Multi-objective Optimization and Meta-learning for SVM Parameter Selection

Péricles B. C. Miranda, Federal University of Pernambuco, pbcm@cin.ufpe.br
Ricardo B. C. Prudêncio, Federal University of Pernambuco, rbcp@cin.ufpe.br
Andre Carlos P. L. F. de Carvalho, University of São Paulo, andre@icmc.usp.br
Carlos Soares, University of Porto, csoares@fep.up.pt

***Abstract*—Support Vector Machines (SVMs) have become a well succeed technique due to the good performance it achieves on different learning problems. However, the performance depends on adjustments on its model. The automatic SVM parameter selection is a way to deal with this. This approach is considered an optimization problem whose goal is to find suitable configuration of parameters which attends some learning problem.**

This work proposes the use of Particle Swarm Optimization (PSO) to treat the SVM parameter selection problem. As the design of learning systems is inherently a multi-objective optimization problem, a multi-objective PSO (MOPSO) was used to maximize the success rate and minimize the number of support vectors of the model. Moreover, we propose the combination of Meta-Learning (ML) with MOPSO to the cited problem. ML is used to recommend SVM parameters, to a given input problem, based on well-succeeded parameters adopted in previous similar problems. In this combination, initial solutions provided by ML are possibly located in good regions in the search space. Hence, using a reduced number of candidate search points, the search process, to find an adequate solution, would be less expensive.

We highlight that, the combination of search algorithms with ML was just studied in the single objective field and the use of MOPSO in this context has not been investigated. In our work, we implemented a prototype in which MOPSO was used to select the values of two SVM parameters for classification problems. In the performed experiments, the proposed solution (MOPSO using ML or Hybrid MOPSO) was compared to a MOPSO with random initialization, obtaining paretos with higher quality on a set of 40 classification problems.

I. Introduction

SVMs have achieved a considerable attention due to its theoretical foundations and good empirical performance when compared to other learning algorithms in different applications [1]. However, the SVM performance strongly depends on the adequate choice of its parameters including, for instance, the kernel function, the values of kernel parameters, the regularization parameter, among others [2]. An exhaustive trial-and-error procedure for selecting good values of parameters is obviously not practical [13].

The process of selecting SVM parameters is commonly treated by different authors as an optimization problem in which a search technique is used to find the adequate configurations of parameters on the problem at hand [3]. Although it represents an automatic mode to select SVM parameters, this approach can still be very expensive, since a large number of candidate configurations of parameters is often evaluated during the search process [1].

An alternative approach to SVM parameter selection is the use of Meta-Learning (ML), which treats the SVM parameter selection as a supervised learning task [1] [4]. Each training example for ML (i.e. each meta-example) stores the characteristics of a past problem and information about the performance obtained by a set of candidate configurations of parameters on the problem. By receiving a set of such meta-examples as input, a meta-learner is able to predict the most suitable configuration of parameters for a new problem based on its characteristics. ML is a less expensive solution compared to the search approach. In fact, once the knowledge is acquired by the meta-learner, configurations of parameters can be suggested for new problems without the need of empirically evaluating different candidate configurations (as performed using search techniques). However, ML is very dependent on the quality of its meta-examples. In the literature, it is usually difficult obtaining good results since meta-features are in general very noisy and the number of problems available for meta-example generation is commonly limited. Hence, the performance of ML for SVM parameter selection may be not so good as the performance of search techniques [5].

A new work, developed by [22], combined ML with a search technique. The proposal is to use ML to suggest an specific number of good solutions and use them as initial population of the search technique. This work used as search technique the single objective version of PSO whose objective was to minimize the error rate. The results showed that the combination of ML with PSO generated better solutions in comparison to PSO with random initialization for SVM parameters selection. Although the good results has been obtained by this combination, the use a single objective search technique is not suitable for SVM parameter selection problem. According to [5], the SVM parameter selection problem is a multi-objective optimization problem. Designing supervised learning systems for classification requires finding a suitable trade-off between several objectives. Typically we want to reduce the complexity of the model and at the same time to obtain a model with high success rate (or low error rate).

In the current work, we propose the combination of a multi-objective search technique based on swarm intelligence and ML to the problem of SVM parameter selection. In this proposal, configurations of parameters suggested by ML are adopted as initial solutions which will be later refined by the search technique. In previous works the search process starts

evaluating solutions randomly sampled from the parameter space (e.g., [6] [7] [8]). In the proposed hybrid method in turn the search process is started from solutions which were well-succeeded in previous similar problems. Hence, we expect that ML guides the search directly to promising regions of the search space, thus speeding up the convergence to good solutions.

In order to evaluate our proposal, we implemented a prototype to select two SVM parameters: the parameter γ of the RBF kernel and the regularization constant C, which may have a strong influence in SVM performance [9].

In our work, a database of 40 meta-examples was produced from the evaluation of a set of 399 configurations of (γ, C) on 40 different classification problems. Each classification problem was described by a number of 8 meta-features proposed in [29]. In the implemented prototype, the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm [11] was used as the search technique to optimize the parameters (γ, C) regarding the two conflicting objectives: complexity and success rate on classification [16][17][5][18]. In our experiments, we evaluated the MOPSO in two different versions: (1) MOPSO with initial population suggested by ML (Hybrid MOPSO) and (2) MOPSO with random initial population. The experiments' results revealed that the hybrid method was able to generate good pareto fronts during the generations when compared to the randomly initialized MOPSO.

Section II brings a brief presentation on the SVM model selection and this problem as a multi-objective problem. Section III presents details of the proposed work. Section IV describes the experiments and obtained results. Finally, Section V presents some conclusions and the future work.

II. SVM Parameter Optimization

The SVM parameter selection task is commonly realized by evaluating a range of different combinations of parameters and retaining the best one in terms of performance and complexity metrics using the problem's dataset [12]. So, different authors proposed search and optimization techniques to automatize and to avoid an exhaustive or a random exploration of parameters [13][3][6][7][8][12][14].

Initial works had faced the SVM parameter selection as a single objective problem [13][15][9] by trying to reduce the error rate. However, as it was discussed in last section, the SVM selection problem is a multi-objective problem which requires suitable solutions for divergent objectives.

Although the use of search techniques automatize the process of parameter selection, this solution may still be very expensive since for each configuration being evaluated during the search it is necessary to train the SVM [1]. The impact of this limitation can be even more drastic depending on the problem at hand and the number of parameters to be optimized.

ML is an alternative that has been studied in recent years to SVM parameter selection [1][19][10][20][21][4]. In this approach, the choice of parameters for a problem is based on well-succeeded parameters adopted to previous similar problems. In ML, it is necessary to maintain a set of meta-examples where each meta-example stores: (1) a set of features (called meta-features) describing a learning problem; and (2) evaluations of a set of candidate parameters on the problem. A meta-learner is then used to acquire knowledge from a set of such meta-examples in order to recommend (or predict) the adequate configurations of parameters for new problems based on the problems' characteristics.

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 in terms of computational cost. However, ML is very dependent on the quality of its meta-examples. In the literature, it is usually difficult obtaining good results since meta-features are in general very noisy and the number of problems available for meta-example generation is commonly limited. Hence, the performance of ML for SVM parameter selection may be not so good as the performance of search techniques [5].

Thus, as we mentioned before, a recent study was performed combining ML with optimization algorithms [22] in such a way that ML is used to recommend parameters which will be later refined by a search technique. This research handled the SVM parameter selection task as a single objective problem and achieved good results.

As the SVM parameter selection problem is a multi-objective problem, this work proposes to use the combination of ML with multi-objective search techniques.

A. Multi-Objective Optimization

In contrast to single objective approaches, 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 decision variable vector which satisfies constraints and optimizes a vector of functions whose elements represent objective functions.

A general multi-objective optimization minimization problem can be defined as [23]:

$$\text{minimize } \vec{f}(\vec{z}) := [f_1(\vec{z}), f_2(\vec{z}), ..., f_n(\vec{z})], \tag{1}$$

subject to:

$$g_i(\vec{z}) \leq 0 \quad i = 1, 2, ..., p, \tag{2}$$

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

where $\vec{z} = (z_1, z_2, ..., z_m) \in \mathbb{R}^n$ is the vector on the decision search space; n is the number of objectives and $g_i(\vec{z})$ and $h_j(\vec{z})$ are the constraint functions and $p + q$ is the number of constrains of the problem.

Given two vectors $\vec{z}$, $\vec{u} \in \mathbb{R}^n$, $\vec{z}$ dominates $\vec{u}$ (denoted by $\vec{z} \prec \vec{u}$) if $\vec{z}$ is better than $\vec{u}$ in at least one objective and $\vec{z}$ is not worse than $\vec{u}$ in any objective. $\vec{z}$ is not dominated if does not exist another current solution $\vec{z}_i$ in the current population, such that $\vec{z}_i \prec \vec{z}$.

The set of non-dominated solutions in the objective space is known as Pareto Front. This work treats the SVM Parameters Selection problem as a multi-objective problem, with two

conflicting objectives, whose goal is to generate SVM model with high performance rate and low complexity. The first objective is to maximize the success rate, and the second objective is to minimize the number of support vectors.

III. Developed Work

SVMs have a strong generalization power, however the performance and complexity of these algorithms depends on an adequate choice of their parameters. Many approaches have been developed to deal with this problem such as optimization techniques and ML. However as we mentioned before, although these solutions, individually, automate the parameter selection process, some other problems may arise. A new approach developed by [22] showed that the combination between search techniques and ML performs well and overcome the cited problems. However, this approach was restricted for single objective.

This work proposes a new method to automate the design of SVMs based on the combination of ML and Multi-Objective search techniques. In this proposal, a meta-learner suggests the initial search points from well-succeeded parameters of a problem which is similar to the problem at hand. As discussed in [24], good solutions to a particular search problem can be used to indicate promising regions of the search space for similar problems. Hence, we expected that the initialization provided by ML enables the search technique to speed up its convergence to good solutions.

Figure 1 presents the general architecture of the proposed solution. Initially, the Meta-Learner module retrieves a predefined number of past meta-examples stored in a Database (DB), selected on the basis of their similarity to the input problem. The process of suggesting meta-examples is not trivial as in a single objective approach. As we are dealing with a multi-objective problem, the dominance evaluation, showed in the previous section, defines the set of non-dominated solutions among all configurations. After that, the Meta-learner is able to perform the suggestion of non-dominated meta-examples.

Following, the Search module adopts as initial search points the configurations of parameters which were well-succeeded on the retrieved meta-examples. The Search module iterates its search process by generating new candidate configurations to be evaluated in the SVM. The output configuration of parameters will be the best one generated by the Search module up to its convergence or another stopping criteria.

In the current work, we implemented a multi-objective particle swarm optimization algorithm (MOPSO) to select two specific SVM parameters: the γ parameter of RBF kernel and the regularization parameter C. The choice of RBF kernel is due to its flexibility in different problems compared to other kernels [25][26]. It is known that the γ and C parameters have an important influence in learning performance since they control the linearity and complexity of the induced SVM respectively [26]. As it will be seen, the current prototype was implemented to select the parameters (γ, C) for classification problems according two conflicting objectives: success rate and number of support vectors. The success rate is the most direct way to evaluate the quality of a SVM model and the number of support vectors influences the space and time complexity of the SVM, so it is important to have a SVM model which has few number of support vectors [5]. Details of implementation will be presented in the next subsections.

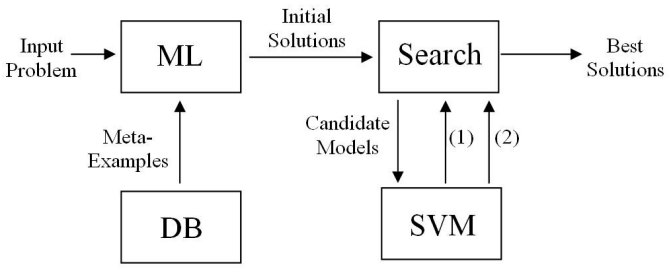

Fig. 1. General Architecture, where (1) is the success rate and (2) the number of support vectors.

A. Search Module

In our prototype, we implemented the version of the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm originally proposed by [11] and adapted here to perform the search for configurations (γ, C). The objective functions evaluated the quality and complexity of each configuration of parameters on a given classification problem. In our work, given a SVM configuration, we defined the objective functions as the success rate (SR) and the number of support vectors (NSV) obtained by the SVM in a 10-fold cross validation experiment. So, the objectives of MOPSO were to find the non-dominated configurations of (γ, C) trying to maximize the SR and minimize the NSV for a given classification problem.

In our MOPSO implementation, each particle i represents a configuration x_i = (γ, C), indicating the position of the particle in the search space. Each particle also has a velocity which indicates the current search direction performed by the particle. MOPSO basically works by updating the position and velocity of each particle in order to progressively explore the best regions in the search space. The update of position and velocity in the basic MOPSO is given by the following equations:

$$\vec{v_i}(t+1) = w\vec{v_i}(t) + c_1 r_1(\vec{p_i}(t) - \vec{x_i}(t)) + c_2 r_2(\vec{n_i}(t) - \vec{x_i}(t)), \quad (4)$$

$$\vec{x_i}(t+1) = \vec{x_i} + \vec{v_i}(t+1), \quad (5)$$

In Equation 4, $\vec{p_i}(t)$ 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 PSO equation, however, the process of updating the $\vec{n_i}(t)$ makes each particle moving in direction of the best global positions achieved from the repository of non-dominated solutions (Pareto Front) through roulette wheel. The parameters ω, $c1$ and $c2$ control the trade-off between exploring good global regions in the search space and refining

the search in local regions around the particle. In equation4, r_1 and r_2 are random numbers used to enhance the diversity of particle positions. In our prototype, we fixed MOPSO parameters using ω = 0.8, $c1$ = 2 and $c2$ = 2.

In our work, the MOPSO was implemented to perform a search in a space represented by a discrete grid of SVM configurations, consisting of 399 different settings of parameters γ and C. By following the guidelines provided in [26], 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 x 21 = 399 different combinations of parameters in the search space.

B. Meta-Database

To create meta-examples, we collected 40 datasets corresponding to 40 different classification problems, available in the WEKA project [27] and UCI Repository [28]. Each meta-example is related to a single classification problem and stores: (1) a vector of meta-features describing the problem; and (2) the objective grid which stores the success rate and number of support vectors obtained by the SVM in the search space of configurations (γ, C). The objective grid consists of 399 different settings of parameters γ and C.

1) Meta-Features: In this work, we used 8 meta-features to describe the datasets of classification problems. These meta-features were based on the set of features defined in [29], which proposed 16 meta-features for classification problems. However, according the restrictions (numerical, no missing values, no binary attributes) of the selected databases, we adopted the meta-features listed in Table I, which is divided in 3 parts: Simple, Statistical and Information Theory meta-features.

TABLE I
META-FEATURES FOR CLASSIFICATION PROBLEMS.

Simple
Number of examples
Number of attributes
Number of classes
Statistical
Mean correlation of attributes
Skewness
Kurtosis
Geometric mean of attributes
Information Theory
Entropy of class

2) Objective Grid: The objective grid stores the success rate obtained by the SVM on a problem considering different SVM configurations. For each of the 399 configurations, a 10-fold cross validation experiment was performed to collect SVM performance and the number of support vectors. The obtained 399 objective values were stored in the objective grid. In these experiments, we deployed the Scikits Learn library [26] to implement the SVMs and to perform the cross-validation experiments.

We highlight here that the objective grid is equivalent to the search space explored by MOPSO. By generating a performance grid for a problem, we can evaluate which configurations of parameters were the best ones in the problem (i.e., the best points in a search space) and we can use this information to guide the search process for new similar problems.

C. Meta-Learner

Given a new input problem described by the vector $\vec{i} = (i_1, ..., i_p)$, the Meta-Learner selects the k most similar problems according to the distance between meta-attributes. The distance function (dist) implemented was the Euclidean Distance, defined as:

$$dist(vec_i, vec_l) = \sum_{j=1}^{p} \sqrt{(i_j - l_j)^2}. \quad (6)$$

After that, we apply the dominance evaluation in the 399 configurations and generate a pareto front for each the k most similar problems. In order to suggest an initial population, we select one random solution of each produced Pareto Front. Random non-dominated solutions of different problems were sampled in order to enhance diversity of the initial population.

IV. EXPERIMENTS

In this section, we present the experiments which evaluated the proposed solution on the set of 40 classification problems considered in our work. The proposed solution was evaluated by following a leave-one-out methodology described bellow.

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 DB to be selected by the ML module. A number of k meta-examples were suggested by the ML module as the initial MOPSO population (in our experiments, we adopted k = 5). The MOPSO then optimized the SVM configurations for the problem left out up to the number of 10 generations. In each generation, a Pareto Front (repository of non-dominated solutions) is formed and to evaluate its quality we applied three metrics. This procedure was repeated 30 times to guarantee reliability.

This experiment was divided in two parts: 1) the number of wins of the algorithms per generation regarding all problems, where the algorithm which achieved better quality (according to an specific metric) is the winner and 2) the mean of the metrics values of all problems for each generation. As a basis of comparison, we used for each value of k a randomly initialized population for MOPSO. Despite its simplicity, the random initialization has the advantage of performing a uniform initial exploration of the search space. Finally, we highlight that each evaluated version of MOPSO was executed 30 times and the average results were recorded.

A. Metrics

In our experiments, we evaluated the results (i.e., the Pareto Fronts) obtained by the algorithms Hybrid MOPSO (HMOPSO) and MOPSO for each problem according to different quality metrics usually adopted in the literature of multi-objective optimization. The following metrics were adopted in this paper: Hypervolume, Spacing and Maximum Spread. One should attempt that each metric considers a different aspect of the Pareto Front.

1) Hypervolume HV*:* this metric was proposed by Zitzler and Thiele [30] and is defined as the hypervolume in the space of objectives covered by the obtained Pareto front ($\mathcal{P}^*$). For MOP with w-objectives, *HV* is defined by:

$$HV = \left\{ \bigcup_i a_i \,|\, vecv_i \in \mathcal{P}^* \right\}, \tag{7}$$

where $vecv_i$ ($i = 1, 2, ..., n$) 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 $vecv_i$ 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 hypervolume is desired.

*2) Spacing (*S*):* this metric was proposed by Schott [31]. S estimates the diversity of the achieved Pareto Front. S is derived by computing the relative distance between adjacent solutions of the Pareto Front as follows:

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2}, \tag{8}$$

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. $S = 0$ means that all solutions of the Pareto Front are equally spaced. Hence, values of S near zero are preferred.

3) Maximum Spread (MS): it 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 Eq. (9).

$$MS = \sqrt{\sum_{m=1}^{P} (max_{i=1}^{n} f_m^i - min_{i=1}^{n} f_m^i)^2}, \tag{9}$$

where n is number of solutions in the Pareto front and k is the number of objectives.

This measure can be used to compare the techniques and thus define which of them covers a bigger extension of the search space. Hence, large values of this metric are preferred.

B. Algorithms Settings

In this section, we present the values of the parameters adopted for the HMOPSO and the MOPSO algorithms. For that, we adopted the same values suggested by Coello *et. al* [11]:

- number of particles: 5
- mutation rate: 0.5
- inertia factor ω: linearly decreases from 0.9 to 0.4
- constants c_1 and c_2: 1.49
- Pareto Front limit: 10 solutions
- number of generations: 10

C. Results

As we mentioned before, two analysis were realized: the number of wins of the algorithms (considering the performance metrics) per generation for all problems and the mean of the metrics values of all problems for each generation. Figures 2, 3 and 4 show the number of wins of the HMOPSO, MOPSO and Random regarding Hypervolume, Spacing and Maximum Spread respectively. This analysis intend to count the number of classification problems that each technique won per generation. According the first experiment, the Random approach lost in all problems, for this we only present the wins of HMOPSO and MOPSO.

Figure 2 shows that the pareto front of HMOPSO overcame the pareto front formed by MOPSO, in all generations, increasing the area under the curve or Hypervolume metric. The same success is showed in Figure 3 for the Spacing metric. In most of the problems, the HMOPSO generated well-distributed pareto fronts regarding the pareto fronts of MOPSO. Depending on the problem, good solutions can be positioned closely decreasing the Spacing metric.

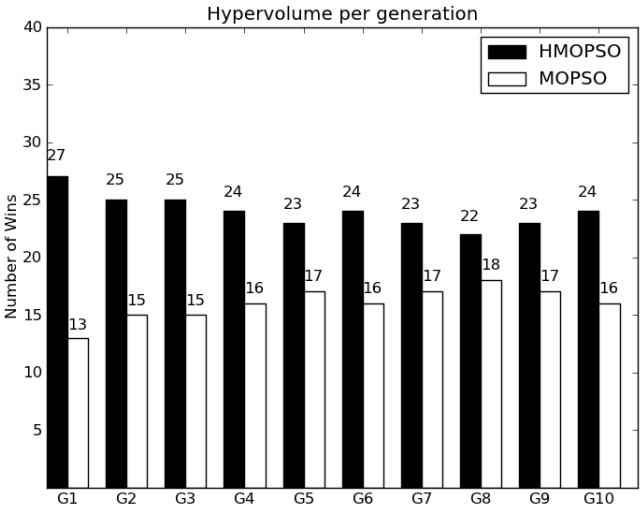

Fig. 2. Number of wins regarding Hypervolume using $k = 5$ per 10 generations.

The good results obtained by HMOPSO for Spacing and Hypervolume was not observed when we considered the Maximum Spread metric. As MOPSO is initialized randomly, its diversity helped to produce solutions more spread in comparison to HMOPSO. Despite of this drawback, along the search, the HMOPSO found solutions which were as spread as the solutions found by MOPSO.

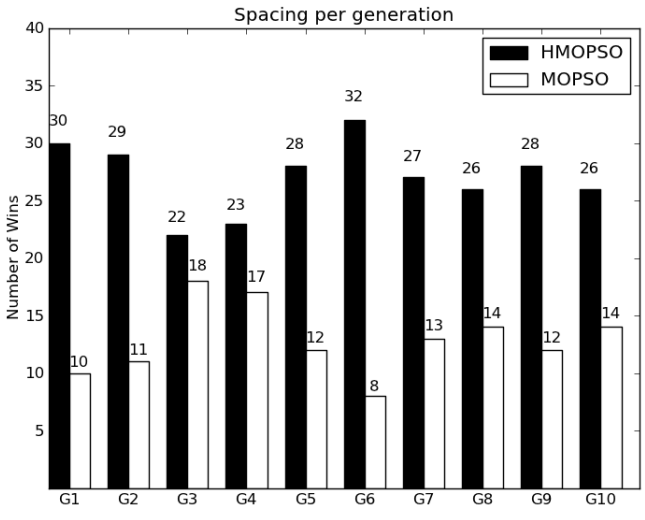

Fig. 3. Number of wins regarding Spacing using $k = 5$ per 10 generations.

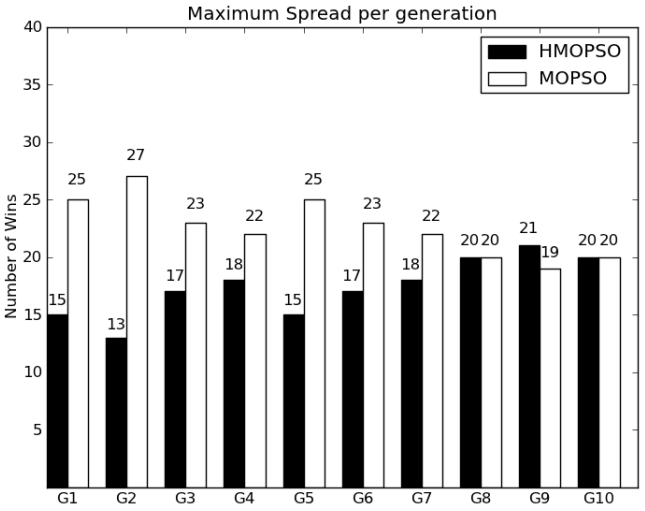

Fig. 4. Number of wins regarding Maximum Spread using $k = 5$ per 10 generations.

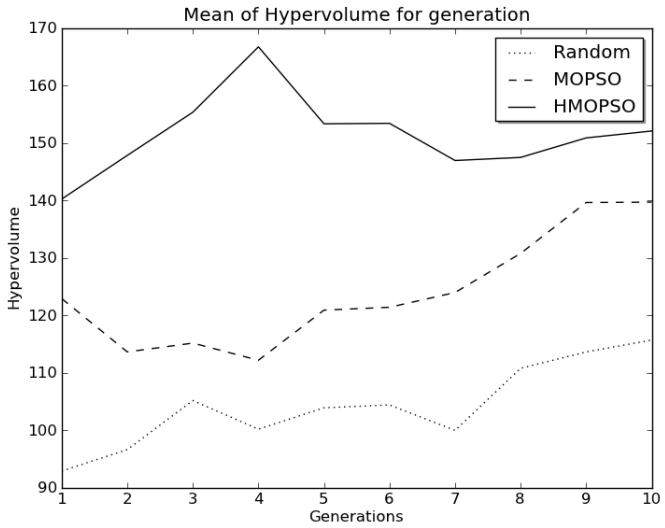

Fig. 5. Average of Hypervolume metric values per generation using $k = 5$ per 10 generations.

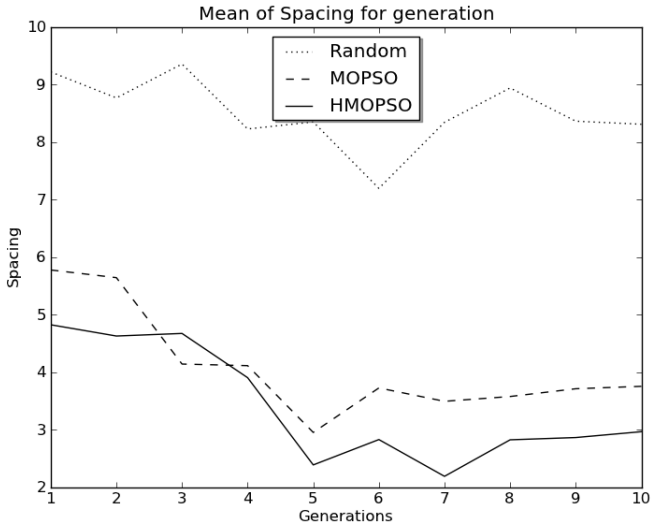

Fig. 6. Average of Spacing metric values per generation using $k = 5$ per 10 generations.

Figures 5, 6 and 7 show the mean of the metrics' values per generation including all problems. These figures compare the Random approach, MOPSO and HMOPSO regarding the quality of the pareto front per generation.

In all cases the Random approach was outperformed by MOPSO and HMOPSO. Moreover, as it can be seen, the HMOPSO achieved interesting results for Hypervolume and Spacing, overcoming the MOPSO in most of the generations. The influence of ML-based suggestions along the optimization had a positive effect in most of the selected problems. For instance, Figure 5 shows the mean performance of the pareto fronts according to Spacing metric, and as we can see, the HMOPSO refined the search forming a well distributed pareto front. Besides generate a better spaced pareto than MOPSO, the HMOPSO augmented the area under curve or HV metric of its pareto front, overcoming the MOPSO in all generations, see in Figure 6.

Regarding Maximum Spread, although the HMOPSO has been overcome in initial generations, it achieved pareto fronts with a similar quality in comparison to MOPSO, considering the last generations.

To compare them adequately, we realized two statistical tests: normality and hypothesis test. The Anderson-Darling method was used to test the data's normality using 5% of significance. After verifying the non-normality of the data, we applied the Wilcoxon test, a non-parametric test, to verify the hypothesis. To perform this test we present the results of the HMOPSO as μ_1 and the MOPSO as μ_2, defining the following hypothesis for Spacing metric:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu 1 < \mu_2.$$

The null hypothesis considered is that both samples are

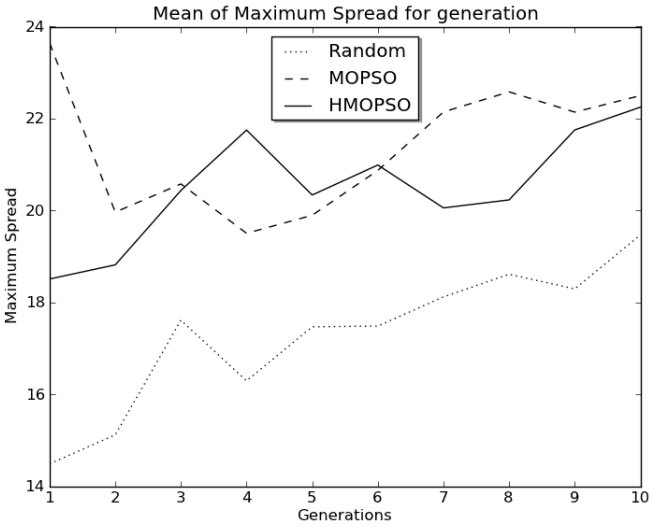

Fig. 7. Average of Maximum Spread metric values per generation using $k = 5$ per 10 generations.

equal, and the alternative one is that the first sample is inferior (spacing must be minimized) to the second sample, using 5% of significance.

In the case of Hypervolume and Maximum Spread, the hypothesis is defined as follows:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu 1 > \mu_2.$$

The null hypothesis considered is that both samples are equal, and the alternative one is that the first sample is superior (Hypervolume and Maximum Spread must be maximized) to the second sample, using 5% of significance.

If the *p-value* results in a value less than 5% of significance, so the null hypothesis is rejected and the alternative accepted. Table II shows the values of p and as it can be seen, the results of *p-value* for Spacing and Hypervolume metrics presented a value less than 5%, rejecting the null hypothesis. However for Maximum Spread, the *p-value* was not enough to reject the null hypothesis. Thus, we cannot prove HMOPSO's results was better in comparison to MOPSO. Hence, we conclude that HMOPSO's values of spacing and hypervolume showed in Figure 5 and 6 are statistically better than MOPSO ones with 95% of certainty.

TABLE II
RESULTS OF WILCOXON TEST USING SIGNIFICANCE LEVEL OF 5%

	Metrics		
	Hypervolume	**Spacing**	**Maximum Spread**
p	0.005062	0.009344	0.168806

V. CONCLUSION

This work combined Meta-Learning and a multi-objective search technique to the problem of SVM parameter selection. The MOPSO was used as the search technique to select the parameter γ of the RBF kernel and the regularization parameter C. In our implementation, 40 classification problems were used to create the meta-database generating meta-examples. In the performed experiments, we observed that the proposed approach was able to generate better paretos compared to a randomly initialized MOPSO, according Spacing and Hypervolume metrics, using 95% of reliability. Regarding maximum spread, our approach lost in initials generations, achieving equality to MOPSO in the last generations.

In future work, we intend to augment the number of meta-examples since reducing the noisy data we believe that the performance of the proposed approach can be improved. Also, different search techniques can be considered in the future implementations in order to trying to improve the maximum spread. Moreover, other quality measures could be studied to evaluate better the pareto front.

Finally, we intend to evaluate the proposed solution in other case studies, such as in the SVM parameter selection for regression problems.

ACKNOWLEDGMENT

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

REFERENCES

[1] C. Soares, P. Brazdil, and P. Kuba. A meta-learning approach to select the kernel width in support vector regression. (4):195–209, 2000.

[2] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods., Cambridge University Press, 2000.

[3] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. *NIPS*, 204–210, 1998.

[4] S. Ali and K. Smith-Miles. On optimal degree selection for polynomial kernel with support vector machines: Theoretical and empirical investigations. *KES Journal*, 1(1):1–18, 2007.

[5] G. Narzisi. An Experimental Multi-Objective Study of the SVM Model Selection problem.

[6] S. Lessmann, R. Stahlbock, and S. Crone. Genetic algorithms for support vector machine model selection. *International Joint Conference on Neural Networks*, 3063–3069, 2006.

[7] F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 2005.

[8] A. Lorena and A. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomputing*, pages 16–18.

[9] S.S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. IEEE Transactions on Neural Networks, 2002.

[10] P. Kuba, P. Brazdil, C. Soares, and A. Woznica. Exploiting sampling and meta-learning for parameter setting support vector machines. *Proceedings of the IBERAMIA*, pages 217–225, 2001.

[11] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[12] G. Cawley. Model selection for support vector machines via adaptive step-size tabu search. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, 28(8):434–437, 2001.

[13] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 131–159, 2002.

[14] B. de Souza, A. de Carvalho, and R. Ishii. Multiclass svm model selection using particle swarm optimization. *Sixth International Conference on Hybrid Intelligent Systems*, 2006.

[15] T. Glasmachers and C. Igel. Gradient-based adaptation of general gaussian kernels. Neural Comput., 2005.

[16] C. Igel. Multiobjective Model Selection for Support Vector Machines, in Proc. of the 3rd Int. Conf. on Evolutionary Muilt Criterion Optimzation, 2005, pp. 534-546.

[17] C. Igel and T. Suttorp Multi-objective optimization of support vector machines, in Yaochu Jin (Ed.), Multi-objective Machine Learning Studies in Computational Intelligence, Vol. 16, pp. 199-220, Springer-Verlag, 2006

[18] Y. Jin and B. Sendhoff, Pareto-Based Multi-Objective Machine Learning : An Overview and Case Studies, IEEE Transactions on Systems, Man and Cybernetics: Part C, vol. 38, no. 3, pp. 397-415, 2008.

[19] S. Ali and K. A. Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, 173–186, 2006.

[20] C. Soares and P. Brazdil. Selecting parameters of svm using meta-learning and kernel matrix-based meta-features. *SAC*, 2006.

[21] S. Ali and K. A. Smith. Matching svm kernel's suitability to data characteristics using tree by fuzzy c-means clustering. *Third International Conference on Hybrid Intelligent Systems*, 2010.

[22] T. Gomes, R. B. C. Prudencio, C. Soares, A. Rossi, A. Carvalho. Combining meta-learning and search techniques to svm parameter selection, in: Brazilian Symposium on Neural Networks, 2010, pp. 7984.

[23] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. Wiley, jun 2001.

[24] S. Louis and J. McDonnell. Learning with case-injected genetic algorithms. IEEE Transactions on Evolutionary Computation, 2004.

[25] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[26] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. Scikit-Learn: A Toolkit of Machine Learning in Python, Available: http://scikit-learn.org.

[27] WEKA, The University of Waikato, Available: http://www.cs.waikato.ac.nz/ml/weka/.

[28] Frank, A. Asuncion, A. UCI Machine Learning Repository, Avaiable: http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

[29] Kate A. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection, 2005.

[30] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study," in *Conference on Parallel Problem Solvin from Nature Algorithms (PPSN V)*, 1998, pp. 292–301.

[31] J. R. Schott, "Fault tolerant design using single and multi-criteria genetic algorithms," Master's thesis, Department of Aeronautics and Astrounatics, Massacgusettes Institute of Technology, Bosto, MA, 1995.

[32] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation Journal*, vol. 8(2), pp. 125–148, 2000.

[33] M. Takaki, D. Cavalcanti, R. Gheyi, J. Iyoda, M. d'Amorim, and R. B. Prudêncio, "A comparative study of randomized constraint solvers for random-symbolic testing," in *Proceedings of the Nasa Formal Methods Symposium*, 2008, pp. 56–65.