An Evolutionary Multi-Objective Approach for Prototype Generation

Alejandro Rosales-Pérez*, Hugo Jair Escalante*, Carlos A. Coello Coello†, Jesus A. Gonzalez*, and Carlos A. Reyes-Garcia*
* Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Tonantzintla, Puebla, 72840, Mexico
Email: {arosales, hugojair, jagonzalez, kargaxxi}@inaoep.mx
† CINVESTAV-IPN, Evolutionary Computation Group (EVOCINV),
Mexico City, 07360, Mexico
Email: ccoello@cs.cinvestav.mx

***Abstract*—k-NN is one of the most popular and effective models for pattern classification. However, it has two main drawbacks that hinder the application of this method for large data sets: (1) the whole training set has to be stored in memory, and (2) for classifying a test pattern it has to be compared to all other training instances. In order to overcome these shortcomings, prototype generation (PG) methods aim to reduce the size of the training set while maintaining or increasing the classification performance of *k*-NN. Accordingly, most PG methods aim to generate instances that try to maximize classification performance. Nevertheless, in most cases, the reduction objective is only implicitly optimized. This paper introduces EMOPG, a novel approach to PG based on multi-objective optimization that explicitly optimizes both objectives: accuracy and reduction. Under EMOPG, prototypes are initialized with a subset of training instances selected through a tournament, according to a weighting term. A multi-objective evolutionary algorithm, PAES (Pareto Archived Evolution Strategy), is implemented to adjust the position of the initial prototypes. The optimization process aims to simultaneously maximize the classification performance of prototypes while reducing the number of instances with respect to the training set. A strategy for selecting a single solution from the set of non-dominated solutions is proposed. We evaluate the performance of EMOPG using a suite of benchmark data sets and compare the performance of our proposal with respect to the one obtained by alternative techniques. Experimental results show that our proposed method offers a better trade-off between accuracy and reduction than other methods.**

I. Introduction

A vast number of pattern classification methods have been proposed so far, being the *k*-nearest neighbor (*k*-NN) classifier one of the most well-known [1]. Its popularity relies on its simplicity and good performance. The *k*-NN classifier belongs to the lazy learning family, meaning that no training phase is needed for this method. Instead, *k*-NN represents each training sample as a point in a multi-dimensional feature space, and new samples are classified based on the labels assigned to their closest training samples. In this sense, the standard *k*-NN requires the entire training set to be stored in memory, and it performs as many distance/similarity computations as samples are available in the training set for classifying a single test pattern. These are major concerns when using the *k*-NN classifier on large data sets, which are becoming ubiquitous nowadays.

To overcome the above mentioned shortcomings, a number of techniques have been proposed aiming to reduce the number of instances of the training set, while preserving a good classification performance. There are two main approaches for instance reduction: prototype selection methods [2], which attempt to select a representative subset of samples from the training set, and prototype generation (PG) methods [3], 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 subsume the prototype selection techniques. Hence, we focus on PG in this work.

Among the available methods for PG, techniques based on bio-inspired optimization have reported the best performance in recent years [3]–[8]. These methods aim at optimizing a criterion related to the classification performance of prototypes; in a few cases, a reduction term is also implicitly considered. Whereas satisfactory results have been reported with such methods, the optimization of a single objective may not be the best option for PG, as classification performance and training set reduction are two objectives that are in conflict with each other (i.e., maximizing accuracy may decrease the reduction performance, and viceversa). Therefore, methods that can explicitly deal with both objectives may obtain solutions that offer a better trade-off between classification and reduction performance.

This paper introduces EMOPG: an evolutionary multi-objective approach for the generation of prototypes. EMOPG aims at adjusting the positioning of prototypes in the input space by using PAES (Pareto Archived Evolution Strategy) [9]. A solution is initialized with a set of potentially good instances (prototypes). Then, PAES optimizes the positioning of these instances aiming to explicitly maximize classification and reduction perfor-

mance. A strategy for generating a single solution from the set of non-dominated solutions obtained by PAES is proposed. We report experimental results in a suite of data sets used for benchmarking PG methods and compare the performance of EMOPG to that of alternative techniques [3]. Experimental results reveal that EMOPG compares favorably with most PG methods proposed so far. In fact, our proposed method achieves the best trade-off in terms of reduction and accuracy.

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

II. Previous Related Work

Among the variety of PG methods proposed so far, recent methods based on evolutionary algorithms and related techniques have reported better results than alternative approaches [3]–[8], [10]. In these approaches, the PG method is treated as one of optimization, where the goal is to find (to optimize) the best set of prototypes for pattern classification with k-NN. These methods start from a set of solutions (prototypes) that are iteratively modified by applying ad-hoc operators with the goal of optimizing a criterion related to the classification performance of the prototypes.

A PG method based on particle swarm optimization (PSO) was proposed in [5], where the authors try to minimize the classification error in the training set. The method is run for several times in order to obtain varied solutions (sets of prototypes). When classifying a new object, the outputs of all of the set of prototypes are combined via voting, i.e., an instance is assigned to the most voted class. Another variant of PSO, called adaptive Michigan PSO (AMPSO), has also been used for the generation of prototypes. In AMPSO, each particle of the swarm is associated to a prototype in such a way that the whole population is the set of prototypes that are optimized [6]. Fernandez et al. proposed ENPC (Evolutionary Design of NN classifiers), which consists of an evolutionary algorithm that starts from a single individual that is evolved by applying a variety of operators that merge and split prototypes [8]. The method is able to automatically determine the number of prototypes and requires little information from the user. Escalante et al. proposed a PG method based on genetic programming (GPGP) [7]. The idea consists of exploiting a tree structure to combine instances using arithmetic operators. The objective function combines the accuracy and reduction criteria in a single formula.

Triguero et al. reported a comparative study among the above mentioned methods and several other techniques [3]. The most representative PG methods are considered in that study. Therefore, in Section IV we compare the performance of our proposed method with all of the methods considered in [3].

One should note that in most of the above reviewed works, a single objective (i.e., accuracy) is being optimized. Some methods combine both objectives into a single one or exploit the structure of the optimization strategy to incorporate one of the objectives. In this work, however, we explicitly aim to optimize both objectives by using a multi-objective evolutionary algorithm. Our hypothesis is that by explicitly modeling these aspects, we will be able to obtain solutions that offer a better trade-off between accuracy and reduction.

III. Evolutionary Multi-Objective Approach for Prototype Generation

In this section, we describe the proposed multi-objective optimization approach to the PG problem. First, we provide a brief introduction to multi-objective evolutionary optimization and then we introduce the proposed technique.

A. Evolutionary Multi-Objective Optimization

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 [11], [12].

A 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 the case that the objectives are in conflict, there does not exist a single solution that minimizes simultaneously all of the objectives. Therefore, the notion of optimum in MOPs differs from that in single-objective optimization, since in this case, the focus is on 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.:

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

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.

A large number of evolutionary algorithms for solving multi-objective problems have been proposed so far. Among these, we can find the NSGA-II [13], SPEA2 [14], PAES [9], etc. Interested readers are referred to [11], [12] for a comprehensive review of multi-objective evolutionary algorithms.

B. EMOPG: Evolutionary Multi-Objective Prototype Generation

In this work, the PG task is treated as a multi-objective optimization problem where two objectives are considered: (1) classification performance, via the minimization of the 1-NN classification error in the training set when using the prototypes; and (2) reduction performance, via minimization of the number of prototypes. Both objectives would correspond to $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, respectively, in equation (1). 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.

We adopted a position-adjustment approach for generating the prototypes starting from a subset of training samples [3]. The initial training samples are selected by considering a weight that accounts for the discrimination power of instances. Then, through an optimization process the positions of the initial prototypes (training samples) in the feature space are modified. The multi-objective optimization approach returns a set of solutions from which a single one must be selected. In this regard, we propose an effective strategy for selecting a single solution. Algorithm 1 describes the proposed approach, the rest of this section details the proposed approach.

1) Weighting Instances: The first step of the proposed algorithm is to obtain a weight for each training instance related to their discrimination 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., [15]. 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 closest to the borders are expected to be the most difficult to classify and, therefore, give more information that allows us to discriminate among classes.

The weighting procedure is described in the steps 2 to 5 in Algorithm 1. For each instance $\mathbf{x}_i$, we first determine $\mathcal{N}^{\neq}_{\mathbf{x}_i}$, 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 different class as follows:

Algorithm 1 EMOPG

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: Choose randomly IC prototypes from $\mathbf{X}$ that belong to c_i
9: Add to the set of prototype 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

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

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.

2) 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 the 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 1. First, we determine the maximum number of prototypes for each class, which is determined 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. It is worth indicating that the tournament selection is done with replacement.

One should 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.

C. Evolutionary Multi-Objective Optimization for Position Adjusting

In the problem that we faced, the positions of the prototypes are adjusted through an evolutionary process. Due to the way in which the set of prototypes is initialized, we expect that this set would be a good enough solution, which should be improve it by a local search engine that considers the two considered objectives in our formulation. To this aim, we used the (1+1)-PAES [9] for solving this problem. A description of PAES is presented in Algorithm 2. An initial individual is first created, who serves as a parent to create new solutions. Under the adopted approach, the initial set of prototypes (see steps 6 to 11 from Algorithm 1) is a potential solution to the problem at hand, and it is used as the initial individual for PAES. 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, a series of comparisons are performed in order to determine whether the child individual should be added to an external archive or not, and what solution should be the parent for the next generation. PAES stores the non-dominated solutions found so far during the search in an external archive. A detailed description of PAES is given in [9]. The remainder of this section explains the application of PAES for adjusting the positions of the initial set of prototypes.

Algorithm 2 PAES [9]

Require: $\mathbf{f}(\mathbf{x})$: fitness functions
Ensure: A set of non-dominated solutions
1: Create an empty external archive A
2: Create an initial individual p_0
3: Add p_0 to external archive $A : A = A \cup p_0$
4: **while** stopping criterion is not satisfied **do**
5: Mutate p_t to create c_t
6: **if** $p_t \preceq c_t$ **then**
7: Discard c_t
8: **else**
9: **if** $c_t \preceq p_t$ **then**
10: Replace p_t with c_t, and add c_t to archive $A : A = A \cup c_t$
11: **else**
12: **if** $\exists a \in A \mid a \preceq c_t$ **then**
13: Discard c_t
14: **else**
15: Apply test (p_t, c_t, A) to determine who becomes the new current solution and whether to add c_t to A
16: **end if**
17: **end if**
18: **end if**
19: **end while**

1) Representation: The task of our PG method is to adjust the position of prototypes in the feature space. To achieve this task, the prototypes are encoded in an $N \times d$ 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 for selecting from among the candidate prototypes. With this in mind, each potential solution to the problem (i.e., a set of prototypes) is represented in an $N \times (d+1)$ dimensional vector as follows:

$$\mathbf{x}^{(i)} = \left[f_1^1, \ldots, f_1^d, b_1, \ldots, f_N^1, \ldots, f_N^d, b_N\right] \qquad (4)$$

where $f_i^j \in \mathbb{R}$ 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 is considered 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.

2) Evolutionary Operators: In (1+1)-PAES, the mutation is the only evolutionary operator used for creating an offspring from a parent. There are a number of mutation operators for dealing either with a real-numbers or a binary encoding. Notwithstanding, the adopted representation is a mixed-encoding, involving both real and binary variables. Hence, one could use a standard mutation for a real-numbers encoding and round off the binary parts, but in this case, small changes made to the binary variables by the mutation 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 in 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 [12], and bit-flip mutation [16] was adopted for the binary part.

3) Fitness Functions: In order to evaluate how good an individual is, we need to assess it with the considered optimization criteria. We consider two objectives, related to classification and reduction performance of the prototypes. 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 is captured by a fitness function,

f_2, indicating the relative reduction rate attained by a specific individual. One should recall the constraint that the set of prototypes must have at least one prototype by each class. This constraint is handled in a straightforward fashion following a penalty function approach. The fitness functions for our problem can be stated as follows:

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

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 model, $\mathcal{L}(y_i, y_i^*)$ is a suitable loss function, $\sum_{i=1}^{N} b_i$ is the total number of prototypes chosen for a particular individual, N is the (desired) maximum number of prototypes, and $v(\mathbf{x})$ is a function that indicates the number of classes that are not represented by at least one prototype. 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 f_1 and f_2. PAES returns a set of non-dominated solutions found during the search. The next section describes our approach to select a single set of prototypes from the set of solutions obtained by PAES.

D. Selecting a Single Solution from the Non-dominated Set

PAES returns a set of non-dominated solutions, which is expected to be an approximation to 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.

In order to choose a single solution, we first define what an ideal solution to the problem would be. Recall that the first objective is to reduce the error rate and the second one is to reduce a relative number of prototypes. Hence, an ideal solution would not commit errors to classify the samples and it would have one prototype per class, i.e., $z_{ideal} = [0, {}^{m}/_{N}]$, for a problem with m classes. We adopted a compromise programming approach [17] to choose a solution. We chose this method because it allowed us to pick up a solution that is located at a minimum distance from a given reference point (the ideal point in our case[1]). As distance measure between non-dominated solutions and z_{ideal} we used the Tchebycheff metric [12]. Thus, the solution is chosen through the following expression:

$$S^* = \underset{\mathbf{x}}{\operatorname{argmin}} \left[\max\left\{f_1(\mathbf{x}), \left|f_2(\mathbf{x}) - {}^{m}/_{N}\right|\right\}\right] \tag{7}$$

The next section reports the experiments that we performed in order to assess the performance of EMOPG.

IV. 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 [3]. We present a statistical analysis of the results and compare the performance of our proposal with that of several other methods from the state of the art.

A. Experimental Settings

For our experiments, we used 59 data sets taken from the KEEL repository[2], which were also used in the comparative study of PG methods performed by Triguero et al. [3]. Table I shows some characteristics of these data sets. They are divided according to the number of samples, in small data sets (less than 2000 samples) and large data sets (2000 and more samples). Each of these data sets were previously partitioned into 10 training/test subsets by means of a 10 fold cross validation procedure. In k fold cross validation, the data set is divided in k disjoint subsets, which are used for training and testing. At each iteration, a subset is used as our test set and the rest as our training set. This procedure is repeated k times, until all of the subsets had been used for testing. Thus, for each data set, we applied our proposal (EMOPG) 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. This leads to a total of 590 experiments performed for PG.

For assessing the performance of the PG methods we consider the two widely used criteria: test-set accuracy and training-set reduction. We compare 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 neighbors (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. The parameters from the compared methods were

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

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

TABLE I: Description of the data sets used for the experimental study [3]. For each data set, we show the number of samples, the number of attributes, and the number of classes.

Data set	Samples	Attributes	Classes
Abalone	4174	8	28
Appendicitis	106	7	2
Australian	690	14	2
Autos	205	25	6
Balance	624	4	3
Banana	5300	2	2
Bands	539	19	2
Breast-Cancer	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	2
Magic	19020	10	2
Mammographic	961	5	2
Marketing	8993	13	9
Monks	432	6	2
Movements-libras	360	90	15
Newthyroid	215	5	3
Nursey	12960	8	5
Pageblocks	5472	10	5
Penbased	10992	16	10
Phoneme	5404	5	2
Pima	768	8	2
Ring	7400	20	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

those adopted in [3]. The number of fitness functions evaluation performed by the compared methods could differ than 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.

B. Experimental Results

In this section, we present experimental results obtained by our proposal to show its feasibility to the PG problem. Table II 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 (in terms of test-set accuracy and training-set reduction) when considering: all the data sets (59 data sets), only small data sets (40 data sets), and only large data sets (19 data sets).

From Table II, we can see that GENN, PSO, and EMOPG reached a slightly better accuracy-performance than the 1-NN classifier when all and the large data sets are taken into account. For small data sets, 1-NN showed slightly better performance than EMOPG. GENN further reached the best performance in terms of accuracy for both all and the small data sets. PSO had a performance similar to that of GENN in those data sets. With respect to the large data sets, the performance of GENN and EMOPG is almost the same. It is also remarkable that PSCSA showed the worst performance among the considered methods for all, the small, and the large data sets.

On the other hand, in terms of the reduction rates attained by each method, we can observe that PSCSA obtained the best training-set reduction rate in all cases. EMOPG was the second best method in terms of reduction for both, all and the small data sets, and the third one for the large data sets. Nonetheless, the reduction rates achieved by the three best methods on large data sets is virtually the same, getting rates above 99%. The worst method in terms of reduction was GENN[3]. In fact, GENN is the only method whose reduction rates were below 20%.

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, due to the fact that they had a competitive performance either on accuracy or on reduction. This test is suitable to compare multiple algorithms over multiple data sets [18]. We applied it with a 95% of confidence. Moreover, we should highlight that our goal was to compare the performance of our proposal (EMOPG) with respect to the reference methods. Therefore, the Bonferroni-Dunn test is performed as a post-hoc test. We summarize the results obtained by these tests as follows:

- In terms of test-set accuracy for all, the small and the large data sets, there does not exist a statistically significant difference between EMOPG, GENN, PSO, and 1-NN. EMOPG significantly outperforms all of these methods in terms of reduction.
- EMOPG significantly outperforms LVQTC and PSCSA in all and the large data sets, and it also

[3]One should note that 1-NN is not a PG method. For that reason, it was not taken into consideration for comparing the reduction rate.

TABLE II: Results obtained by EMOPG 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 [3]. The best results are shown in **boldface**.

Method	Accuracy			Reduction Rate		
	All	Small	Large	All	Small	Large
1-NN	74.794 ± 18.478	72.451 ± 16.078	79.728 ± 22.236	00.000 ± 0.000	00.000 ± 0.000	00.000 ± 0.000
AMPSO	70.662 ± 17.676	69.028 ± 15.915	74.103 ± 20.966	95.485 ± 1.861	94.388 ± 0.991	97.973 ± 0.090
GENN	$\mathbf{77.469 \pm 17.712}$	$\mathbf{75.637 \pm 15.447}$	81.327 ± 21.696	17.701 ± 14.926	19.910 ± 14.477	15.758 ± 19.923
LVQTC	70.048 ± 18.740	69.806 ± 17.436	70.558 ± 21.736	96.874 ± 3.125	95.608 ± 2.961	99.752 ± 0.160
MSE	73.776 ± 17.640	72.366 ± 14.809	76.744 ± 22.690	96.538 ± 4.658	95.299 ± 5.104	99.363 ± 0.732
PSCSA	66.904 ± 19.676	66.824 ± 18.742	67.074 ± 22.054	$\mathbf{99.001 \pm 1.365}$	$\mathbf{98.600 \pm 1.469}$	$\mathbf{99.879 \pm 0.169}$
PSO	76.617 ± 16.390	75.012 ± 14.088	79.996 ± 20.438	95.900 ± 1.569	94.974 ± 0.831	97.990 ± 0.083
EMOPG	74.824 ± 17.421	71.730 ± 15.386	$\mathbf{81.337 \pm 19.974}$	98.568 ± 1.312	98.112 ± 1.371	99.528 ± 0.195

significantly outperforms PSCSA in the small data sets with respect to accuracy-performance.

Overall, we can say that EMOPG offers a better trade-off between accuracy and reduction than the alternative methods considered here. GENN obtained very good results in terms of accuracy. However, it had the worst reduction rate. On the other hand, PSCSA was able to obtain better reduction rates than the other methods, but its performance on accuracy was the worst among the considered methods. EMOPG offered a more balanced trade-off between the accuracy/reduction objectives than any other of the considered methods. In fact, the difference in accuracy between GENN and EMOPG was not statistically significant, while the difference in reduction was. Notwithstanding that PSCSA clearly outperforms EMOPG in terms of reduction, the difference between both approaches is less than a 0.50% for all the data sets adopted in our study, and less than a 0.30% for the large data sets. Moreover, the accuracy performance of PSCSA was the worst, being outperformed even by the 1-NN classifier. These results are interesting, since EMOPG was able to reduce the number of samples in the training set, without significantly overfitting.

Figure 1 shows the non-dominated front generated by EMPOG for the Car data set. This non-dominated front is expected to be an approximation to the true Pareto front. From this figure, we can note that there is a trade-off between the reduction rate and the accuracy, such that by reducing the number of prototypes, it is expected that the error increases. From the shape of the front, we can clearly see that solutions are well distributed across the objective space and provide the decision maker with a variety of possibilities including highly accurate (and many prototypes) and very compact (but not so accurate) solutions.

Figure 2 graphically depicts the behavior of several evolutionary and non-evolutionary methods for PG (which were considered in the comparative study performed in [3]) and EMOPG with respect to the training-set reduction and test-accuracy. It can be seen from Figure 2a that for small data sets, EMOPG was outperformed by several methods in terms of test-accuracy. GENN was the best one, but as we previously stated, the difference in accuracy

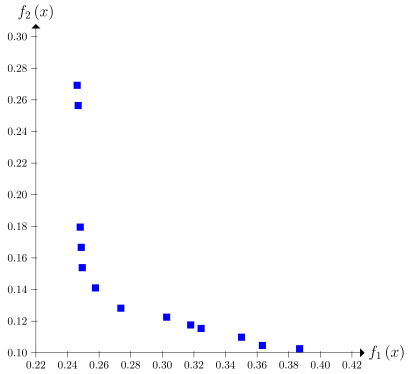
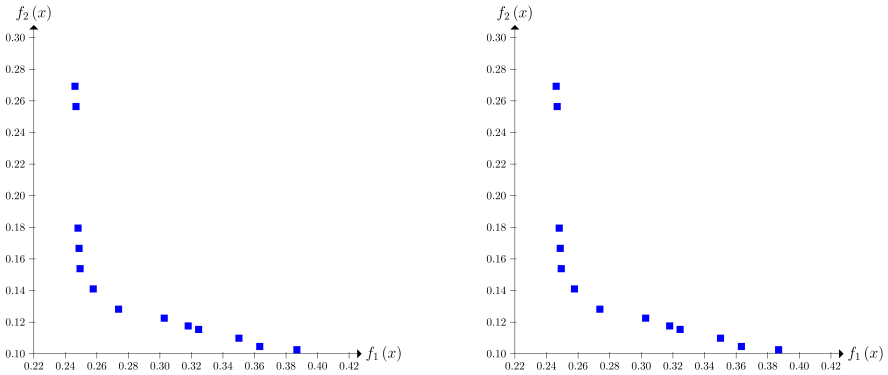


Fig. 1: Non-dominated fronts generated by EMOPG for the Car data set in a particular trial.

between both methods is not statistically significant. However, the training-reduction attained by EMPOG was the second best. Regarding large data sets, from Figure 2b one can note that EMOPG outperforms most of the existing methods in terms of both reduction and accuracy. In fact, the performance of EMOPG on large data sets is better than that of the methods that outperformed it on small data sets. This is interesting, since PG methods are normally applied on large data sets. The accuracy with respect to GENN is similar, but EMOPG has a better reduction rate. In fact, from this figure it can be observed that the performance of EMOPG is the closest to the top right corner, where hypothetically, the best method would be located.

V. Conclusions and Future Work

We have proposed EMOPG, a novel evolutionary multi-objective approach for dealing with the prototype generation (PG) problem. Our approach explicitly aims to optimize the two main criteria that are directly related to the two main drawbacks of the k-NN classifier that affect its use on large data sets. EMOPG obtains prototypes that do not significantly degrade the performance of k-NN, with reduction rates above 98%. This makes this approach applicable to problems from different domains, specially on large data sets. EMOPG outperformed several PG methods reported in the state of the art, and it was not



(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 for both (a) small and (b) large data sets.

significantly worse compared to the best ones. Hence, we can conclude that EMOPG can be seen as a new baseline-to-beat PG technique.

The contributions of our proposed method are as follows: (i) the initialization of the prototypes giving preference to border instances allows to improve the convergence of the algorithm; (ii) the optimization is formulated such that the solutions simultaneously improve the accuracy-performance and the reduction rate; (iii) our proposal obtained competitive performance over a large number of data sets; and (iv) the multiple solutions in the non-dominated front allow the user to choose different solutions based on his/her preferences without performing a new search.

As part of our future work, we would like to extend EMOPG for dealing with the reduction of both the number of samples and the number of features. Including preferences during the optimization is also another interesting path for future research. We want to evaluate EMOPG for different values of k of k-NN. We would also like to study the impact of the evolutionary parameters on the quality of the prototypes found by EMOPG. Finally, we are interested in testing EMOPG on large scale data sets.

Acknowledgements

The first author is grateful for the support from CONACyT scholarship no. 329013.

References

[1] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Sys.*, vol. 14, no. 1, pp. 1–37, 2007.

[2] S. García, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 417–435, 2012.

[3] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Trans. Syst. Man Cy. C*, vol. 42, no. 1, pp. 86–100, Jan. 2012.

[4] I. Triguero, S. Garcia, and F. Herrera, "Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification," *Pattern Recogn.*, vol. 44, pp. 901–916, 2011.

[5] L. Nanni and A. Lumini, "Particle swarm optimization for prototype reduction," *Neurocomputing*, vol. 72, no. 4–6, pp. 1092–1097, 2008.

[6] A. Cervantes, I. M. Galvan, and P. Isasi, "AMPSO: a new particle swarm method for nearest neighborhood classification," *IEEE Trans. Sys. Man Cy. B*, vol. 39, no. 5, pp. 1082–1091, 2009.

[7] H. J. Escalante, K. M. Mendoza, M. Graff, and A. Morales-Reyes, "Genetic programming of prototypes for pattern classification," in *Proc. of IbPRIA 2013*, ser. LNCS, vol. 7887. Springer, 2013, pp. 100–107.

[8] F. Fernandez and P. Isasi, "Evolutionary design of nearest prototype classifiers," *J. Heuristics*, vol. 10, pp. 431–454, 2004.

[9] J. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.

[10] U. Garain, "Prototype reduction using an artificial immune system," *Pattern Anal. Appl.*, vol. 11, no. 3–4, pp. 353–363, 2008.

[11] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. Springer, US, 2007.

[12] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[14] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds. International Center for Numerical Methods in Engineering, 2001, pp. 95–100.

[15] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, pp. 257–286, 2000.

[16] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. Springer, 2010.

[17] M. Zeleny, "Compromise programming," *Multiple criteria decision making*, pp. 262–301, 1973.

[18] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.