# A hybrid CP/MOLS approach for multi-objective imbalanced classification

Nicolas Szczepanski, Gilles Audemard, Laetitia Jourdan, Christophe Lecoutre, Lucien Mousin, Nadarajen Veerapen

# A Hybrid CP/MOLS Approach for Multi-Objective Imbalanced Classification

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

Gilles Audemard
audemard@cril.fr
Univ. Artois, *CRIL*, CNRS, UMR 8188
Lens, France

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

Christophe Lecoutre
lecoutre@cril.fr
Univ. Artois, *CRIL*, CNRS, UMR 8188
Lens, France

Lucien Mousin
lucien.mousin@univ-catholille.fr
Lille Catholic University, Faculté de Gestion, Economie et Sciences
Lille, France

Nadarajen Veerapen
nadarajen.veerapen@univ-lille.fr
Univ. Lille, CNRS, Centrale Lille, *UMR 9189 - CRIStAL*
F-59000 Lille, France

## ABSTRACT

In the domain of partial classification, recent studies about multi-objective local search (MOLS) have led to new algorithms offering high performance, particularly when the data are imbalanced. In the presence of such data, the class distribution is highly skewed and the user is often interested in the least frequent class. Making further improvements certainly requires exploiting complementary solving techniques (notably, for the rule mining problem). As Constraint Programming (CP) has been shown to be effective on various combinatorial problems, it is one such promising complementary approach. In this paper, we propose a new hybrid combination, based on MOLS and CP that are quite orthogonal. Indeed, CP is a complete approach based on powerful filtering techniques whereas MOLS is an incomplete approach based on Pareto dominance. Experimental results on real imbalanced datasets show that our hybrid approach is statistically more efficient than a simple MOLS algorithm on both training and tests instances, in particular, on partial classification problems containing many attributes.

## CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; *Machine learning approaches*;

## KEYWORDS

Partial Classification, Imbalanced Data, Multi-objective Optimization, Local Search, Constraint Programming

**ACM Reference Format:**
Nicolas Szczepanski, Gilles Audemard, Laetitia Jourdan, Christophe Lecoutre, Lucien Mousin, and Nadarajen Veerapen. 2021. A Hybrid CP/MOLS Approach for Multi-Objective Imbalanced Classification. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459310

## 1 INTRODUCTION

In the domain of partial binary classification, predicting the class of new data has been studied extensively during the last decade [17]. As an illustration of such classification tasks, one can wish to predict whether a new patient is ill or not (i.e., to determine the class of this patient), according to known observations from previous patients, which are made up of several attributes (fever, age, ...). The classification is said to be partial because one just wants to know if an observation belongs to a specific class, even if many of them exist. In this paper, we deal with the classification rule mining problem, which is a problem over (discrete and) imbalanced data whose distribution greatly varies over the classes. For instance, a partial classification task can be executed in a medical diagnosis context, where less than 10% of patients are actually ill. This form of classification can be considered as a multi-objective optimization problem where each objective represents a different quality measure.

Learning classifier systems (LCS) are popular approaches to solve such problems. They often combine a genetic algorithm with a learning component (supervised learning, reinforcement learning, or unsupervised learning). Moreover, LCS deals with multi-objective optimization using an aggregating function that is often a simple sum [31]. This kind of reduction to mono-objective optimization has unfortunately some drawbacks. To overcome them, an efficient approach was proposed in [23] where a supervised partial classification task is solved by MOCA-I (Multi-Objective Classification Algorithm for Imbalanced data), based on a multi-objective local search algorithm (MOLS) and a dominance-based multi-objective local search (DMLS). Statistically better results are obtained this way than with single objective approaches like, for example, the famous C4.5 algorithm [24]. More recently, some works have exploited the robustness of automatic algorithm configuration (AAC) approaches in order to tune the parameters of MOCA-I, and thus to improve performances [36].

On the other hand, Constraint Programming (CP) is a framework that has been shown quite effective for modeling and solving various forms of optimization problems, by means of highly efficient inference and search algorithms [7, 25, 34]. It is not coincidence that some data mining problems have been studied in CP. For example, in [32], it is shown that constraint programming techniques can be applied to various pattern mining and rule learning problems as the frequent or discriminative itemset mining problems. More recently, a constraint-based declarative model [3] has been proposed to solve the association rules problem, showing good results compared to the data mining state-of-the-art.

The main goal of this paper is to exploit CP for the partial classification rule mining problem. To the best of our knowledge, this is the first time that such an approach is used to solve this specific problem. Several reasons motivated us to adopt a CP approach. Firstly, modeling the rule mining problem with CP is rather easy and natural, and besides MOLS and CP share the concept of discrete variables (actually, finite domains containing integer values). Secondly, contrary to MOLS, the search conducted by CP (solvers) is strongly guided by the objective, which can be seen as a special dynamic constraint. Lastly, the MOLS and CP paradigms are rather complementary: while CP solving is based on depth-first exploration combined with strong deductions (inferences made by constraint propagation), MOLS handles a Pareto set (of non-dominated solutions) that continually evolves by exploring neighbourhoods. More generally, CP and MOLS search algorithms carry out different kinds of exploration of the search space: one is (typically) complete while the other is incomplete. Thus, using the orthogonality of these two approaches is an appealing option.

In a first step of this work, we noticed that the number of variables and constraints in a pure CP model, for the partial classification rule mining problem, could be huge and memory expensive, resulting in memory overflow on instances containing numerous observations. This is why we have designed and implemented a hybrid CP/MOLS algorithm to counteract this problem. This hybridization is composed of two phases, respectively handled by CP and MOLS. Basically, CP is applied on rather small sub-problems, in order to avoid memory issues, before exploiting solutions delivered by CP during the MOLS phase. Experimental results on real imbalanced datasets show that our approach is statistically more efficient than MOLS or CP, when applied stand-alone, on both training and tests instances.

The paper is organized as follows. In the next section, the supervised imbalanced classification problem is introduced. Then, Section 3 introduces the MOLS and CP approaches for this problem. Next, Section 4 introduces the hybridisation of both approaches. Experimental results are described in Section 5, before concluding.

## 2 THE SUPERVISED IMBALANCED CLASSIFICATION PROBLEM

In this section, we introduce the supervised partial classification problem and the manner the rule mining problem can be encoded. We also define some solution quality metrics for multi-objective optimization of both MOLS and CP.

### 2.1 Presentation

The goal of classification is to predict the class of an observation that is unlabeled (unknown). For this purpose, the classification process is based on observations that have already been encountered before, called labeled observations. Each observation is composed of several attributes that represent a variety of information concerning the problem under consideration. An attribute is defined by a continuous or discrete variable value. As an illustration, a classification task could aim at predicting the illness of new patients among (the classes) {*flu, diabetic, healthy*} by using their symptoms and personal information (fever, cough, age, weight) as attributes. Note that when an attribute is continuous, a discretization technique can be applied.

In this paper, we deal with the partial classification problem. On the one hand, complete classification aims at discriminating between the target classes and consists in forming a partition of the set of observations (necessarily classifying a patient in one of the classes *flu*, *diabetic* or *healthy*. On the other hand, partial classification is restricted to a subset of the target classes and consists in dividing the observations into two complementary parts (for example, determining whether a patient has the *flu* or not).

From labeled observations, the outcome of a classification task is a *classifier* of the form $AT \Rightarrow P$. Each such rule indicates that $P$, a target class, is the logical consequence of the conjunction of some conditions denoted by $AT$, each of them representing a test on an attribute. For example, if the classifier is « $\{fever > 39$ *and* $cough = true\} \Rightarrow ill$ », then the classifier answers « positive » to the prediction « *patient is ill*? » when the attribute values for a given patient logically validate the attribute tests. Otherwise, it returns « negative ».

Numerous metrics have been proposed in the literature to evaluate the efficiency of classifiers, and most of them are based on the confusion matrix presented in Table 1. For each observation, the confusion matrix reports a value among {TP, TN, FN, FP} according to the prediction made by the classifier and the actual class of the observation. The actual class is the known correct class of the observation, and this information is required to check the efficiency of a classifier. Consequently, *True Positives* (TPs) and *True Negatives* (TNs) count well-classified observations, whereas *False Positives* (FPs) and *False Negatives* (FNs) count misclassified observations. As an illustration, the classifier rule $\{fever > 39$ *and* $cough = true\} \Rightarrow ill$ is not activated if the patient attribute values are $\{fever = 40$ *and* $cough = false\}$. However, if the patient is really ill (i.e. the actual class is « ill »), this is a False Negative.

**Table 1: Confusion Matrix**

| | | Predicted Class | |
|---|---|---|---|
| | | positive | negative |
| Actual Class | positive | TP | FN |
| | negative | FP | TN |

Usually, in order to fairly evaluate the ability of a classifier, data are divided into two parts: the training instances representing labeled observations and the test instances corresponding to unlabeled observations. In this way, a solving method exploits the training data set to produce a classifier, and then, confusion matrix-based measures are employed to evaluate this classifier on both training and test instances. Thus, this makes it possible to take into account the efficiency of the solving method to produce a good classifier (using training instances) and the classifier effectiveness on unknown data (using test instances).

This evaluation protocol is significant because a classifier can be effective on training instances but not on testing instances. In practice, this phenomenon is called *overfitting* and means that the classifier is too closely tied to the training data set. This can also mean that the training and test data sets are not similar enough.

Another issue to be addressed by solving methods is the handling of imbalanced data. They are characterised by a number of observations that varies greatly between classes. On top of that, on most datasets, the class to predict is often less frequent than the opposite class. For instance, in a medical context, the diagnosis can be positive on less than 10% of patients. To address this issue, several multi-objective techniques using confusion matrix-based measures and based on a solution encoding have been introduced in the literature [23].

A solution encoding represents the form of the propositional formula encoding the attribute tests $AT$ of a classifier (the prediction $P$ is omitted). In the literature concerning the rule mining problem, several encodings of solutions exist: the tree encoding [33], the Michigan encoding [39], and the Pittsburgh encoding [2]. Later, in this paper, we use the variable-length Pittsburgh encoding [23] which allows a large number of attributes to be handled.

In this encoding, a solution is called a ruleset: this is a disjunction of rules, where each rule is a conjunction of terms. Each term corresponds to an attribute test ($AT$) which is defined by an attribute, an operator and a value, i.e., a triplet of the form (attribute, operator, value). Classically, the operator is relational ($<$, $>$, or $=$), because data (values) are assumed to be discretized and subject to a total order. For example, (`fever`, $>$, 39) is a term representing the test `fever` $>$ 39. This solution encoding is used to predict the class by using the terms and the logical operators *or* and *and* constituting the ruleset. The predicted class of the classifier is determined by the interpretation of a ruleset with the attribute values of an observation.

## 2.2 A Multi-Objective Model

The goal of Multi-Objective Optimisation (MOO) is to jointly optimize several criteria (objective functions) that directly affect the solution quality of a given problem. More precisely, a solution to a MOO problem is ideally one such that all objective functions $f_i(x)$ reach their optimal values. From now on, without any loss of generality, we consider minimization:

$$\underset{x \in D}{\operatorname{argmin}}(f_1(X_1), f_2(X_2), \ldots, f_n(X_n)) \quad (1)$$

In Equation 1, $n$ denotes the number of objectives ($n \geq 2$), $D$ is the set of feasible solutions $x$ represented by the vector of $k$ decision variables $x = (x_1, x_2, \ldots, x_k)$. Moreover, the sets $X_i \subseteq x$ represent several vectors of decision variables which can be different depending on the associated objective function. Note that mixed MOO problems, consisting of both some objective functions to maximize and minimize, can easily be transformed into minimization MOO problems by changing their sign, $f'_i(x) = -f_i(x)$.

The concept of Pareto dominance is used to distinguish between solutions according to the involved criteria. A solution $s_1$ dominates another solution $s_2$ if, and only if, (i) $s_1$ is better than or equal to $s_2$ for all criteria, and (ii) $s_1$ is strictly better than $s_2$ for at least one criterion. A set of non-dominated solutions $\{s_1, s_2, \ldots, s_m\}$, where there is no pair of distinct solutions $(s_i, s_j)$ such that $s_i$ dominates $s_j$, is called a Pareto set, a Pareto front, or an archive in the context of multi-objective local search algorithms. Solving a MOO problem consists in finding a Pareto optimal set $S^* \subset D$, i.e., a Pareto set such that no other feasible solution $x' \in D$ dominates any $x \in S^*$.

For multi-objective local search (MOLS), introduced later, objectives are defined as (following the model described in [23]):

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

The sensitivity corresponds to the proportion of samples detected by the ruleset that are positive with respect to the class under investigation. In contrast, the confidence corresponds to the probability that an observation detected as positive by the ruleset is a true positive. Minimizing the number of terms helps to reduce complexity and avoid the *bloat* effect caused by rulesets containing over-specific rules without any improvement of their quality. Moreover, note that these two metrics are highly susceptible to variation in the distribution of observations.

As almost none of the CP solvers deal with multi-objective optimization, in the CP approach, we use an aggregation based on the *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. Thus, the closer the F-measure is to 1, the better the quality of the classifier (i.e. the ruleset). Moreover, as the *F-Measure* is recommended in partial classification [22], the experimental evaluation of this work uses this measure.

# 3 MODELING AND SOLVING METHODS

In this section, we introduce MOLS and CP.

## 3.1 Multi-objective Local Search

Multi-objective local search (MOLS) algorithms are most often based on Pareto local search (PLS) [30] that gradually improves a Pareto set. In the literature, numerous extensions to PLS have emerged, such as the iterated PLS [10], the stochastic PLS [11], the anytime PLS [12] and the dominance-based multi-objective local search (DMLS) [5]. As opposed to single-objective local search, focusing on only one solution, DMLS maintains multiple candidates and non-dominated solutions. First, this algorithm starts by creating an archive of several initial solutions. Note that this *initialization* phase is exploited later in our hybridisation. Thereafter, these solutions are improved by iteratively executing four distinct phases: *selection*, *exploration*, *archiving* and *perturbation*.

Specifically, at each iteration, the first step of DMLS selects the solutions from the archive that will be explored (*selection* phase). Thereafter, the *exploration* phase seeks new candidate solutions to add to the archive by successively exploring the neighborhoods of the selected solutions. Such additions are realized according to the improving and/or non-dominated criteria. Once new solutions have been added to the archive, a filtering (i.e. the *archiving* phase) is performed to keep only non-dominated solutions. However, in some cases, no new improved solution can be added to the archive, meaning that solutions of the current archive are local optima. To remedy this problem, one aims at escaping from these local optima by moving to another area of the search space. The last phase, called *perturbation*, is dedicated to this task by generating new random solutions, or by carrying out some kick moves.

This paper takes up the MOLS algorithm of [23], specially designed for imbalanced data. It should be noted that, in this previous work, the MOLS method used (MOCA-I) has already been compared against 13 other classification methods using the KEEL software. MOCA-I has shown its superiority on all the compared algorithms. As with any local search algorithm, it is based on a neighborhood relation, which implicitly associates a set of neighbors with each solution [4]. For the classification rule mining problem, the neighborhood of a ruleset $r$ is the set of rulesets having one test attribute that differs from $r$. More precisely, this difference can be one fewer or more term, or a term with a different value or operator.

Moreover, each phase of MOLS accepts various parameters (perturbation methods, selection and exploration strategies, the maximal archive size, ...). Thus, some works have exploited this parameter variety, as e.g., in [36, 37], where optimized versions of MOLS have been proposed for the imbalanced classification problem by means of the automatic algorithm configuration (AAC) paradigm. Note that we have chosen the best identified parameters for MOLS from this literature, i.e., a initial population of solution of size 100, an archive of max size 500, a restart strategy, a random selection of solution and an exploration of all neighbors. The sequential MOLS algorithms have been rewritten from [23] into a new library called MH-builder.

## 3.2 Constraint Programming

In this section, we present a CP model for the partial classification rule mining problem, under the form of a constraint optimization problem (COP) [25], before giving a few details about the solving process. Concerning related work, note that other data mining problems have already been studied in CP. The authors of [32] show that constraint programming techniques can be applied to various pattern mining and rule learning problems (itemset mining, maximal frequent itemset mining, discriminative itemset mining, and so on). More recently, in [3], a constraint-based declarative model has been introduced to solve the association rules (AR) problem and show good results compared to the data mining state-of-the-art. For interested readers, others pattern mining problems are processed with CP in [20].

*3.2.1 Modeling Phase.* Algorithm 1 depicts a simplified version of the CP model we propose for the partial classification rule mining problem. To develop this model, we have used PyCSP$^3$ [27], a recent Python modeling library. Roughly speaking, this model allows us to extract a ruleset while maximizing the F-measure. For simplicity, the number of rules and the number of terms, `nRules` and `nTerms`, constituting the target ruleset are considered as pre-defined. Each problem instance is then characterized by the following data:

- some information concerning the labeled observations: `nObservations`, their actual classes (`actual`) and their attribute values;
- the number of attributes `nAttributes`;
- the number of values `nValues` w.r.t. the attributes.

First, the variables of the model are introduced. Six multi-dimensional arrays of variables are declared by using the function `VarArray` while specifying, as parameters, the size of each dimension (the `size` parameter) and the domain of each variable (the `dom` parameter). The three first arrays will represent the terms `attributes`, `operators`, `values` forming each rule of the ruleset. The three next arrays will permit to determine, for each observation, whether each attribute test, each rule and the ruleset is positive (satisfied) or not. Note that each such variable can be assigned to either 0 (negative) or 1 (positive). Two stand-alone variables are also introduced, `tp` and `fp`, by calling the `Var` function: they represent the number of true positives and false negatives, respectively.

Second, the constraints of the model are introduced. This is made possible by calling the `satisfy` function, passing constraints as parameters. A first global constraint `allDifferent` is posted (line 21) to ensure that all attributes in any rule are distinct. Next, at lines 23 and 24, a list (group) of table constraints is introduced. Due to lack of space, the auxiliary function `aux_table` is not described here, but its role is to return the list of 4-tuples that can be accepted for a given sequence of four variables. In other words, this allows us to establish a link between a specific observation and a specific term. Then, the two next lists of constraints (lines 25 to 28) encode, for each observation, the value of the ith `rule` (resp. the value of the `ruleset`) as a conjunction of `terms` (resp. as a disjunction of `rules`). For example, in any solution of the model, `ruleset[0]` is equal to 1 (resp. 0) when the first observation (with index 0) is identified as being `positive` (resp. `negative`). The two last constraints count, respectively, the number of `tp` and `fp`, according to the actual classes of observations (`actual`) and the values of the `ruleset`.

Third, the objective of the model is introduced. This objective consists in maximizing an expression denoting the F-measure (at lines 33 to 36).

*3.2.2 Solving Phase.* Solving a CP model involves the use of a constraint solver (or a SAT solver, after translating the model in propositional form). Typically, such solvers are complete, performing a depth-first exploration with backtracking, taking a decision (usually, a variable assignment) at each step and running a filtering process called constraint propagation. To address the issue of heavy-tailed runtime distributions [18], the search is restarted regularly, following a geometric progression (or the Luby sequence). The order in which variables are chosen during the depth-first traversal of the search space is decided by a variable ordering heuristic; a classical generic heuristic is `dom/wdeg` [6], combined with a mechanism simulating a certain form of intelligent backjumps [26]. The order in which values are chosen when assigning variables is decided by a value ordering heuristic; for COPs, it is highly recommended to use first the value present in the last found solution, which is a technique known as solution(-based phase) saving [8, 38].

Backtrack search for COP relies on Branch and Bound (B&B) whose principle is equivalent, assuming a minimization problem, to adding a special objective constraint $\texttt{obj} < \infty$ to the constraint network (although it is initially trivially satisfied), and to update the limit of this constraint whenever a new solution is found. It means that any time a solution S is found with cost $B = \texttt{obj}(S)$, the objective constraint becomes $\texttt{obj} < B$. Hence, B&B provides a sequence of better and better solutions until no more exist, guaranteeing that the last found solution is optimal.

For our experiments, we have used the ACE constraint solver, which is the new avatar of AbsCon[1].

[1] http://www.cril.univ-artois.fr/software/abscon.html

```
from pycsp3 import *
... # first, data are loaded (code not inserted here, for simplicity)
# attributes[i][j] is the attribute involved in the j^th term of the i^th rule
attributes = VarArray(size=[nRules, nTerms], dom=range(nAttributes))
# operators[i][j] is the operator involved in the j^th term of the i^th rule
operators = VarArray(size=[nRules, nTerms], dom={EQ, LT, GT})
# values[i][j] is the value involved in the j^th term of the i^th rule
values = VarArray(size=[nRules, nTerms], dom=range(nValues))
# terms[k,i,j] is 1 iff the k^th observation satisfies the j^th term of the i^th rule
terms = VarArray(size=[nObservations, nRules, nTerms], dom={0, 1})
# rules[k,i] is 1 if the k^th observation satisfies the i^th rule
rules = VarArray(size=[nObservations, nRules], dom={0,1})
# ruleset[k] is 1 if the k^th observation satisfies the ruleset (i.e., at least one of the rules)
ruleset = VarArray(size=nObservations, dom={0,1})
# tp is the number of true positives
tp = Var(dom=range(nObservations))
# fp is the number of false positives
fp = Var(dom=range(nObservations))
satisfy(
    # in each rule, involved attributes must be different
    [AllDifferent(attributes[i]) for i in range(nRules)],
    # determining if observations satisfy terms
    [(terms[k][i][j], attributes[i][j], operators[i][j], values[i][j]) in aux_table(k,i,j)
        for k in range(nObservations) for i in range(nRules) for j in range(nTerms)],
    # determining if observations satisfy rules
    [rules[k][i] == conjunction(terms[k][i]) for k in range(nObservations) for i in range(nRules)],
    # determining if observations satisfy the ruleset
    [ruleset[k] == disjunction(rules[k]) for k in range(nObservations)],
    # computing the values of variables tp and fp
    Count([ruleset[k] for k in range(nbObservations) if actual[k] == POSITIVE], value=1) == tp,
    Count([ruleset[k] for k in range(nbObservations) if actual[k] == NEGATIVE], value=1) == fp,
)
maximize(
    # maximizing the F-measure
    (2×(tp/(tp+fp))×(tp/(tp+fn)))/((tp/(tp+fp))+(tp/(tp+fn)))
)
```

**Algorithm 1: A PyCSP$^3$ Model for the Partial Classification Rule Mining Problem**

## 4 HYBRIDIZATION

In this section, the hybrid CP/MOLS approach we propose, HYB in short, is presented. Of course, some forms of hybridization between CP and LS (Local Search) have already been proposed in the literature. For example, LS has been used to guide local exploration by minimizing, at each step, the total number of conflicts when solving constraint satisfaction problems [28]. The original constraint-based local search proposed in [16, 29] also combines CP and LS: constraints are used to describe and control LS [16, 29]. The hybridization we propose consists in considering the (partial) solutions computed by CP, during a limited time period, as an initial population for MOLS. Note that this kind of hybridization technique has already been applied between mixed integer linear programming (MILP) and CP for the airline planning problem [19].

Our hybrid approach is composed of two main phases. First, CP is applied on some relevant sub-problems in order to collect a set of partial solutions. These solutions are then used to establish an initial population, on which MOLS can be started.

### 4.1 CP Phase

This phase exploits the ability to deal with the classification rule mining problem by considering different ruleset sizes. Hereafter, the specific instance derived from our CP model where a ruleset of $i$ rules, each one composed of $j$ terms, is encoded, will be denoted by $m(i, j)$. Our approach consists in solving incrementally several related instances over time by starting from $m(1, 1)$ until an instance of maximal size $m(i^{max}, j^{max})$ is reached, where $i^{max}$ and $j^{max}$ denote the maximal values used to stop the process (e.g., 6 rules of 5 terms). The incremental nature of this approach permits to solve a current instance while using the solution(s) found just earlier (e.g., using the solution obtained for $m(1, 1)$ as a partial solution for $m(1, 2)$).

The entire process is composed of three steps. First, as mentioned above, we solve a sequence of related instances. Note that any instance of the sequence (except the first one) is built from the previous one by increasing, either the number of rules $i$ or the number of terms $j$. More precisely, this sequence is defined by the equation:

$$m(i,j) = \begin{cases} m(i,j+1), & \text{if } i+j \text{ is even} \\ m(i+1,j), & \text{if } i+j \text{ is odd} \end{cases} \quad (2)$$

Hence, the first elements of the sequence are: $m(1,1)$, $m(1,2)$, $m(2,2)$, $m(2,3)$, ...

It is important to note that, by construction, the quality (i.e., the F-measure) of solutions can only increase. Since we solve instances under assumptions (i.e., from assignments corresponding to previous solutions), the solution obtained for the last instance of the sequence is not necessarily optimal. This is why, in a second step, we solve this instance without any assumption (but we use the last solution that has been found for this instance, as a soft guide for the value ordering heuristic [38]).

Finally, from the best found solution, whose F-measure is $f$, we build a final instance while constraining the objective to be around $f$. By solving it, we hopefully obtain a certain number of diverse solutions.

### 4.2 MOLS Phase

The initialization step of MOLS involves generating an initial population of solutions. This step is simply replaced by considering the output (set of solutions) of the CP phase. Note that for our experiments, we use the same parameter values as in [23], that is to say, a production of 100 rulesets with a maximal size of 2 rules of 2 terms each.

## 5 EXPERIMENTAL RESULTS

In this section, the performance of the CP, MOLS and hybrid approaches are compared using a proper statistical test, executed on several data sets.

### 5.1 Protocol

This work follows the recommendations made by Demsar to compare several learning algorithms over multiple data sets [9]. The principle is to get a solid measure of the overall performance of any algorithm over several independent data sets. For this purpose, we can use the Friedman statistical test [15] in order to detect the differences existing between various algorithms over several problems. More precisely, this test is based on the ranking positions of algorithms over all problems. A post-hoc all pairwise comparison of algorithms is performed using the Wilcoxon statistical test on all computed F-Measures. Specifically, the Wilcoxon test allows us to determine whether two series of numbers are equivalent or not, from a pairwise comparison. Therefore, a solving method is considered as better than another one if its rank is better and if the two series of numbers are not equivalent in the Wilcoxon's sense. In contrast, two solving methods are equivalent if the series of numbers are equivalent, regardless of their rank.

As we deal with the supervised classification problem, the product classifier must be evaluated on unknown data, which are different from the training data. Recall that it permits to recognize an overfitting issue. Since it is almost impossible to find multiple and distinct datasets from the same classification problem, datasets are split into training and test sets as for classical machine learning algorithms [22], by following the $k$-fold cross-validation protocol.

The $k$-fold cross validation produces $k$ training sets and $k$ test sets by splitting the dataset (i.e. the observations) into $k$ same-size folds. Each training set is simply a combination of $k-1$ folds while the remaining fold becomes the test set. Classifiers are built from training data sets, but importantly, their evaluations are realized on both training and test instances. This way, the effectiveness of apparently good classifiers can be demonstrated on unknown data.

In our case, experiments have been carried on the basis of 5-fold cross validation. Moreover, in order to analyze the behavior of algorithms as fairly as possible, MOLS and HYB have been run from 30 distinct seeds, on each of the 5 folds, leading to 150 measures for each algorithm and each dataset. Note that the pure CP algorithm only needs one run, on each fold, to be launched because it is deterministic (i.e., the same result is always computed, with the same performance, as no random process is involved). Finally, to make sure that each algorithm has a chance to reach local optima, the timeout has been set to 20 minutes. Experiments have been conducted on two Intel XEON E5-2687W computers of 24 cores of 3.0GHz with 64GB of RAM each.

### 5.2 Datasets

The datasets come from various sources [14, 21, 35] and represent several supervised classification problems with various degrees of imbalance (27.85% to 2.90%). Table 2 exposes the main features of each dataset and confirms their imbalanced aspect in the column $d_{asy}$, which denotes the proportion of observations to be expected in the target class. These datasets come from the UCI repository[2], except for the `lucap0` dataset coming from a Machine Learning Challenge [21]. The last three datasets `a1a`, `lucap0` and `w1a` are binary (i.e., the number of possibles values for any attribute is always 2), and, are more challenging because they contain a higher number of observations and attributes. Because we are interested in the partial classification problem, these datasets have been converted into partial classification datasets using the methodology presented in [13], and discretized using the 10-bin discretization method of the KEEL software [1].

**Table 2: Description of the datasets (number of observations, attributes, numerical attributes and degree of asymmetry).**

| Name | #obs | #att | #num | $d_{asy}$ | ref |
|---|---|---|---|---|---|
| haberman | 306 | 3 | 3 | 27.42% | [14] |
| ecoli2d | 336 | 7 | 7 | 15.48% | [14] |
| ecoli1d | 336 | 7 | 7 | 22.92% | [14] |
| yeast3d | 1484 | 8 | 8 | 10.35% | [14] |
| abalone9-18d | 731 | 8 | 7 | 5.65% | [14] |
| yeast2vs8d | 482 | 8 | 8 | 4.85% | [14] |
| a1a | 1605 | 123 | 0 | 24.61% | [35] |
| lucap0 | 2000 | 144 | 0 | 27.85% | [21] |
| w1a | 2477 | 300 | 0 | 2.90% | [35] |

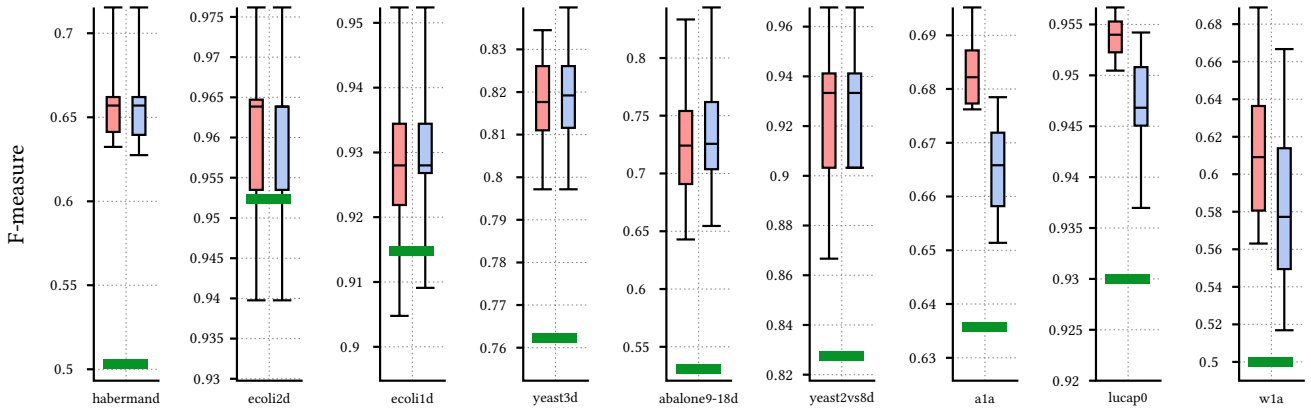


**Figure 1: *F-measure* of the training instances by dataset. The hybrid and MOLS methods are respectively represented by the left (red) and right (blue) box-and-whisker plots (150 runs each) while the CP method is represented by the green lines (median of 5 runs).**

## 5.3 Results

Figure 1 reports the quality of the classifiers produced by the three approaches (CP, MOLS and HYB) on the nine training datasets. The quality directly depends on the performance of the underlying optimization methods, and is given in term of *F-measure*. For each dataset, we have two box-and-whisker plots: the methods HYB and MOLS are respectively represented by the left (red) and right (blue) box-and-whisker plots, each one based on 150 *F-measure* values. The bottom and top of each box represent the first and third quartiles, while the band inside the box corresponds to the second quartile (the median). The ends of the whiskers represent minimum and maximum values. Note that, per dataset, there is also a green line segment indicating the median of the 5 *F-measure* values (for the 5 folds per dataset) obtained with CP.

On the one hand, it is clear that CP alone is far less effective than both MOLS and HYB. Indeed, green lines are often quite low, and always underneath the medians of the box-and-whisker plots. On the other hand, one can observe that hybridization can pay off: CP combined with MOLS, i.e., HYB, globally outperforms MOLS. More precisely, whereas MOLS and HYB seem to be somewhat equivalent on the first 6 datasets, there are some significant differences on the last 3 datasets in favor of the hybridization method. It is worthwhile to recall that these last datasets (`a1a`, `lucap0` and `w1a`) are challenging (due to their large number of attributes and observations; see Table 2).

As explained in Section 5.1, we use the Friedman statistical test [15] to emphasize significant differences between several algorithms over various datasets. Here, this statistical test has two main objectives: firstly, to confirm the previous results, given by Figure 1, about MOLS and HYB (on the training dataset) and, secondly, to check whether the best method on training instances is always the best one on unknown data (i.e., on test instances).

Table 3 depicts the result of this statistical test per dataset (first column) on both training and test instances, respectively in the second and third columns. The Friedman statistical test can output `EQUIVALENT`, `MOLS` or `HYBRID` according to the two specific series of numbers and ranks that are associated with the two methods. Of course, these series represent the 150 *F-measure* values for MOLS and HYB. The Wilcoxon signed rank test is used with a significance level of $\alpha = 0.01$. We obtain `EQUIVALENT` when the Wilcoxon test ensures that the probability of having MOLS *F-measures* greater than HYB *F-measures* is equal to the probability of having HYB *F-measures* greater than MOLS *F-measures*. But we obtain either MOLS or HYB when the two series are not equivalent in the Wilcoxon's sense; in that case, the best method is the one with the best ranking according to the Friedman statistical test.

**Table 3: Friedman statistical test with $\alpha = 0.01$**

| Name | Training | Test |
|---|---|---|
| haberman | EQUIVALENT | EQUIVALENT |
| ecoli2d | EQUIVALENT | EQUIVALENT |
| ecoli1d | EQUIVALENT | EQUIVALENT |
| yeast3d | EQUIVALENT | EQUIVALENT |
| abalone9-18d | EQUIVALENT | EQUIVALENT |
| yeast2vs8d | EQUIVALENT | EQUIVALENT |
| a1a | HYBRID | HYBRID |
| lucap0 | HYBRID | HYBRID |
| w1a | HYBRID | HYBRID |

On the training instances, the statistical tests clearly confirm the results presented in Figure 1. Interestingly, on the test instances, they prove that the hybrid method is the most robust approach, because either HYB is equivalent to MOLS or HYB outperforms MOLS. The latter case is true for the datasets `a1a`, `lucap0` and `w1a` that have the particularity of involving binary attributes, which, technically, reduces the size of the table constraints in the CP model (and consequently, improve the performance of the CP phase of the hybridization).

[2] http://archive.ics.uci.edu/ml/index.php

**Table 4: Average values and standard deviations (given between parentheses) of F-measures obtained with the hybrid approach.**

| Name | Training | Test |
|---|---|---|
| haberman | 0.661 (0.026) | 0.389 (0.122) |
| ecoli2d | 0.959 (0.007) | 0.794 (0.094) |
| ecoli1d | 0.928 (0.008) | 0.755 (0.075) |
| yeast3d | 0.817 (0.009) | 0.725 (0.043) |
| abalone9-18d | 0.226 (0.029) | 0.344 (0.236) |
| yeast2vs8d | 0.924 (0.019) | 0.528 (0.194) |
| a1a | 0.683 (0.006) | 0.647 (0.022) |
| lucap0 | 0.953 (0.001) | 0.945 (0.009) |
| w1a | 0.610 (0.032) | 0.407 (0.138) |

Table 4 displays the F-measures obtained by HYB on both the training and test datasets. This allows us to assess the effectiveness of HYB on unknown data. The F-measures remain rather similar (when passing from training to test instances), except for `haberman`, `yeast2vs8d` and `w1a` which are subject to overfitting. For such datasets, one may then try to refine the k-fold cross-validation protocol.

## 6 CONCLUSION

In this paper, we have presented an original form of hybridization mixing Constraint Programming (CP) and Multi-Objective Local Search (MOLS), for solving the partial classification rule mining problem. More precisely, we have shown that it was interesting to apply CP on particular sub-problems in order to build a relevant initial population for MOLS. Experimental results on real imbalanced datasets show that our hybrid approach is statistically better than MOLS or CP, when applied stand-alone, on both training and tests instances.

In the near future, we plan to build upon this work, exploring a few perspectives. Firstly, we intend to improve the CP model by, e.g., compressing the table constraints (using various techniques from the literature). Secondly, as hybridization seems to provide a substantial advantage on large instances, we project to analyse the reasons (e.g., features of the datasets) behind this observable efficiency.

## REFERENCES


[1] Jesús Alcalá-Fdez, Luciano Sanchez, Salvador Garcia, Maria Jose del Jesus, Sebastian Ventura, Josep Maria Garrell, José Otero, Cristóbal Romero, Jaume Bacardit, Victor M Rivas, et al. 2009. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing* 13, 3 (2009), 307–318.

[2] Jaume Bacardit and Josep Maria Garrell. 2003. Evolving Multiple Discretizations with Adaptive Intervals for a Pittsburgh Rule-Based Learning Classifier System. In *Genetic and Evolutionary Computation — GECCO 2003*, Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska, and Julian Miller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1818–1831.

[3] Mohamed-Bachir Belaid, C. Bessiere, and Nadjib Lazaar. 2019. Constraint Programming for Association Rules. In *SDM*.

[4] Aymeric Blot, Marie-Éléonore Kessaci, and Laetitia Jourdan. 2018. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics* 24, 6 (2018), 853–877.

[5] Aymeric Blot, Marie-Eléonore Marmion, Laetitia Jourdan, and Holger H. Hoos. 2019. Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems. *Evolutionary Computation* 27, 1 (2019), 147–171.

[6] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. 2004. Boosting systematic search by weighting constraints. In *Proceedings of ECAI'04*. 146–150.

[7] Rina Dechter. 2003. *Constraint processing*. Morgan Kaufmann.

[8] Emir Demirovic, Geoffrey Chu, and Peter Stuckey. 2018. Solution-Based Phase Saving for CP: A Value-Selection Heuristic to Simulate Local Search Behavior in Complete Solvers. In *Proceedings of CP'18*. 99–108.

[9] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* 7 (Dec. 2006), 1–30.

[10] Madalina Drugan and Dirk Thierens. 2012. Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics* 18 (10 2012). https://doi.org/10.1007/s10732-012-9205-7

[11] Madalina M. Drugan and Dirk Thierens. 2010. Path-Guided Mutation for Stochastic Pareto Local Search Algorithms. In *PPSN XI*. Springer Berlin Heidelberg, Berlin, Heidelberg, 485–495.

[12] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. 2015. Anytime Pareto local search. *European Journal of Operational Research* 243 (06 2015). https://doi.org/10.1016/j.ejor.2014.10.062

[13] Alberto Fernández, Salvador García, Julián Luengo, Ester Bernadó-Mansilla, and Francisco Herrera. 2010. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *IEEE Transactions on Evolutionary Computation* 14, 6 (2010), 913–941.

[14] Andrew Frank. 2010. UCI machine learning repository. *http://archive. ics. uci. edu/ml* (2010).

[15] Milton Friedman. 1937. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *J. Amer. Statist. Assoc.* 32, 200 (1937), 675–701. https://doi.org/10.1080/01621459.1937.10503522

[16] Philippe Galinier and Jin-Kao Hao. 2004. A General Approach for Constraint Solving by Local Search. *J. Math. Model. Algorithms* 3 (03 2004), 73–88. https://doi.org/10.1023/B:JMMA.0000026709.24659.da

[17] Liqiang Geng and Howard J Hamilton. 2006. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)* 38, 3 (2006), 9.

[18] Carla Gomes, Bart Selman, Nuno Crato, and Henry Kautz. 2000. Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems. *Journal of Automated Reasoning* 24, 1 (2000), 67–100.

[19] Mattias Grönkvist. 2004. A Constraint Programming Model for Tail Assignment. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Jean-Charles Régin and Michel Rueher (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 142–156.

[20] Mattias Guns. 2012. Declarative Pattern Mining using Constraint Programming (Een declaratieve aanpak tot pattern mining door middel van constraint programming). (2012).

[21] Isabelle Guyon, Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov. 2008. Design and analysis of the causation and prediction challenge. In *Causation and Prediction Challenge*. 1–33.

[22] Haibo He and Eduardo Garcia. 2009. Learning from Imbalanced Data. *IEEE Trans. on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284.

[23] Julie Jacques, Julien Taillard, David Delerue, Clarisse Dhaenens, and Laetitia Jourdan. 2015. 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* 34 (2015), 705–720.

[24] Julie Jacques, Julien Taillard, David Delerue, Laetitia Jourdan, and Clarisse Dhaenens. 2013. The benefits of using multi-objectivization for mining pittsburgh partial classification rules in imbalanced and discrete data. In *GECCO 15th*. ACM, 543–550.

[25] Christophe Lecoutre. 2009. *Constraint Networks: Techniques and Algorithms*.

[26] Christophe Lecoutre, Lakdhar Sais, Sébastien Tabary, and Vincent Vidal. 2009. Reasonning from Last Conflict(s) in Constraint Programming. *Artificial Intelligence* 173, 18 (2009), 1592–1614.

[27] Christophe Lecoutre and Nicolas Szczepanski. 2020. PYCSP3: Modeling Combinatorial Constrained Problems in Python. *CoRR* abs/2009.00326 (2020). https://arxiv.org/abs/2009.00326 Available from https://github.com/xcsp3team/pycsp3.

[28] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. [n. d.]. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. ([n. d.]).

[29] Alexander Nareyek. 2000. Using Global Constraints for Local Search. *Lecture Notes in Computer Science* 2062 (07 2000).

[30] Luis Paquete, Marco Chiarandini, and Thomas Stützle. 2004. *Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study*. Vol. vol. 535. 177–199. https://doi.org/10.1007/978-3-642-17144-4_7

[31] Jaume Bacardit Peñarroya. 2004. Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time.

[32] Luc De Raedt, Tias Guns, and Siegfried Nijssen. 2010. Constraint Programming for Data Mining and Machine Learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1671–1675.

[33] Alan paul Reynolds and Beatriz de la Iglesia. 2007. Rule Induction for Classification Using Multi-objective Genetic Programming. In *Evolutionary Multi-Criterion Optimization*, Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 516–530.

[34] Francesca Rossi, Pascal van Beek, and Toby Walsh (Eds.). 2006. *Handbook of Constraint Programming*. Elsevier.

[35] Bernhard Schölkopf, Christopher JC Burges, Alexander J Smola, et al. 1999. *Advances in kernel methods: support vector learning*. MIT Press.

[36] Sara Tari, Holger H. Hoos, Julie Jacques, Marie-Eléonore Kessaci, and Laetitia Jourdan. 2020. Automatic Configuration of a Multi-objective Local Search for Imbalanced Classification. In *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I (Lecture Notes in Computer Science)*, Thomas Bäck, Mike Preuss, André H. Deutz, Hao Wang, Carola Doerr, Michael T. M. Emmerich, and Heike Trautmann (Eds.), Vol. 12269. Springer, 65–77. https://doi.org/10.1007/978-3-030-58112-1_5

[37] Sara Tari, Nicolas Szczepanski, Lucien Mousin, Julie Jacques, Marie-Eleonore Kessaci, and Laetitia Jourdan. 2020. Multi-objective Automatic Algorithm Configuration for the Classification Problem of Imbalanced Data. In *IEEE Congress on Evolutionary Computation CEC*.

[38] Julien Vion and Sylvain Piechowiak. 2017. Une simple heuristique pour rapprocher DFS et LNS pour les COP. In *Proceedings of JFPC'17*. 39–45.

[39] Stewart W. Wilson. 1995. Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3, 2 (1995), 149–175. https://doi.org/10.1162/evco.1995.3.2.149