Neurocomputing 143 (2014) 27–43

Contents lists available at ScienceDirect

Neurocomputing

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

A hybrid meta-learning architecture for multi-objective optimization of SVM parameters

Péricles B.C. Miranda [a,b,*], Ricardo B.C. Prudêncio [a], André P.L.F. de Carvalho [c], Carlos Soares [d]

[a] Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
[b] Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco, Recife, Brazil
[c] Depto. de Ciências da Computação, Universidade de São Paulo, São Carlos, Brazil
[d] INESC TEC, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal

ARTICLE INFO

Article history:
Received 5 February 2014
Received in revised form
29 May 2014
Accepted 8 June 2014
Communicated by V. Palade
Available online 18 June 2014

Keywords:
Multi-objective optimization
Particles swarm optimization
Meta-learning
Support vector machines
Parameter selection

ABSTRACT

Support Vector Machines (SVMs) have achieved a considerable attention due to their theoretical foundations and good empirical performance when compared to other learning algorithms in different applications. However, the SVM performance strongly depends on the adequate calibration of its parameters. In this work we proposed a hybrid multi-objective architecture which combines meta-learning (ML) with multi-objective particle swarm optimization algorithms for the SVM parameter selection problem. Given an input problem, the proposed architecture uses a ML technique to suggest an initial Pareto front of SVM configurations based on previous similar learning problems; the suggested Pareto front is then refined by a multi-objective optimization algorithm. In this combination, solutions provided by ML are possibly located in good regions in the search space. Hence, using a reduced number of successful candidates, the search process would converge faster and be less expensive. In the performed experiments, the proposed solution was compared to traditional multi-objective algorithms with random initialization, obtaining Pareto fronts with higher quality on a set of 100 classification problems.

1. Introduction

Support Vector Machines (SVMs) represent a class of very competitive algorithms that have been successfully applied by the Machine Learning community to different learning problems. Despite the potential good results that can be yielded, the SVM performance strongly depends on the adequate choice of its parameters, and an exhaustive trial-and-error procedure for selecting good values of parameters is not practical for computational reasons [1].

The selection of SVM parameters is treated by different authors as an optimization task in which a search technique is used to find adequate configurations of parameters for the given learning problem at hand. Different optimization techniques have been applied in literature to SVM parameter selection, including for instance Evolutionary Algorithms (EA) [2], Particle Swarm Optimization (PSO) [3] and Tabu Search [4]. Previous works commonly used single objective techniques for SVM parameter selection, however this is not totally adequate since this task is inherently a Multi-Objective Optimization (MOO) problem [5]. In this context, we can mention the use of Multi-Objective EA (MOEA) [5,6], Multi-Objective PSO (MOPSO) [7] and Gradient-Based techniques [1], which considered multiple objectives in the parameter selection process. Although the application of search techniques represents an automatic solution to select SVM parameters, this approach can be very expensive, since a large number of candidate configurations of parameters are often evaluated during the search process [8].

An alternative approach to SVM parameter selection is the use of Meta-Learning (ML), which treats parameter selection as a supervised learning task [8,9]. Each training example for ML (i.e. each meta-example) stores the characteristics of a past learning problem (i.e., its number of training instances, its class entropy, etc.) and the performance obtained by a set of candidate configurations of parameters on the problem. By receiving a set of such meta-examples, a meta-learner can predict configurations of parameters for a new problem based on its characteristics.

ML can provide good or intermediate results with a relatively small number of suggested configurations of parameters [11]. Optimization algorithms in turn can produce better results compared to ML, but with high computational costs. Although ML can be a lower cost alternative, it is dependent on the chosen learning problems in the meta-base and the characteristics adopted to

* Corresponding author.
E-mail addresses: pbcm@cin.ufpe.br (P.B.C. Miranda), rbcp@cin.ufpe.br (R.B.C. Prudêncio), andre@icmc.usp.br (A.P.L.F. de Carvalho), csoares@fep.up.pt (C. Soares).

http://dx.doi.org/10.1016/j.neucom.2014.06.026
0925-2312/© 2014 Elsevier B.V. All rights reserved.

describe them [10]. If the chosen characteristics do not represent the problems well or the similarity between problems in the meta-base is not sufficiently high, the meta-learning suggestions can be harmed.

In order to take advantage of the above strategies, in [11,10] the authors developed a hybrid approach in which ML and search techniques were combined to SVM parameter selection. In the hybrid approach, ML was adopted to suggest a number of solutions (configurations of parameters) which are adopted as the initial population of the search technique. The search technique then just refined promising solutions returned by ML, speeding up the optimization process. In [11], the authors adopted a classical version of PSO and the Tabu Search algorithm as search techniques, with the objective of minimizing the error rate obtained by the SVM in regression problems. In [10], in turn, Genetic Algorithms were adopted to optimize the SVM parameters to classification problems. In both cases, an instance-based algorithm was adopted as the meta-learner. The hybrid approaches proposed in previous works had only a single objective considered, when in fact, the use of multiple objectives would be more adequate to the SVM parameter selection.

In order to avoid the above limitation, we created a hybrid architecture to the SVM parameter selection in order to deal with multiple objectives. More specifically, we proposed to adopt a ML initialization of MOO search techniques. In our proposal, a meta-learning technique is initially used to suggest a Pareto front of parameters for a given problem based on non-dominated configurations applied to similar problems. To the best of our knowledge, this is the first attempt of producing Pareto fronts by adopting a meta-learning strategy. The configurations of parameters in the suggested Pareto front are optimized by a MOO technique. We highlight that although the focus of our work is the SVM parameter selection problem, the concepts adopted in the hybrid MOO approach can be extended to other contexts (i.e., the reuse of knowledge acquired in previous instances of MOO problems in order to improve the optimization process to new instances).

In this work we implemented a prototype to analyze the influence of ML on the optimization process considering multiple criteria. Six well-known Multi-Objective Particle Swarm Optimization (MOPSO) techniques were implemented for comparison: MOPSO, MOPSO-CDR, MOPSO-CDRS, CSS-MOPSO, m-DNPSO and MOPSO-CDLS. We focused on particle swarm algorithms due to their simplicity and good performance in difficult optimization tasks. These techniques were applied in our work to select two SVM parameters: the parameter γ of the RBF kernel and the regularization constant C, which may have a strong influence in the SVM performance [1]. In our work, a dataset of 100 meta-examples was produced from the evaluation of a set of 399 configurations of (γ, C) on 100 different classification problems. Each classification problem was described by a number of 8 meta-features proposed in [12]. All the implemented MOO algorithms were used to optimize the parameters (γ, C) regarding the success rate and number of support vectors (which indicates complexity) observed in the SVMs. These algorithms were used in two different scenarios: with the initial population suggested by Meta-Learning, leading to hybrid methods, and with a random initial population. According to experimental results, the hybrid methods were able to generate better solutions along the iterations when compared to the traditional approach.

This paper is organized as follows. Section 2 presents the SVM parameter selection problem. Section 3 presents details of the proposed work, followed by Section 4 which presents the implementation details. Section 5 describes the experiments and in Section 6 presents the obtained results. Finally, Section 7 presents some conclusions and the future work.

2. SVM parameter selection

The SVM parameter selection task is often performed by evaluating a range of different combinations of parameters and retaining the best one in terms of an *objective function* [4]. One important issue to be considered is the objective function to be optimized, which is in general a functional, estimating the SVM performance using the problem's dataset (e.g., leave-one-out and cross-validation estimates, error bounds, model complexity, among others). Another important issue to be taken into consideration is the strategy adopted to explore the space of parameters. Regarding the second issue, an exhaustive procedure to explore the parameter space can lead to good results, however it is a strategy that can be computationally expensive.

In order to avoid an exhaustive or a random exploration of parameters and improve the search process, different authors have deployed search and optimization techniques such as gradient-based techniques [1], Evolutionary Algorithms [2,13], Tabu Search [4] and Particle Swarm Optimization [3]. In this context, each search technique deploys specific search operators and mechanisms to explore the search space, aiming to reach optimized parameters with good values for the chosen objective function.

The initial works commonly considered the SVM parameter selection a single objective problem, however this is not totally adequate since this task is inherently a Multi-Objective Optimization (MOO) problem [5]. Thus, many authors have applied multi-objective optimization techniques as Multi-Objective EA (MOEA) [5] and Multi-Objective PSO (MOPSO) [7] for the given problem. Although the application of MOO search techniques represents an automatic and suitable solution to select SVM parameters, this approach can be very expensive and with a low convergence, since a large number of possible configurations of parameters are often evaluated during the search [8,14] and also a great amount of objectives to be analyzed.

Alternatively, *Meta-Learning* has been proposed and investigated in recent years to SVM parameter selection [9,14–17]. In this approach, the choice of parameters for a problem is based on well-succeeded parameters adopted to previous similar problems. For this, it is necessary to maintain a set of *meta-examples* where each meta-example stores: (1) a set of characteristics (called *meta-features*) describing a learning problem; and (2) the best configuration of parameters (among a set of candidates) empirically evaluated on the problem. A *meta-learner* is then used to acquire knowledge from a set of such meta-examples in order to recommend (predict) adequate configurations of parameters for new problems based on their meta-features. Meta-learning is able to predict not only one configuration of parameters but also to recommend *rankings* of configurations (as performed in [14]). This is interesting since the user has more alternatives if the first configuration recommended by meta-learning does not achieve adequate results. In this way, using ML, SVM models can be suggested for new problems without executing the SVM on each candidate configuration of parameters making this approach more economic than optimization techniques in terms of computational cost [10]. However, ML is very dependent on the quality of its meta-examples and on the choice of meta-features.

As we mentioned before, recent studies were performed combining ML with optimization algorithms [10,11] in such a way that ML is used to recommend parameters which will be later refined by a search technique. The first work, developed by Gomes and Prudencio [11], combined ML with Particle Swarm Optimization algorithms for the SVM parameter selection problem. This proposal uses ML to recommend a certain number of solutions as initial population for the optimization algorithm. The inspiration for this methodology arose from the fact that similar problems have similar search spaces. Thus, the search space of a problem with similar characteristics can provide useful pieces of information about optimal regions of a new problem [18]. The ML module is an important step in the whole process that can increase the convergence velocity of the optimization algorithm toward good solutions.

The implementation of the optimization algorithm had to be configured to optimize the settings chosen C, and γ of the RBF kernel. So that each particle represents a configuration (C, γ), which indicates its position in the search space. As stated earlier, the objective function used to evaluate the quality of the solutions is the SVM of kernel RBF, whose output is the error rate in the regression with cross-validation 10-fold. Thus, the goal of the algorithm is to find the configuration (C, γ) that minimizes the error rate for a given problem.

The work developed by Reif and Shafait [10] followed the same ideology, using ML and case-based reasoning to provide initial solutions for genetic algorithms in order to optimize parameters for SVM and Random Forest (*RF*) algorithms in classification problems. For the SVM, the same parameters considered in [11] were optimized, C and γ. The *RF* algorithm had 5 parameters optimized and 15 meta-features (numerical and categorical) were used. The k-NN technique was used to define the similarity between meta-examples and the cross-validation 10-*fold* was used as objective function.

Both works handled the SVM parameter selection task as a single objective problem and achieved good results. As the SVM parameter selection problem can also be multi-objective problem, the current work proposes to use the combination of ML with multi-objective search techniques. In this section we present the advantages and limitations of the search and the meta-learning approaches to SVM parameter selection. In our work, we combine the two approaches in such a way that meta-learning is used to recommend parameters which will be later refined by a search technique.

3. Proposed solution

In the current paper, we proposed a hybrid architecture which combines meta-learning and optimization algorithms for multi-objective parameter selection. The current paper shares with [11] the idea of using meta-learning to suggest initial solutions for an optimization algorithm (see Fig. 1). In [11], initially, the Meta-Learner module retrieves a predefined number of past meta-examples stored in the Meta-base, selected on the basis of their similarity to the input problem. Following, the Search module adopts as initial search points the most successful configurations of parameter values on the retrieved meta-examples. The Search module iterates its search process by generating new candidate configurations to be evaluated in the learning strategy module (e.g., a SVM classifier, or a combination of classifiers). The configuration of parameter values produced will be the best one generated by the Search module up to its convergence or according to another stopping criterion.

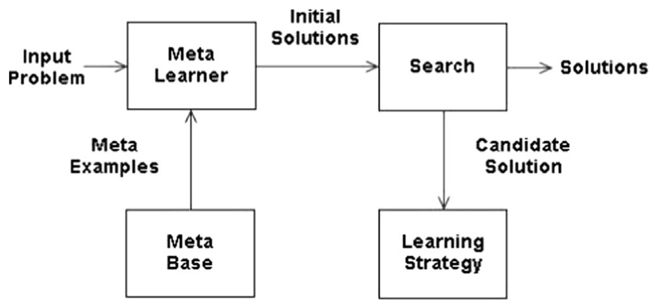


Fig. 1. General architecture.

However, different from [11], in order to deal with multiple objectives, we initially proposed a new multi-objective meta-learning approach which is able to suggest a Pareto front from the retrieved problems (instead of single solutions as in [11]). This new meta-learning approach is original. Also, we combined the meta-learning with MO optimization algorithms, which was not considered in [11]. In the current work, 6 multi-objective algorithms were evaluated. Finally, we highlight that the current paper brings new ideas and contributions that can be specifically relevant for multi-objective optimization field.

In the next sub-sections each module of the proposed solution will be detailed.

3.1. *Search module*

In contrast to single objective approaches, the multi-objective optimization (MOO) aims to optimize more than one objective at the same time. MOO can be defined as the problem of finding a vector of decision variables that satisfies constraints and optimizes a vector of functions whose elements represent objective functions.

Considering the search space of solutions $\mathcal{S}$ and a set of objective functions $\vec{f}(\vec{x}):=[f_1(\vec{x}),f_2(\vec{x}),\dots,f_k(\vec{x})]$, where $\vec{x}=(x_1,x_2,\dots,x_n)\in\mathbb{R}^n$ is the vector of parameters in $\mathcal{S}$. The optimization task is to find a set of solutions $S^{opt}\subset\mathcal{S}$ considering the vector of objectives $\vec{f}(\vec{x})$. A general multi-objective optimization minimization problem can be defined as [19]

$$\text{minimize } \vec{f}(\vec{x}):=[f_1(\vec{x}),f_2(\vec{x}),\dots,f_k(\vec{x})], \tag{1}$$

subject to:

$$g_i(\vec{x})\le 0,\quad i=1,2,\dots,p, \tag{2}$$

$$h_j(\vec{x})=0,\quad j=1,2,\dots,q, \tag{3}$$

where $\vec{x}=(x_1,x_2,\dots,x_n)\in\mathbb{R}^n$ is the vector on the decision search space; k is the number of objectives and $g_i(\vec{x})$ and $h_j(\vec{x})$ are the constraint functions and $p+q$ is the number of constrains of the problem. Given two vectors $\vec{x},\vec{y}\in\mathcal{S}$, $\vec{x}$ dominates $\vec{y}$ (denoted by $\vec{x}\prec\vec{y}$) if $\vec{x}$ is better than $\vec{y}$ in at least one objective and $\vec{x}$ is not worse than $\vec{y}$ in any objective. $\vec{x}$ is not dominated (incomparable) if does not exist another current solution $\vec{y}$ in the current population, such that $\vec{y}\prec\vec{x}$. The set of non-dominated solutions in the objective space is known as Pareto front (PF).

In an iterative optimization technique, the search process starts with an initial set $S^{ini}\subset\mathcal{S}$ of possible solutions, which progressively move to better regions. Given a learning problem d, the optimized solutions S^{opt} found during the search can be seen as a function of initial search points:

$$S^{opt}\leftarrow Search(d,S^{ini}) \tag{4}$$

In a conventional search proposal, S^{ini} is initialized randomly and uniformly in $\mathcal{S}$. Ideally, the search algorithm should be less dependent of the initial population. In order to minimize this dependency, the set S^{ini} is recommended by using meta-learning.

3.2. *Meta-base*

The meta-base is derived from a set of learning problems $\mathcal{D}$, which can be collected from a data repository. Formally, $\mathcal{M}$ is a set of meta-examples derived from $\mathcal{D}$. Each meta-example $e_i\in\mathcal{M}$ is related to a learning problem $d_i\in\mathcal{D}$ and stores: (1) a vector of z meta-features $\vec{c}_i=(c_i^1,\dots,c_i^z)$ describing a problem d_i; (2) the incomparable solutions $(x_i^1,\dots,x_i^*)\subset\mathcal{S}$, with their respective performance in the problem. Two points are considered in the meta-examples creation: (1) meta-features adopted to describe the learning problem and (2) definition of suitable solutions for the problem.

The meta-features form a set of descriptive aspects of the datasets (e.g., number of attributes, number of *outliers*, number of categorical attributes, among others). Several researches have been carried out to propose meta-features, including general measures, statistics, information theory and model-based (see [20,21,9]). According to Prudêncio and Ludermir [22], the meta-features should be appropriate to the class of problem to be solved (e.g. classification or regression). Besides, the meta-features should not be derived from subjective analysis, as visual inspection of graphics or charts.

Finally, the set of solutions $(x_i^1, \ldots, x_i^*)$ stored in the meta-example is defined through the application of a set of candidate solutions $S^C \subset \mathcal{S}$ in the problem. The most suitable solutions in S^C are chosen considering their performance on the objectives in $\vec{f}(\vec{x})$.

3.3. Meta-learner

After the meta-base creation, the meta-leaner $\mathcal{L}$ is responsible to suggest suitable solutions from meta-examples that are similar to the input problem. Formally, $\vec{c}$ is the description of the input problem d and $\mathcal{M}$ is the set of meta-examples. The meta-learner generates the set S^{rec} of recommended solutions:

$$S^{rec} \leftarrow \mathcal{L}(\vec{c}, \mathcal{M}). \tag{5}$$

Initially, given a description $\vec{c}$, the meta-leaner selects the most similar meta-examples from $\mathcal{M}$. The similarity of meta-examples is defined by the meta-feature values. The solutions stored into the selected meta-examples are inserted in the set to be recommended S^{ini}. The set of solutions recommended by the meta-learning is adopted as initial population in the search module. Thus, the hybrid strategy can be formally defined as

$$Hybrid(d, \vec{c}, \mathcal{M}) = Search(d, \mathcal{L}(\vec{c}, \mathcal{M})) \tag{6}$$

In the next section, we present an implemented prototype which followed the hybrid solution described here. Experiments evaluating the implemented prototype will be presented in Section 5.

4. Implementation

In this work we adopted a hybrid multi-objective architecture to select parameters for SVMs in classification problems. Although the implementation and the study case are focused on parameter selection of SVMs considering classification problems, we emphasize that the proposed architecture can be applied to select parameters of other algorithms and for regression as well.

In this work we used the framework *Scikit Learn* to implement the SVMs [23]. This *toolbox* provides several implementations of machine learning algorithms, all being well documented and validated. Two specific parameters are considered: parameter γ from RBF kernel and the parameter of regularization C. In the next sub-sections the implementations of each architecture module are detailed.

4.1. Search module

In this module we adopted six multi-objective particle swarm optimization algorithms and all of them were adapted to search for configurations $\vec{x} = (\gamma, C)$. We considered two objective functions: the success rate in classification (SR), to evaluate the performance of the SVM model, and the number of support vectors (NSV), which indicates the complexity of the SVM model [5]. The search process aims to find configurations $\vec{x} = (\gamma, C)$ which maximizes the SR and minimizes the NSV for a given classification problem.

In this implementation each solution or particle represents a configuration (γ, C), indicating a position of a particle in the search space. Each particle also has a velocity which indicates its current direction. In order to emphasize the differences between the single and multi-objective algorithms, we present as follows the canonical multi-objective (see Algorithm 1). All changes are written in *italic*.

Algorithm 1. Pseudo-code of the canonical multi-objective PSO algorithm.

```
Initialization:
Create particles with random positions ($\vec{x_i}$) and velocities ($\vec{v_i}$).
Assign to $\vec{p_i}(t)$ the current particle position.
Assign to $\vec{n_i}(t)$ the best position found by the neighbors.
Update repository of non-dominated solutions (PF*).
while stop criteria not reached do
  for all particle $p$ do
    Select social leader from PF*.
    Evaluate fitness $\vec{f}(\vec{x_i})$.
    Update velocity and position using the velocity and
   position equations
    (see Eqs. 7, 8).
    Update pbest values.
  end for
  Update PF*.
end while
```

The multi-objective optimization algorithm updates the position and velocity of each particle in order to progressively explore the best regions in the search space. The canonical equations to update position and velocity are presented below:

$$\vec{v}_i(t+1) = \vec{v_i}(t) + c_1 r_1(\vec{p}_i(t) - \vec{x_i}(t)) + c_2 r_2(\vec{n}_i(t) - \vec{x_i}(t)), \tag{7}$$

$$\vec{x}_i(t+1) = \vec{x}_i + \vec{v}_i(t+1), \tag{8}$$

In Eq. (7), $\vec{p}_i(t)$ is the best position achieved by the particle so far, and $\vec{n}_i(t)$ is the best position achieved by any particle in the population so far. This equation seems to be similar to the single objective PSO equation, however, the process of updating the $\vec{n}_i(t)$ makes each particle move in direction of the best global positions achieved from the set of non-dominated solutions. The parameters ω, c_1 and c_2 control the trade-off between exploring good global regions in the search space and refining the search in local regions around the particle. In Eq. (7), r_1 and r_2 are random numbers used to enhance the diversity of particle positions. As we mentioned before, we implemented six variations of MOPSO algorithms aiming to investigate the influence of ML suggestions. In the next sections we present the algorithms considered here.

4.1.1. MOPSO

The MOPSO was proposed by Coello Coello and Lechuga [24]. The difference between the MOPSO algorithm and the Algorithm 1 is the manner how the repository of incomparable solutions is represented. The repository of the MOPSO algorithm is called External Archive (EA). In the EA, the space of incomparable solutions is an adaptive grid which is composed by hypercubes. A hypercube is a geographical region which contains a certain number of solutions. For each hypercube is assigned a grade which depends on the number of solutions into the hypercube. The hypercube's grade influences in the selection process of the social leader. The hypercube with higher grade has more chances to be selected by the *roulette wheel* mechanism. After selecting a hypercube, a solution is randomly selected from that and is the new social leader. The hypercube grade increases as the number of solutions on it decreases. This selection process foments the diversity. The great problem of the MOPSO algorithm is the grid management. As new solutions are inserted or removed from hypercube, the grade of each hypercube is recalculated making the process very expensive.

4.1.2. m-DNPSO

The m-DNPSO was proposed by Hu et al. [25] and it aims to reduce the computational cost of the social leader selection presented previously in the MOPSO algorithm. The main characteristic of the m-DNPSO is to optimize one objective at a time. Initially, two objectives are considered: f_1 and f_2. f_1 is defined as the neighborhood's objective and f_2 is the global objective. The choice of the objectives that are considered is arbitrary. The m-DNPSO proposes two approaches to select the cognitive and social leaders. In order to determine the cognitive leader, it is evaluated the Euclidean distance of each particle in the swarm regarding to the solutions into the EA considering the objective f_1. The social leader is chosen among the solutions considering f_2. The m-DNPSO mechanism to select leaders reduced the computational cost presented by the MOPSO. However, this mechanism is more adequate for two objectives, and the choice of which objective to be considered by the leaders selection can impact in the algorithm performance.

4.1.3. CSS-MOPSO

The main deficiencies identified in the algorithms discussed previously are the computational cost, premature convergence and low diversity. The CSS-MOPSO was proposed by Chiu et al. [26] and implements the cross-searching strategy (CSS) which foments the local search, aiming a greater exploitation. The CSS strategy, when applied to Algorithm 1, ignores the cognitive component presented in Eq. (7) and considers two social components $\vec{n}_1(t)$ and $\vec{n}_2(t)$. Eq. (9) shows equation used in the CSS-MOPSO

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 r_1(\vec{n}_{1i}(t) - \vec{x}_i(t)) + c_2 r_2(\vec{n}_{2i}(t) - \vec{x}_i(t)). \quad (9)$$

The $\vec{n}_1(t)$ is selected based on the *Datum* point which represents the intersection between the perpendicular lines relative to the two extreme solutions in the EA. The solution into the EA which has the smaller angle θ (angle between the line that connects a solution from the EA to the *Datum* point) is selected as $\vec{n}_1(t)$. Fig. 2 illustrates the selection of $\vec{n}_1(t)$.

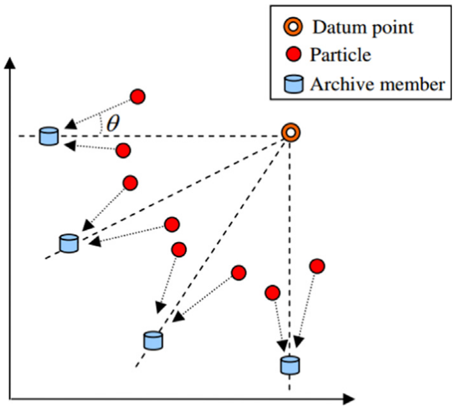


Fig. 2. Selection process of $\vec{n}_1(t)$.

The selection of $\vec{n}_2(t)$ is according to the fitness value considering an objective f_i. Initially, all particles into the swarm are ordered by the fitness value considering the f_i objective. For each particle is assigned a serial number. Considering all particles with an even serial number, for each particle is defined as $\vec{n}_2(t)$ the closest solution in the EA with whose fitness value of f_i is greater than the particle's fitness. The same mechanism is performed for particles with an odd serial number. However, the solution in the EA is selected as $\vec{n}_2(t)$ if it is the closest and with the fitness value of f_i is smaller than the particle's fitness. Fig. 3 illustrates the selection of $\vec{n}_2(t)$ for the objective f_1.

This algorithm presented better results considering the diversity of the Pareto front, identifying new solutions that were not presented before in other algorithms. However, the CSS strategy is limited for problems with more than two objectives.

4.1.4. MOPSO-CDLS

The MOPSO-CDLS was proposed by Tsou et al. [27] and it is based on the approach proposed by Raquel et al. [28] to select the social and cognitive leaders. This new mechanism uses the crowding distance (CD) that is used to select the leaders from the *EA*. The CD represents the density of the region in which a given solution is inserted. The CD is important to analyze how the solutions are distributed, allowing us to investigate the closeness degree among them. Fig. 4 shows how the CD is evaluated. The CD of the *i*th solution (solid circles) is the average of the length of cuboid' side (dotted square). The solutions located in crowded

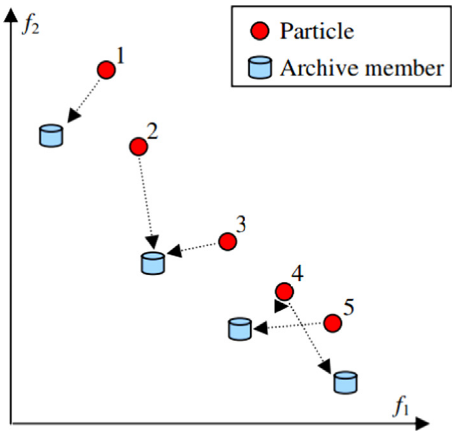


Fig. 3. Selection process of $\vec{n}_2(t)$.

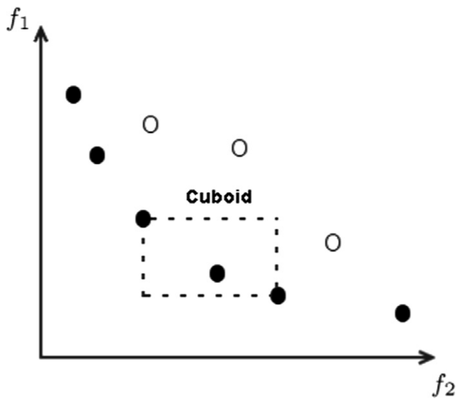


Fig. 4. *Crowding distance* evaluation considering two objective functions f_1 and f_2.

regions have smaller CD value; on the other hand, solutions located far from crowded regions have greater CD value.

Two situations can occur: the social leader of a particle is randomly chosen among the 10% less crowded solutions, if the particle is dominated by these solutions; the social leader is randomly chosen from the entire external archive, otherwise.

The cognitive leader of each particle is updated if the new position dominates the current cognitive leader. If these solutions are incomparable, the cognitive leader is randomly chosen. MOPSO-CDLS uses a local search mechanism in the *EA*. The inclusion of CD in the process of selecting leaders generated Pareto fronts with more spread and well distributed solutions.

4.1.5. MOPSO-CDR

The MOPSO-CDR was proposed by Santana et al. [29] in 2009. It was inspired on the MOPSO-CDLS algorithm and it incorporates a roulette wheel selection based on the crowding distance to select the social leader ($\vec{n}(t)$) and to prevent an excessive number of non-dominated solutions in the EA. Solutions with lower Crowding Distance have more chance to be selected as a social leader. Furthermore, MOPSO-CDR presents a novel procedure to update the cognitive leader ($\vec{p}_i$). The $\vec{p}_i$ of a particle is updated if the new position of the particle dominates the current $\vec{p}_i$. If the new position and the $\vec{p}_i$ are incomparable, the *EA* is used. The algorithm searches in the *EA* for the nearest solution to the $\vec{p}_i$ and for the nearest solution to the new position. If the closer solution in the *EA* to the new position is in a less crowded region than the closer solution in the *EA* to the $\vec{p}_i$, the new position will be chosen as the new $\vec{p}_i$. Otherwise, the old $\vec{p}_i$ remains.

4.1.6. MOPSO-CDRS

MOPSO-CDRS was proposed by Miranda and Bastos-Filho [30] in 2011 and it was inspired on MOPSO-CDR. This algorithm aims to improve diversity and uniformity of the solutions. The MOPSO-CDRS has two operation modes: basic and speciation. In the *Basic* mode, the algorithm selects the social and the cognitive leaders exactly in the same manner of the MOPSO-CDR algorithm [29]. In the *Speciation* mode, the swarm is divided into $m+1$ sub-swarms, where m is the number of objectives of the MOP. All sub-swarms have the same number of particles. Each sub-swarm has a distinct task and performs the search guided by a different objective. One sub-swarm continues executing the MOPSO-CDR algorithm, while the other k sub-swarms select the leaders according to a specific objective. At each iteration, the algorithm checks if it is necessary to change the operation mode based on the evaluation of the *EA* using two metrics, *Maximum Spread* and *Spacing*. The algorithm analyzes the *Maximum Spread* when it is in the *Basic* mode, whereas it analyzes the *Spacing* when it is in the *Speciation* mode. The algorithm changes the operation mode when a stagnation process occurs regarding the analyzed metric.

4.2. Meta-base

The meta-base creation involved 100 datasets corresponding to 100 different classification problems available for downloading in the *UCI Machine Learning* repository [31]. The list of classification problems used in this work is presented in Table 1.

In all datasets, the missing values were replaced by the average values and the data sets had the order of his examples changed randomly to minimize any tendency of the data collection of the original set. These problems correspond to datasets associated with different application domains. A variety of fields is positive in our context, because the characteristics of datasets tend to have a good variation.

Table 1
Classification problems used for meta-examples generation.

Blood	Northix	Credit-a	Mamography	First-order theorem proving
Leaf	Vehicle	Lung Cancer	Optdigits	Spoken Arabic Digit
Colic	Soybean	Hill Valley	Wholesale customers	Istanbul Stock Exchange
Heart	Splice	Hepatitis	Musk v.2	Image Segment.
Colon	Libras	Breast-w	Ozone Lvl. Detection	Robot Execution Failures
SECOM	Dexter	Column-3c	Bank Marketing	Activity Recognition
Iris	Letter	Parkinson	Parfum data	Daily and Sports Activities
Sick	Yeast	Column-2c	Ionosphere	Cardiotocography
Sonar	Cancer	Kr-vs-kp	MicroMass biodegrad.	Energy efficiency
Ecoli	Gisette	Fertility	Vertebral Column	Reuter-50-50
QSAR	PEMS-SF	Haberman	Banknote auth.	Demospongiae
Steel	ISOLET	Dorothea	Japanese Vowels	Primary Tumor
Seeds	Madelon	Mushroom	Seismic bumps	Heart-Statlog
Zoo	Arcene	Red Wine Qual	LSVT Voice Rehabilit.	Australian Sign
Lymph	Segment	Semeion	Haberman's Survival	Hypothyroid
Autos	Balance Scale	White Wine Qual	UJI Pen Chars. v.2	EEG Eye State
Vote	Breast Tissue	Spambase	Page Blocks Classif.	MAGIC Gamma Telescope
Wine	Wine Quality	CNAE-9	Climate Model Simulation Crashes	User Knowledge Modeling
Glass	SPECTF Heart	p53 Mutants	Statlog Vehicle	Handwritten Digit
ILPD	Pen Digits	Prina Diabetes	Thoracic Surg. Data	Amazon Commerce reviews set

Table 2
Meta-features for classification problems.

Simple
Number of examples
Number of attributes
Number of classes
Statistics
Mean of the attributes correlation
Geometric mean of attributes
Mean of skewness
Mean of kurtosis
Information theory
Class entropy

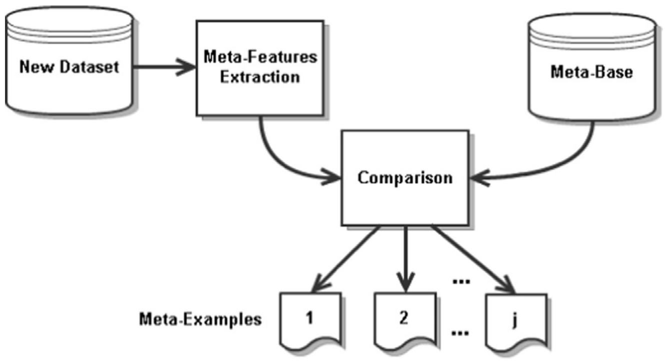


Fig. 5. Selection process of the *j* most similar meta-examples compared to the new dataset.

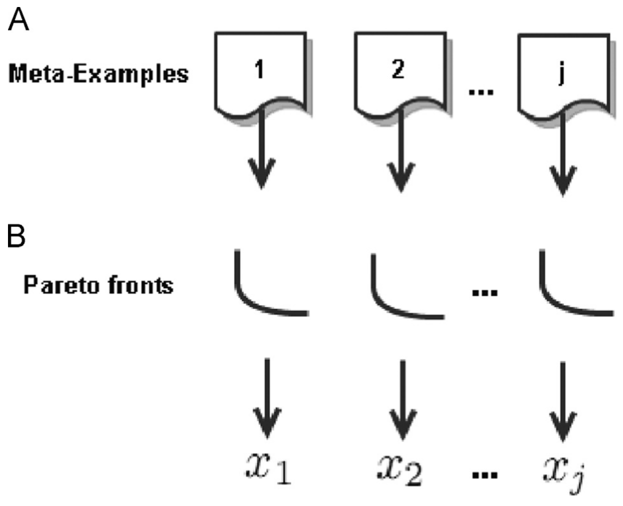


Fig. 6. Suggestion process of *j* solutions based on CD.

As we mentioned before, the meta-base is a repository of meta-examples $\mathcal{M}$, where each meta-example is a tuple composed by a vector $\overrightarrow{c}$ of meta-features and a set of configurations of parameters associated with their respective fitness values considering the vector of objectives $\overrightarrow{f}$. Here, we used 8 meta-features to describe the datasets of classification problems. These meta-features were selected from the set of features defined in [12]. The meta-features are categorized as Simple, Statistics and Information Theory (see Table 2).

The category of simple values is composed by the number of examples, attributes and classes of a dataset. The category of statistical values is composed by the mean correlation of attributes; mean of Skewness, which measures the asymmetry of the distribution regarding the central axis; mean of Kurtosis, which measures the dispersion (characterized by the flatness of the distribution curve) and the geometric mean of the attributes which evaluates the mean of the data standard deviation. Finally, the information theory category measures the randomness of the instances; being composed by the class entropy which defines the degree of uncertainty of classification [12].

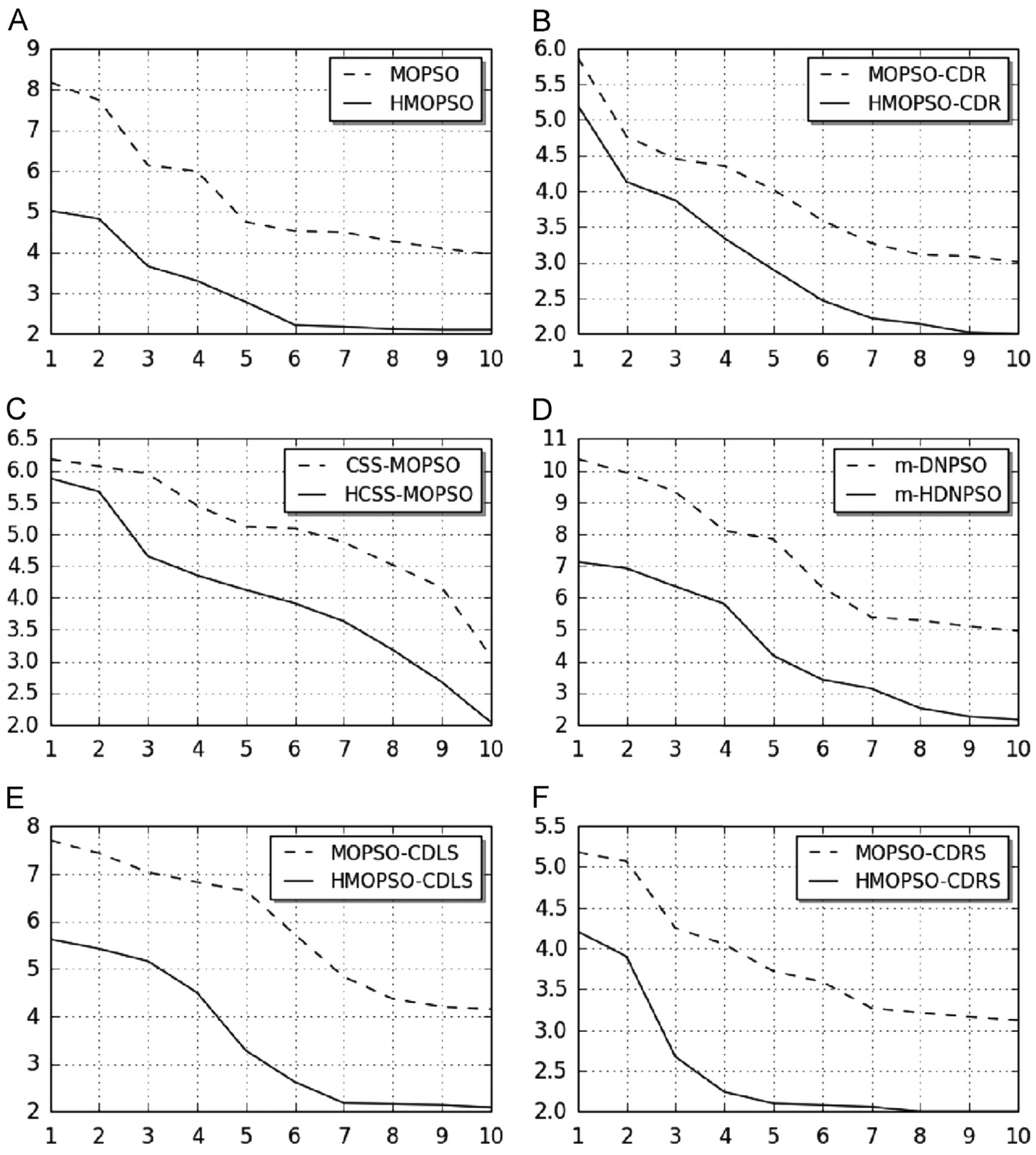


Fig. 7. Mean of *Spacing* × Iteration, using 10 particles and considering 100 classification problems.

In our work, each meta-example is composed by a set of 399 different configurations of parameters γ and C. By following the guidelines provided in [23], we considered the following exponentially growing sequences of γ and C as potentially good configurations: the parameter γ assumed 19 different values (from 2^{-15} to 2^{3}) and the parameter C assumed 21 different values (from 2^{-5} to 2^{15}), thus yielding $19 \times 21 = 399$ different combinations of parameters in the search space. In order to evaluate the fitness values of each configuration of a Meta-example, we performed SVM in the cross-validation 10-*fold* experiment.

4.3. *Meta-leaner*

A new ML mechanism of suggestion was created and it is executed in two steps; the first step is to select the j most similar problems (meta-examples) from the meta-database for a given input problem (see Fig. 5). The similarity is evaluated considering the Euclidean distance (using the meta-features values) between the input problem and each meta-example in the Meta-base. After the comparison step, all meta-examples are sorted by the similarity value. The j most similar meta-examples J pass for the next step.

In the second step, as it can be seen in Fig. 6, we applied the dominance mechanism (discussed in Section 4.1) for each meta-example in J, filtering only the incomparable solutions (procedure A in Fig. 6). Thus, for each meta-example in J was generated one Pareto front. After that, all Pareto fronts are sorted by *Crowding Distance* (presented in Section 4.1.4), which represents the density of the region in which a given solution is inserted. The CD is important to analyze how the solutions are distributed, allowing us to investigate the closeness degree among them.

Once all Pareto fronts are sorted by CD values, we select one configuration of parameters randomly, of each Pareto front, which belongs to the group of solutions with higher CD value. This mechanism selects a solution, located in low-density regions, from each j most similar problems (procedure B in Fig. 6). This can generate an initial population with higher spreading, making the searching process more diverse.

A critical point in this mechanisms is to define j, the number of similar problems to be selected. If j assumes a high value, it is

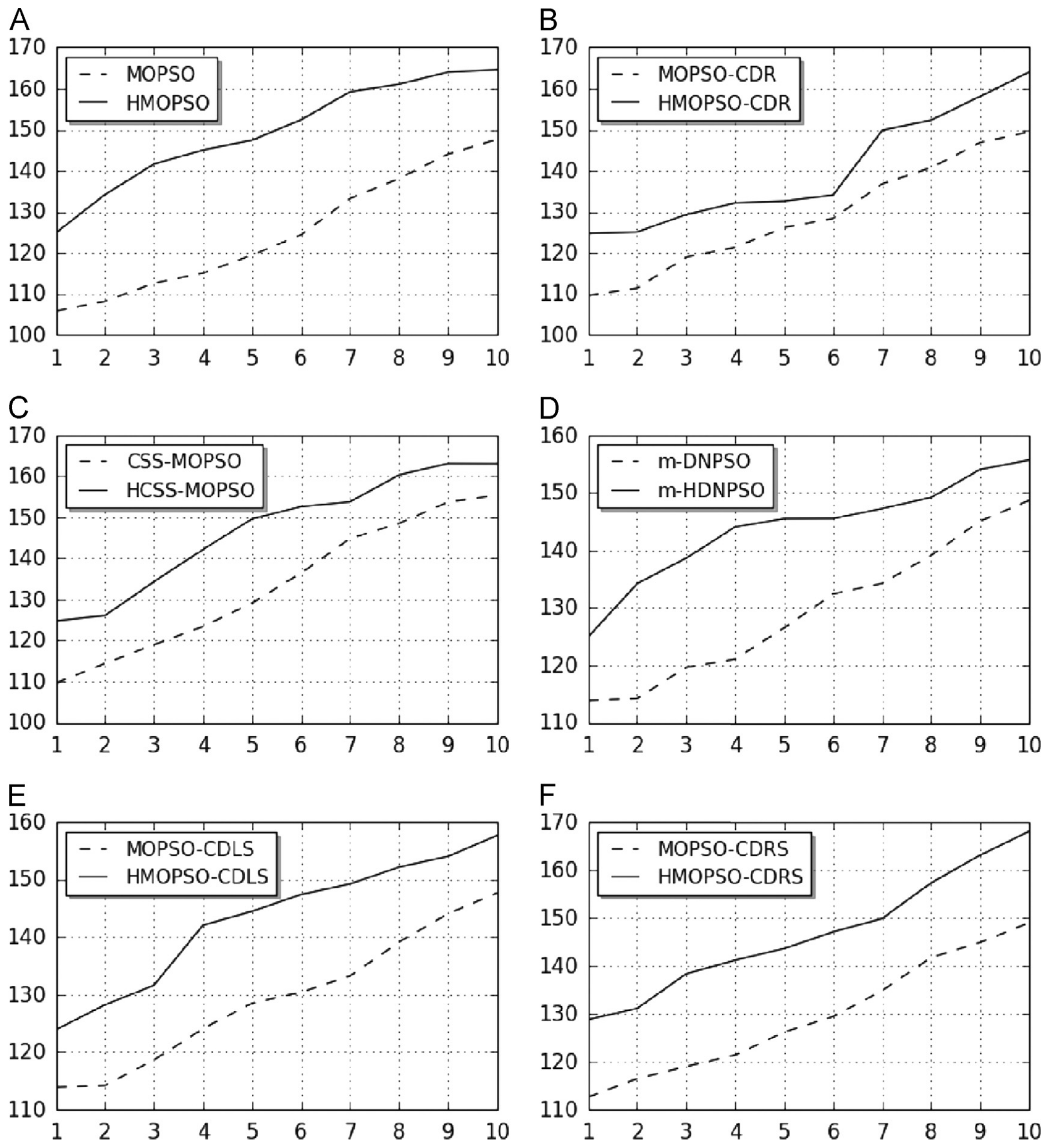


Fig. 8. Mean of *hypervolume* × Iteration, using 10 particles and considering 100 classification problems.

possible that non-similar problems be selected. In the case in which j assumes a low value, similar problems can be inconsiderate, impacting in the quality of the initial population.

5. Experiments

In this section, we present the experiments which evaluated the proposed solution on the set of 100 classification problems considered in our work. The proposed solution was evaluated by following a leave-one-out methodology described below.

At each step of leave-one-out, one meta-example was left out to evaluate the implemented prototype and the remaining 39 meta-examples were considered in the meta-base to be selected by the meta-learner module. A number of j meta-examples were suggested by the meta-learner module as the initial population of the MO algorithms (in our experiments, we adopted $j=10$). The MO algorithms then optimized the SVM configurations for the problem left out up to the number of 10 iterations. In each iteration, a Pareto front (repository of non-dominated solutions) is formed and to evaluate its quality we applied four metrics. This procedure was repeated 30 times to guarantee reliability.

This experiment was divided into three parts: initially, the results are analyzed in *Perspective* 1, which presents the mean of the metrics values of all problems for each iteration. After that, in *Perspective* 2, we present the number of wins of the algorithms per iteration regarding all problems, where the algorithm which achieved better quality (according to a specific metric) is the winner.

Although the analysis of Pareto front's quality is important, we also performed an evaluation considering the test error in the classification of the non-dominated solutions (*Perspective* 3). This perspective allows to investigate whether the non-dominated solutions really solve the problem adequately. No use the hybrid approach presents superior Pareto fronts regarding the non-hybrid techniques if its solutions do not present a significantly better classification performance.

In order to evaluate the results under this perspective, it was necessary to select a non-dominated solution from a Pareto front and evaluate its classification error for the test set. In this

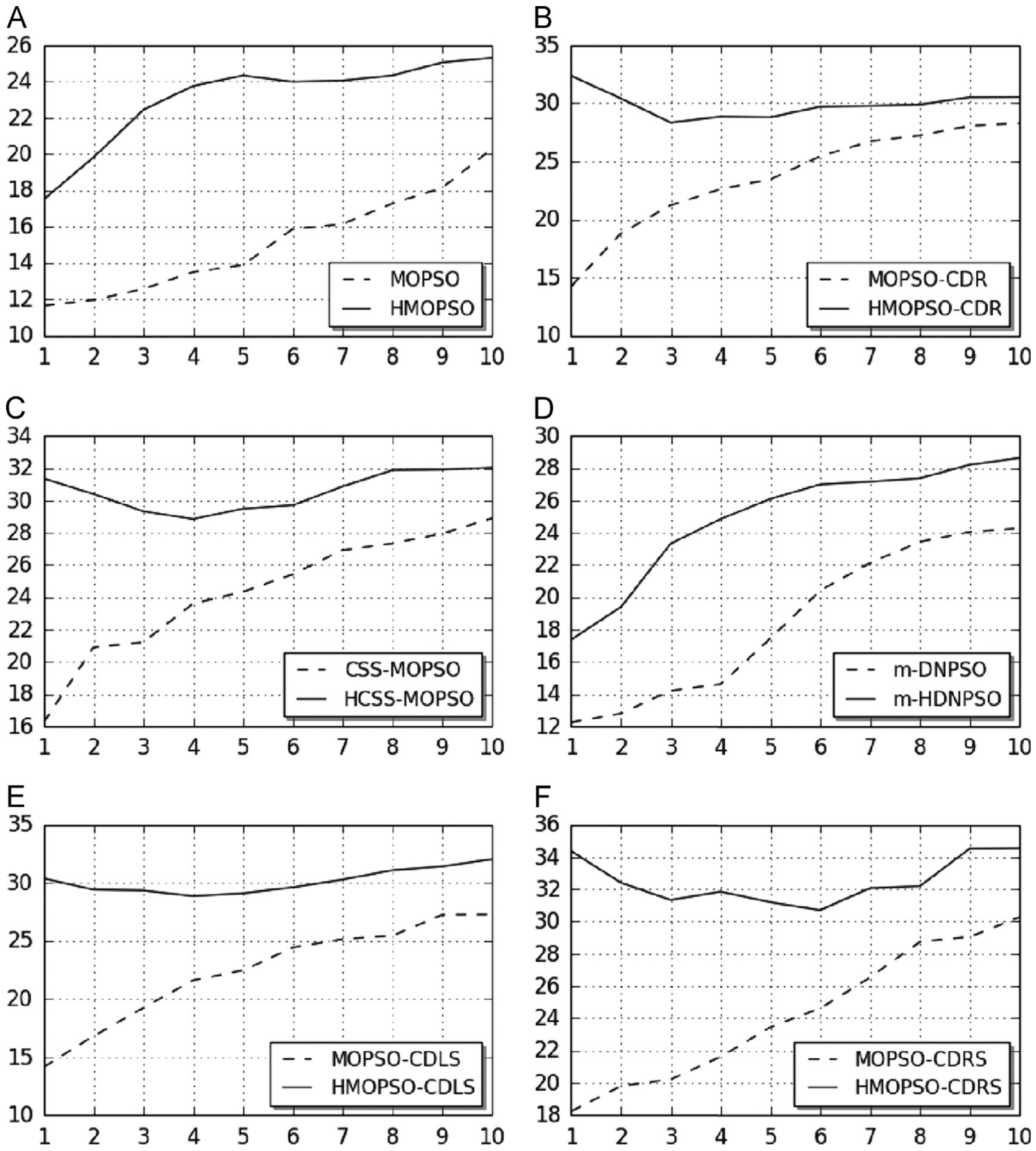


Fig. 9. Mean of *Max. Spread* × Iteration, using 10 particles and considering 100 classification problems.

experiment, we used the ranking method Borda count [35] to select a non-dominated solution from the Pareto. Once the solution was selected, it is evaluated regarding its classification performance using the fast leave-one-out test proposed by [34]. During the simulation, this procedure was applied to all problems per iteration.

As a basis of comparison, we used for each value of j a randomly initialized population for the algorithms (without meta-learning). Despite its simplicity, the random initialization has the advantage of performing a uniform initial exploration of the search space. Finally, we highlight that each algorithm was executed 30 times and the average results were recorded.

5.1. Settings

Each algorithm considered here presents its own set of parameters. In the case of MOPSO, the mutation rate is 0.5, the number of fractions for the adaptive grid is 30 and the inertia factor decreases linearly from 0.4 to 0.0 [24]. In the MOPSO-CDLS the inertia factor decreases linearly from 0.9 to 0.4 [27]. In the m-DNPSO, $m = 10$ and the inertia factor is generated randomly at each iteration in the interval $[0.5; 1.0]$ [29]. The CSS-MOPSO uses 0.01 as standard deviation for the Gaussian mutation and the inertia factor decreases linearly from 0.9 to 0.4 [26]. In the MOPSO-CDR, the mutation rate is 0.05 and the inertia factor decreases linearly from 0.9 to 0.4. The MOPSO-CDRS presents the same configuration of the MOPSO-CDR, and the saturation limit for *Spacing* and *Maximum Spreading* is 0.1%. In all algorithms, the cognitive and social acceleration constants are equal to 1.49. Moreover, the parameter j, relative to the number of suggestion performed by the meta-learning (population size) is $j = 10$. We used 10 iterations as stopping criteria of the algorithms.

5.2. Metrics

In our experiments, we evaluated the results (i.e., the Pareto fronts) obtained by all hybrid and traditional algorithms for each problem according to different quality metrics usually adopted in

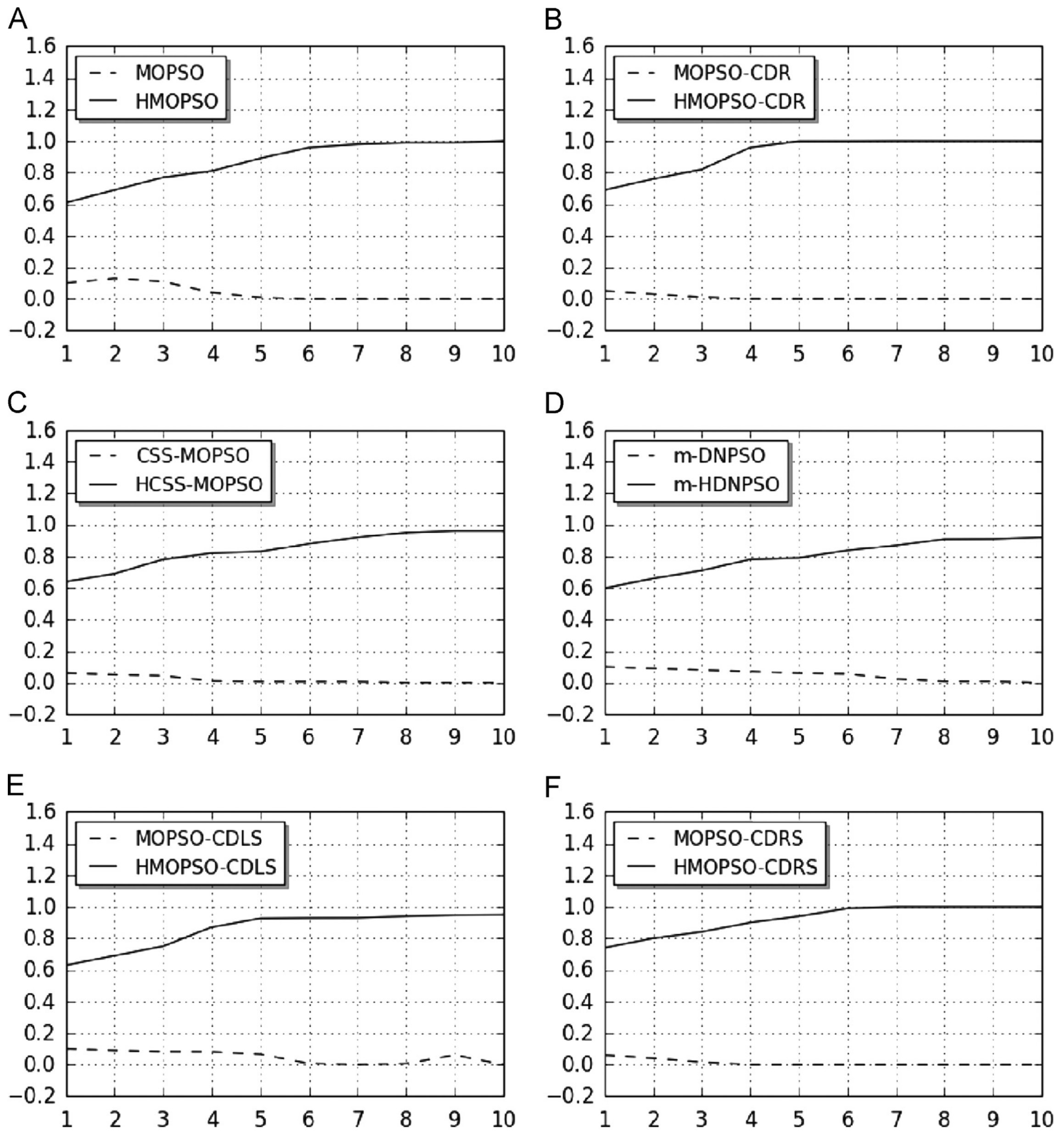


Fig. 10. Mean of *Coverage* × Iteration, using 10 particles and considering 100 classification problems.

the literature of MOO. The adopted metrics were Hypervolume, Maximum Spread, Spacing and Coverage. Each metric considers a different aspect of the Pareto front.

The hypervolume (*HV*) was proposed by Zitzler and Thiele [32] and is defined by the hypervolume in the space of objectives covered by the obtained Pareto front ($\mathcal{P}^*$). For MOP with *k*-objectives, *HV* is defined by

$$HV = \left\{ \bigcup_i a_i \mid x_i \in \mathcal{P}^* \right\}, \tag{10}$$

where x_i $(i = 1, 2, \ldots, n)$ is a non-dominated solution of the Pareto front (P^*), n is the number of solutions in the Pareto front and a_i is the hypervolume of the hypercube delimited by the position of solution x_i in the space of objectives and the origin. In practice, this metric gives the size of the dominated space, which is also called the "area under the curve". A large value of *HV* is desired.

The maximum spread (*MS*) was proposed by Zitzler et al. [32] and evaluates the maximum extension covered by the non-dominated solutions in the Pareto front. *MS* is computed by using the following equation:

$$MS = \sqrt{\sum_{m=1}^{k} \left(\max_{i=1}^{n} f_m^i - \min_{i=1}^{n} f_m^i \right)^2}, \tag{11}$$

where n is the number of solutions in the Pareto front and k is the number of objectives. This measure determines which solution covers a bigger extension of the search space; hence, large values of this metric are preferred.

The spacing *SP* estimates the diversity of the achieved Pareto front. *SP* is derived by computing the relative distance between adjacent solutions of the Pareto front as follows:

$$SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2}, \tag{12}$$

where n is the number of non-dominated solutions, d_i is the distance between adjacent solutions to the solution v_i and $\overline{d}$ is the average distance between the adjacent solutions. $SP = 0$ means that all solutions of the Pareto front are equally spaced. Hence, values of *SP* near zero are preferred.

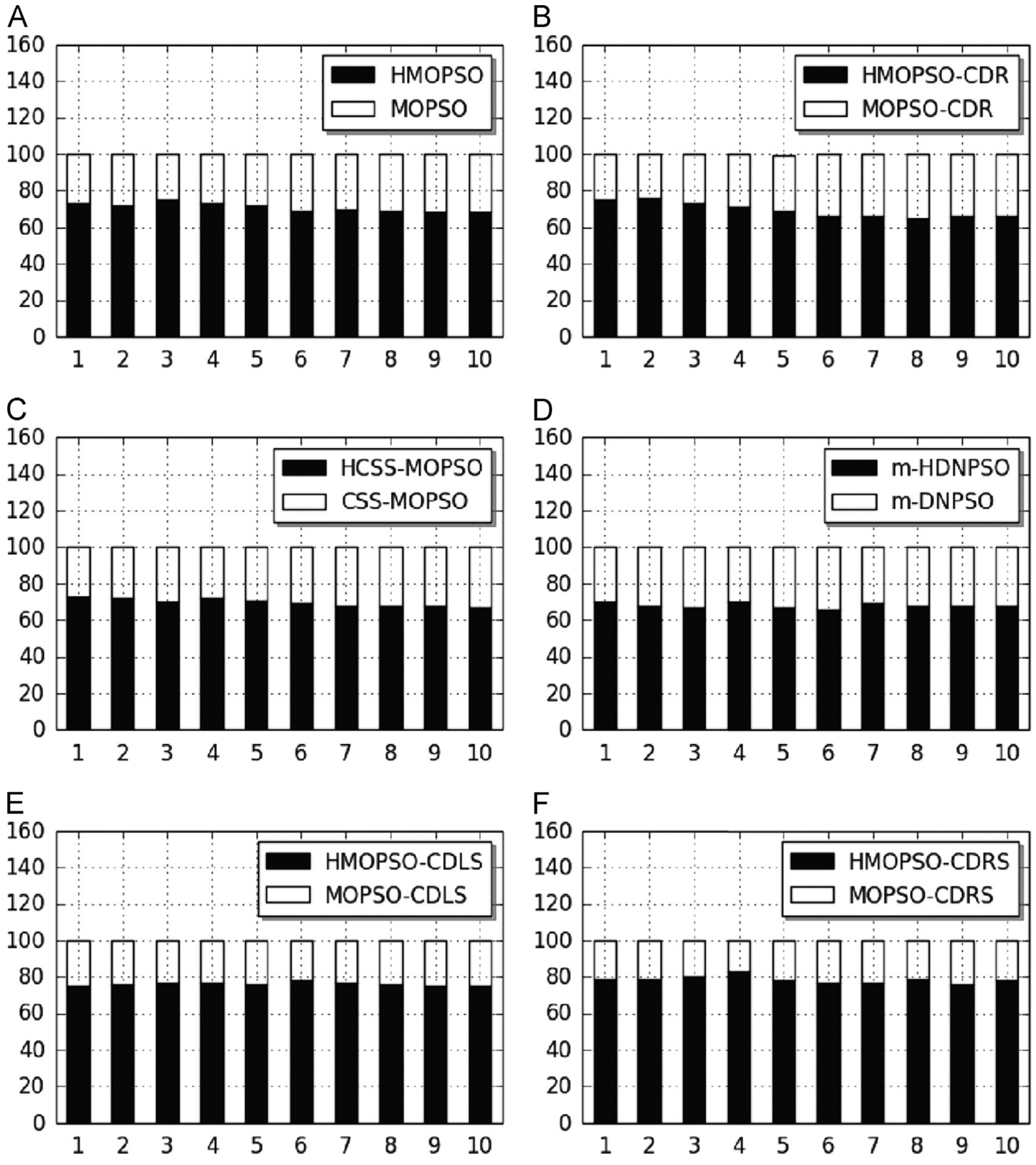


Fig. 11. Number of wins of *Spacing* × Iteration, using 10 particles and considering 100 classification problems.

The coverage CV was proposed by Zitzler et al. [32]. CV is evaluated by using the following equation:

$$CV(A,B)=\frac{|\{b\in B;\ \exists a\in A : a\succcurlyeq b\}|}{|B|}, \tag{13}$$

where A and B are two sets of non-dominated solutions. The $CV(A,B)=1$ means that all solutions in B are weakly dominated by A. On the other hand, $CV(A,B)=0$ means that none of the solutions in B are weakly dominated by A. Note that both $CV(A,B)$ and $CV(B,A)$ have to be evaluated, since $CV(A,B)$ is not necessarily equal to $1-CV(B,A)$. If $0<CV(A,B)<1$ and $0<CV(B,A)<1$, then neither A totally dominates B nor B totally dominates A.

6. Results

In order to analyze our results adequately we performed statistical analysis. Initially, we applied the Kolmogorov–Smirnov test to verify whether the data follows a normal distribution. As the test result was negative, we applied the Wilcoxon test [33] to verify our hypothesis: the hybrid approaches are superior to traditional approaches. All the following analyses used this methodology.

Initially we present the results of *Perspective* 1. Fig. 7 presents the results regarding to *SP*. As it can be seen, the meta-learning generated good initial solutions for all the algorithms, contributing for the optimization process. In two algorithms, HMOPSO-CDR and HMOPSO-CDRS we can see that in the first iteration, the Pareto front presented worse quality when compared to MOPSO-CDR and MOPSO-CDRS respectively. This happened due to the diverse nature that already exists in both algorithms. As the meta-learning also uses a diverse mechanism to suggest, the diversity is increased in these two algorithms. However, this bad beginning of HMOPSO-CDR and HMOPSO-CDRS did not influence the simulation as a whole.

Although the meta-learning contribution is visible, it becomes necessary the application of the Wilcoxon test to guarantee, with

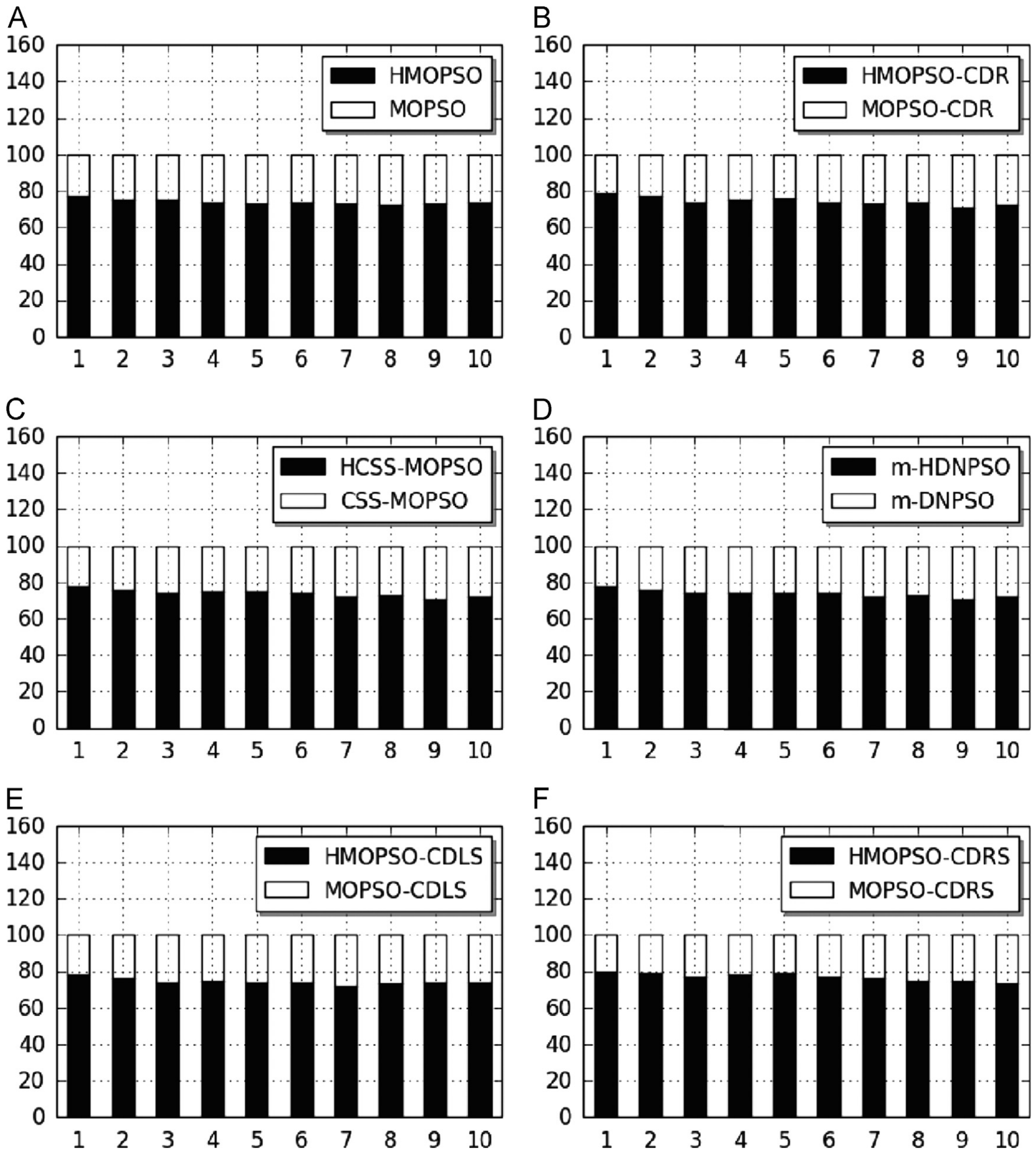


Fig. 12. Number of wins of *Hypervolume* × Iteration, using 10 particles and considering 100 classification problems.

95% of reliability, the superiority of hybrid techniques over traditional ones considering all simulation.

Fig. 8 presents the results considering *hypervolume*. Here, we can clearly see the influence of the meta-learning suggestions in the first iteration. As the mechanism increases the diversity, the solutions tend to be distant from each other expanding the area under the Pareto front. These initial solutions provided by meta-learning contributed for the refinement of the solutions resulting in better Pareto fronts when compared with the traditional algorithms. Considering all iterations, all hybrid algorithms achieved, with 95% of reliability, better results than their traditional versions.

Fig. 9 presents the results for *Max. Spread*. Considering the first iteration of all algorithms, it is visible that meta-learning provided a good start for all of them. Despite this advantage, the nature of MOPSO and m-DNPSO algorithms (both have no diversity mechanisms) did not favor the convergence to a Pareto front with good spread in the last iterations. Thus, the two hybrid approaches are considered equal to their traditional algorithms, considering the last three iterations. On the other hand, the results presented in Fig. 9B, C, E and F show that the suggested initial MA mechanisms and diversity of the native algorithms favored the spreading of the solutions since early iterations to the end.

Fig. 10 presents the results for *Coverage*. This metric is very important because it informs the percentage of dominance between Pareto fronts. The results show the average percentage of dominance achieved by the algorithms per iteration.

As it can be seen in Fig. 10, the hybrid algorithms generated Pareto fronts with a high percentage of dominance since the first iterations. The algorithms HMOPSO, HMOPSO-CDR and HMOPSO-CDRS reached 100% of dominance at the end of the simulation, with respect to their traditional approaches.

Besides *Perspective* 1, we also analyzed the results in *Perspective* 2, which considers the number of wins each algorithm achieved among all problems per iteration. Fig. 11, for instance, presents the number of wins for the *SP* metric. In order to compute

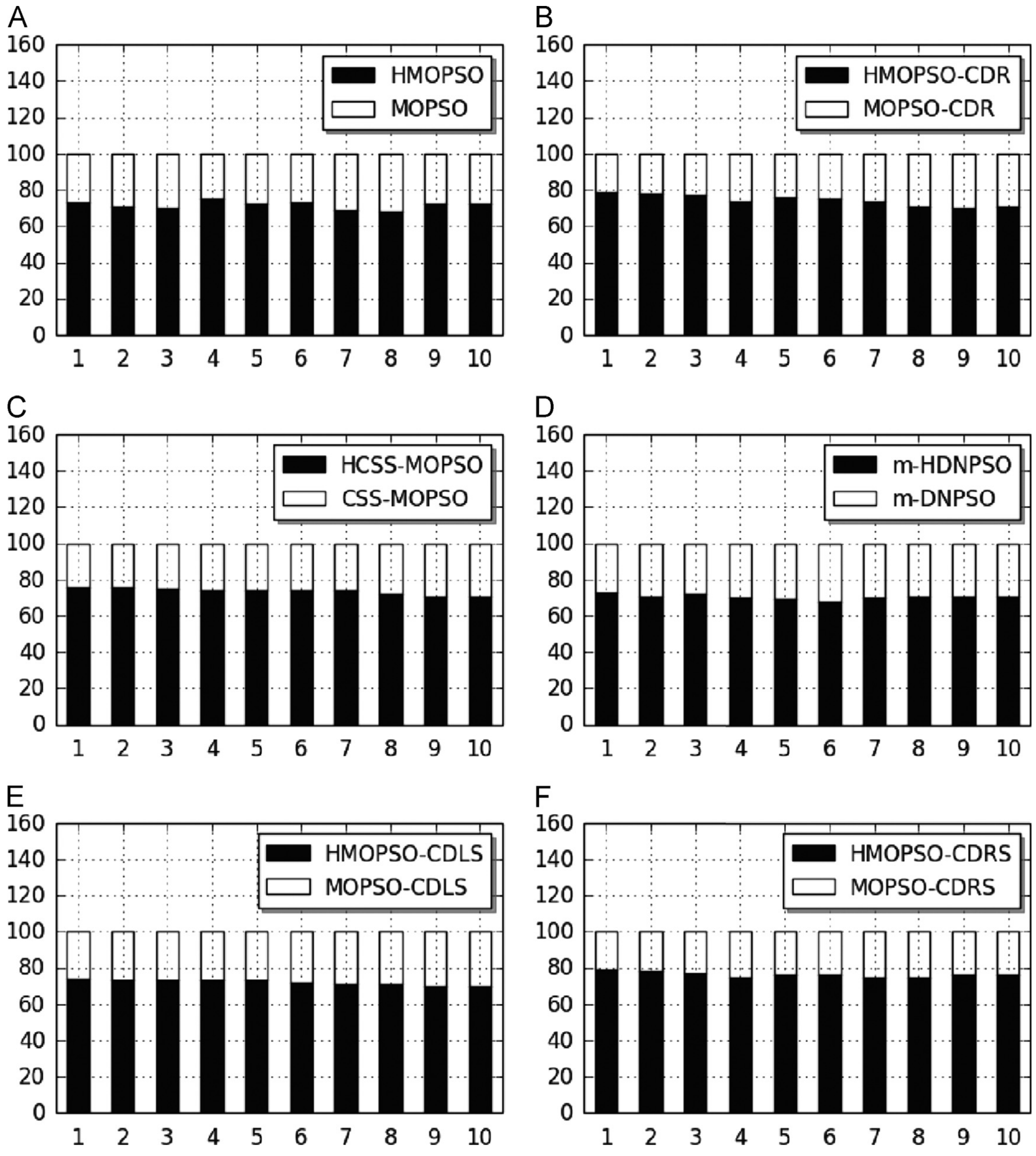


Fig. 13. Number of wins of *Max. Spread* × Iteration, using 10 particles and considering 100 classification problems.

a win for a given problem, we compare the SR achieved by the traditional algorithm with the SR achieved by its hybrid version. If the SR value of the hybrid algorithm is statistically better than the SR value of the traditional algorithm, a win is assigned for the hybrid algorithm. In this figure, the black segment of each bar indicates the number of statistical wins the hybrid approach received for the SP metric among the 100 problems at each iteration. The same methodology was applied to produce Figs. 12, 13 and 14, respectively for metrics Hypervolume, Max. Spread and Coverage.

The results shown in this perspective were in general positive for all metrics considered, making clear the superiority of hybrid algorithms. In the *SP* metric, for instance (Fig. 11), all hybrid algorithms overcome at least 65 of the 100 classification problems in all iterations. The algorithm HMOPSO-CDRS (see Fig. 11F) had the highest number of wins compared to the others, winning in at least 76 classification problems per iteration.

Considering the Hypervolume (Fig. 12), the number of victories of hybrid algorithms was also very expressive. The algorithms HMOPSO, HMOPSO-CDR, HCSS-MOPSO, m-HDNPSO, HMOPSO-CDLS and HMOPSO CDRS won, on average, 74, 74.5, 74, 73.8, 74.4 and 76.9 classification problems, respectively. Concerning the Max. Spread (Fig. 13), in turn, the HMOPSO-CDR, HCSS-MOPSO, HMOPSO-CDLS and HMOPSO-CDRS won 74.5, 73.7, 72 and 76.3 of problems considering all iterations, respectively.

The best results were found considering the Coverage metric (Fig. 14). As it can be seen, the high number of wins among all hybrid algorithms since the first iterations are consequence of the suggestions performed by the ML. The suggestions favored the generation of Pareto fronts with better quality for most classification problems involved. Among the hybrid algorithms, three of them (see Fig. 14A, B and F) achieved total dominance in all the problems involved in the last iterations.

Finally, Fig. 15 presents the mean test error for the 100 classification problems at each iteration. As previously explained, for each classification problem, the algorithms generate a Pareto front per iteration. A non-dominated solution is selected from each Pareto front and its classification test error is evaluated in the

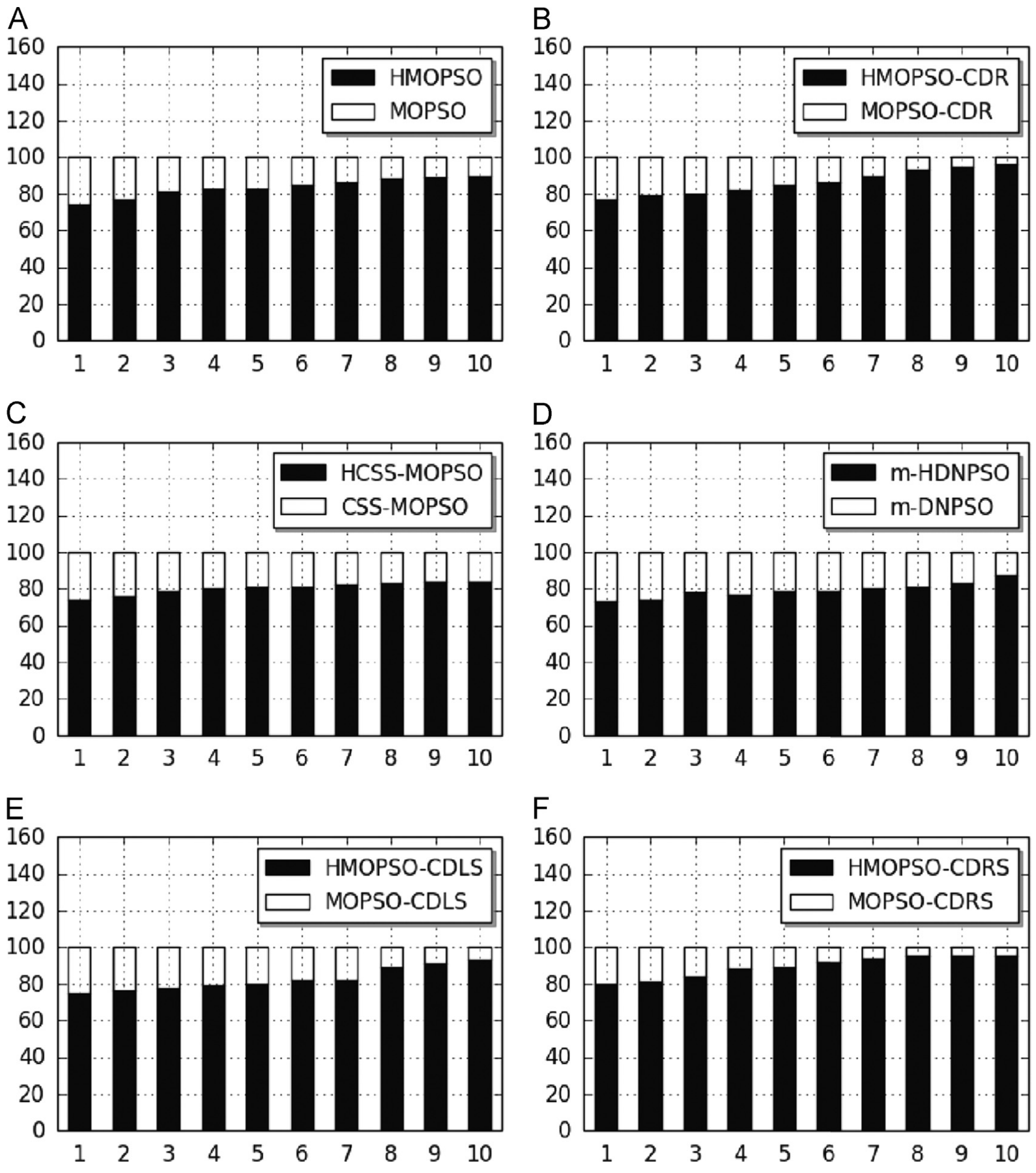


Fig. 14. Number of wins of *Coverage* × Iteration, using 10 particles and considering 100 classification problems.

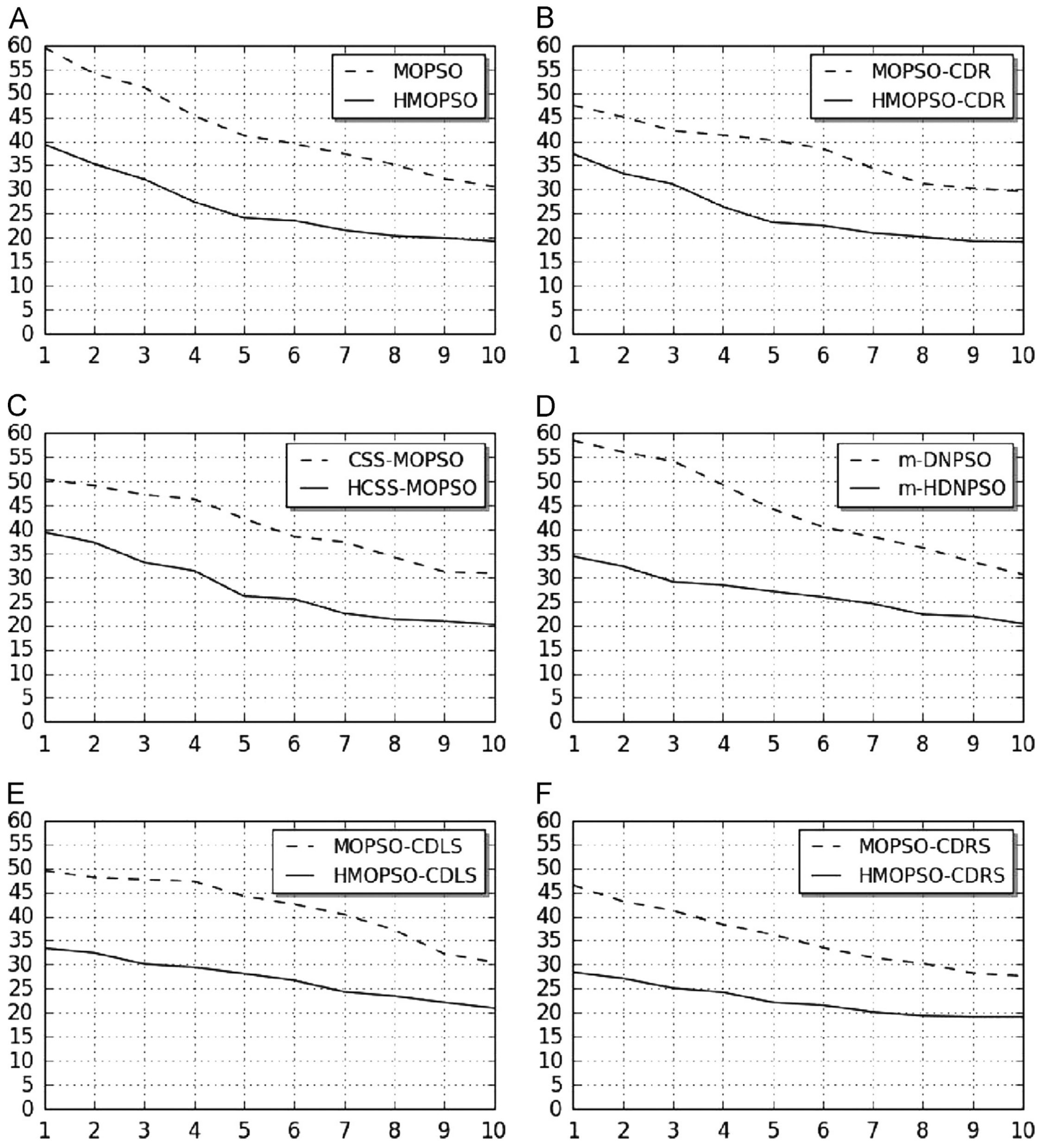


Fig. 15. Mean of testing error × Iteration, using 10 particles and considering 100 classification problems.

current classification problem. As it can be seen in Fig. 15, the hybrid algorithms outperformed the traditional algorithms since the first iteration, confirming the positive influence of the ML technique in the optimization process. We also performed the Wilcoxon test to evaluate this experimental result and the predictive performance of the hybrid approaches overcame the traditional approaches with 95% of confidence considering all experiments.

This information confirms that our proposal has not only generated Pareto fronts with a superior quality, but also the solutions inside the front also have high predictive quality and can be also useful for the problems considered here.

7. Conclusion

In this paper, Meta-Learning (ML) and multi-objective particle swarm optimization algorithms were combined to select the parameter γ of the RBF kernel and the regularization parameter *C*. The ML component selects the solutions that will constitute the initial population for the search algorithm. In the experiments performed, it was possible to observe that the hybrid approaches were able to generate better Pareto fronts when compared to the randomly initialized algorithms, according to four evaluation metrics. Besides, we also investigated the quality of the non-dominated solutions regarding their test classification error and all hybrid approaches obtained better predictive results than the traditional approaches. For future works, we intend to collect more classification problems in order to increase the number of meta-examples in the meta-base and we believe that the performance of the proposed approach can be improved as more meta-examples are considered. Also, supplementary search techniques can be used and different swarm sizes can be tested. Besides, we intend to use other quality metrics to perform better analysis of the Pareto fronts generated, as well as evaluate the proposed solution in other case studies, such as in the SVM parameter selection for regression problems.

Acknowledgments

The authors would like to thank CNPq, CAPES and FAPESP (Brazilian Agencies) for their financial support.

References

[1] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1–3) (2002) 131–159, http://dx.doi.org/10.1023/A:1012450327387.

[2] S. Lessmann, R. Stahlbock, S. Crone, Genetic algorithms for support vector machine model selection, in: IJCNN '06, International Joint Conference on Neural Networks, 2006, pp. 3063–3069. http://dx.doi.org/10.1109/IJCNN.2006.247266.

[3] M.N. Kapp, R. Sabourin, P. Maupin, A pso-based framework for dynamic svm model selection, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, ACM, New York, NY, USA, 2009, pp. 1227–1234. http://dx.doi.org/10.1145/1569901.1570066.

[4] G.C. Cawley, Model selection for support vector machines via adaptive step-size tabu search, in: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, 2001, pp. 434–437.

[5] T. Suttorp, C. Igel, Multi-objective optimization of support vector machines, in: Y. Jin (Ed.), Multi-Objective Machine Learning Studies in Computational Intelligence, vol. 16, Springer, Berlin, Heidelberg, 2006, pp. 199–220.

[6] Y. Lei, X. Huai-Tie, Multi-stage multi-objective evolving svms ensemble using nsga-ii, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), vol. 3, 2010, pp. 856–860.

[7] B. Feres de Souza, A.C.P.L.F. de Carvalho, R. Calvo, R.P. Ishii, Multiclass svm model selection using particle swarm optimization, in: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems, HIS '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 31–. http://dx.doi.org/10.1109/HIS.2006.50.

[8] C. Soares, P. Brazdil, P. Kuba, A meta-learning method to select the kernel width in support vector regression, Mach. Learn. 54 (3) (2004) 195–209, http://dx.doi.org/10.1023/B:MACH.0000015879.28004.9b.

[9] S. Ali, K.A. Smith-Miles, A meta-learning approach to automatic kernel selection for support vector machines, Neurocomputing 70 (2006) 173–186, http://dx.doi.org/10.1016/j.neucom.2006.03.004.

[10] M. Reif, F. Shafait, A. Dengel, Meta-learning for evolutionary parameter optimization of classifiers, Mach. Learn. 87 (3) (2012) 357–380, http://dx.doi.org/10.1007/s10994-012-5286-7.

[11] T.A. Gomes, R.B. Prudêncio, C. Soares, A.L. Rossi, A. Carvalho, Combining meta-learning and search techniques to select parameters for support vector machines, Neurocomputing 75 (1) (2012) 3–13, http://dx.doi.org/10.1016/j.neucom.2011.07.005.

[12] P.B. Brazdil, R.J. Henery, Analysis of results, Machine Learning, Neural and Statistical Classification, Ellis Horwood, Upper Saddle River, NJ, USA, 1994, pp. 175–212.

[13] F. Friedrichs, C. Igel, Evolutionary tuning of multiple svm parameters, Neurocomputing 64 (2005) 107–117, http://dx.doi.org/10.1016/j.neucom.2004.11.022.

[14] C. Soares, P.B. Brazdil, P. Kuba, A meta-learning method to select the kernel width in support vector regression, Mach. Learn. 54 (3) (2004) 195–209, http://dx.doi.org/10.1023/b:mach.0000015879.28004.9b.

[15] C. Soares, P.B. Brazdil, Selecting parameters of svm using meta-learning and kernel matrix-based meta-features, in: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC '06, ACM, New York, NY, USA, 2006, pp. 564–568. http://dx.doi.org/10.1145/1141277.1141408.

[16] S. Ali, K.A. Smith, Matching SVM kernel's suitability to data characteristics using tree by fuzzy C-means clustering, Design and Application of Hybrid Intelligent Systems, IOS Press, Amsterdam, The Netherlands, 2003, pp. 553–562.

[17] S. Ali, K. Smith-Miles, On optimal degree selection for polynomial kernel with support vector machines: theoretical and empirical investigations, Int. J. Know.-based Intell. Eng. Syst. 11 (1) (2007) 1–18.

[18] S. Louis, J. McDonnell, Learning with case-injected genetic algorithms, IEEE Trans. Evol. Comput. 8 (4) (2004) 316–328, http://dx.doi.org/10.1109/TEVC.2004.823466.

[19] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.

[20] K.A. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv. 41 (1) (2009), http://dx.doi.org/10.1145/1456650.1456656 6:1–6:25.

[21] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, 1st ed., Springer Publishing Company, Incorporated, 2008.

[22] R.B. Prudêncio, T.B. Ludermir, Meta-learning approaches to selecting time series models, Neurocomputing 61 (2004) 121–137, http://dx.doi.org/10.1016/j.neucom.2004.03.008, Hybrid Neurocomputing: Selected Papers from the Second International Conference on Hybrid Intelligent Systems.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[24] C.A. Coello Coello, M. Lechuga, Mopso: a proposal for multiple objective particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, CEC '02, vol. 2, 2002, pp. 1051–1056.

[25] X. Hu, R. Eberhart, Y. Shi, Particle swarm with extended memory for multi-objective optimization, in: Swarm Intelligence Symposium, 2003, SIS '03, Proceedings of the 2003 IEEE, 2003, pp. 193–197. http://dx.doi.org/10.1109/SIS.2003.1202267.

[26] S.-Y. Chiu, T.-Y. Sun, S.-T. Hsieh, C.-W. Lin, Cross-searching strategy for multi-objective particle swarm optimization, in: IEEE Congress on Evolutionary Computation, 2007, CEC 2007, 2007, pp. 3135–3141. http://dx.doi.org/10.1109/CEC.2007.4424872.

[27] Ching-Shih Tsou, Shih-Chia Chang and Po-Wu Lai (2007). Using Crowding Distance to Improve Multi-Objective PSO with Local Search, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, FelixT.S.Chan and Manoj KumarTiwari (Ed.), ISBN: 978-3-902613-09-7, InTech, DOI: 10.5772/5098. Available from: http://www.intechopen.com/books/swarm_intelligence_focus_on_ant_and_particle_swarm_optimization/using_crowding_distance_to_improve_multi-objective_pso_with_local_search.

[28] C.R. Raquel, P.C. Naval Jr., An effective use of crowding distance in multi-objective particle swarm optimization, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO '05, ACM, New York, NY, USA, 2005, pp. 257–264. http://dx.doi.org/10.1145/1068009.1068047.

[29] R. Santana, M.R. Pontes, C.J.A. Bastos-Filho, A multiple objective particle swarm optimization approach using crowding distance and roulette wheel, in: Ninth International Conference on Intelligent Systems Design and Applications, 2009, ISDA '09, 2009, pp. 237–242. http://dx.doi.org/10.1109/ISDA.2009.73.

[30] C.J.A. Bastos-Filho, P. Miranda, Multi-objective particle swarm optimization using speciation, in: 2011 IEEE Symposium on Swarm Intelligence (SIS), 2011, pp. 1–6. http://dx.doi.org/10.1109/SIS.2011.5952572.

[31] D.N.A. Asuncion, UCI machine learning repository, 2007. URL: ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.

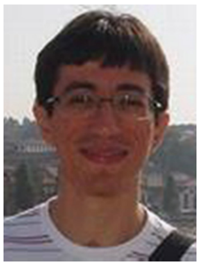
[32] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evol. Comput. 8 (2) (2000) 173–195, http://dx.doi.org/10.1162/106365600568202.

[33] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83, http://dx.doi.org/10.2307/3001968.

[34] Gavin C. Cawley, Nicola L.C. Talbot, Efficient leave-one-out cross-validation of kernel Fisher discriminant classifiers, Pattern Recognit. 36 (11) (2003) 2585–2592.

[35] Emerson Peter, The original Borda count and partial voting, Soc. Choice Welf. 40 (2) (2013) 353–358.

Péricles B.C. Miranda received his B.Sc. degree in Computer Engineering from Universidade de Pernambuco, his M.Sc. degree in Computer Science from Universidade Federal de Pernambuco, Brazil and a Ph.D. candidate in the same university. He is a lecturer at the Department of Statistics and Informatics, Universidade Federal Rural de Pernambuco, Brazil. His main interest areas are Machine Learning, Meta-learning, Optimization Algorithms and Hybrid Intelligent Systems.



Ricardo B.C. Prudêncio received his B.Sc. degree in Computer Science from the Universidade Federal do Ceara and his M.Sc. and Ph.D. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He is a lecturer at the Center of Informatics, Universidade Federal de Pernambuco, Brazil. His main interests are Machine Learning, Meta-learning, Hybrid Intelligent Systems, Time Series Forecasting and Text Mining.

André P.L.F. de Carvalho received his B.Sc. and M.Sc. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He received his Ph.D. degree in Electronic Engineering from the University of Kent, UK. Prof. André de Carvalho is a Full Professor at the Department of Computer Science, Universidade de São Paulo, Brazil. He has published around 80 Journal and 200 Conference refereed papers. He has been involved in the organization of several conferences and journal special issues. His main interests are Machine Learning, Data Mining, Bioinformatics, Evolutionary Computation, Bio-inspired Computing and Hybrid Intelligent Systems.

Carlos Soares received his B.Sc. degree in Systems Engineering and Informatics from Universidade do Minho, Portugal. He received his M.Sc. degree in Artificial Intelligence and his Ph.D. in Computer Science from Universidade do Porto, Portugal. He is a lecturer at Faculdade de Economia da Universidade do Porto. His main interests are Machine Learning, Data Mining, Meta-learning and Data Streams.