meaning that this method does not have a training phase. Instead, unknown samples are classified based on the labels assigned to their most similar training samples. Hence, the standard k-NN requires to store in memory the entire training set, as well as to perform as many distance/similarity estimations as samples are available in the training set when classifying a single test pattern. These are major concerns when using the k-NN classifier on large data sets.

A number of studies have been proposed in order to overcome the aforementioned shortcomings. These studies aim at reducing the number of instances of the training set, while trying to preserve a good classification performance. In the literature, we can find two main approaches for instance reduction: prototype selection (PS) methods [10, 13], which attempt

to select a representative subset of samples from the training set, and prototype generation (PG) methods [17], whose goal is to generate a small set of artificial prototypes to replace the original training set. Both approaches have benefits and limitations, although an important advantage of PG methods is that they are not limited to select samples from the training set only, but they can also modify the values of these samples, resulting in a change of their position in the multi-dimensional feature space. Therefore, PG methods subsume PS techniques, raising our particular interest on their study. Another aspect that can have an impact in both, classification performance and the computational cost of k-NN is the dimensionality of the problem, since the higher the number of features the larger the cost of the similarity estimation. Furthermore, some features can be redundant or irrelevant and it is often useful to eliminate them. PG and feature selection procedures are usually applied separately. However, we believe that the joint reduction of instances and features is a promising solution for k-NN's limitations.

In recent years, the interest of using techniques based on bio-inspired optimization for approaching the PG problem has significantly grown [1,6–8,12,16,17]. These methods try to optimize a criterion related to the classification performance of a k-NN classifier that uses the prototypes. There are studies that also consider a reduction criterion in a single aggregated objective function, aiming to find effective prototypes with maximum reduction. However, there are two main issues when using an aggregated objective function. The first one is related to the determination of an appropriate preference for each objective to embed both criteria in a single formula, which could be non trivial. The second one is that a single solution can be obtained from this approach (the one that optimizes the aggregated criterion), which could result in little insight gained about the problem. The latter is an important issue when objectives are in conflict, i.e., maximizing accuracy may decrease the reduction performance, and viceversa. Therefore, we believe that by explicitly optimizing three objectives: classification performance, instance and feature reduction criteria, simultaneously in a multi-objective optimization approach, it is possible to obtain a number of solutions which represent acceptable trade-offs among the objectives, which could enable the user to make a better decision when facing a particular classification problem.

We propose EMOPG+FS, an evolutionary multi-objective approach that seeks to reduce the number of training samples, through the generation of prototypes, and the number of features, by means of selecting a subset of them, while preserving a good classification performance for k-NN. EMOPG+FS adopts a positioning adjustment approach for generating the prototypes, i.e., each pattern is represented as a point in the feature space, whose position

is modified by an optimization process. To this end, we used PAES (Pareto Archived Evolution Strategy) [11] as our multi-objective optimization technique. Our method was initially introduced in [14]. Here, we extend the previous work by explaining in depth our proposed algorithm and performing an analysis of its behavior as well as an experimental assessment of the method using a suite of benchmark data sets. Experimental results provide evidence of the suitability of using a multi-objective approach for dealing with the PG problem.

The remainder of this paper is organized as follows. Section 2 briefly reviews the most relevant previous related work on PG, focusing on evolutionary computation approaches. Section 3 explains in detail the formulation of the PG problem as a multi-objective optimization one and introduces our proposed approach. Section 4 reports experimental results that provide an experimental validation of the suitability of our proposal. Finally, Section 5 presents our conclusions, and outlines some possible paths for future work.

2 Related Work

A wide variety of PG methods have been proposed so far, see e.g. [17] for a compendium. However, in recent studies, PG methods based on bio-inspired optimization have reported better results than alternative techniques [1, 6–9, 12, 16, 17]. These methods start from a set of solutions (sets of prototypes) and modify them according to certain operators, through an iterative search procedure that aims to optimize a criterion related to the classification performance of prototypes.

A PG method based on particle swarm optimization (PSO) was proposed in [12]. A standard PSO method was designed to minimize the classification error in the training set. The method is run for several times in order to obtain varied solutions. When classifying a new object the outputs of the prototypes set are combined via voting. Another variant of PSO, adaptive Michigan PSO (AMPSO), has also been used for generating prototypes. In AMPSO each swarm particle is associated to a prototype in such a way that the whole population is the set of prototypes to optimize [1]. The ENPC (Evolutionary Design of NN classifiers) method was proposed in [8], ENPC is an evolutionary algorithm that starts from a single individual that is evolved by applying a variety of operators to combine and split prototypes. This method is able to automatically determine the number of prototypes and requires little information from the user. In [6], genetic programming is used to generate prototypes: a prototype is encoded as a tree combining training instances via arithmetic operators, and a standard genetic program is implemented with the aim of maximizing classification per-

formance of k-NN, when using the prototypes. In [16] a hybrid method based on differential evolution was proposed in which, after applying a PS method, the positioning of prototypes is updated with differential evolution. Finally, in [7] a multi-objective genetic algorithm is proposed for PG. Although being effective, this method does not perform any feature reduction step.

The above methods have reported acceptable results (see, e.g., [17]). However, they treat the classification-performance and reduction objectives as a single (aggregated) objective. As mentioned above, this form of optimization has drawbacks that may lead to solutions that do not offer a good tradeoff among the individual objectives. Furthermore, in most PG studies, the feature selection process has been disregarded, in spite of the fact that reducing the data dimensionality directly leads to important savings in computational cost. This paper proposes a multi-objective optimization method, EMOPG-FS, that treats the objectives separately, and incorporates a feature reduction process in addition to the PG mechanism.

3 EMOPG+FS: Evolutionary Multi-Objective Prototype Generation and Feature Selection

In this section, we describe the proposed multi-objective optimization approach to the PG problem. We initially provide a short introduction to multi-objective optimization followed by the description of the proposed approach.

3.1 Evolutionary Multi-Objective Optimization

A general MOP can be formulated as follows:

$$\begin{aligned} &\text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_l(\mathbf{x})]^T \\ &\text{subject to } \mathbf{x} \in \mathcal{X} \end{aligned} \tag{1}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ is a vector of decision variables, $f_i(\mathbf{x})$, $i = 1, \ldots, l$, are the $l-$objective functions, and $\mathcal{X}$ is the set of feasible solutions. In a MOP, the objectives can be in conflict causing that there does not exist a single solution that minimizes simultaneously all of the objectives. Hence, the notion of optimum in MOPs consists of finding solutions that provide a good trade-off among the objectives. Pareto optimality provides a framework to determine such trade-offs. We say that a solution $\mathbf{x}^1$ dominates a solution $\mathbf{x}^2$ (denoted by

$\mathbf{x}^1 \preceq \mathbf{x}^2$) if and only if $\mathbf{x}^1$ is not worse than $\mathbf{x}^2$ in any objective, and there exists at least one objective for which is better, i.e.:

$$\forall i : f_i\left(\mathbf{x}^1\right) \leq f_i\left(\mathbf{x}^2\right) \wedge \exists i : f_i\left(\mathbf{x}^1\right) < f_i\left(\mathbf{x}^2\right) \tag{2}$$

A solution $\mathbf{x}^*$ is a Pareto optimal solution if there does not exist another solution $x' \in \mathcal{X}$ such that $\mathbf{x}' \preceq \mathbf{x}^*$. The set of all Pareto optimal solutions is known as the Pareto optimal set, and the image of this in objective space is referred to as the Pareto Front.

Evolutionary algorithms are stochastic search techniques which mimic the principles of Darwin's evolutionary theory. These algorithms are well-suited for solving multi-objective problems (MOPs), due to the fact that they work with a population of solutions, allowing them to obtain a widespread set of non-dominated solutions in a single run. Furthermore, they are less susceptible to the shape and continuity of the Pareto front than mathematical programming techniques [2, 3]. From the seminal work of Schaffer [15] in 1985, the interest on using evolutionary algorithms for handling multi-objective problems has significantly increased. Among the most well known multi-objective evolutionary algorithms (MOEA), we can find SPEA2 [22], PAES [11], NSGA-II [4], and MOEA/D [21]. A comprehensive review of MOEAs can be found in [2, 3].

In this work, we use (1+1)-PAES [11] for solving the prototype generation problem through an adjustment position approach. PAES can be considered the simplest evolutionary multi-objective algorithm reported in the state of the art, and we describe it in Algorithm 1.

PAES starts by creating an initial individual, who serves as a parent to create new solutions. The next step is to create an offspring from the parent individual, which is achieved by applying a mutation operator over the parent. After that, PAES compares the parent and the child solution in order to determine if the child dominates the parent or viceversa or whether neither the child dominates the parent nor the parent dominates the child. It also determines whether the child should be included into the external archive and what solution should serve as a parent for the next generation. For doing this, PAES takes into consideration whether a solution is located in a crowded region of the feature space or not. At each iteration, PAES stores the non-dominated solutions found so far during the search in an external archive. A detailed description of PAES is out of the scope of this paper, but interested readers are referred to [11]. The remainder of this section explains the application of this MOEA for handling the adjustment of the positions of the initial set of prototypes and performing a features selection step.

Algorithm 1 PAES [11]

Require: $\mathbf{f}(\mathbf{x})$: fitness functions

Ensure: A set of non-dominated solutions

Create an empty external archive A
Create an initial individual p_0
Add p_0 to external archive $A : A = A \cup p_0$
while stopping criterion is not satisfied **do**
    Mutate p_t to create c_t
    if $p_t \preceq c_t$ **then**
        Discard c_t
    else
        if $c_t \preceq p_t$ **then**
            Replace p_t with c_t, and add c_t to archive $A : A = A \cup c_t$
        else
            if $\exists a \in A \mid a \preceq c_t$ **then**
                Discard c_t
            else
                Apply test (p_t, c_t, A) to determine who becomes the new current solution and whether to add c_t to A
            end if
        end if
    end if
end while

3.2 EMOPG+FS

In this work, the prototype generation (PG) task is simultaneously treated with feature selection. The aim is to reduce both instances and dimensionality while preserving the classification performance. We approach the problem with multi-objective optimization where the three considered objectives are: (1) the minimization of the 1-NN classification error in the training set when using the prototypes, (2) the reduction in the number of prototypes, and (3) the reduction in the number of features. Moreover, we constrain the set of feasible solutions ($\mathcal{X}$) to be formed by all possible sets of prototypes that have at least one prototype per class and are described by at least one feature.

Figure 1 shows the general scheme of EMOPG+FS. It starts weighting each instance. The weight of each instance is based on the distance to the boundary of each class in order to give preference to those instances that are more discriminative. Based on these weights, EMPOG+FS selects a subset of training samples, in which such samples are represented as points in a multi-dimensional space. Next, their positions are adjusted through an op-

timization process. Simultaneously to the position adjustment, EMPOG+FS also performs a selection of the features to be used in the classification task. The outcome of the multi-objective optimization process is a set of solutions that satisfy a good trade-off among the objectives. A single solution must be chosen from this set for its use; this is known as the decision making step. In this regard, we propose a strategy to select a single solution from the non dominated set of solutions. Algorithm 2 describes the EMPOG+FS approach, and each stage of this algorithm is detailed next.

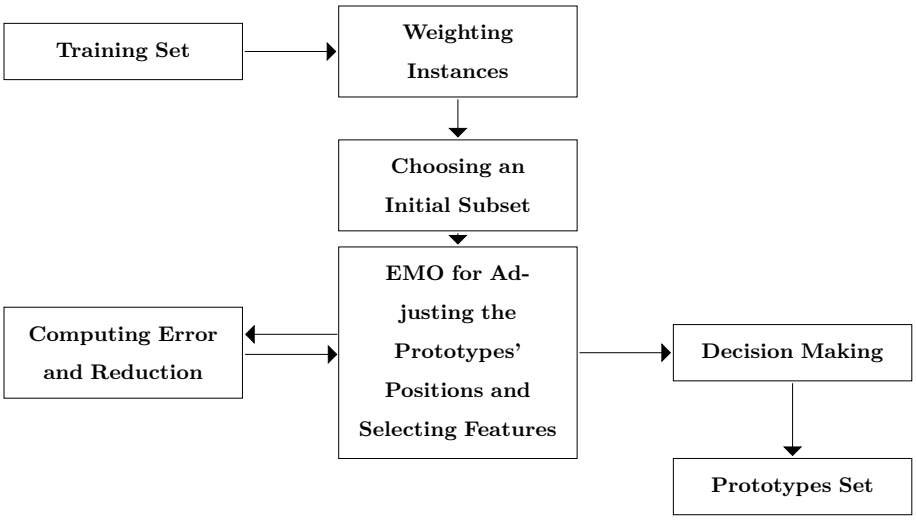


Fig. 1: General scheme of EMOPG+FS that shows its main stages.

Weighting Instances In the first step of the proposed algorithm we obtain a weight for each training instance related to its discriminative power. The goal is to use this weight for the selection of initial prototypes for the evolutionary algorithm. We consider an instance weighting scheme inspired by the criteria considered for condensation-based instance selection, see e.g., [18]. The main idea behind the weighting scheme is that instances that are closer to instances from different classes have a higher weight than those that are farther away, due to the fact that instances closer to the borders are expected to be the most difficult to classify and, therefore, give more information that allows us to discriminate among classes.

Algorithm 2 EMOPG+FS

Require: $\mathbf{X}$: training set, N: maximum number of prototypes, k : number of nearest neighbors, IC: number of instances competing in a tournament, MOEA's parameters

Ensure: A set of prototypes

1: Let $\mathbf{N} = [n_1, \ldots, n_m]$ be the number of instances for each class, such that $\sum_{i=1}^{m} n_i = N$ for m classes
{Weight each instance in the training set based on its k nearest neighbors from other classes}
2: **for** each instance $\mathbf{x}_i \in \mathbf{X}$ **do**
3: Find the k nearest neighbor from other classes
4: Compute the weight of the instance $\mathbf{x}_i$ using equation (3)
5: **end for**
{Construct an initial set of N prototypes giving preference to border instances}
6: **for** each class $c_i \in \mathbf{C}$ **do**
7: **while** the cardinality of prototypes from $c_i < n_i$ **do**
8: Randomly choose IC prototypes from $\mathbf{X}$ that belong to c_i
9: Add to the set of prototypes the prototype with the highest weight among the IC prototypes
10: **end while**
11: **end for**
12: Apply a multi-objective evolutionary algorithm for adjusting the positions of the prototypes
13: Select a single solution from the resulting non-dominated front based on some preference

This procedure is described in steps 2 to 5 in Algorithm 2. For each instance $\mathbf{x}_i$, we first determine $\mathcal{N}_{\mathbf{x}_i}^{\neq}$, the set of its k nearest neighbors of different classes. Next, we assign a weight to each instance $\mathbf{x}_i$ depending on its closeness with its neighbors of a different class as follows:

$$w(\mathbf{x}_i) = \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{N}_{\mathbf{x}_i}^{\neq}}^{k} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} \tag{3}$$

where $\|\mathbf{x}_i - \mathbf{x}_j\|$ is the Euclidean norm between $\mathbf{x}_i$ and $\mathbf{x}_j$. This weight is used to generate the initial prototypes as described below.

Constructing an Initial Set of Prototypes The second step of our proposal is to generate an initial set of prototypes through the selection of samples from the training set. As we previously stated, border instances could provide useful information that help to discriminate among classes. Therefore, the initial set of prototypes should give more preference to such instances. One way of doing this would consist in choosing the instances with the highest weights according to equation (3). Nonetheless, under this approach, the chosen prototypes could belong to a specific region of the feature space, reducing the diversity among the prototypes for the further optimization process. To overcome this shortcoming, we propose an initialization process inspired on the tournament selection from evolutionary algorithms.

The procedure for choosing an initial set of N prototypes is described in steps 6 to 11 in Algorithm 2. First, we determine the maximum number of prototypes for each class, which is found in a stratified fashion, looking to preserve, in the prototypes set, the original proportions of examples of each class as in the training set. After that, for each class, we select IC prototypes at random, and the one with the highest weight is added to the initial set of prototypes. This process is repeated until the maximum number of prototypes is reached. We should highlight that the tournament selection process is done with replacement. One should also note that by proceeding in this manner, we guarantee that there is at least one prototype per class in the set of initial prototypes. The motivation for using this sort of initialization instead of a random one, is to help the optimization process to converge faster.

3.3 Evolutionary Multi-Objective Optimization for Position Adjustment

Once the initial set of prototypes is selected, their position is adjusted. To do so, the multi-objective evolutionary algorithm PAES is used. In the rest of this section, we detail the formulation of the adjustment-position problem as one of multi-objective optimization which is solved using an evolutionary algorithm.

Representation The task of our PG method is to adjust the position of prototypes in the feature space as well as performing a selection of the most descriptive features for the problem at hand. To achieve this task, the prototypes are encoded in an $N \times d + d$ dimensional vector, where N is the maximum allowable number of prototypes and d is the dimensionality of each prototype in feature space. In addition to the position of the prototypes in feature space, and for the sake of reducing the initial number of prototypes as much as possible, the proposed encoding also considers a mechanism that allows selecting candidate prototypes. With this in mind, each potential solution to the problem (i.e., a set of prototypes) is represented in an $N \times (d + 1) + d$ dimensional vector as follows:

$$\mathbf{x}^{(i)} = \left[g_1^1, \ldots, g_1^d, \ldots, g_N^1, \ldots, g_N^d, b_1, \ldots, b_N, b_{N+1}, \ldots, b_{N+d}\right] \tag{4}$$

where $g_i^j \in \mathbb{R}$ represents the j^{th} feature value of the i^{th} prototype, and $b_i \in \{0, 1\}$ is a variable that indicates whether the corresponding prototype/feature is considered as part of the solution or not.

Alternatively, one can see the above representation as a matrix, where each row represents a prototype, and each column represents the features that describe a particular prototype plus a binary value. One should note that the class label is not encoded in the adopted

representation. This is because the initial set of prototypes is chosen from the training set. Therefore, each sample has a class label, which is defined *a priori*, and remains unchanged during the evolutionary search.

Evolutionary Operators (1+1)-PAES uses the mutation operator as the only evolutionary operator for creating an offspring from a parent. In the literature, there are a number of mutation operators that allow us to deal either with real-numbers, integer representations, binary encoding, or any other sort of encoding (interested readers are referred to [20]). Nevertheless, EMOPG+FA adopts a mixed-encoding representation that involves two types of variables: a real part and a binary part. One could obviously use a standard mutation for real-numbers encoding and after performing mutation, such variables that correspond to the binary part can be rounded off. This could be a naive fashion for handling, in a single and well-known operator, the mixed-encoding problem. However, in this case, small changes made to binary variables by the operator may be lost after performing the rounding-off process. To overcome this limitation, we propose to mutate both real and binary variables independently. Therefore, the individual is decomposed into two parts: the real part and the binary part. For each part, an *ad-hoc* mutation operator is applied. For the real-numbers part, we used polynomial-based mutation [3], while bit-flip mutation [20] was adopted for the binary part.

Fitness Functions The next step is to determine how good a solution is for the problem at hand and the fitness function is in charge of this. In order to do so we need to assess the solution with the considered optimization criteria. As we previously stated, we are interested not only in the classification performance of the generated prototype, but also in the reduction attained both in the prototype set size and in the feature set size. Therefore, these three criteria are the fitness functions of our multi-objective prototype generation and feature selection formulation. The first objective is assessed through a fitness function, f_1, that accounts for the error incurred by the prototypes when used with a 1-NN rule to classify the training set. The second objective, f_2, measures the relative samples reduction rate attained by a specific individual. Finally, the third objective, f_3, counts the number of features required to describe the samples in the set of prototypes. Besides the definition of the objective function, we should also recall that the set of prototypes must have at least one prototype for each class and they must be described by at least one feature. These constraints should be taken into consideration during the evaluation of the objective functions. We handle this in a straightforward fashion by adopting a penalty function approach, i.e., we penalize the

solutions that violate any of these constraints. The fitness functions for our problem can be stated as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= \frac{1}{P}\sum_{i=1}^{P}\mathcal{L}(y_i, y_i^*) + v(\mathbf{x}) \\ f_2(\mathbf{x}) &= \frac{\sum_{i=1}^{N} p_i}{N} + v(\mathbf{x}) \\ f_3(\mathbf{x}) &= \sum_{i=1}^{d} f_i + v(\mathbf{x}) \end{aligned} \tag{5}$$

where P is the number of samples in the training set, y_i is the class label, y_i^* is the class predicted by the set of prototypes, $\mathcal{L}(y_i, y_i^*)$ is a suitable loss function, $\sum_{i=1}^{N} p_i$ is the total number of prototypes chosen for a particular individual, N is the (desired) maximum number of prototypes, $\sum_{i=1}^{d} f_i$ is the number of features required to describe the prototypes, and $v(\mathbf{x})$ is a function that indicates the number of classes that are not represented by at least one prototype and described by at least one feature. The 0/1 loss function was used for our purposes, due to the fact that it is well suited for classification tasks. This loss function is defined as:

$$\mathcal{L}(y_i, y_i^*) = \begin{cases} 1 & \text{if } y_i^* \neq y_i \\ 0 & \text{if } y_i^* = y_i \end{cases} \tag{6}$$

Thus, the goal of PAES is to search the space of prototypes aiming to simultaneously optimize the three aforementioned criteria. At the end of the search stage, PAES returns the set of non-dominated solutions found. The next section describes our approach to select a single set of prototypes from the set of solutions obtained by PAES.

3.4 Selecting a Single Solution from the Non-dominated Set

By working with multi-objective optimization problems, one normally expects that more than one solution will be needed to satisfy the trade-offs among the objectives. Hence, the outcome of a MOEA is a set of non-dominated solutions, which is expected to be an approximation of the true Pareto optimal set. In the absence of user preferences, all of them are equally acceptable solutions to the problem at hand. In our case, each of these non-dominated solutions represents a set of prototypes to be used as a reduced data set for the 1NN classifier. Thus, it does not make any sense working with all solutions for classifying a test pattern. Instead of that, one can try to choose a single solution to be used as the final set of prototypes for the classification task.

In order to choose a single solution, we first define what an ideal solution to the problem would be. We should recall the three criteria that we look for in EMOPG+FS. The first one is to reduce the error rate, the second one is to reduce the number of prototypes, and the third one is to reduce the number of features required to describe the prototypes. Taking this into consideration, an ideal set of prototypes should correctly classify each instance, i.e., it should not commit errors when classifying the samples. Moreover, a single prototype should appropriately represent its corresponding class. Thus, an ideal solution should have at most as many prototypes as classes the problem has. Finally, with respect to the third goal, an ideal solution should only require a single feature to perfectly discriminate among the classes. These aspects are what define our ideal solution or our ideal outcome from the prototype generation and feature selection approach. Here, we used our ideal solution to choose the most similar solution in the resulting Pareto set. In order to determine which is the most similar solution to the ideal one, a distance computation is performed between the ideal solution and each solution in the resulting Pareto set. The solution located at the minimum distance[3] is chosen to be used as the set of prototypes for classifying unseen test patterns using the 1-NN rule.

We used the Tchebycheff metric [3] as our distance measure between non-dominated solutions and z_{ideal}. Thus, the solution is chosen through the following expression:

$$S^* = \underset{\mathbf{x}}{\operatorname{argmin}} \left[\max \left\{\left|f_i\left(\mathbf{x}\right) - z_i\right|\right\}\right] \forall_{i \in \{1,2,3\}} \tag{7}$$

where $f_i\left(\mathbf{x}\right)$ is the i^{th} criterion considered in EMOPG+FS and $\mathbf{z} = [z_1, z_2, z_3]$ is our ideal solution. The following section reports the experiments that we performed in order to assess the performance of EMOPG.

4 Experiments and Results

This section describes the experimental study performed over a suite of benchmark data sets widely used for the evaluation of PG methods [17]. We present a statistical analysis of the results and compare the performance of our proposal with respect to several other methods from the state of the art.

[3] This allows us to choose the solution nearest to our ideal solution. If a designer, however, has a preference for one particular objective, he/she can pick up another solution from the non-dominated set.

4.1 Experimental Settings

For our experiments, we considered a suite of 59 data sets taken from the KEEL repository[4], which were also used in the comparative study of PG methods performed by Triguero et al. [17]. Table 1 shows some characteristics of these data sets. In [17], the authors divided the datasets according to the number of samples into two categories: in small data sets (less than 2000 samples) and large data sets (2000 and more samples). Moreover, each of these were previously partitioned into 10 training/test subsets by means of a 10 fold cross validation strategy. In k fold cross validation, the data set is divided into k disjoint subsets, which are used for training and testing. This procedure is repeated k times, and at the i^{th} iteration, the i^{th} subset is used as the test set and the rest are used as the training set.

Here, we performed experiments in order to evaluate the performance of EMOPG+FS over the suite of data sets. For this set of experiments, we applied our EMOPG+FS to each data set 10 times, each time for each of the training partitions, in order to generate a corresponding prototypes set, whose performance is assessed by using the corresponding test set.

For assessing the performance of the PG methods we considered two criteria: test-set accuracy and training-set reduction. We compared the experimental results obtained by our proposed method with those obtained by other evolutionary and non-evolutionary methods for PG and by the 1-NN classifier.

Regarding the parameters configuration used in our experiments, we fixed the maximum number of prototypes (N) to be a 5% of the training set size, the number of nearest neighbor (k) used to weight instances was fixed to 5, the number of instances competing in a tournament (IC) was set to 2, the distribution index for the crossover was set to 10, and the mutation rate was set to 0.06, the stopping criterion for PAES was to perform 20,000 fitness functions evaluations, and the external archive keeps at most 20 solutions. These parameters were empirically chosen by testing different combinations of such parameters (i.e., those providing the best performance were selected). The parameters from the comparative methods were those adopted in [17]. The number of fitness function evaluations performed by the comparative methods could differ from those performed in our proposal, but we adopted these parameters under the assumption that they were the ones that gave the best performance for each method.

[4] These data sets are available at http://sci2s.ugr.es/keel/datasets.php

Table 1: Description of the data sets used for the experimental study [17]. For each data set, we show the number of samples, the number of attributes, and the number of classes.

Data set	Samples	Attributes	Classes	Data set	Samples	Attributes	Classes
Abalone	4174	8	28	Mammographic	961	5	2
Appendicitis	106	7	2	Marketing	8993	13	9
Australian	690	14	2	Monks	432	6	2
Autos	205	25	6	Movements-libras	360	90	15
Balance	624	4	3	Newthyroid	215	5	3
Banana	5300	2	2	Nursey	12960	8	5
Bands	539	19	2	Pageblocks	5472	10	5
Breast-Cancer	286	9	2	Penbased	10992	16	10
Bupa	345	6	2	Phoneme	5404	5	2
Car	1728	6	4	Pima	768	8	2
Chess	3196	36	2	Ring	7400	20	2
Cleveland	297	13	5	Saheart	462	9	2
Coil2000	9822	85	2	Satimage	6435	36	7
Contraceptive	1473	9	3	Segment	2310	19	7
Crx	125	15	2	Sonar	208	60	2
Dermatology	366	33	6	Spambase	4597	57	2
Ecoli	336	7	8	Spectheart	267	44	2
Flare-Solar	1066	9	2	Splice	3190	60	3
German	1000	20	2	Tae	151	5	3
Glass	214	9	7	Texture	5500	40	11
Haberman	306	3	2	Thyroid	7200	21	3
Hayes-Roth	133	4	3	Tic-tac-toe	958	9	2
Heart	270	13	2	Titanic	2201	3	2
Hepatitis	155	19	2	Twonorm	7400	20	2
Housevotes	435	16	2	Vehicle	846	18	4
Iris	150	4	3	Vowel	990	13	11
Led7digit	500	7	10	Wine	178	13	3
Lymphography	148	18	2	Wisconsin	683	9	2
Magic	19020	10	2	Yeast	1484	8	10
Zoo	101	16	7				

4.2 Experimental Results

In this section, we present experimental results obtained by our proposal to show its feasibility for the PG problem. Table 2 shows the average and standard deviation of the results obtained by our proposal (EMOPG) and by the reference studies. This table shows separately the results obtained when considering: all the data sets (59 data sets), only small data sets (40

Table 2: Results obtained by EMOPG+FS in terms of classification performance and reduction rate averaged over different data sets for all, for the small and for the large data sets. It also shows the results obtained by other PG methods, see [17]. The best results are shown in **boldface**.

Method	**Accuracy**		
	All	Small	Large
1-NN	74.79 ± 18.48	72.45 ± 16.08	79.73 ± 22.24
AMPSO	70.66 ± 17.68	69.03 ± 15.92	74.10 ± 20.97
GENN	$\mathbf{77.47 \pm 17.71}$	$\mathbf{75.64 \pm 15.45}$	81.33 ± 21.70
LVQTC	70.05 ± 18.74	69.81 ± 17.44	70.56 ± 21.74
MSE	73.78 ± 17.64	72.37 ± 14.81	76.74 ± 22.69
PSCSA	66.90 ± 19.68	66.82 ± 18.74	67.07 ± 22.05
PSO	76.62 ± 16.39	75.01 ± 14.09	79.99 ± 20.44
EMOPG+FS	76.70 ± 16.89	74.26 ± 14.70	$\mathbf{81.82 \pm 20.24}$

data sets), and only large data sets (19 data sets). It also shows the results of other prototype generation methods reported in the state of the art.

From Table 2, we can see that GENN, PSO, and EMOPG+FS reached a slightly better accuracy-performance than the 1-NN classifier. Both in all and in small data sets, GENN had the best performance with respect to the reference methods. PSO had a performance similar to that of GENN in those data sets. Regarding large data sets, both GENN and EMOPG+FS had almost the same performance. On the other hand, PSCSA showed the worst performance among the considered methods for all cases.

Table 3 shows the results in terms of the reduction rates attained both in prototypes and features. Here, we can note that PSCSA is the best one for reducing the prototype set size. EMOPG+FS showed to be the second best for both all and the small data sets, and ranked fourth for the large data sets. On the other hand, GENN is the one that showed the worst rates with respect to the reduction in the prototype set size, which was the only one that attained a reduction rate lower than a 20%. We would like to remark that none of the comparative methods performs a dimensionality reduction step, while EMOPG+FS does. In this regard, EMOPG+FS is able to reduce the feature set size in almost a 50%; this represents computational-savings during the similarity computation phase.

In order to assess if there exists a statistically significant difference among the evaluated methods, we used the Friedman test. This test is conducted on GENN, LVQTC, PSCSA, PSO, 1-NN, and EMPOG+FS, due to the fact that they had a competitive performance

Table 3: Results obtained by EMOPG+FS in terms of the reduction rate in number of prototypes and number of features averaged over different data sets for all, for the small and for the large data sets. It also shows the results obtained by other PG methods, see [17]. The best results are shown in **boldface**.

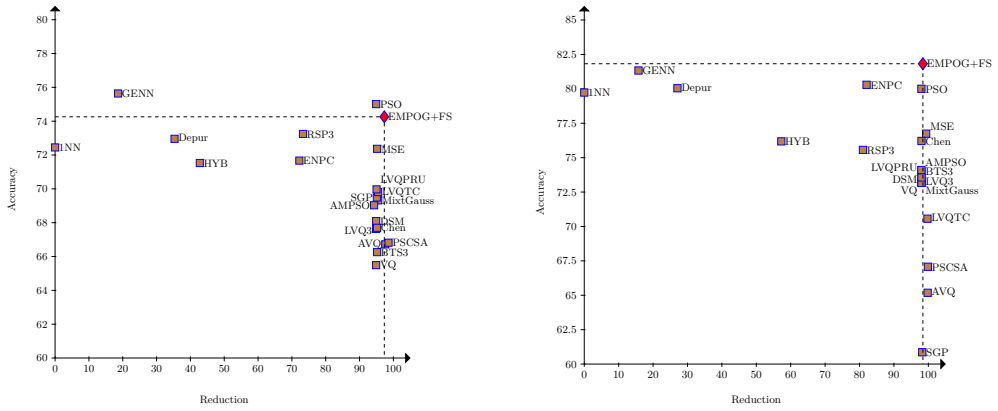
Method	**Reduction Rate**					
	Prototypes			**Features**		
	All	Small	Large	All	Small	Large
AMPSO	95.49 ± 1.86	94.39 ± 0.99	97.97 ± 0.09	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
GENN	17.70 ± 14.93	19.91 ± 14.48	15.76 ± 19.92	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
LVQTC	96.87 ± 3.13	95.61 ± 2.96	99.75 ± 0.16	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
MSE	96.54 ± 4.66	95.30 ± 5.10	99.36 ± 0.73	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
PSCSA	$\mathbf{99.00 \pm 1.37}$	$\mathbf{98.60 \pm 1.47}$	$\mathbf{99.88 \pm 0.17}$	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
PSO	95.90 ± 1.57	94.97 ± 0.83	97.99 ± 0.08	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00
EMOPG+FS	97.68 ± 1.35	97.32 ± 1.49	98.42 ± 0.46	$\mathbf{56.80 \pm 23.16}$	$\mathbf{60.47 \pm 19.14}$	$\mathbf{49.09 \pm 29.00}$

either on accuracy or on reduction. This test is suitable to compare multiple algorithms over multiple data sets [5]. We applied it with a 95% of confidence. Moreover, we should highlight that our goal was to compare the performance of our proposal (EMOPG+FS) with respect to the reference methods. Therefore, the Bonferroni-Dunn test was performed as a post-hoc test. We summarize the results obtained by these tests as follows:

- In terms of accuracy performance for all, small, and large data sets, there is not a statistically significant difference between EMOPG+FS, GENN, PSO, and 1-NN. EMOPG+FS performs significantly better than all of these methods in terms of prototype set size reduction.
- EMOPG+FS significantly outperforms PSCSA in all, small, and large data sets in terms of accuracy performance, but it is statistically worst in terms of the prototype set size reduction.

Figure 2 shows in a graphical fashion a comparison between EMOPG+FS and other evolutionary and non-evolutionary approaches for prototype generation. It shows, separately, the performance of each of these methods both in small and large data sets. It can be seen from Figure 2a that for small data sets, EMOPG+FS was outperformed by two methods in terms of test-accuracy. GENN was the best one, but the difference in accuracy between both methods is not statistically significant. Regarding large data sets, from Figure 2b one can note that EMOPG+FS outperforms most of the existing methods in terms of both reduction

and accuracy. This is interesting, since PG methods are normally applied to large data sets. The accuracy with respect to GENN is similar, but EMOPG+FS has a better reduction rate. In fact, from this figure it can be observed that the performance of EMOPG+FS is the closest to the top right corner, where hypothetically, the best method would be located.



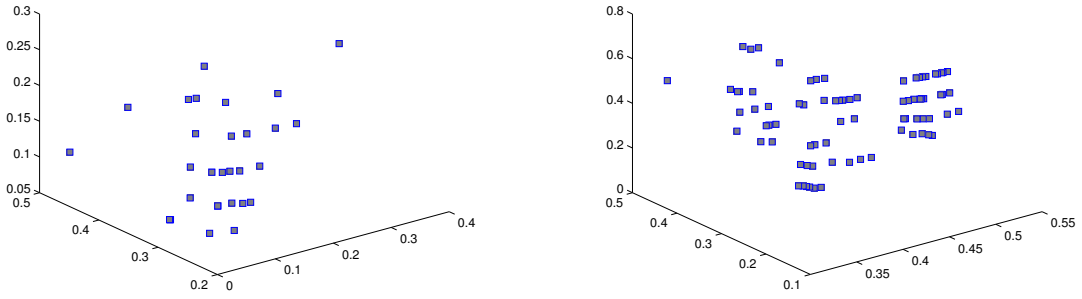
(a) Small (b) Large

Fig. 2: Trade-off between training-reduction and test-accuracy reached by several evolutionary and non-evolutionary methods for prototype generation and EMPOG+FS for both (a) small and (b) large data sets.

Finally, in Figure 3 we can observe an approximation of the Pareto front generated by EMOPG+FS. These figures show that there exists a trade-off among the considered objectives, yielding that there does not exist a single solution that would minimize all objectives simultaneously. The advantage of using this multi-objective approach is that it provides the user with more information about the problem in order to make a decision about which solution should be chosen for a given data set that satisfies his/her requirements.

4.3 Discussion

From the experiments, we can note that several prototypes generation methods seek to find a reduced set of prototypes while trying to preserve the classification performance. GENN is a prototype generation method which offers prototypes that preserve or even improve the 1-NN rule in accuracy rate in classification performance. Nevertheless, the reduction rate reached by this method (in terms of the generated prototype set size) is the worst one. This is particularly depicted as it is the only one with a reduction rate in prototypes set size



(a) German (b) Magic

Fig. 3: Non-dominated fronts generated by EMOPG+FS for two data sets in a particular trial. The left plot shows a small data set and the right one shows a large data set.

lower that a 20%, while most of the considered methods for the comparison are able to reach reduction rates above a 90%.

On the other hand, PSCSA is the best one when the reduction in the prototype set size is the most important issue. PSCSA outperforms the second best in more than 1% for all cases. This one percent can represent a considerable number of samples in data sets of large scale. However, as we can see, the accuracy performance of PSCSA is the worst, getting classification rates below a 70%, while the others, in most cases, attain rates above this 70%. Therefore, this shows the necessity of considering both criteria as equally important.

There exist other methods that offer a more balanced trade-off between accuracy and prototype set size reduction. Among these, we can find PSO, LVQTC, and EMOPG+FS. However, the main difference between EMOPG+FS and the others is that the former is not only limited to reduce the prototypes set size at the time that it tries to preserve the classification performance, but it also tries to reduce the dimensionality of the problem at hand. The latter could represent computational savings during the computation of the similarity between a test pattern and the prototypes. From the numerical results, we can note that EMOPG+FS is the one that offers the most balanced trade-off when the three criteria are taken into consideration without significant over-fitting.

5 Conclusions and Future Work

We presented EMOPG+FS, an evolutionary multi-objective approach for dealing jointly with prototype generation (PG) and feature selection problems, while keeping a good accuracy performance. Approaching the problem of generating a new set of prototypes with multi-objective evolutionary algorithms allows to find solutions that satisfy good trade-offs among the considered criteria. Experimental results showed that our proposal is able to obtain representative prototypes and to considerably reduce the dimensionality of the data set, without significantly degrading the performance of k-NN. Besides this, the performance of EMOPG+FS over different data sets from different domains gives evidence of the suitability of using it as a general method for this task.

Our future work involves extending our proposal to deal with user preferences during the optimization process. We would also like to evaluate the performance of EMOPG+FS using different values of k for the k-NN classifier. Our current experimental study has been carried out using a set of standard benchmark datasets, which were previously pre-processed. However, dataset from real world problems can have noise/missing datasets. Analyzing the behavior of EMOPG+FS on such dataset is also part of the future work. Moreover, studying the impact of the evolutionary parameters on the quality of the solutions generated by EMOPG+FS and testing EMOPG+FS on very large scale data sets are other potential paths for future research.

Acknowledgments

The third author acknowledges support from CONACyT project no. 221551.

References

1. Cervantes, A., Galvan, I.M., Isasi, P.: AMPSO: a new particle swarm method for nearest neighborhood classification. IEEE Trans. Sys. Man Cy. B 39(5), 1082–1091 (2009)
2. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary algorithms for solving multi-objective problems. Springer, US, second edn. (2007)
3. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley (2001)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
6. Escalante, H.J., Graff, M., Morales-Reyes, A.: Pggp: Prototype generation via genetic programming. Applied Soft Computing 40, 569–580 (2016)
7. Escalante, H.J., Marín-Castro, M., Morales-Reyes, A., Graff, M., Rosales-Pérez, A., y Gómez, M.M., Reyes, C.A., Gonzalez, J.A.: MOPG: A Multi-Objective Evolutionary Algorithm for Prototype Generation. Pattern Analysis and Applications (2016), (in press)

8. Fernandez, F., Isasi, P.: Evolutionary design of nearest prototype classifiers. J. Heuristics 10, 431–454 (2004)
9. Garain, U.: Prototype reduction using an artificial immune system. Pattern Anal. Appl. 11(3–4), 353–363 (2008)
10. García, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. 34(3), 417–435 (2012)
11. Knowles, J., Corne, D.: Approximating the nondominated front using the pareto archived evolution strategy. Evol. Comput. 8(2), 149–172 (2000)
12. Nanni, L., Lumini, A.: Particle swarm optimization for prototype reduction. Neurocomputing 72(4–6), 1092–1097 (2008)
13. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Martinez-Trinidad, J.F.: Prototype selection methods. Computación y Sistemas 13(4), 449–462 (2010)
14. Rosales-Pérez, A., Gonzalez, J., Coello-Coello, C., Reyes-Garcia, C., Escalante, H.: Evolutionary multi-objective approach for prototype generation and feature selection. In: Bayro-Corrochano, E., Hancock, E. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Lecture Notes in Computer Science, vol. 8827, pp. 424–431. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-12568-8_52
15. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms. pp. 93–100. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (1985)
16. Triguero, I., Garcia, S., Herrera, F.: Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recogn. 44, 901–916 (2011)
17. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man Cy. C 42(1), 86–100 (Jan 2012)
18. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. 38, 257–286 (2000)
19. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Sys. 14(1), 1–37 (2007)
20. Yu, X., Gen, M.: Introduction to Evolutionary Algorithms. Springer (2010)
21. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions onEvolutionary Computation 11(6), 712–731 (2007)
22. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., Tsahalis, D.T., Périaux, J., Papailiou, K.D., Fogarty, T. (eds.) Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems. pp. 95–100. International Center for Numerical Methods in Engineering (2001)