Multi-objective Automatic Algorithm Configuration for the Classification Problem of Imbalanced Data

1st Sara Tari
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
sara.tari@univ-lille.fr

2nd Nicolas Szczepanski
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
nicolas.szczepanski@univ-lille.fr

3rd Lucien Mousin
Lille Catholic University
Faculté de Gestion, Economie et Sciences
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
lucien.mousin@univ-catholille.fr

4th Julie Jacques
Lille Catholic University
Faculté de Gestion, Economie et Sciences
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
julie.jacques@univ-catholille.fr

5th Marie-Eléonore Kessaci
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
marie-eleonore.kessaci@univ-lille.fr

6th Laetitia Jourdan
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
laetitia.jourdan@univ-lille.fr

***Abstract*—Classification problems can be modeled as multi-objective optimization problems. MOCA-I is a multi-objective local search designed to solve these problems, particularly when the data are imbalanced. However, this algorithm has been tuned by hand in order to be efficient on particular datasets. In this paper, we propose a methodology to automatically configure a multi-objective algorithm for solving a supervised partial classification problem. This methodology is based on a multi-objective approach of automatic algorithm configuration and requires a clear definition of the experimental protocol. Therefore, we present a k-fold cross-validation protocol to train and test the configuration model. To the best of our knowledge, it is the first time that multi-objective automatic algorithm configuration is performed on optimization algorithms to solve classification problems. Experimental results on real imbalanced datasets show that our approach can find efficient configurations of MOCA-I with less effort in comparison with the ones found exhaustively by hand.**

***Index Terms*—multiobjective optimization, automatic configuration, partial supervised classification, imbalanced data**

I. Introduction

Classification is a supervised learning problem where known observations are classified within one or more classes, and the goal is to predict the class(es) of a new observation. Applications are, for example, the prediction of the diagnosis of a patient's illness, the belonging to a species of flower.... However, the data may be imbalanced, which means that the number of observations in each class varies a lot between each other. In this case, the data are said imbalanced, and various techniques have been proposed to tackle this particular type of classification problem. An efficient approach was proposed by [1] to solve supervised partial classification problem with imbalanced and discrete data. The authors transformed the original classification problem into a multi-objective optimization and MOCA-I [2], a multi-objective metaheuristic based on multi-objective local search designed to solve this problem proved his great performance on nine imbalanced and discrete datasets of the literature. Nevertheless, MOCA-I was tuned by hand that requires a lot of experimental efforts while automatic algorithm configuration (AAC) approaches could be used for this task.

AAC consists of automating the configuration of highly parameterized algorithms such as metaheuristics to solve both single and multi-objective optimization problems. For example, AAC has been applied to configure stochastic local search algorithms to solve the multi-objective permutation flowshop scheduling problem or the multi-objective traveling salesman problem [3], [4]. In order to solve multi-objective problems, Blot et al. [5] have proposed a pure multi-objective AAC (MO-AAC) approach where multiple optimization performance indicators are optimized simultaneously. In [4], the authors showed that the MO-AAC approach is more appropriate than the classical AAC approach to configure a multi-objective algorithm to solve classical multi-objective combinatorial problems of the literature. The MO-AAC approach has never been tested and used on classification problems.

In this paper, we propose to use MO-AAC to automatically configure the MOCA-I algorithm that solves a supervised partial classification problem with imbalanced and discrete data. The contributions are: (i) the definition of an experimental protocol based on k-fold cross-validation and (ii) an approach

This work is part of the PATHACOV project, funded by the Interreg France-Wallonie-Vlaanderen program, with the support of the European regional development funds.

based on MO-AAC to find efficient configurations of our multi-objective local search algorithm.

The paper is organized as follows. Section II presents the supervised partial classification problem and the MOCA-I algorithm. Section III describes the principles of multi-objective automatic algorithm configuration as well as MO-paramILS. In section IV, we detail our methodology that corresponds to an adaptation of the classical MO-AAC protocol to the context of the partial supervised classification problem. Section V gives the experimental setup and discusses the obtained results. In the last section, we conclude and provide some perspectives on this work.

II. The Supervised Partial Classification Problem and the MOCA-I Algorithm

In this section, we briefly define the multi-objective optimization in order to present the supervised partial classification problem and its modeling as a multi-objective optimization problem. Then, the different strategies of a multi-objective local search (MOLS) are described. Finally, we present MOCA-I that is the adaptation of a MOLS to the multi-objective modeling proposed to solve the supervised partial classification problem.

A. Multi-objective Optimization

In multi-objective combinatorial optimization problems, multiple objective functions are optimized simultaneously. The *Pareto dominance* is a multi-objective approach where no predominance is given between the objectives. Therefore, a solution s_1 *dominates* a solution s_2, if s_1 is better or equal to s_2 according to all objectives and there is at least one objective according to which s_1 is strictly better than s_2. When s_1 is neither dominated by, nor dominating, s_2, the solutions are said *non-dominated*. A set S of non-dominated solutions is called a *Pareto set*, *Pareto front*, or an *archive* in a multi-objective local search context.

B. The Supervised Partial Classification Problem and Its Modeling as a Multi-Objective Problem

In a context of binary classification, a supervised classification task aims to predict the *class* (category, for example, ill or healthy) to which an observation belongs. Observations are described by a variety of information, the *attributes*, that can be of various forms depending on the problem under consideration. For example, in a classification task aiming to determine whether an individual has the flu and in which each observation corresponds to a patient, attributes describe a list of symptoms that each patient may have. A classification task uses labeled observations to generate a *classifier*, that describes a class by using the attributes. The resulting classifier is used to predict the class on new unlabeled observations. In future work, we will work on medical data furnished by different European hospitals. Since most prevalent diseases such as diabetes occur only on 6% of the population, medical datasets are often imbalanced: the class to predict is less frequent than the opposite class. It causes most classification algorithms to fail [6]. Moreover, we focus on partial classification: the aim is to predict observations matching a subset of the class. An example of such a task could be to predict only patients positive to the flu without aiming to find healthy individuals.

Several metrics exist in the literature to assess the efficiency of classifiers, most of which are based on the confusion matrix presented in Table I. The confusion matrix reports values in function of the predictions performed by the classifier and the actual class of observations. *True positives* (TP) and *true negatives* (TN) counts well-classified observation, whereas *false positives* (FP) and *false negatives* (FN) counts misclassified observations. As an illustration, a healthy individual detected as infected by the flu by the classifier is labeled as a *false positive*, while an individual infected with flu not detected by the classifier is labeled as a *false negative*.

TABLE I: Confusion Matrix.

		Prediction	
		positive	negative
Class	positive	TP	FP
	negative	FN	TN

Using the same observations to build and evaluate classifiers can lead to a classifier that mostly describes these observations, and thus has a weak generalization capacity on unknown observations. To limit this phenomenon called *overfitting*, classifiers are commonly evaluated on both known and unknown data. The observations of a given dataset are split into disjoint training and test sets, then the classifier is built using the training set and evaluated using the test set.

1) Modeling as a multiobjective optimization problem: Since a classifier is a combination of attributes, finding the best classifier induces a combinatorial explosion. This characteristic makes optimization techniques suitable to tackle this problem. This part describes the modeling used in this paper: solution representation, objectives, and neighborhood.

We use a *Pittsburgh* representation, where a solution is a ruleset. Each ruleset is composed of rules, which are a conjunction of terms. A term corresponds to an attribute test, composed of an attribute, an operator ($<, >, =$), and a value. The encoding length of a solution may vary, reducing the sensitivity of the model to the number of attributes under study compared to fixed-length encoding.

Since it is a partial classification task, all solutions predict the same outcome of the target class. Thus, it avoids inconsistencies when using rulesets, making this representation well-suited to this case. For a given ruleset, an observation is classified as positive to the target class if it triggers at least one rule.

The values of the confusion matrix are computed using all the individuals in the dataset on which MOCA-I is applied.

To optimize machine learning metrics that are in conflict, and thus attain rulesets that are interesting in a classification context, three objectives are used:

- the *sensitivity* $\frac{TP}{TP+FN} \in [0, 1]$,
- the *confidence* $\frac{TP}{TP+FP} \in [0, 1]$,
- the number of terms

The sensitivity, to be maximized, represents the proportion of samples positive to the class under investigation that are detected by the ruleset. In contrast, the confidence, to be maximized, represents the proportion of samples predicted as positive that are real positives. The number of terms to be minimized aims to reduce complexity and avoid the *bloat* effect that leads rulesets to contain over-specific rules without any improvement of the quality. This effect can quickly occur in variable-length encodings.

Since we tackle the problem using a local search algorithm, a neighborhood relation, which assigns a set of neighboring solutions to each solution, has to be defined. The neighborhood of a ruleset corresponds to the set of all rulesets having one difference in an attribute test: an additional term, a suppressed term, or a difference on the value or operator of a term.

C. Multi-objective Local Search Algorithms

A multi-objective local search (MOLS) is an algorithm where solutions in an archive evolve using a neighborhood relation [7]. A MOLS alternates between several phases: the `selection` of one or more solutions into the archive, the neighborhood `exploration` of the selected solutions, and the `archiving` of new solutions found. A `perturbation` phase may be applied in order to diversify the search. In the following, the different components used in the experiments to iterate these phases are described.

a) Selection: The selection strategy determines which solutions of the archive will be explored being either `all` solutions or one solution at random (`rand`).

b) Exploration: According to a given strategy, the neighborhood of each selected solution is explored, and some visited neighbors are kept. For each selected solutions, *all* explores the whole neighborhood of solutions and keeps all the improving and the non-dominated neighbors while *imp ndom* randomly explores the neighborhood until one improving solution is found and keeps this neighbor as well as the non-dominated neighbors identified so far.

c) Archiving: This phase adds the solutions kept during the exploration phase into the archive following the Pareto rule removing all dominated solutions. The archive is bounded, and new solutions are discarded when the maximal size of the archive is reached.

d) Perturbation: The perturbation phase corresponds to a *restart*, that restarts the search from a new initial population.

D. The MOCA-I Algorithm

The above multi-objective modeling has been implemented and manually configured in previous work, as the MOCA-I algorithm: multi-objective classification algorithm for imbalanced data [2]. In the subsection above, we described basic components of MOLS, in particular, the ones instantiated in our experiments that correspond to those previously considered to manually configure MOCA-I. Additionally, we present some mechanisms specific to MOCA-I that were also previously considered. We consider an additional parameter that specifies the maximum number of rules in rulesets and is thus specific to MOCA-I. Since the purpose of MOCA-I is to produce a classifier, we return a single ruleset among the ones from the final archive. This ruleset is the one of best *F-Measure* (F_1), a machine learning metric computed as follows:

$$F_1 = \frac{2 \times Confidence \times Sensitivity}{Confidence + Sensitivity}$$

F_1 corresponds to the harmonic mean, and thus a trade-off, between confidence and sensitivity. Moreover, the initial population is dependent on the problem and consists of generating rulesets from existing observations in order to obtain rules that match at least one observation [2].

III. Multi-objective Automatic Algorithm Configuration

Automatic algorithm configuration (AAC) aims to automatically determine a configuration (parameter setting) that optimizes the performance of a given algorithm for a given class of problem instances, regularly relatively to a given computational budget. The procedure that configures the target algorithm is called a *configurator* such as *irace* [8], *SMAC* [9], *paramILS* [10] or its multi-objective version *MO-paramILS* [5].

AAC often consists of two phases: the training phase and the test phase. The training phase optimizes the configuration of the target algorithm according to a performance metric for a configuration space Θ on a given set of training instances. This process corresponds to an optimization problem in which the solution is a configuration, and the quality of this solution depends upon the objective being optimized. The test phase assesses the configuration resulting from the training phase on a set of test instances disjoint from the set of training instances. Since it involves a training phase and a test phase, AAC also corresponds to a machine learning process, involving similar challenges: the configuration returned by the process may overfit the training instances and thus not being adapted to the instances to solve (test instances).

A. Multi-objective AAC

AAC generally refers to a mono-objective optimization problem in which the objective is the running time or the quality of the solution. In multi-objective optimization, different performance indicators are used to evaluate the quality of the Pareto front [11]. These indicators often focus on three main properties: the *accuracy*, the *diversity*, and the *cardinality* of the Pareto front. AAC can be easily performed on a multi-objective optimization problem if the objective is to optimize the quality of the indicator chosen [3]. An alternative, called multi-objective AAC, consists of using simultaneously multiple performance indicators to optimize the configuration of a target algorithm. In this context, MO-AAC produces a Pareto set of configurations instead of a single one. Since AAC procedures are stochastic processes, it is usual to perform several runs of their training phase. Consequently, in a multi-objective AAC context, the training phase generates several fronts of configurations, noted $\{\{\theta^*\}\}$. To obtain a single Pareto front of optimal configurations ($\{\theta^*\}$), a validation phase is then performed using (usually) the training instances. Finally, the test phase assesses the optimal configurations of $\{\theta^*\}$ on the test instances for all given performance objectives.

B. MO-paramILS

MO-paramILS [5] is a recent multi-objective extension of ParamILS, a configurator that optimizes a single performance metric using a local search method named iterated local search (ILS). It follows the principles of paramILS by using an ILS to optimize the configuration of the target algorithm, and the main difference between the two configurators lies in the multi-objective aspect of this ILS.

The training phase of MO-paramILS starts from an archive of r randomly generated solutions and evolves an archive of configurations during the search process. Moreover, the incumbent (best-encountered configuration) corresponds to an archive of solutions. During the perturbation phase of the ILS, a single configuration is randomly selected and modified by performing s random moves. As in paramILS, a restart mechanism that can occur with a probability $p_{restart}$ provides additional diversification. The only change in the restart mechanism is that it replaces the current archive with a single configuration randomly chosen.

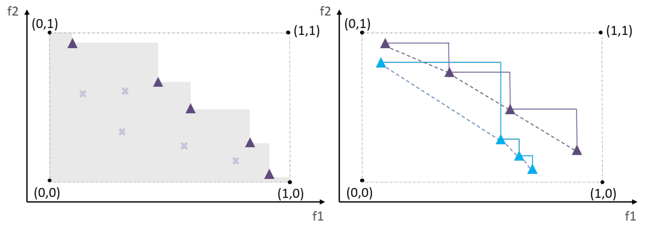

Fig. 1: The normalized hypervolume indicator correspond to the gray area (left side). Illustration of the δ' spread indicator for two Pareto fronts (right side).

The performance metrics being optimized traditionally correspond to the hypervolume [12] and the Δ' spread indicator. The hypervolume indicator (HV), to be maximized, computes the volume of the search space that contains dominated solutions and assesses both the accuracy and the diversity of Pareto sets. In the case of objective values normalized in $[0,1]$, HV measures the volume between a Pareto set and the point $(0,0)$, as depicted in the left part of Figure 1. The Δ' spread indicator, to be minimized, is a variant of the $Delta$ spread indicator [13] and measures the diversity of solutions in Pareto sets. It removes the euclidian distances from the extreme position of the Pareto front to obtain more discriminating values and allow a proper comparison as depicted in the right part of Figure 1. This variant is computed as follows :

$$\Delta' = \frac{\sum_{i=1}^{|S|-1} |d_i - \overline{d}|}{(|S|-1)\overline{d}}$$

where $\overline{d}$ corresponds to the average over the euclidian distance $d_i, i \in [1, |S|-1]$ between adjacent solutions of the ordered front S.

The hypervolume reflects the quality of the Pareto front and gives an idea about the diversity of the solutions in the Pareto front, therefore, considering Δ' ensures a good diversity of the front. Let us note that in the case of MO-paramILS, performance metrics have to be minimized; thus, we consider 1-HV instead of HV.

IV. The Proposed MO-AAC Approach for Classification Problem

In this section, we propose a way to adapt the classical MO-AAC approach [4] used in MO-paramILS to partial classification tasks.

A. Cross-validation for MOCA-I

As MOCA-I performs a supervised classification task, the produced ruleset must be assessed on unknown data to limit the overfitting effect. As finding several datasets for the same classification problem is almost impossible, datasets are split into training and test sets as for classical machine learning algorithms. In particular, rulesets are constructed and assessed following a k-fold cross-validation protocol, as is previous work assessing the efficiency of the algorithm [14].

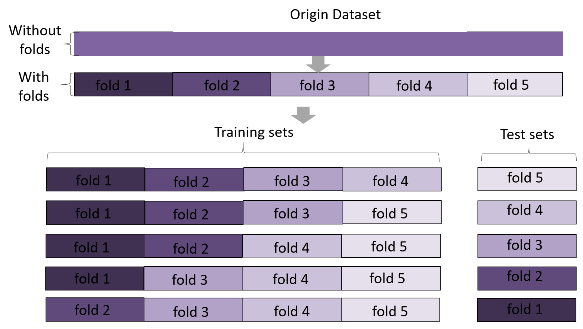

Fig. 2: Illustration of the k-fold cross-validation with $k = 5$.

k-fold cross validation produces k training sets and k test sets by splitting the dataset into k same-size folds. As depicted in Figure 2, each training sets correspond to a unique combination of $k-1$ folds. The remaining fold corresponds to the associated test set. Each training set is then used to construct (at least) a classifier whose efficiency is assessed on the associated test set.

B. MO-AAC Approach

1) Training and Test Instances: Let us recall that in the context of AAC, the set of training instances must be disjoint from the set of test instances while being similar in terms of properties. For classical optimization problems, generators of instances often exist that allows producing instances of

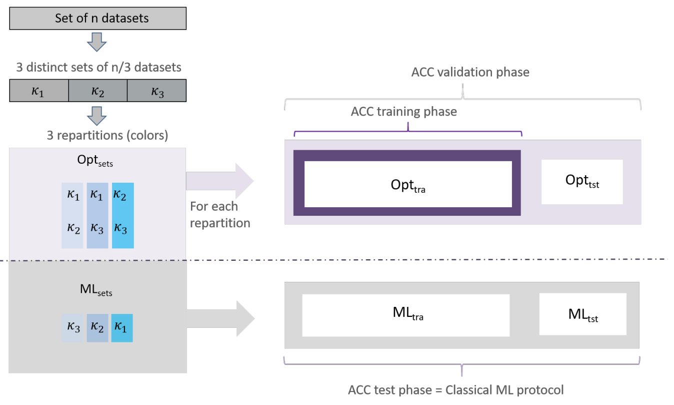

Fig. 3: Repartition of datasets into groups.

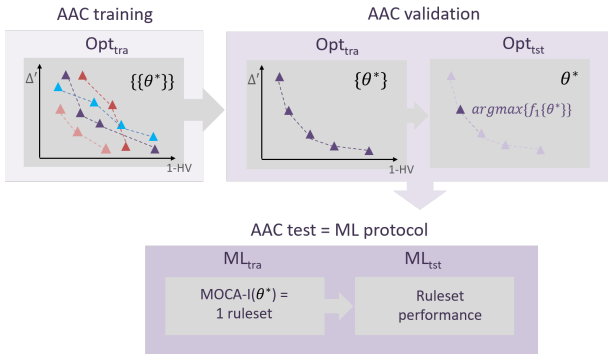

Fig. 4: Illustration of the MO-AAC Protocol. For each phase, the produced configurations are drawn.

the same properties as the tackled instances. However, in a classification context, datasets usually correspond to real data, and finding similar datasets is a tedious, if not impossible, task.

Here, we propose a methodology inspired by a κ-fold cross-validation by using a subset of our datasets to optimize the configuration of the remaining datasets. We repeat the process for each subset in order to find a configuration for each dataset.

Let us note that we deliberately chose not to use some training sets resulting from a given dataset to optimize the configuration on the remaining ones as it majorly contains the same observations and thus would increase the risk to overfit the data.

We divide our set of n datasets into κ equal groups and use $\kappa - 1$ groups for the training and validation phases of the configurator while using the remaining set for the test phase. We repeat the process κ times so that each group is used exactly once in the test phase. As depicted in Figure 3, we call Opt_{sets} the datasets used for the training and validation phases of the configurator, and ML_{sets} the disjoint set of datasets of the test phase.

Following the k-fold cross-validation principle presented in the previous section, we consider that a dataset comprises k training sets and k test sets. We then call Opt_{tra} the set of $k \times (\kappa - 1) \times \frac{n}{\kappa}$ training sets contained in Opt_{sets}, and Opt_{tst} the test sets of Opt_{sets}. The training phase of the configurator only uses Opt$_{tra}$, whereas the validation phase uses Opt$_{tra}$ and Opt$_{tst}$. For ML$_{sets}$, we call ML$_{tra}$ the training datasets used during the test phase of the configurator and ML$_{tst}$ the test sets.

2) *MO-AAC Protocol:* The protocol described in this section is repeated for each couple (Opt$_{sets}$, ML$_{sets}$) and is illustrated in Figure 4.

a) AAC Training: Configurations are being optimized on Opt$_{tra}$ datasets using HV and Δ' as performance indicators. We compute these performance indicators on the final archive of solutions after the removal of dominated solutions in terms of sensitivity and confidence since the number of terms is principally useful to guide the search. Moreover, MOCA-I returns the solution of best *F-Measure*, a metric that corresponds to a trade-off between sensitivity and confidence. As in the classical protocol, we conduct several runs of the training phase to reduce the impact of the stochastic bias, leading to $\{\{\theta^*\}\}$ a set containing several Pareto sets of configurations.

b) AAC Validation: Each configuration of $\{\{\theta^*\}\}$ is run once on each Opt$_{tra}$ dataset. The average values of the performance metrics for each configuration lead to a single Pareto set of *survival* configurations (θ^*). Contrary to the standard protocol, we only select one configuration θ^* among this set for the AAC test phase. This configuration corresponds to the one that leads MOCA-I to the best *F-Measure* on Opt$_{tst}$ dataset, to limit the selection of a configuration that leads MOCA-I to produce rulesets that overfit the data.

c) AAC Test:: Following the presented approach in section IV-B1, the selected configuration is then used to construct the rulesets on ML$_{tra}$ datasets, and their efficiency is assessed on Opt$_{tst}$ datasets.

V. Experiments

In this section, the datasets used in the experiments are first introduced. Then, further details on the experimental setup and the methodology used to compare the results of the exhaustive configuration with the ones obtained with the MO-ACC protocol, are given. Finally, these results are analyzed and discussed.

A. Datasets

The datasets come from the literature but have been transformed into binary classification datasets according to the method proposed in [15] and discretized using the 10-bin discretization method of KEEL software [16]. Note that the resulting datasets correspond to the ones previously tackled with the original MOCA-I algorithm (see Table II). These datasets are unbalanced, confirmed by their degree of asymmetry (d_{asy}), which refers to the proportion of observations having the class to predict. In these experiments, we produced five *training* sets and five *test* sets per dataset in order to follow a 5-fold cross-validation protocol.

TABLE II: Description of the dataset (number of observations, attributes, numerical attributes and degree of asymmetry).

Name	#obs.	#att.	#num.	d_{asy}	ref.
haberman	306	3	3	27.42%	[17]
ecoli1	336	7	7	22.92%	[17]
ecoli2	336	7	7	15.48%	[17]
yeast3	1484	8	8	10.35%	[17]
yeast2vs8	482	8	8	4.85%	[17]
abalone9vs18	731	8	7	5.65%	[17]
abalone19	4174	8	7	0.77%	[17]
lucap0	2000	144	0	27.85%	[18]
a1a	1605	123	0	24.61%	[19]

B. Experimental protocol

Table III presents the configuration space of MOCA-I which represents 96 distinct configurations $((2\times3+1\times2)(3\times2\times2))$.

TABLE III: Configuration space for MOCA-I.

Parameter	Values
Initial population	{50, 100, 200}
Maximum archive size	{100, 300, 500}
Maximum number of rules	{5,10,20}
MOLS selection strategy	{`rand`, `all`}
MOLS exploration strategy	{`imp ndom`, `all`}

Since the configuration space is quite limited, we are able to perform the exhaustive runs in order to assess the efficiency and detect the drawbacks of the proposed AAC approach. In particular, in order to be fair with the AAC approach, we perform 6 runs per folds of datasets and aggregate the performance to have a total of 30 runs per dataset for each configuration. For each dataset, we use an experimentally determined reference runtime as stopping criterion (see Table IV).

For MO-paramILS, as explained in Section IV, 3 groups of datasets are required. Table IV gives the groups (A, B, and C) that have been defined in order to have equivalent runtimes considering the whole approach (training, validation, and test phases). Table V gives the parameters used by MO-ParamILS to train and validate the model.

TABLE IV: Groups of datasets. Runtime (in seconds) of MOLS is given for each dataset.

Group	Dataset	
	Name	runtime
A	lucap0	1 753.8
	ecoli1	35.5
	yeast2vs8	4.8
B	a1a	864.9
	ecoli2	20.3
	yeast3	282.5
C	haberman	27.1
	abalone9vs18	423.1
	abalone19	941.3

C. Results

Experiments have been conducted in parallel on two (24 × 3.0GHz, 64GB RAM) Intel XEON E5-2687W machines.

TABLE V: MO-AAC experimental protocol

Phase	Parameters
Training	No default configuration
	1 random configuration
	10 MO-ParamILS runs
	100 MOLS run budget
	max 10 MOLS run per configuration
Validation	1 run per instance
Test	30 runs per dataset (6/training set)

Group	arch	explore	init pop	ruleset	select	HV	Δ'	F1
AB	**500**	**`imp ndom`**	**100**	**20**	**`rand`**	0.382	0.520	**0.710**
	500	`all`	200	20	`rand`	0.404	0.502	0.674
	500	`imp ndom`	50	10	`rand`	0.388	0.508	0.664
AC	**500**	**`all`**	**100**	**10**	**`rand`**	0.541	0.435	**0.465**
	500	`imp ndom`	50	10	`rand`	0.511	0.458	0.461
BC	**500**	**`all`**	**100**	**10**	**`rand`**	0.557	0.458	**0.450**
	500	`all`	200	20	`rand`	0.557	0.467	0.444

TABLE VI: Non-dominated configurations of the validation for each training group with the returned configuration in bold.

Table VI reports the survival configurations of the AAC-validation phase and gives in bold the one leading to the best average F-measure. Let us remark that this AAC configuration corresponds to the already best-known configuration for the groups AC and BC.

Figure 5 (top) reports the performance of the tested configuration according to the hypervolume and spread metrics on the Opt$_{tra}$ instances for the AAC-validation phase of each group (AB, AC, and BC). All available configurations are plotted: the Pareto ones are plotted with blue circles, the reference configuration with the blue triangle, and the other ones with a gray circle. In red, the survival configurations are plotted with an empty square while the best of them is plotted with a plain one. For the groups AC and BC, the two survival configurations are among the non-dominated configurations while for the group AB, they are very close to the Pareto front. This result shows the performance of our AAC approach.

Figure 5 (bottom) gives the performance of the configuration according to the hypervolume and spread metrics on the ML$_{tra}$ instances for the AAC-test phase of each group (A, B and C). Here, only the selected AAC configuration is highlighted among the other ones. For the groups B and C, the AAC configuration achieves a good hypervolume and spread, following the tendency of the validation phase on the associated training groups. Nevertheless, for group A the quality achieved by the AAC configuration dramatically differs from the one of the validation phase. This variation highlights that MO-paramILS may have overfitted the datasets for this particular group of datasets.

Figure 6 provides boxplots of the average *F-Measure* of all configurations and reports the average *F-Measure* achieved by the AAC configuration on each dataset. For each group, the quality of the *F-Measure* achieved using the AAC configuration compared to the exhaustive configuration is slightly correlated to the quality of the optimization metrics. Furthermore, we performed statistical tests (Friedman and Wilcoxon)

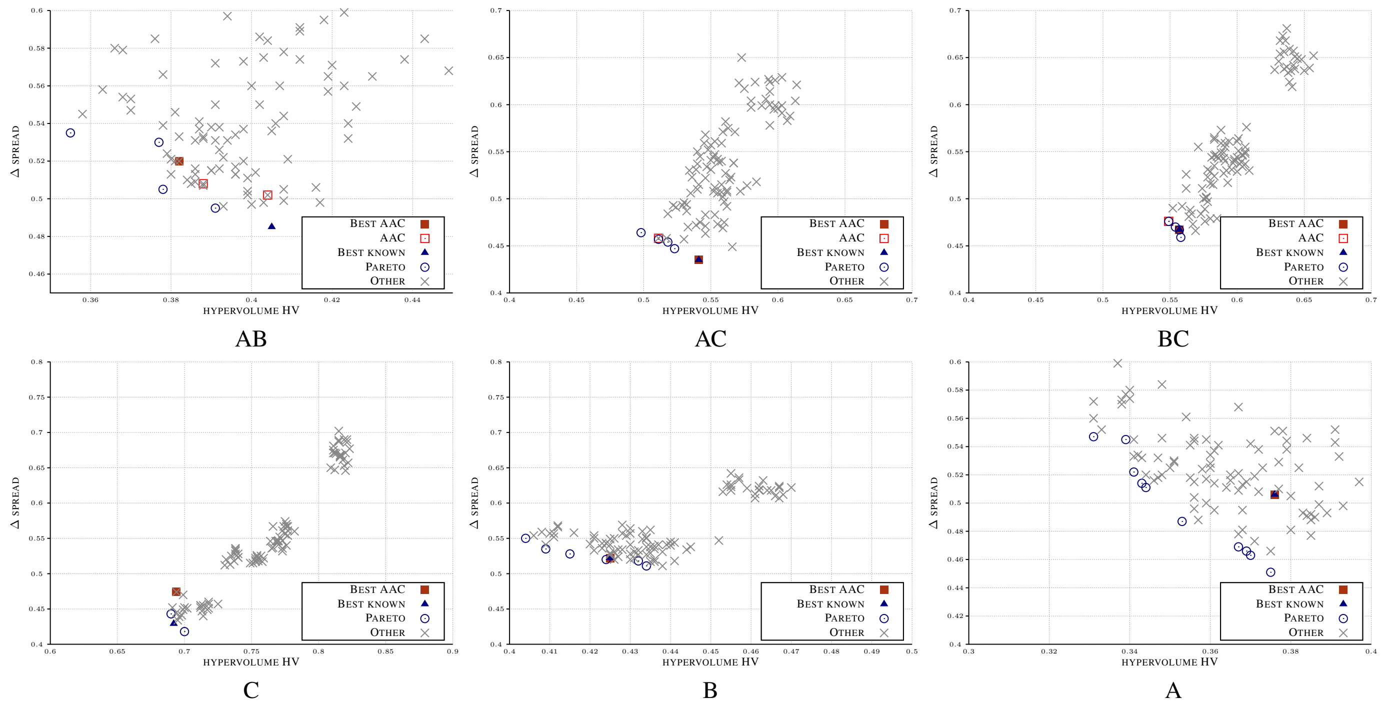

Fig. 5: Performance on Opt$_{tra}$ instances (top) and on ML$_{tra}$ instances (bottom).

to compare the performance between the configuration. For 5 datasets over 9, our AAC approach was able to find a configuration statistically equivalent to the best one. These datasets are `lucap0`, `a1a`, `yeast3`, `abalone19`, and `yeast2`. The supplementary material provided in https://bit.ly/2SfOfa3 shows that on these datasets, the average test *F-Measure* of the exhaustive configurations is strongly correlated with the average training *F-Measure*. On the other datasets, these values are often uncorrelated, showing a correlation between the efficiency of the AAC and the characteristics of datasets.

The comparisons of the AAC configurations with the exhaustive ones show the proposed approach is interesting. However, this approach must be refined by using Opt$_{sets}$ and ML$_{sets}$ of similar characteristics.

VI. Conclusion

In this work, we proposed a methodology to automatically configure MOCA-I, a multi-objective local search algorithm designed to solve supervised classification problems. Automatically configure such an algorithm is challenging since overfitting can occur in both the AAC process and classification task. In order to assess the quality of our approach, the resulting configurations were compared to the exhaustive configuration of MOCA-I on imbalanced datasets from the literature. We were able to find good configurations for most datasets, despite their limited number used in the training phase of the configurator and with a significantly reduced computational effort compared to the exhaustive runs.

In future work, we plan to investigate the effect of using different performance metrics in the automatic algorithm configuration process for the supervised partial classification of imbalanced data. In particular, we will focus on the effect of machine learning metrics during this process. One of our aims will be to determine whether such metrics should be used alone or combined with optimization metrics.

To improve the quality of the rulesets produced by MOCA-I, we will add more MOLS components to the configuration space, with a particular focus on selection, exploration, and perturbation strategies. Doing so will result in a larger configuration space, which will probably require to adjust the MO-paramILS protocol.

We will apply the methodology resulting from these studies on health datasets to detect lung cancer using biomarkers. This kind of data greatly varies from those used in this study, and finding similar datasets may be even harder in this context. Thus, we need to further think about ways of selecting the datasets for the training phase of the AAC configurator.

References

[1] J. Jacques, J. Taillard, D. Delerue, L. Jourdan, and C. Dhaenens, "The benefits of using multi-objectivization for mining pittsburgh partial classification rules in imbalanced and discrete data," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 543–550.

[2] J. Jacques, J. Taillard, D. Delerue, C. Dhaenens, and L. Jourdan, "Conception of a dominance-based multi-objective local search in the context of classification rule mining in large and imbalanced data sets," *Applied Soft Computing*, vol. 34, pp. 705–720, 2015.

[3] M. López-Ibáñez and T. Stützle, "The automatic design of multiobjective ant colony optimization algorithms," *IEEE Trans. Evolutionary Computation*, vol. 16, no. 6, pp. 861–875, 2012.

[4] A. Blot, M. Marmion, L. Jourdan, and H. H. Hoos, "Automatic configuration of multi-objective local search algorithms for permutation problems," *Evolutionary Computation*, vol. 27, no. 1, pp. 147–171, 2019.

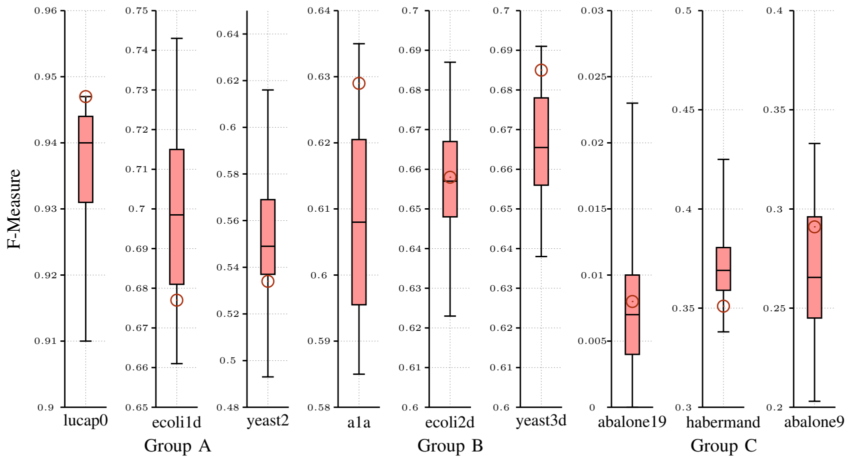

Fig. 6: Boxplots of the average *F-Measure* using all configurations on each dataset. The circle corresponds to the average value of the AAC configuration.

[5] A. Blot, H. H. Hoos, L. Jourdan, M.-É. Kessaci-Marmion, and H. Trautmann, "Mo-paramils: A multi-objective automatic algorithm configuration framework," in *International Conference on Learning and Intelligent Optimization*. Springer, 2016, pp. 32–47.

[6] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[7] A. Blot, M.-É. Kessaci, and L. Jourdan, "Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation," *Journal of Heuristics*, vol. 24, no. 6, pp. 853–877, 2018.

[8] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[9] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523.

[10] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "Paramils: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009.

[11] J. Knowles and D. Corne, "On metrics for comparing nondominated sets," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 1. IEEE, 2002, pp. 711–716.

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

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] J. Jacques, J. Taillard, D. Delerue, L. Jourdan, and C. Dhaenens, "Moca-i: discovering rules and guiding decision maker in the context of partial classification in large and imbalanced datasets," in *International Conference on Learning and Intelligent Optimization*. Springer, 2013, pp. 37–51.

[15] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 913–941, 2010.

[16] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[17] A. Frank, "Uci machine learning repository," *http://archive. ics. uci. edu/ml*, 2010.

[18] I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov, "Design and analysis of the causation and prediction challenge," in *Causation and Prediction Challenge*, 2008, pp. 1–33.

[19] B. Schölkopf, C. J. Burges, A. J. Smola *et al.*, *Advances in kernel methods: support vector learning*. MIT press, 1999.