Performance prediction using support vector machine for the configuration of optimization algorithms

Abdellatif El Afia
ENSIAS
Mohammed V University, Rabat, Morocco
Email: a.elafia@um5s.net.ma

Malek Sarhani*
ENSIAS
Mohammed V University, Rabat, Morocco
Email: malek.sarhani@um5s.net.ma

***Abstract*—The aim of this paper is to propose a machine learning approach for predicting the performance of each configuration of optimization algorithms. Our approach consists of advocating making the decision of finding the most suitable configuration on a per-instance analysis based on a supervised machine learning model. That is, it consists of building a support vector machine (SVM) model to predict the performance of each configuration on each instance and then to select the adapted setting depending on the instance. Furthermore, feature selection has been used as a pre-processing step to select the relevant features in order to enhance the predictive capacity of SVM. The experiment consists of predicting algorithm performance metrics for two well known optimization problems using SVM in its continuous and binary form depending on the metric of each problem.**

I. Introduction

Nowadays, the configuration of algorithms is one of the main challenges within optimization problems; the reason lies in the fact that the algorithm performance depends heavily on the chosen parameter values. The problem of configuration in optimization algorithms is as old as the algorithms themselves. However, it has been done in most cases manually following the conventional propositions proposed in the literature. However, these parameters depend on the problem type (unimodal, multi-modal...). That is, each problem has its own specificity and then the necessity of automating this pre-processing step. Even if it is known that the configuration of algorithms is a necessary step in most optimization problems, little effort is spent in the automation of this step.

The aim of the configuration of algorithms is to search for a suitable algorithm in the space of available configurations. More specifically, its aim is to choose the configuration which can lead to the best performance of the algorithm on an instance corresponding to the metric. This problem could be viewed as a general design whose components are selected according to the problem to be solved.

It has been often observed that there is no best configuration (or algorithm in the algorithm selection problem) which can dominate for all problem instances. That is, different configurations may perform better on different instances [1]. Therefore, in contrast of traditional configuration approaches which aims to identify the best algorithm for a given class of instances, we advocate making this decision on a per-instance machine learning. Instance-specific optimization algorithms configuration generalizes both instance algorithm tuning as well as algorithm selection problems. That is, concerning the former, it can be applied to any kind of parameters (continuous or discrete). For the latter, this approach is adapted by considering the first parameter as the algorithm which is chosen, and the remaining parameters as the parameters for these algorithms.

this problem can be viewed as an expert system which can learn from its own experience based on previous results in order to improve its performance. Machine learning algorithms are frequently used for this purpose as they may used to automate the algorithm applicability to each problem instance as they can extract target knowledge from experiences. This automation is very useful to build optimization software with enhanced performance. Below, in Algorithm 1 we present the main steps which enable to build such framework [2]:

Our contribution in this paper is to propose a model based support vector machine incorporating feature selection to predict each configuration performance in order to choose the promising ones. Our approach is adapted also the algorithm selection problem.

The remainder of the paper is organized as follows. Section 2 provides literature review. Section 3 presents the formulation of the configuration problem. Section 4 describes our approach to predict algorithm's performance. Section 5 describes the experimental results and Section 6 is dedicated to conclusion.

II. Literature review

In this section, we start by mentioning that the configuration of algorithms can be done in two ways: the off-line configuration involves the adjustment of parameters before running the algorithm while the online configuration consists of adjusting the algorithm parameters while solving the problem. On the one hand, in [3], we have presented a brief review of the online configuration of an example of these algorithms which is particle swarm optimization (PSO) [4]. In this paper we interest on the offline configuration of optimization algorithms. Firstly, the autonomous search book edited by [5] has surveyed the main approaches which can be used for parameters tuning of optimization algorithms especially in constrained problems. In particular, [6] has presented the three most used procedures for tuning algorithms which are: Racing procedures, ParamILS and Sequential Minimization Optimization (SMO).

The formal definition of the off-line tuning problem has been popularized by [7]. Furthermore, The author have proposed a machine learning approach named as the racing procedures to solve this problem. The main interest in our previous paper [8] was given to finding the best configuration of one of these algorithms which is particle swarm optimization. Hidden Markov Model has been used to evaluate each configuration in order to maximize the utility function metric.

Concerning the use of machine learning in offline configuration, it can be integrated in racing procedures and SMO. For SMO, it uses the model predictive distribution for θ to compute its expected positive improvement over the best configuration seen so far which is large for configurations θ with low predicted cost and for those with high predicted uncertainty.

In particular, concerning the issue of performance prediction, IBM has just been presented a work which aims at using the surrogate regression (decision tree) to predict algorithm performance [9]. The modelization has been given using genetic programming.

A necessary step while using machine learning for tuning is feature extraction and feature selection: purpose is to extract a set of predictors of the problem [7].

More practically. Nowadays, it is known that one of the main challenges within solvers is to propose an automatic design for the configuration of their proposed algorithms. Two recently proposed solvers have interested in this issue which are SatZilla [1] and Sunny [10]. The former is an algorithm portfolio for propositional satisfiability problem (SAT) that select solvers on a per-instance basis according to the performance prediction , while the latter is an algorithm selector designed for constraint programming (CP) in general.

Concerning the algorithm performance, different metrics can actually be adopted to evaluate the algorithm effectiveness. For both SatZilla and Sunny solvers, the proposed performance evaluation was the algorithm's runtime. It has been computed for each instance. Concerning feature selection, it has been done especially by removing the correlation between them which can be done just by classical statistical approaches such as principal component analysis. Moreover, [11] Applied backward feature selection approach using Weka software in order to filter features as a pre-processing step for instance specific algorithm Configuration. The authors affirmed that a subset features may be enough to reach similar performance of the proposed Sunny solver that uses all the available features.

In this paper, we interest on the performance prediction phase. Machine learning methods which are adapted to this problem are the supervised ones as they can be used to predict the performance of a single algorithm under different parameter settings and choose the best setting on a per-instance basis. As the use of machine learning for predicting algorithm performance rely on learning problem features, support vector machine (SVM) can be adopted to provide a model with better generalization capacity to elaborate more accurate prediction of algorithm performance, In contrast to artificial neural network (ANN) which is based on the empirical risk minimization, SVM is based on the structural risk minimization. Indeed, it is important that the learner does not overfit. So, we choose support vector machine as an adequate machine learning technique to extract knowledge from all collected features of data. Our motivation behind the use of SVM is its successful applications in many fields, which are similar to our problem. That is, by using a complete set of historical data rather than a sample, SVM may analyze the data set in its entirety for knowledge discovery. It has been widely investigated in many similar classification and regression problems.

A common problem when using machine learning in general and SVM in particular consists in using features that do not properly characterize dataset of a given problem. Thus, feature selection (FS) may be useful in some dataset. Indeed, FS may improve the performance of the predictors by eliminating irrelevant inputs. it may also achieves data reduction for accelerated training, increases computational efficiency [12]. Also, it reduces the risk of over-fitting and enhance generalization capacity of the model.

III. The proposed configuration framework

Without lack of generality, we present a methodology for building an algorithm configuration based on feature learning. This approach can be adapted to the algorithm selection approach by replacing an algorithm by the set of algorithms and replacing a configuration by a candidate algorithm.

The classical off-line algorithm configuration or parameter tuning problem has been defined by [7]. However, in this paper, as in sequential minimization optimization (SMO) [13], we interest on finding the adapted configuration on each instance as it is rare to obtain a configuration which dominate in all instances.

Our configuration approach is based on three steps as presented in Algorithm 2. In the first one, we generate all required elements for configuring the algorithm by machine learning. In the second one, we construct a model to predict each algorithm configuration performance. In the last one, we use that model to select the promising configuration on each instance.

This step is in accordance with Algorithm 1 which presents the main steps for the automatic configuration of solvers. That is, the instance generation correspond to step 1 and feature extraction to step 2 and 3 (the configuration is considered as a feature while building the SVM model). The fitting phase correspond to steps 4 and 5 and the selection phase to step 6.

In this paper, we interest on the second phase of Algorithm 2 which consists of building a model to predict the algorithm

Algorithm 1: The proposed optimization algorithm configuration framework

```
Data: The algorithm (A), The tuned parameters P,
      Candidate configurations (C), Metric (M)
Step 1: Generation phase ;
        - Problem instances generation;
        - Feature extraction;
Step 2: Fitting phase ;
        - Feature selection ;
        - Building SVM model with the selected
        features ;
        - Predicting each configuration performance by
        fitting the model ;
Step 3: Selection phase ;
        - Choose the best configuration on each
instance
Result: The best combinaison of configurations
```

performance on each problem instance. That is, our aim is to predict each algorithm configuration performance using SVM including feature selection and therefore to choose the best one.

IV. Prediction of algorithm performance using machine learning

The aim of this section is to present how SVM has been used to predict each algorithm configuration performance.

Our approach consists firstly of selecting the relevant features based on the prediction of SVM accuracy. Then, the prediction of performance is done using SVM. Our approach can be executed in both binary and continuous forms of SVM (this approach has been recently investigated in a continuous problem [14] in which it takes the historical informations to predict the future values).

Furthermore, we present the main steps which can be used to test our approach which are feature pre-processing, feature selection, training SVM model and testing it.

A. Feature pre-processing

The first step while using SVR for prediction is to normalize the dataset within the range of [0,1] (as proposed by Libsvm package [15]). Indeed, this phase is an important step in Machine Learning, which can significantly improve the prediction accuracy of the learned hypothesis. Moreover, while using machine learning, the examples should be necessary divided into the two subsets: the training data and the testing data. That is, the training set is used to build the model and the testing set is used to evaluate it on a new data.

This preliminary step can be summarized in Algorithm 3 (where ”[n]” is the integer part of the number ”n”).

B. Feature selection

In our approach, We have inspired from feature selection methods to automatically select from among candidate features of each dataset. This idea has been mentioned by [16]. However, it has not been developed by the authors. In this paper, we interest on wrappers which have been popularized by [17] as they hybridize between a machine learning method for classification and an algorithm adapted to search in the space of feature subsets which is a binary space [12]. Furthermore, while using a wrapper approach, we can obtain simultaneously the predicted values and the corresponding features. In this paper, FS is grounded on support vector machine which has proved important predictive ability in feature selection. Concerning the searching algorithm, we have chosen the particle swarm optimization (PSO). In this paper, we interest in using the wrapper FS approach which has been proposed in [18] to select the relevant features based on predictive accuracy on the training set. The optimization algorithm is based on binary particle swarm optimization (PSO) as depicted in [18]:

C. Support vector machine

The support vector machine (SVM) is a recent tool from the artificial intelligence field which use statistical learning theory. It has been successfully applied to many fields and it recently of increasing interests of researchers: It has been first introduced by [19] and was applied firstly to pattern recognition (classification) problems, recent research has yielded extensions to regression problems.

Concerning the algorithm configuration problem, it can be used in most problems in its continuous form (support vector regression) to predict the algorithm performance which can be the running time or the obtained cost by each algorithm configuration. However, for some constrained problems. The performance can be a binary function (positive and negative values) in which the negatives ones may lead to failure and the positive ones are those made along a successive search on feasible solutions [20].

D. Training and testing the SVM model

In this section, our main interest is given in this section to SVR (the regression version of SVM) as most performance measurement are continuous ones (runtime). In this paper, we have adopted the Libsvm package to train the SVR model as described in Algorithm 4:

Algorithm 2: Training support vector regression by Libsvm

```
Data: Training set $(x_1, y_1), ..., (x_n, y_n)$ with n instances,
      the SVR parameters ($\sigma$ , $C$ and $\epsilon$ )
Step 1: Solve the dual problem ;
Step 2: Compute weight vector w ;
Step 3: Compute $b^*$ value ;
Step 4: Compute f value ;
Result: Weight vector w, Lagrangian multipliers $\alpha$ and
        $\alpha^*$, $b^*$ and function f
```

After building the SVR model on the training set, we test its prediction capacity on the testing set. The prediction accuracy is measured on the testing set by the mean absolute percentage error (MAPE) and the mean square error (MSE).

$$MAPE = 100.\frac{\sum |(prediction - real)/real|}{n} \quad (1)$$

$$MSE = 1/n \sum (prediction - real)^2 \quad (2)$$

Where n is the number of instances of the testing set.

E. The proposed approach for the prediction of algorithm performance

The aim of this section is to illustrate the used process to test Our approach for predicting the algorithm configuration performance. That is, the input correspond to the initial dataset and by following the four steps, the algorithm is tested on the testing set as depicted in Algorithm 6.

V. Experiment

In this section, the parameter setting is presented in the beginning. After, deal with three dataset which belong to the propositional satisfiability problem and we investigate it on constrained satisfaction problem:

A. Parameters setting

We compare our approach with SVM model when using just FS and with the native SVR model without using FS and with the decision tree in both regression and classification form (as proposed by Matlab software) which has been proposed in sequential minimization optimization (SMO). These problems belong to the algorithm selection problem which can be modeled as an instance configuration problem as described in the introduction.

We choose the values of SVM as follows: $C = 1000$ and $\sigma = 0.25$. Furthermore, an additional parameter ϵ for SVR is set to 0.1 (these parameters can be set using an optimization algorithm as in [21])

In our experiment, 75% of samples are used in training while the remaining 25% samples are used in testing.

B. Performance measurement of our approach on the propositional satisfiability problem

First of all, the paper takes the dataset evaluated by the SatZilla solver These dataset have been used in the SAT competition. In this paper, we evaluate our approach on three of these data sets (which has been respectively named as ”Indu” (industrial), ”Hand” (handmade), ”Rand” (random)). In each one of them, a number of training instances and features have been generated. Concerning feature extraction of SAT problem, it have been basically based on 91 features. These features can be classified into the following categories: problem size features, graph-based features, balance features, proximity to horn formula features, DPLL probing features, and local search probing features) as detailed in [22]. Our adopted datasets have included other features ([2],...).

For instances generation, it is based on a collection of SAT instances that include each algorithm instance from these three datasets.

Concerning the choice of performance measurement, the algorithm runtime has been adopted in this experiment. That is, the aim of this dataset is to predict each algorithm (solver) performance on the different instances. Therefore, the algorithm ID is considered as a feature to determine the corresponding algorithm for each instance of the problem.
Then the dataset can be presented as follows: Y correspond to the algorithm runtime and X to the features of each problem in addition to the algorithm ID.

C. Results

The following table shows different performance measurement obtained for the three methods: (”SVR-FS” correspond to our approach, ”SVR” means that we use SVR model without using feature selection and ”Tree” means the regression tree) for the ”Indu” dataset.

TABLE I
Comparison of results for ”Indu” experiment

	SVR-FS	SVR	Tree
MAPE	158.2031	159.2822	**128.4223**
MSE	**5.04183e+06**	$5.04246e+06$	$5.9208e+06$

To provide a graphical view of the forecasting performance of the proposed model, we depict in Figs 1 and 2 the obtained plots.

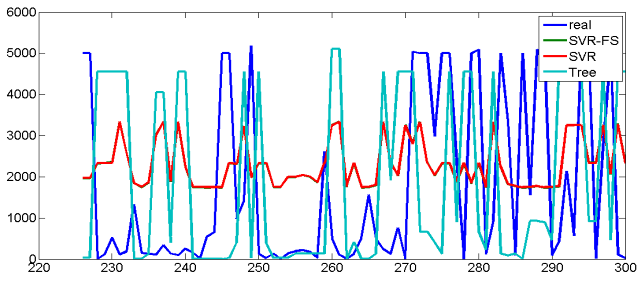

Fig. 1. Comparison of predicted values for ”Indu” dataset

In these plots, we compare the predicted values of the three models on the testing set with the real ones. Furthermore, we show the errors of each model on the testing set.

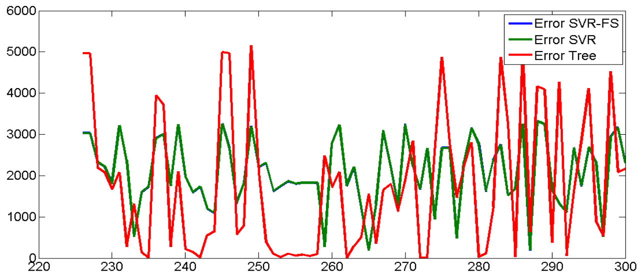

Fig. 2. Comparison of errors for ”Indu” dataset

Furthermore, to better illustrate feature selection impact of SVR, we display Fig. 3 a more zoomed comparison of errors to present the slightly difference between "SVR-FS" and "SVR" (in the instance 263):

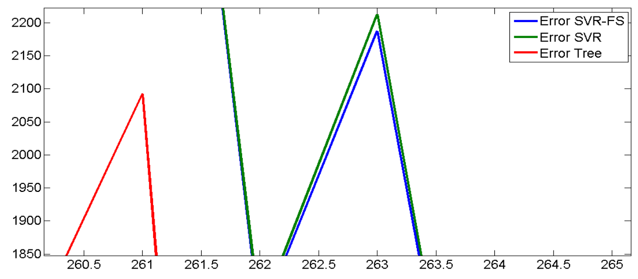

Fig. 3. Errors difference "Indu" dataset

In the same manner, we depict the different obtained performance measurement in Table II for the "Hand" dataset:

TABLE II
COMPARISON OF RESULTS FOR "HAND" EXPERIMENT

	SVR-FS	SVR	Tree
MAPE	39.1014	39.1225	**21.0181**
MSE	**5.05064e+08**	$5.05081e+08$	$5.05064e+08$

Furthermore, the obtained results are presented in Fig. 4.

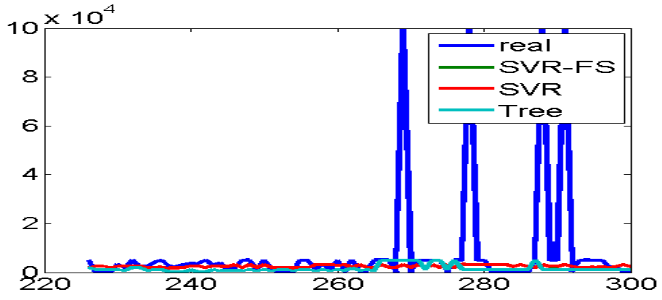

Fig. 4. Comparison of predicted values for "Hand" dataset

The corresponding errors are depicted in Fig. 5

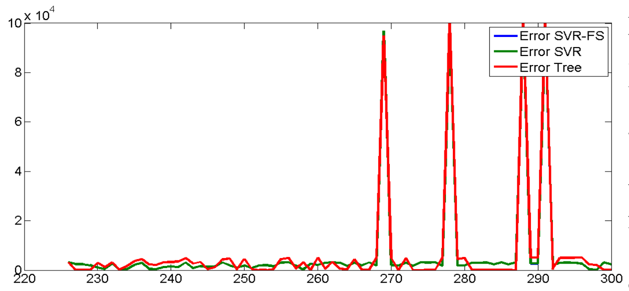

Fig. 5. Comparison of errors for "Hand" dataset

TABLE III
COMPARISON OF RESULTS FOR "RAND" EXPERIMENT

	SVR-FS	SVR	Tree
MAPE	632.3718	**627.8543**	1.7821e+03
MSE	**5.32186e+06**	$5.32703e+06$	$6.7584e+06$

The results of the last dataset are shown in Table III.

Furthermore, their corresponding predicted values are depicted in Fig 6.

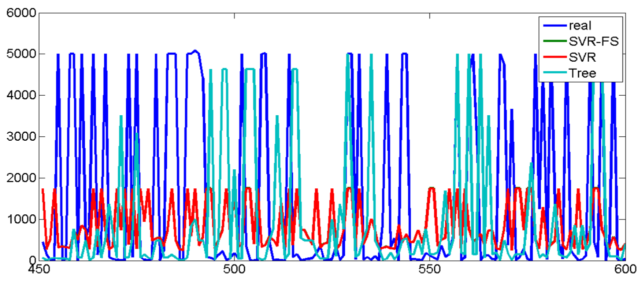

Fig. 6. Comparison of predicted values for "Rand" dataset

The errors are presented in 7:

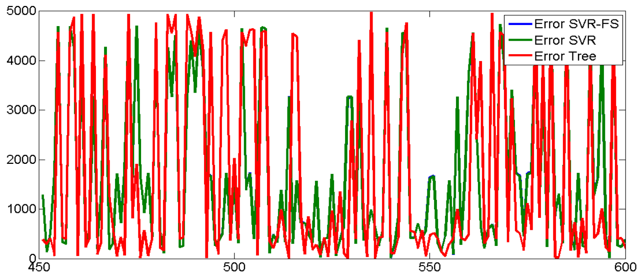

Fig. 7. Comparison of errors for "Rand" dataset

On the one hand, by comparing the two approaches, we can conclude that:
In "Indu" and "Hand" datasets, "SVR-FS" has given a little small MAPE and MSE than "SVR". Even if "SVR-FS" and "SVR" have given better MSE than "Tree", the corresponding MAPE is worst. This means that "Tree" has a smaller percentage of error. but, in large instances, SVR gives more accurate values. In "Rand" dataset, "SVR-FS" and "SVR" have clearly given better results than "tree".

On the other hand, as we can see from Figs. 1, 4 and 6, the testing present a large deviation. That is, the std deviation for "Indu", "Hand" and "Rand" datasets are respectively $2.0750e{+}03$, $2.2085e{+}04$ and $2.1358e{+}03$. The characteristic of these datasets had a great impact on the learning capacity of these methods and thus the necessity to extend the SVR to deal with these kind of sparse dataset.

To summarize, feature selection has slightly improve SVR accuracy and SVR have given competitive results comparing to regression tree.

D. Preliminary investigation of the approach on the constrained satisfaction problem

In this experiment, we deal with constrained satisfaction problem which can be modeled as a classification problem. That is, by fixing the allowed running time for each configuration (algorithm), we verify if it can find a solution to the problem. In this dataset, the two classes are "ok" if the problem instance has been solved with the configuration (positive response) and "memout" (negative ones). These classes can be considered as a metric for this problem.

TABLE IV
COMPARISON OF RESULTS FOR THE SECOND EXPERIMENT

	SVM-FS	SVM	Tree
MAPE	$65.2174\%(\frac{330}{506})$	$65.2174\%(\frac{330}{506})$	**78.06%** $(\frac{395}{506})$

As we can see from Table IV, the results corresponding to SVM are worse than on decision tree. The reason for the inferiority of SVM may be related to the parameters setting of the algorithm.

VI. Conclusion and Future Work

In this paper, we have proposed an approach which uses support vector machine to predict each configuration performance including feature selection. Experimental results on the algorithm selection problem, which is included in the configuration problem, have shown that has competitive computational performance than the regression tree especially in continuous problems. However, due to the very high deviation of the dataset in regression problem, the error is still high. Then, future research could attempt to propose a support vector regression model which is adapted to this kind of sparse dataset in continuous problems. Also, we can more examine the binary SVM on other dataset.

References

[1] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Satzilla-07: the design and analysis of an algorithm portfolio for sat," in *Principles and Practice of Constraint Programming–CP 2007*. Springer, 2007, pp. 712–727.

[2] ——, "Satzilla: portfolio-based algorithm selection for sat," *Journal of Artificial Intelligence Research*, pp. 565–606, 2008.

[3] O. Aoun, M. Sarhani, and A. El Afia, "Hidden markov model classifier for the adaptive particle swarm optimization," in *Recent developments of metaheuristics*, L. Amodeo, E.-G. Talbi, and F. Yalaoui, Eds. Springer, 2017, ch. 1.

[4] J. Kennedy and R. . Eberhart, "Particle swarm optimization," *Proceedings of IEEE international conference neural networks*, vol. IEEE, pp. 1942–8, 1995.

[5] Y. Hamadi, "Autonomous search," in *Combinatorial Search: From Algorithms to Systems*. Springer, 2013, pp. 99–122.

[6] H. H. Hoos, "Automated algorithm configuration and parameter tuning," in *Autonomous search*. Springer, 2012, pp. 37–71.

[7] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, ser. Studies in Computational Intelligence, J. Kacprzyk, Ed. Springer, 2006, vol. 197.

[8] O. Aoun, M. Sarhani, and A. El Afia, "Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems," in *14-th IFAC Symposium on Control in Transportation Systems*. Istanbul, Turkey: IFAC (Elsevier), 2016.

[9] C. Ansótegui, Y. Malitsky, H. Samulowitz, M. Sellmann, and K. Tierney, "Model-based genetic algorithms for algorithm configuration," *24th International Joint on Artificial Intelligence, Buenos Aires, Argentina*, 2015.

[10] R. Amadini, M. Gabbrielli, and J. Mauro, "Portfolio approaches for constraint optimization problems," in *Learning and Intelligent Optimization*. Springer, 2014, pp. 21–35.

[11] R. Amadini, F. Biselli, M. Gabbrielli, T. Liu, and J. Mauro, "Feature selection for sunny: a study on the algorithm selection library," in *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, 2015, pp. 25–32.

[12] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157 – 1182, 2003.

[13] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[14] M. Sarhani, A. El Afia, and R. Faizi, "Hybrid approach based support vector machine for electric load forecasting incorporating feature selection," *International journal of big data intelligence*, vol. Accepted, 2016.

[15] C. Chang and C. ling, *LIBSVM: A Library for Support Vector Machines*, 2001.

[16] D. Bridge, E. O'Mahony, and B. O'Sullivan, "Case-based reasoning for autonomous constraint solving," in *Autonomous Search*. Springer, 2012, pp. 73–95.

[17] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 273 – 324, 1997.

[18] M. Sarhani and A. El Afia, "Facing the feature selection problem with a binary pso-gsa approach," in *Recent developments of metaheuristics*, L. Amodeo, E.-G. Talbi, and F. Yalaoui, Eds. Springer, 2016.

[19] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classiffers," *, 5th Annual ACM Workshop on COLT, Pittsburgh PA*, pp. 144–152, 1992.

[20] S. L. Epstein and S. Petrovic, "Learning a mixture of search heuristics," in *Autonomous Search*. Springer, 2012, pp. 97–127.

[21] M. Sarhani and A. El Afia, “Simultaneous feature selection and parameter optimisation of support vector machine using adaptive particle swarm gravitational search algorithm,” *International journal of metaheuristics*, vol. 5, no. 1, 2016.

[22] E. Nudelman and K. Leyton-Brown, “Understanding random sat: Beyond the clauses-to-variables ratio,” in *Principles and Practice of Constraint Programming–CP 2004*. Springer, 2004, pp. 438–452.