Pruning Optimum-Path Forest Classifiers Using Multi-Objective Optimization

Douglas Rodrigues
Department of Computing
Federal University of São Carlos
São Carlos, Brazil
E-mail: douglasrodrigues.dr@gmail.com

André Nunes Souza
Department of Electrical Engineering
São Paulo State University
Bauru, Brazil
E-mail: andrejau@feb.unesp.br

João Paulo Papa
Department of Computing
São Paulo State University
Bauru, Brazil
E-mail: papa@fc.unesp.br

***Abstract*—Multi-objective optimization plays an important role when one has fitness functions that are somehow conflicting with each other. Also, parameter-dependent machine learning techniques can benefit from such optimization tools. In this paper, we propose a multi-objective-based strategy approach to build compact though representative training sets for Optimum-Path Forest (OPF) learning purposes. Although OPF pruning can provide such a nice representation, it comes with the price of being parameter-dependent. The proposed approach cope with that problem by avoiding the classifier to be hand-tuned by modeling the task of parameter learning as a multi-objective-oriented optimization problem, which can be less prone to errors. Experiments on public datasets show the robustness of the proposed approach, which is now parameterless and user-friendly.**

I. Introduction

Optimization techniques have been widely used in several research areas, since many problems usually refer to the task of finding the minimum or maximum of a given function. Some challenges such as the allocation of resources, product delivery for logistic companies, and cutting-and-packing problems with direct application in industries are among the most widely pursued tasks.

In regard to optimization techniques, a considerable attention has been given to nature-inspired meta-heuristics, i.e., approaches that aim at solving several problems using concepts based on physical process, social dynamics and/or the behavior of living beings [1], [2], [3], [4], [5], [6], [7]. Since such techniques are quiet elegant to solve optimization problems, they have been applied for solving multi-objective optimization problems, where the idea of a unique global optimal solution is replaced by a non-dominated solution set, the so-called Pareto-optimal set [8], [9], [10], [11], [12], [13].

Meta-heuristic multi-objective optimization techniques have become popular to solve many optimization problems in the field of engineering [14], [15], [16], [17], [18], [19], [20]. However, such techniques have a wider range of applications, mainly in the context of machine learning-oriented problems, which are usually composed of several multi-objective tasks [21]. It is very common to face problems in which we need to find out the best set of parameters (e.g., a neural network architecture) that lead to both high recognition rates and low computational burden.

Parameter-dependent machine learning techniques are often preferable to cope with real-world problems, since they can be adjusted to fit better to a given application. Support Vector Machines (SVMs) [22], Neural Networks (NNs) [23] and Optimum-Path Forest (OPF) [24], [25], [26] are some examples of parameter-dependent techniques, just to name a few. Although OPF comprises a collection of classifiers, being some of them parameterless, new problems may require some of them to be parameterized. Papa et al. [27] proposed to design compact though representative training sets by means of learning the most important samples during training, thus discarding the remaining ones preserving similar or even higher accuracy [28], [29], [30]. Such process is ruled by a parameter that controls the desired loss in accuracy with respect to the final pruned training set when compared to the original one. Later on, Nakamura et al. [31] modeled the problem of finding the OPF pruning parameter automatically as a mono-objective optimization problem. In fact, they combined both information of training set size and accuracy in a single equation (fitness function) for further optimization.

Actually, since a good recognition accuracy does require a considerable training set size (for most applications where training is complex), these two criteria fit perfectly into a multi-objective optimization problem, since they are conflicting with each other. Therefore, the main contribution of this paper is apply meta-heuristic multi-objective algorithms to the Optimum-Path Forest pruning algorithm in order to obtain compact and representative training sets without the need for the desired loss and the maximum number of iteration parameters. The experiments showed the robustness of the proposed approach in a number of datasets.

The remainder of this paper is organized as follows. Section II presents the theoretical background about multi-objective optimization, while Section III presents the OPF classifier and its pruning strategy. Section IV discusses the experiments, and Section V states conclusions and future works.

II. Multi-Objective Optimization

The multi-objective optimization problem aims at finding the global minimum $\mathbf{x}^* \in \mathcal{S}$ that minimizes a set of M functions represented by $\mathbf{f}$, i.e.:

$$\mathbf{x}^* = \arg\min_{\forall \mathbf{x} \in \mathcal{S}} (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})), \tag{1}$$

subject to:

$$g_i(\mathbf{x}) = 0 \quad \forall i = 1, 2, \ldots, p, \tag{2}$$

$$h_i(\mathbf{x}) \geq 0 \quad \forall i = 1, 2, \ldots, q, \tag{3}$$

where p and q represent the number of equality $g(\cdot)$ and inequality constraints $h(\cdot)$, and $\mathcal{S} \in \mathbb{R}^N$ stands for the search space.

In a multi-objective problem, there is no single solution that is optimal with respect to all objectives when considering conflicting objectives. Thus, the solution to a multi-objective optimization problem is no longer a scalar value, but a vector in the form of a "trade-off" known as *Pareto-optimal set*.

Firstly, we define the *Pareto Dominance*, where a solution vector $\mathbf{x}^a$ is said to dominate another solution vector $\mathbf{x}^b$ (i.e., $\mathbf{x}^a \prec \mathbf{x}^b$) if $f(x_i^a) \neq f(x_i^b), \forall i = \{1, 2, \ldots, N\}$, and $\exists i \in \{1, 2, \ldots, N\}$ such that $f(x_i^a) < f(x_i^b)$. In regard to the Pareto Dominance, a solution vector $\mathbf{x}^a$ is considered Pareto-optimal if, for every $\mathbf{x}^b$, $f_j(\mathbf{x}^a) \neq f_j(\mathbf{x}^b)$, $j = 1, 2, \ldots, M$, and if there exists at least one $j \in \{1, 2, \ldots, M\}$ such that $f_j(\mathbf{x}^a) < f_j(\mathbf{x}^b)$. Therefore, the Pareto-optimal set $\mathcal{P}^*$ considering a multi-objective optimization problem $\mathbf{f}(\mathbf{x})$ with respect to all Pareto-optimal solutions is thus defined as follows:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{S} \mid \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}'), \ \forall \mathbf{x}' \in \mathcal{S}\}. \tag{4}$$

The Pareto-optimal front PF^* with respect to a multi-objective optimization problem $\mathbf{f}(\vec{x})$ and the Pareto-optimal set $\mathcal{P}^*$ is defined as follows:

$$PF^* = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\}. \tag{5}$$

One way to solve multi-objective optimization problems is to combine all objectives into a single-objective problem. Hence, the idea of scalarized multi-objective optimization is to convert a problem of minimizing the vector $\mathbf{x}$ into a scalar optimization problem [32]. The weighted-sum method is one of the most used approach, in which several objective functions are combined into a single one through a weight vector. Thus, a problem with multiple objective functions is reduced to a single optimization problem subject to the original constraints, and the choice of the value of each weight is performed according to a preference assigned to each objective function:

$$\mathbf{x}^* = \arg\min_{\forall \mathbf{x} \in \mathcal{S}} \left(\sum_{k=1}^{M} w_k f_k(\mathbf{x}) \right), \tag{6}$$

with $\sum_{i=1}^{M} w_i = 1$. A single point of the Pareto front will be produced by a given a weight vector. A sufficiently large number of weight vectors generate a good approximation to the true Pareto front. If the weights are positive for all objectives, the solutions to the problem are Pareto optimal [32], [33]. The weights are calculated as follows:

$$w_i = \frac{u_i}{\sum_{i=1}^{M} u_i}, \tag{7}$$

where $u_i \sim \mathcal{U}(0, 1)$.

III. Optimum-Path Forest

Let $\mathcal{D} = \mathcal{D}^{tr} \cup \mathcal{D}^{ts}$ be a λ-labeled dataset such that $\mathcal{D}^{tr}$ and $\mathcal{D}^{ts}$ stand for the training and testing sets, respectively. Additionally, let $\mathbf{s} \in \mathcal{D}$ be an n-dimensional sample that encodes features extracted from a certain data, and $d(\mathbf{s}, \mathbf{v})$ be a function that computes the distance between two samples $\mathbf{s}$ e $\mathbf{v}$, $\mathbf{v} \in \mathcal{D}$.

Let $\mathcal{G}^{tr} = (\mathcal{D}^{tr}, \mathcal{A})$ be a graph derived from the training set, such that each node $\mathbf{v} \in \mathcal{D}^{tr}$ is connected to every other node in $\mathcal{D}^{tr} \backslash \{\mathbf{v}\}$, i.e. $\mathcal{A}$ defines an adjacency relation known as *complete graph* (Figure 1a illustrates such training graph), in which the arcs are weighted by function $d(\cdot, \cdot)$. We can also define a path π_s as a sequence of adjacent and distinct nodes in $\mathcal{G}^{tr}$ with terminus at node $\mathbf{s} \in \mathcal{D}^{tr}$. Notice a *trivial path* is denoted by $\langle s \rangle$, i.e. a single-node path.

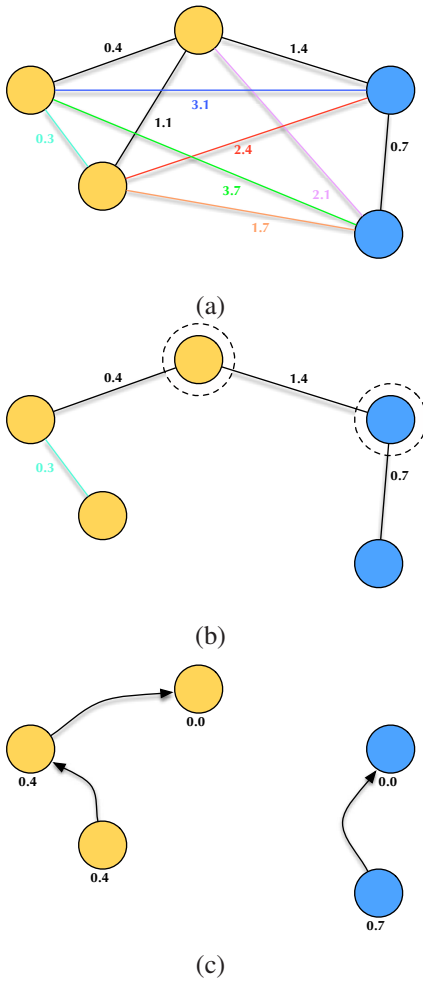

Fig. 1. Illustration of the OPF working mechanism: (a) a two-class (orange and blue labels) training graph with weighted arcs, (b) a MST with prototypes highlighted, and (c) optimum-path forest generated during the training phase with costs over the nodes (notice the prototypes have zero cost).

Let $f(\pi_s)$ be a path-cost function that essentially assigns a real and positive value to a given path π_s, and $\mathcal{S}$ be a set of prototype nodes. Roughly speaking, OPF aims at solving the following optimization problem:

$$\min f(\pi_s),\ \forall\ \mathbf{s} \in \mathcal{D}^{tr}. \tag{8}$$

The good point is that one does not need to deal with mathematical constraints, and the only rule to solve Equation 8 concerns that all paths must be rooted at $\mathcal{S}$. Therefore, we must choose two principles now: how to compute $\mathcal{S}$ (prototype estimation heuristic) and $f(\pi)$ (path-cost function).

Since prototypes play a major role, Papa et al. [24] proposed to position them at the regions with the highest probabilities of misclassification, i.e. at the boundaries among samples from different classes. In fact, we are looking for the nearest samples from different classes, which can be computed by means of a Minimum Spanning Tree (MST) over $\mathcal{G}^{tr}$. The MST has interesting properties, which ensure OPF can be errorless during training when all arc-weights are different to each other [34]. Figure 1b depicts a MST with prototypes highlighted.

Finally, with respect to the path-cost function, OPF requires f to be a smooth one [35]. Previous experience in image segmentation led the authors to use a chain code-invariant path-cost function, that basically computes the maximum arc-weight along a path, being denoted as f_{max} and given by:

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{if } \mathbf{s} \in \mathcal{S} \\ +\infty & \text{otherwise,} \end{cases} \\ f_{max}(\pi_s \cdot (\mathbf{s}, \mathbf{t})) &= \max\{f_{max}(\pi_s), d(\mathbf{s}, \mathbf{t})\}, \end{aligned} \tag{9}$$

where $\pi_s \cdot (\mathbf{s}, \mathbf{t})$ stands for the concatenation between path π_s and arc $(\mathbf{s}, \mathbf{t}) \in \mathcal{A}$. In short, by computing Equation 9 for every sample $\mathbf{s} \in \mathcal{D}^{tr}$, we obtain a collection of optimum-path trees (OPTs) rooted at $\mathcal{S}$, which then originate an optimum-path forest. A sample that belongs to a given OPT means it is more strongly connected to it than to any other in $\mathcal{G}^{tr}$. Roughly speaking, the OPF training step aims at solving Equation 9 in order to build the optimum-path forest, as displayed in Figure 1c. A gentle implementation of the aforementioned procedure is given by *Algorithm* 1.

Algorithm 1: OPF with Complete Graph - Training Algorithm

Input: A λ-labeled training graph $\mathcal{G}^{tr} = (\mathcal{D}^{tr}, \mathcal{A})$ and the distance function d.
Output: An optimum-path forest P, label map L, cost map C, and a list of nodes ordered by their costs (ascending order) K.

```
1  S ← SelectPrototypes(G^tr);
2  for s ∈ S do
3      C_s ← 0;
4      P_s ← NIL;
5  for s ∈ D^tr \ S do
6      C_s ← ∞;
7      P_s ← NIL;
8  Q ← BuildPriorityQueue(D^tr, C); K ← ∅;
9  while Q ≠ ∅ do
10     Remove from Q a sample s whose C_s is minimum;
11     K ← K ∪ {s};
12     for v ∈ D^tr \ s do
13         tmp ← max{C_s, d(s, v)};
14         if (tmp < C_v) then
15             C_v ← tmp;
16             P_v ← s;
17             L_v ← λ(s);
18 return [P, L, C, K];
```

Line 1 calls $SelectPrototypes$ function, which computes the minimum spanning tree over the input graph, selects prototypes as the connected elements with different classes (Figure 1b), and finally it outputs the prototype set $\mathcal{S}$. Lines $2-4$ and $5-7$ initialize the prototypes and remaining samples, respectively, where $C_\mathbf{s}$ stands for the cost of sample s, and $P_\mathbf{s}$ denotes its predecessor in the optimum-path forest. Line 8 creates a priority queue based on the input graph and the cost of each sample (for such purpose, LibOPF implements a binary heap).

The main loop in Lines $9-17$ is in charge of the OPF competition process, in which Line 10 removes a sample s from the priority queue whose cost is minimum, and Line 11 inserts s in the ordered list K (such list will be used to speed up the classification phase). The inner loop in Lines $12-17$ evaluates all neighbors of $\mathbf{s}$ in order to conquer them, and line 13 computes f_{max} as described by Equation 9. When sample $\mathbf{v}$ is conquered by $\mathbf{s}$ (Lines $15-17$), the cost (Line 15), predecessor (Line 16) and label map (Line 17) of $\mathbf{v}$ are updated.

The next step concerns the testing phase, where each sample $\mathbf{t} \in \mathcal{D}^{ts}$ is classified individually as follows: $\mathbf{t}$ is connected to all training nodes from the optimum-path forest learned in the training phase (Figure 2a), and it is evaluated the node $\mathbf{v}^* \in \mathcal{D}^{tr}$ that conquers $\mathbf{t}$, i.e. the one that satisfies the following equation:

$$C_\mathbf{t} = \min_{\mathbf{v} \in \mathcal{D}^{tr}} \max\{C_\mathbf{v}, d(\mathbf{v}, \mathbf{t})\}. \tag{10}$$

The classification step simply assigns $L(\mathbf{t}) = \lambda(\mathbf{v}^*)$, as depicted in Figure 2b. Roughly speaking, the testing step aims at finding the training node $\mathbf{v}$ that minimizes $C_\mathbf{t}$.

The example displayed in Figure 2 shows an interesting situation: although $\mathbf{t}$ is closest to a sample from "yellow" class, it has been labeled to the another class, which emphasizes OPF is not a distance-based classifier, but instead it uses the "power of connectivity" among samples. The OPF with complete graph degenerates to a nearest neighbor classifier only when all training samples are prototypes. Actually, such situation is considerably difficult to face, thus indicating a high degree of

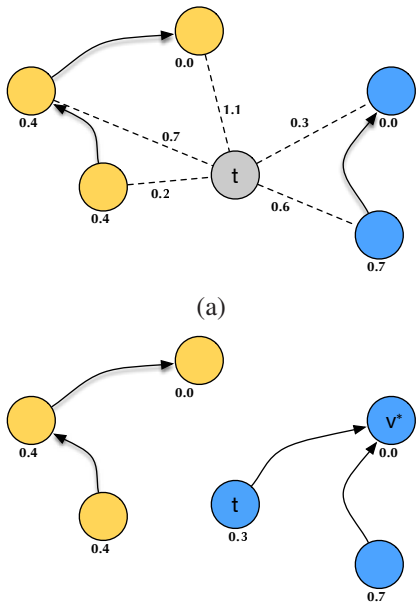

Fig. 2. Illustration of the OPF classification mechanism: (a) sample **t** is connected to all training nodes, and (b) **t** is conquered by $\mathbf{v}^*$ and it receives the "blue" label.

overlapping among samples, which means the features used for that specific problem may not be adequate enough to describe it. Algorithm 2 implements the OPF classification procedure. Notice this algorithm uses the ordered list of notes K, where $k_i \in K$ stands for a node in $\mathcal{D}^{tr}$, to speed up the classification step, as proposed by Papa et al. [26].

Algorithm 2: OPF Classification Algorithm

Input: Classifier $[P_1, C_1, L_1, K]$, test set $\mathcal{D}^{ts}$, and the distance function d .
Output: Label L_2 and predecessor P_2 maps defined for $\mathcal{D}^{ts}$, and accuracy value Acc.
Auxiliary: Cost variables tmp and $mincost$.

1 **for** *each* $t \in \mathcal{D}^{ts}$ **do**
2 $i \leftarrow 1$, $mincost \leftarrow max\{C_1(k_i), d(k_i, t)\}$;
3 $L_2(t) \leftarrow L_1(k_i)$, $P_2(t) \leftarrow k_i$;
4 **while** $i < |K|$ *and* $mincost > C_1(k_{i+1})$ **do**
5 Compute $tmp \leftarrow max\{C_1(k_{i+1}, d(k_{i+1}, t\}$;
6 **if** $tmp < mincost$ **then**
7 $mincost \leftarrow tmp$;
8 $L_2(t) \leftarrow L_1(k_{i+1})$, $P_2(t) \leftarrow k_{i+1}$;
9 $i \leftarrow i + 1$;
10 Compute accuracy Acc according to [24];
11 **return** $[L_2, P_2, Acc]$;

A. Learning with Pruning of Irrelevant Patterns

Large datasets usually present redundancy, so at least in theory it should be possible to estimate a reduced training set with the most relevant patterns for classification. The use of a training set $\mathcal{D}^{tr}$ and an evaluating set $\mathcal{D}^{ev}$ has allowed OPF to learn relevant samples for $\mathcal{D}^{tr}$ from the classification erros in $\mathcal{D}^{ev}$, by swapping misclassified samples of $\mathcal{D}^{ev}$ and non-prototype samples of $\mathcal{D}^{tr}$ during a few iterations [24]. In this learning strategy, $\mathcal{D}^{tr}$ remains the same size and the classifier instance with the highest accuracy is selected to be tested on the unseen set $\mathcal{D}^{ts}$. Algorithm 3 implements this learning procedure.

Algorithm 3: OPF Learning Algorithm

Input: A λ-labeled training and evaluating sets $\mathcal{D}^{tr}$ and $\mathcal{D}^{ev}$, respectively, number T of iterations, and distance function d.
Output: Optimum-path forest P_1, cost map C_1, label map L_1, ordered set K and $MaxAcc$.
Auxiliary: Arrays FP and FN of sizes c for false positives and false negatives, set S of prototypes, and list LM of misclassified samples.

1 Set `MaxAcc` $\leftarrow -1$;
2 **for** *each iteration* $I = 1, 2, \ldots, T$ **do**
3 $LM \leftarrow 0$ and compute the set $\mathcal{S} \subset \mathcal{D}^{tr}$ of prototypes;
4 $[P_1, C_1, L_1, K] \leftarrow$ Algorithm 1$(\mathcal{D}^{tr}, \mathcal{S}, d)$;
5 **for** *each class* i **do**
6 $FP(i) \leftarrow 0$ and $FN(i) \leftarrow 0$.
7 $[L_2, P_2, Acc] \leftarrow$Algorithm 2$([P_1, C_1, L_1, K], \mathcal{D}^{ev}, d)$;
8 **if** $Acc > MaxAcc$ **then**
9 $[P_1^*, C_1^*, L_1^*, K^*] \leftarrow [P_1, C_1, L_1, K]$;
10 $MaxAcc \leftarrow Acc$;
11 **while** $LM \neq 0$ **do**
12 $LM \leftarrow LM \backslash \{t\}$;
13 Replace t by a non-prototype sample randomly selected from $\mathcal{D}^{tr}$;
14 **return** $[P_1^*, C_1^*, L_1^*, Z^*]$ *and* $MaxAcc$;

The efficacy of Algorithm 3 increases with the size of $\mathcal{D}^{tr}$, because more non-prototype samples can be swapped by misclassified samples of $\mathcal{D}^{ev}$. However, for sake of efficiency, we need to choose a reasonable maximum size for $\mathcal{D}^{tr}$. After learning the best training samples for $\mathcal{D}^{tr}$, we may also mark paths in P_1 used to classify samples in $\mathcal{D}^{ev}$ and define their nodes as *relevant samples* in a set $\mathcal{R}$. The "irrelevant" training samples in $\mathcal{D}^{tr} \backslash \mathcal{R}$ can then be moved to $\mathcal{D}^{ev}$. Algorithm 4 applies this idea repetitively, while the loss in accuracy on $\mathcal{D}^{ev}$ with respect to the highest accuracy obtained by Algorithm 3 (using the initial training set size) is less or equal to a maximum value *MLoss* specified by the user or there are no more irrelevant samples in $\mathcal{D}^{tr}$.

In Algorithm 4, Lines $1-3$ compute learning and classification using the highest accuracy classifier obtained for an initial training set size. Its accuracy is returned in Acc and used as reference value in order to stop the pruning process when the loss in accuracy is greater than a user-specified *MLoss* value,

Algorithm 4: OPF Pruning Algorithm

Input: Training and evaluating sets, $\mathcal{D}^{tr}$ and $\mathcal{D}^{ev}$, labeled by λ, distance function d, maximum loss $MLoss$ in accuracy on $\mathcal{D}^{ev}$, and number T of iterations.

Output: OPF classifier $[P_1, C_1, L_1, Z]$ with reduced training set.

Auxiliary: Set $\mathcal{R}$ of relevant samples, and variables Acc and tmp.

1 $[P_1, C_1, L_1, K] \leftarrow$ Algorithm 3$(\mathcal{D}^{tr}, \mathcal{D}^{ev}, T, d)$;
2 $[L_2, P_2, Acc] \leftarrow$ Algorithm 2$([P_1, C_1, L_1, K], \mathcal{D}^{ev}, d)$;
3 $tmp \leftarrow Acc$ and $\mathcal{R} \leftarrow \emptyset$;
4 **while** $(Acc - tmp) \leq MLoss$, $\mathcal{R} \neq \mathcal{D}^{tr}$ **do**
5 $\mathcal{R} \leftarrow \emptyset$;
6 **for** *each sample* $t \in \mathcal{D}^{ev}$ **do**
7 $s \leftarrow P_2(t) \in \mathcal{D}^{tr}$;
8 **while** $s \neq NIL$ **do**
9 $\mathcal{R} \leftarrow \mathcal{R} \cup s$;
10 $s \leftarrow P_1(s)$;
11 Move samples from $\mathcal{D}^{tr} \backslash \{\mathcal{R}\}$ to $\mathcal{D}^{ev}$;
12 $[P_1, C_1, L_1, K] \leftarrow$ Algorithm 3$(\mathcal{D}^{tr}, \mathcal{D}^{ev}, T, d)$;
13 $[L_2, P_2, Acc] \leftarrow$Algorithm 2$([P_1, C_1, L_1, K]), \mathcal{D}^{ev}, d)$;
14 **return** $[P_1, C_1, L_1, K]$;

or when all training samples are considered relevant. The main loop in Lines $4 - 13$ essentially marks the relevant samples in $\mathcal{D}^{tr}$ by following the optimum paths used for classification (Lines $6 - 10$) backwards, moves irrelevant samples to $\mathcal{D}^{ev}$, and repeats learning and classification from a reduced training set until it reaches the above stopping criterion.

IV. Experimental Results

In this section, we present the experimental results with respect to the proposed approach to optimize OPF pruning. In order to minimize both the OPF classification error and the training set size, we considered the following optimization techniques: Multi-objective Black Hole Algorithm (MOBHA) [36], Multi-objective Cuckoo Search (MOCS) [37], Multi-objective Firefly Algorithm (MOFFA) [38] and Multi-objective Particle Swarm Optimization (MOPSO) [39]. Additionally, we employed ten benchmark datasets, being eight from UCI repository[1] and two private: NTL-Comercial and NTL-Industrial. Notice we set the number of possible solutions, $m = 10$, and the number of iterations $T = 10$. We used 50%, 30% and 20% for the training, evaluating and testing set percentages, respectively. In regard to the source-code, we used the LibOPT [40].

The proposed approach aims at modeling the problem of automatically tuning the $MLoss$ parameter and the number of iterations T by means of a multi-objective problem, which means we are going to use the OPF classification error over the evaluating set and the training set size as the fitness functions to be minimized. Mathematically speaking, the aforementioned problem can be formulated as follows:

$$\mathbf{x}^* = \arg \min_{\forall \mathbf{x} \in \mathcal{S}} \left(w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) \right), \tag{11}$$

where $f_1(\mathbf{x})$ stands for the the OPF classification error over the evaluating set, and $f_2(\mathbf{x})$ denotes the training set size. As such, we opted to approach the problem as a scalarized multi-objective problem, as previously discussed in Section II.

Table I displays the number of samples, number of features, and number of classes for each dataset. Notice we decided to evaluate the robustness of the proposed approach under different scenarios.

TABLE I
Description of the benchmarking datasets.

Dataset	No. samples	No. features	No. classes
German Numer	1,000	24	2
Ionosphere	351	34	2
MPEG-7	1,400	180	70
Pendigits	7,494	16	10
Satimage	4,435	36	6
Sonar	208	60	2
Splice	1,000	60	2
SVM-Guide 2	391	20	3
NTL-Comercial	4952	8	2
NTL-Industrial	3182	8	2

Table II presents the parameter configuration for each meta-heuristic optimization technique. The parameters were set based on empirical studies conducted over a number of experiments.

TABLE II
Parameters used for each technique.

Technique	Parameters
PSO	$c_1 = 1.7, c_2 = 1.7, w = 0.7$
CS	$\alpha = 0.1, p_a = 0.25$
FFA	$\alpha = 0.2, \beta = 1, \gamma = 1$
BHA	–

In order to evaluate the techniques compared in this work, we used a measure F that considers the results of both fitness functions as follows:

$$F = \frac{\text{accuracy over the test set}}{\text{training set size}}. \tag{12}$$

Therefore, since one aims at using smaller training sets and obtaining higher accuracy rates, the greater the value of F, the better the technique is.

Table III presents the OPF classification accuracy over the test set and the training set size. However, Table IV displays the F values for each optimization technique considered in this work, where the values in bold stand for the best results. Notice OPF denotes the standard classifier, i.e., without pruning samples. It is worth noting that MOBHA obtained the best

[1] http://archive.ics.uci.edu/ml/

results in German Numer, MPEG-7 and Pendigits datasets. MOCS achieved the best F values in SVM-Guide 2, NTL-Comercial and NTL-Industrial datasets. Concerning Splice and Sonar datasets, the highest values belong to MOFFA, while the better results in Ionosphere and Satimage datasets were achieved by MOPSO.

Clearly, one can observe the multi-objective techniques obtained better results than standard OPF in all datasets. Also, different optimization techniques obtained the best results in distinct datasets, although they have performed similarly in all situations. Another interesting point to observe concerns the stability of the methodology employed in this work. The results presented in Table IV stand for a single run over the datasets. Although Equation 11 initializes w_1 and w_2 with random values, we have observed that different runs, i.e., experiments with other values for these variables, did not influence the final results. Probably, the results would be different if we have used more fitness functions, as well as the decision variables ($MLoss$ and T) seem to stabilize both the accuracy and training set size after some time.

Table V displays the classification time over the testing set (miliseconds) regarding the proposed approach to find $MLoss$ and T. One can observe that MOBHA obtained the lowest classification time in German Numer, MPEG-7, Pendigits, SVM-Guide 2 and NTL-Comercial datasets, thus obtaining a gain of 322%, 272%, 419%, 322% and 454%, respectively, in terms of computational load. With respect to Ionosphere, Satimage, Sonar and NTL-Industrial datasets, MOPSO obtained the lowest classification time with gains of 390%, 363%, 313% and 413%, respectively. MOCS obtained the best classification time in Splice dataset achieving a gain of 368%. Therefore, we can highlight the gain in efficiency concerning the proposed multi-objective OPF pruning algorithm.

V. Conclusions

The very basic idea of OPF pruning is to learn the most representatives samples in order to create a compact training set. However, OPF pruning is parameter-dependent ($MLoss$ and T), and finding the proper values for such parameters can be a hard task.

In this paper, we proposed to use well-known meta-heuristic algorithms in order to find near-optimal values concerning the aforementioned parameters. The experiments were conducted over 10 datasets in order to show the robustness of the proposed approach. We have observed that all meta-heuristic algorithms obtained F values considerably better than standard OPF, indicating a good compactness of the training sets, being MOBHA and MOCS the ones that obtained the best results in the majority datasets. Also, we can highlight the improvements in the computational load.

In regard to future works, we intend to work on a many-objective optimization model by adding more fitness functions, such as pruning less representative OPF prototypes. We believe that keeping more prototypes, we can also obtain more accurate results.

Acknowledgment

The authors are grateful to FAPESP grants #2014/12236-1 and #2016/19403-6, Capes, and CNPq grant #306166/2014-3.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.

[2] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[3] J. Kennedy and R. Eberhart, *Swarm Intelligence*. M. Kaufman, 2001.

[4] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[5] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal Bio-Inspired Computing*, vol. 2, no. 2, pp. 78–84, 2010.

[6] X.-S. Yang and D. S., "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, pp. 330–343, 2010.

[7] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3, pp. 267–289, 2010.

[8] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings of the Fifth International Conference in Genetic Algorithm*, 1993, pp. 416–423.

[9] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proceedings of the IEEE Conference on Evolutionary Computation*, vol. 1, 1994, pp. 82–87.

[10] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, 1994.

[11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[12] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, 1999, pp. 98–105.

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

[14] S. Sivasubramani and K. S. Swarup, "Multi-objective harmony search algorithm for optimal power flow problem," *International Journal of Electrical Power and Energy Systems*, vol. 33, no. 3, pp. 745–752, 2011.

[15] S. N. Omkar, J. Senthilnath, R. Khandelwal, G. N. Naik, and S. Gopalakrishnan, "Artificial bee colony (ABC) for multi-objective design optimization of composite structures," *Applied Soft Computing*, vol. 11, no. 1, pp. 489–499, 2011.

[16] R. Akbari, R. Hedayatzadeh, K. Ziarati, and B. Hassanizadeh, "A multi-objective artificial bee colony algorithm," *Swarm and Evolutionary Computation*, vol. 2, pp. 39–52, 2012.

[17] H. C. Lau, L. Agussurja, S.-F. Cheng, and P. J. Tan, "A multi-objective memetic algorithm for vehicle resource allocation in sustainable transportation planning," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, 2013, pp. 2833–2839.

[18] K. Khalili-Damghani, A.-R. Abtahi, and M. Tavana, "A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems," *Reliability Engineering and System Safety*, vol. 111, no. 0, pp. 58–75, 2013.

[19] Y.-J. Zheng, Q. Song, and S.-Y. Chen, "Multiobjective fireworks optimization for variable-rate fertilization in oil crop production," *Applied Soft Computing*, vol. 13, no. 11, pp. 4253–4263, 2013.

[20] M. K. Marichelvam, T. Prabaharan, and X.-S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 301–305, April 2014.

[21] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 3, pp. 397–415, May 2008.

TABLE III
TRADE-OFF BETWEEN THE OPF CLASSIFICATION ERROR AND TRAINING SET SIZE.

	German Numer	Ionosphere	MPEG-7	Pendigits	Satimage	Sonar	Splice	SVM-Guide 2	NTL-Comercial	NTL-Industrial
MOBHA	61.03%/167	85.53%/51	88.65%/251	99.25%/928	91.95%/666	86.63%/36	70.70%/137	76.12%/64	63.19%/601	70.37%/414
MOCS	60.00%/175	85.53%/52	89.61%/263	99.44%/960	92.15%/665	84.69%/31	70.91%/134	78.61%/64	66.75%/615	75.11%/436
MOFFA	63.97%/174	88.58%/50	89.25%/254	99.44%/962	92.28%/672	88.35%/31	70.25%/132	79.95%/68	64.32%/637	70.43%/414
MOPSO	62.86%/178	89.47%/48	89.49%/256	99.36%/946	92.64%/630	87.04%/31	72.11%/147	74.55%/65	62.76%/614	70.08%/418
OPF	59.64%/500	81.58%/174	91.85%/700	99.67%/3744	93.15%/2214	85.98%/103	65.86%/499	74.16%/194	62.97%/2475	57.91%/1590

TABLE IV
F VALUES CONSIDERED THE DATASET AND TECHNIQUES EMPLOYED IN THIS WORK.

	German Numer	Ionosphere	MPEG-7	Pendigits	Satimage	Sonar	Splice	SVM-Guide 2	NTL-Comercial	NTL-Industrial
MOBHA	**0.0037**	0.016771	**0.003532**	**0.00107**	0.001381	0.024064	0.005161	0.011894	0.001051	0.0017
MOCS	0.003429	0.016448	0.003407	0.001036	0.001386	0.027319	0.005292	**0.012283**	**0.001085**	**0.001723**
MOFFA	0.003676	0.017716	0.003514	0.001034	0.001373	**0.0285**	**0.005322**	0.011757	0.001010	0.001701
MOPSO	0.003532	**0.01864**	0.003496	0.001050	**0.001471**	0.028077	0.004905	0.011469	0.001022	0.001677
OPF	0.001193	0.004689	0.001312	0.000266	0.000421	0.008348	0.001320	0.003823	0.000254	0.000364

TABLE V
OPF CLASSIFICATION TIME [MS] OVER THE TEST SET.

Datasets	OPF	MOBHA	MOCS	MOFFA	MOPSO
German Numer	0.012205	**0.003800**	0.004236	0.004243	0.004359
Ionosphere	0.001462	0.000454	0.000447	0.000488	**0.000375**
MPEG-7	0.080667	**0.029715**	0.033082	0.030930	0.032747
Pendigits	0.481865	**0.115130**	0.120072	0.122654	0.122084
Satimage	0.251095	0.073188	0.075933	0.077176	**0.069209**
Sonar	0.000653	0.000448	0.000228	0.000246	**0.000209**
Splice	0.018213	0.005239	**0.004956**	0.005084	0.005645
SVM-Guide 2	0.001660	**0.000516**	0.000828	0.001063	0.000808
NTL-Comercial	0.185389	**0.040857**	0.041952	0.044368	0.043448
NTL-Industrial	0.071641	0.017673	0.018219	0.018176	**0.017353**

[22] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

[24] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.

[25] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares, "Efficient supervised optimum-path forest classification for large datasets," *Pattern Recognition*, vol. 45, no. 1, pp. 512–520, 2012.

[26] J. P. Papa, S. E. N. Fernandes, and A. X. Falcão, "Optimum-path forest based on k-connectivity: Theory and applications," *Pattern Recognition Letters*, vol. 87, pp. 117–126, 2017.

[27] J. Papa, A. Falcão, G. de Freitas, and A. Avila, "Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 396–400, 2010.

[28] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[29] N. Jankowski and M. Grochowski, *Comparison of Instances Seletion Algorithms I. Algorithms Survey*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 598–603.

[30] E. Pkalska, R. P. W. Duin, and P. Paclk, "Prototype selection for dissimilarity-based classifiers," *Pattern Recognition*, vol. 39, no. 2, pp. 189–208, 2006.

[31] R. Nakamura, C. Pereira, J. Papa, and A. Falcão, "Optimum-path forest pruning parameter estimation through harmony search," in *24th SIBGRAPI Conference on Graphics, Patterns and Images*, 2011, pp. 181–188.

[32] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer US, 1998.

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

[34] C. Allène, J.-Y. Audibert, M. Couprie, and R. Keriven, "Some links between extremum spanning forests, watersheds and min-cuts," *Image Vision Computing*, vol. 28, no. 10, pp. 1460–1471, 2010.

[35] A. X. Falcão, J. Stolfi, and R. A. Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[36] K. Jeet and R. Dhir, "Software clustering using hybrid multi-objective black hole algorithm," in *International Conference on Software Engineering and Knowledge Engineering*, 2016, pp. 650–653.

[37] K. N. Abdul Rani, W. F. Hoon, M. F. Abd Malek, N. A. Mohd Affendi, L. Mohamed, N. Saudin, A. Ali, and S. C. Neoh, "Modified cuckoo search algorithm in weighted sum optimization for linear antenna array synthesis," in *IEEE Symposium on Wireless Technology and Applications*, 2012, pp. 210–215.

[38] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Engineering with Computers*, vol. 29, no. 2, pp. 175–184, 2013.

[39] M. C. Bhuvaneswari, *Application of Evolutionary Algorithms for Multi-Objective Optimization in VLSI and Embedded Systems*. Springer India, 2015.

[40] J. P. Papa, G. H. Rosa, D. Rodrigues, and X.-S. Yang, "Libopt: An open-source platform for fast prototyping soft optimization techniques," *ArXiv e-prints*, 2017, http://adsabs.harvard.edu/abs/2017arXiv170405174P.