Automatic Algorithm Selection for the Quadratic Assignment Problem Using Meta-learning and Fitness Landscape Measures

Augusto Dantas *, Aurora Pozo †

Department of Computer Science
Federal University of Paraná
Curitiba, Brasil
Email: *aldantas@inf.ufpr.br, †aurora@inf.ufpr.br

***Abstract*—The performance of a meta-heuristic on a search problem relies on the features of the problem instance, meaning that it does not exist a single method that always achieves the best performance. Therefore, approaches that automatically select an algorithm can improve the robustness of an application. Among the possible alternatives, the selection can be done with meta-learning, which aims at mapping the characteristics (meta-features) of problem instances to the performance of a set of algorithms. Some works have proposed the use of Fitness Landscape Analysis (FLA) to characterize the instances, where the structure of a problem is analysed by extracting measures of the search space, normally with a sample of solutions. This dissertation is composed by a series of incremental studies on meta-learning algorithm selection for the Quadratic Assignment Problem. We experimented with a variety of meta-features, giving emphasis on the FLA based metrics, and with a set of classic and novel meta-heuristics. The reported results show that this meta-learning approach is suitable for algorithm selection, yielding good and balanced classification performance with a low addition in computational effort.**

***Index Terms*—Algorithm Selection, Meta-learning, Combinatorial Optimization, Fitness Landscape Analysis**

Student level: MSc
Date of conclusion: February 13, 2019
Examining board members:

- Prof. Dr. Aurora Trinidad Ramirez Pozo - Universidade Federal do Paraná
- Prof. Dr. Luiz Eduardo Soares de Oliveira - Universidade Federal do Paraná
- Prof. Dr. Roberto Santana Hermida - Universidad del Pais Vasco

On-line version of the dissertation: http://www.inf.ufpr.br/aldantas/dissertation.pdf
Journal publication:

- A. Dantas and A. Pozo, "On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem," *Theoretical Computer Science*, vol. 805, pp. 62 – 75, 2020.

Conference publications:

- A. L. Dantas and A. T. R. Pozo, "A meta-learning algorithm selection approach for the quadratic assignment problem," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8.
- A. Dantas and A. Pozo, "Selecting algorithms for the quadratic assignment problem with a multi-label meta-learning approach," in *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, Oct 2018, pp. 175–180.

I. Introduction

In optimization problems, the goal is to seek the minimization or maximization of one or more functions through the attribution of a set of variables, sometimes subjected to constraints [1]. The problems can be classified into two major categories according to the nature of the variables (continuous or discrete). A Combinatorial Optimization Problem (COP) is the search for a solution in a discrete and finite search space. COPs have great practical importance, making them the object of study of several works [2].

The Quadratic Assignment Problem (QAP) is one of the most challenging COP. The QAP was initially derived as a mathematical model of assigning a set of economic activities to a set of locations [3]. The goal is to minimize the total flow between the facilities and the total distance between the locations. There are several real world problems that can be modeled as QAP, such as the typewriter keyboard design, the location of the hospital departments, the backboard wiring, among others [4].

Because the QAP is an NP-hard problem [5], the use of exact methods is unfeasible for instances with large sizes. As an alternative, we can use meta-heuristic approaches to obtain good solutions in a reasonable computational time. Meta-heuristics are generic methodologies that act as guiding strategies for heuristic search operators [1]. Several meta-heuristics have been proposed, each one with distinct properties. Some of them can be based on single-solution perturbation, like the Tabu Search [6], or can be population evolutionary approaches,

like the Genetic Algorithm [7]. Additionally, some meta-heuristics are constructive methods, such the Ant Colony Optimization [8].

Although the existing meta-heuristics have achieved satisfying results, they have different biases. This means that there is not a single algorithm able to outperform the others in all cases [9]. Therefore, some algorithms may be more suitable than others on different problems [10].

Because of that, there has been an increasing interest in studying ways to automatically choose the most suitable meta-heuristic for a particular problem instance. However, selecting an appropriate algorithm for a given problem is a difficult task [11] that requires expert knowledge on search algorithms [12]. In the machine learning community, the task of automated algorithm selection has been tackled by meta-learning (MtL). Meta-learning aims at understanding the relationship between the characteristics of the problem and the performances of the solving algorithms [13]. This, in turn, allows the selection of the most promising algorithm using an inductive learning process. Although the term was originally used for applications on classification and data mining problems, the concept has been extended to other application domains [13] such as regression, time-series forecasting, sorting, constraint satisfaction, optimization and recommender systems [14].

The success of a meta-learning approach relies mostly on the quality of the characteristics used to represent the instances, i.e., the meta-features. Some of them can be statistical properties of the instance definition, such as matrix and graph properties. Additionally, studies have shown that measures based on Fitness Landscape Analysis (FLA) can be used for this goal [15]. FLA is a technique that aims at gathering knowledge about the internal structure of a problem. This is done by analysing the shape of the function landscape formed by a set of sampled solutions, given their objective values and neighborhoods [16].

In this dissertation, we conducted a series of studies on automatic algorithm selection for the Quadratic Assignment Problem using meta-learning and FLA based features. The main contribution is the empirical analysis of the set of meta-features that we used for this task. Additionally, our meta-heuristic pool (the meta-labels) includes popular well-known methods (the Max-Min Ant System and the Robust Tabu Search) and some recently proposed methods (the Breakout Local Search and the Breakout Memetic Search).

In our first study [17], we investigated the algorithm selection problem as a single label classification for the instances from the QAPLIB benchmark [18]. The experiments were focused on improving not only the overall accuracy, but also the prediction performance on individual meta-labels. The best results were achieved after performing exhaustive feature selection and by employing a cascade classification scheme. Albeit the promising results, the additional work that was required for achieving good classification performance is undesirable for this type of application.

Besides, because more than one algorithm can have the same performance for a given instance, it is reasonable to handle this task as a multi-label learning problem. Therefore, in the next study [19], with the same experimental setup as before (QAP instances and meta-heuristic pool), we evaluated the selection using multi-labeled datasets. The results indicated that it is possible to achieve high classification performance without requiring much manual effort as we did in the single label experiments.

Following that, in our final study [20], we evaluated a larger set of FLA based meta-features. Moreover, we employed a different sampling methodology for extracting the features, highlighting the required computational time. Additionally, we added to the dataset new 17 large QAP instances. With high classification performance, this study shows the viability of building meta-learning datasets without the need of an expensive sampling methodology, which is a recurring problem of meta-learning using FLA. Hence, we demonstrate that the algorithm selection approach yields better optimization solutions across the instances instead of only using single meta-heuristic, and is faster than running all meta-heuristics.

The rest of this paper is organized as follows: in Section II we present the relevant related works. Section III defines the Quadratic Assignment Problem. Section IV explain the meta-learning concepts, giving details on the meta-features that we used and the algorithms to be selected. The main experimental process and results are given in Section V, whereas Sections VI and VII respectively show the runtime analysis and the meta-learning output performance. At last, we draw some conclusions and indicate future works in Section VIII.

II. Related Works

One of the first research on Algorithm Selection Problem (ASP) was proposed by Rice [21], which focus on selecting an algorithm from a portfolio that is likely to perform best based on measurable features of the problem instances. The understanding of problem instances and algorithm performance is a task that can be tackled with meta-learning approach. MtL has been used effectively in algorithm portfolio to predict the algorithm that likely performs best for unseen problems [13].

A pioneer work regarding automated algorithm selection for QAP has been done in [13], in which three approaches were evaluated for this task. First, a Multi Layer Perceptron was trained to predict the percentage deviation from the known optimal solutions, and then choose the algorithm with the least distance. Next, a Probabilistic Neural Network was used to select the best performing algorithm for each instance as a single label classification problem. Finally, the author used a Self Organizing Map to cluster the instances based on their features, and then select the algorithm for a new instance based on the assigned cluster.

More recently, the authors in [22] investigated algorithm selection for QAP using a K-Nearest Neighbors (KNN) model. With different parameters settings, a total of 10 algorithms configurations were considered. However, in order to label the entries of the dataset, the algorithms were clustered into 6 groups. Hence, the KNN had the task of predicting which cluster of algorithms is better for a given instance. The best

results were achieved when using $k = 1$, which means that the model was highly dependent on the existence of an instance very similar to the one being tested.

Another related work was presented in [15], where 117 instances, two meta-heuristic algorithms and a total of 34 meta-features from FLA were considered. Here, the labels were based on the dominance of the performances of the algorithms, which was determined by taking snapshots of the solution costs at determined times during the optimization. The best classification results were obtained by using Support Vector Machine, showing the representative power of the gathered FLA information.

Albeit showing promising results, the two first works made the experiments on small sets of instances. In [13], only 28 instances were analysed, whereas in [22] the size of the dataset was 47. Moreover, the mentioned works did not evaluate the individual classes performances. In [15], the only reported evaluation metric was the accuracy, which may be deceiving.

Besides, all of them only tackle the algorithm selection as a single label problem. Instead, the algorithm selection problem can be treated as a multi-label classification task, since there are instances in which more than one algorithm achieve equivalent performance. In [23], a multi-label approach is proposed for selecting meta-heuristics, but applied to the Traveling Salesman Problem. The authors investigated three transformation methods for dealing with multi-label learning. The first consists in replicating the features of the instances with many labels and assign to each copy only one of the labels. The second method simply removes from the dataset the multi-labeled instances. Finally, the third method transforms the original problem to several single-label problems and combines the predictions. Also, 4 classification models were evaluated, being that the best results were achieved by a Decision Tree with the third transformation method. Furthermore, this work demonstrated that multi-label algorithm selection for combinatorial optimization problems is a promising research direction.

Hence, our goal was to investigate the selection of meta-heuristics for the QAP, both in single and multi-label configurations, and with different sets of meta-features and meta-labels. Moreover, in our experiments we took into account not only the accuracy, but also the performance on individual classes and the computational effort.

III. Quadratic Assignment Problem

The Quadratic Assignment Problem can be described as the problem of assigning a set of n facilities to a set of n locations, the goal in QAP is to assign each facility into a unique location in order to minimize the total flow and distance between the associations. The problem can be formally defined as

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} \tag{1}$$

where S_n represents all possible permutations of a set $N = \{1, 2, \dots n\}$, f_{ij} and d_{ij} are the correspondent values in the flow and distance matrices, respectively, and the product $f_{ij}d_{\phi(i)\phi(j)}$ is the singular cost of assigning the facility i to the location $\phi(i)$ and the facility j to the location $\phi(j)$.

The instances are represented by two matrices that defines the flow between facilities and the distances between locations. Figure 1 exemplifies an instance of size 5 and one permutation as a possible solution. Considering that each element on the permutation represents a facility, their positions indicate the respective assigned locations. Therefore, in the example, the facility 3 is assigned to location 1, facility 4 to the location 2 and so on. Hence, the cost of this solution, according to (1), would be equal to 2140.

Fig. 1: Example of an instance and a solution

(a) Flow matrix

F	**1**	**2**	**3**	**4**	**5**
1	0	4	2	1	3
2	4	0	1	3	0
3	2	1	0	4	2
4	1	3	4	0	1
5	3	0	2	1	0

(b) Distance matrix

D	**1**	**2**	**3**	**4**	**5**
1	0	50	60	94	50
2	50	0	22	50	36
3	60	22	0	44	14
4	94	50	44	0	50
5	50	36	14	50	0

(c) Permutation

3	4	5	1	2

Across our experiments, we used all 135 instances [1] from the standard benchmark library QAPLIB [18]. Additionally, in our last conducted work, we included 17 new instances from [24], which were designed to be hard for heuristic search.

IV. Meta-learning

The field of meta-learning, especially with respect to algorithm selection and configuration, has been an active area of research since the seminal work of Rice (1976) [21]. According to [25], meta-learning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes.

Approaches based on meta-learning heavily rely on the characterization of the problem instances, i.e., the meta-data of the problem. One important component of the meta-data are the meta-features, that are informative properties of the problem instance that affect the performance of the algorithms. The other component is the meta-labels, which define or rank the most suitable algorithms for each case.

Under theses concepts, the algorithm selection problem is addressed like a traditional learning task. A meta-model is induced, which can be described like the meta-data capable to explain which algorithm works better than others in problems with specific characteristics. Then, this meta-model is used to predict the best algorithm(s) for a new problem. Figure 2 sumarizes the sequence of activities of the meta-learning approach.

[1] the instance `esc16f` was removed because its flow matrix has only zeroes.

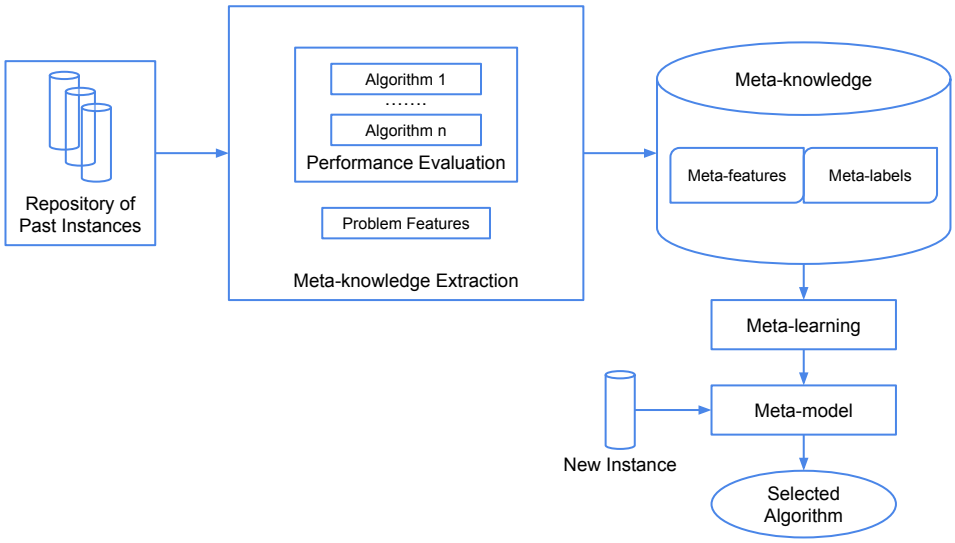

Fig. 2: Meta-learning sequence of activities, adapted from [25]

A. *Meta-features*

Meta-features are characteristics of a problem instance that ideally represent its inner information. They can be extracted directly from the instance definition, called the static features, or from a sampled set of solutions, which are based on Fitness Landscape Analysis (FLA).

1) Static: The static features are obtained by solely analysing the information contained in the instance definition. Therefore, they are the most easily extracted meta-features, specially when considering the required computing time. In our works, we used seven simple characteristics that may represent the QAP instances, which are described next. Apart from the instance size, the other features are matrix measures.

- Size (n): this meta-feature is straightforward the size of the instances, i.e., the number of facilities/locations associations.

- Flow (fd) and distance dominance (dd): the dominance is defined as the coefficient of variation of values and is obtained by $100 * \frac{\mu}{\sigma}$, with μ and σ being the mean and standard deviation of the matrices values, respectively. A high dominance indicates that the weight of the relationships are concentrated only on few pairs of items [26].

- Flow (fsp) and distance (dsp) sparsity: it is the amount of values 0 related to the total number of cells (n^2).

- Flow (fas) and distance (das) asymmetry: is the relative number of cells (i, j) that are different from cells (j, i) to the total number of pairs $\left(\frac{n^2-n}{2}\right)$.

2) Fitness Landscape Analysis: In FLA, the goal is to acquire some knowledge about the internal structure of a problem. A common way to achieve this is to analyse the properties of a collection of sampled solutions, given their costs and neighborhoods. Therefore, FLA is highly dependent on the operator that is used to gerante the neighbor solutions, also called the move operator. In this work, we use the swap operator, that exchanges the values between two elements found in positions i and j, as shown in Figure 3, and, for an instance of size n, produces $\frac{n(n-1)}{2}$ neighbors. This is the same operator used during the local search phase in the meta-heuristics described in Section IV-B.

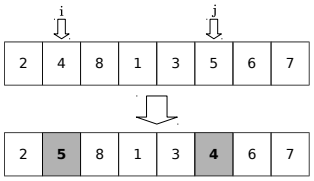

Fig. 3: The swap operator

FLA methods can be separated in two major categories, based on which type of sampling they use. It can be achieved by local methods, that examine the immediate changes found in paths made by random or directed walks, or global methods, that seek to obtain a higher overview of a landscape, which in done by analysing a sample of solutions (usually a set of local optima) [16]. In our studies, we focused on the later category.

The sampling methodology is an important aspect of FLA, because it is the most computationally expensive step. Initially, we collected the sample using the same strategy proposed by [27], which is also used in the related work from [13].

In this strategy, the Iterated Local Search (ILS) [26] is executed 1000 times, with 500 iterations each, and all achieved unique local optima solutions are stored. The best solutions within this set are called the pseudo global optima, which are required to compute some of the features, since it is unfeasible to know the actual global optima set. Next, the Best Improvement local search is randomly repeated until 5000 unique local optima are found, or until it exceeds a certain amount of execution time (which we set to 5 minutes). At the end, two sets of local optima are generated: the ILS set and the BI set, and the same measures are extracted from both of them (more details are given in Section V).

Then, in the later studies we wanted to investigate another set of meta-features using a possibly cheaper sampling strategy. Hence, we used the Metropolis-Hasting algorithm (MtH) [28].

The MtH aims at sampling the space by giving more importance to solutions with better fitness values, thus avoiding the sample of solutions belonging to the same plateau [28]. Therefore, the algorithm iteratively adds to the sample a solution with better fitness than the previous by a random factor. This process is shown in Algorithm 1, where m is the sample size, $f(\gamma_k)$ is the fitness of individual γ_k, u is the factor of acceptance and the α function is given by

$$\alpha(x, y) = min\left\{1, \frac{y}{x}\right\} \tag{2}$$

Algorithm 1: Metropolis-Hastings

```
Result: The sample γ with m solutions
γ_1 ← random solution
for k = 2 to m do
    repeat
        φ ← random solution
        u ← random number from uniform (0,1) distribution
    until u <= α(f(γ_{k-1}), f(φ))
    γ_k ← φ
    k ← k + 1
end
```

The MtH is executed until the sample reaches the size of m. Then, the Best Improvement local search is applied on every solution from the sample and all unique local optima are stored. Therefore, this strategy results in two sets of solutions: the base set, given by the MtH, and the local optima set from the BI executions.

Next we describe the FLA meta-features that we used along our studies. Some of them, such as the Fitness Distance Correlation, are used in all experiments, some others only appear in the last study. In Section V we detail which meta-features are used in each study.

- Fitness Distance Correlation (FDC): considering a set of solutions, the FDC measures the distance correlation between the distances of each solution to the closest global optimum and their costs [29]. The distance in this context is the number of different assignments on two solutions, also known as the Hamming Distance. Let c_i and d_i be respectively the cost and distance of the i_{th} solution from a set of size m, the FDC is calculated by

$$FDC = \frac{\frac{1}{m}\sum_{i=1}^{m}(c_i - \mu_c)(d_i - \mu_d)}{\sigma_c.\sigma_d} \tag{3}$$

in which μ_c and μ_d are the average cost and distance, with σ_c and σ_d being the respective standard deviations. However, the global optima solutions are usually unknown. Hence, we use instead a set of pseudo-global optima, which are the best solutions from the sampled local optima.

- Number of pseudo global optima: is the size of the set of the best unique local optima solutions.

- Average distance to optima: this is the distance to the closest (pseudo) global optimum solution (the μ_d itself).

- Accumulated Escape Probability: it is an evolvability measure that classifies the hardness of a problem for an Evolutionary Algorithm (the higher, the easier) [30]. It is the average of the escape probability of every fitness levels of the solution from the sample. The escape probability here is calculated as the amount of equal or better neighbor solutions for a given point divided by the neighborhood size.

- Dispersion Metric: the dispersion of a set of solutions is the average pairwise Hamming distance of the set. The Dispersion Metric is given by the dispersion of the t best solutions of the sample minus the dispersion of the first t solutions from the same sample. The key idea is to measure the change in dispersion when improving the fitness of the solutions [31]. In this work, we set t to be 5% of the considered sample size.

- Average descent: this is the number of movements performed by the local search until reaching the local optima from the starting point.

- Optima fitness coefficient: this metric is given by dividing the standard deviation of the fitnesses of all optima solutions by their mean.

Additionally, in our first conducted study, we included two meta-features belonging to the group of FLA by local method, namely the autocorrelation coefficient (acc) and the correlation length (acl), which are metrics that represent the ruggedness of a landscape. The former is the average variation of fitness caused by a single step along the neighborhood, whereas the later is the number of required steps to make the correlation values statistically different [15]. Although this type of FLA is more computational expensive, the authors from [32] proposed the calculation of the acc and acl metrics in polynomial time and without the need for sampling. They also provided the calculated values for all instances from QAPLIB.

B. Meta-labels

The meta-labels represent the most suitable algorithms for a problem instance. Therefore, to build the learning dataset, they are executed on all instances and have their performance measured. In our experiments, we compared the algorithms by means of average solution cost.

The meta-label is then defined by the performances of the competing meta-heuristics, which were chosen because of their reported results, properties of exploration and availability of implementation. Our first studies use the algorithms Breakout Local Search (BLS) [33], the Max-Min Ant System with Best Improvement Local Search (MMASBI) [27] and the Robust Tabu Search (RO-TS) [34]. In the last work, due to the relatively poor performance of the RO-TS, we replaced it to the Breakout Memetic Algorithm (BMA) [35].

They all have in common the appliance of a local search as an intensification mechanism. More precisely, they employ the Best Improvement (BI) local search, in which the whole neighborhood is first scanned and then it moves to the best solution among them [4].

Because we are using the swap operator, it is possible to make the BI faster by, instead of calculating the $O(n^2)$ cost function (Equation 1) for each neighbor, we calculate just the

cost $\delta(\phi, i, j)$ of swapping the elements from positions i and j in permutation ϕ, with the linear equation [36]:

$$\begin{aligned}\delta(\phi,i,j) = d_{ii} * (f_{\phi(j)\phi(j)} - f_{\phi(i)\phi(i)}) + d_{ij} * (f_{\phi(j)\phi(i} - f_{\phi(i)\phi(j)}) + \\ d_{ji} * (f_{\phi(i)\phi(j)} - f_{\phi(j)\phi(i)}) + d_{jj} * (f_{\phi(i)\phi(i} - f_{\phi(j)\phi(j)}) + \\ \sum_{k=1,k\neq i,j}^{n} (d_{ki} * (f_{\phi(k)\phi(j)} - f_{\phi(k)\phi(i)}) + d_{kj} * (f_{\phi(k)\phi(i)} - f_{\phi(k)\phi(j)}) + \\ d_{ik} * (f_{\phi(j)\phi(k)} - f_{\phi(i)\phi(k)}) + d_{jk} * (f_{\phi(i)\phi(k)} - f_{\phi(j)\phi(k)}))\end{aligned} \tag{4}$$

This computational cost can be even further reduced by using information from preceding iterations. For the cases where the swapping indexes $\{u, v\}$ are different from the previous indexes $\{i, j\}$ that were used to generate the current permutation ϕ', such as that $(\{u, v\} \cap \{i, j\} = \emptyset)$, the movement cost can be calculated in constant time by [36]:

$$\begin{aligned}\delta(\phi', u, v) = \delta(\phi, i, j) * (d_{ru} - d_{rv} + d_{sv} - d_{su}) * (f_{\phi(j)\phi(u)} - f_{\phi(j)\phi(v)} + \\ f_{\phi(i)\phi(v)} - f_{\phi(i)\phi(u)}) * \\ (d_{ur} - d_{vr} + d_{vs} - d_{us}) * (f_{\phi(u)\phi(j)} - f_{\phi(v)\phi(j)} + \\ f_{\phi(v)\phi(i)} - f_{\phi(u)\phi(i)})\end{aligned} \tag{5}$$

Next, we briefly explain the meta-heuristics that we are learning to select from.

1) Breakout Local Search: The BLS algorithm is similar to the Iterated Local Search in the way that it initially applies a local search procedure until a local optimum is found, followed by a perturbation operation to jump to new search regions. The difference is that BLS may apply three distinct perturbation moves, depending on the search stage. The more often executed one is the directed perturbation, which is based on tabu search principles, meaning that it favors movements which improve the cost function and that have not been recently applied, thus performing an exploitation search. The other two are the recency-based perturbation, that favors the least recently performed moves, and the random perturbation. Neither of them takes into account the cost degradation, resulting in a more explorative behavior [33].

2) Max-Min Ant System: Following the Ant Colony Optimization concepts, the MMAS is a constructive algorithm that probabilistically chooses the assignments based on two information: the heuristic (problem-specific) and the pheromone trail (based on experience of the solutions constructed so far) [8]. The main difference introduced by MMAS is that it imposes maximum and minimum limits to the allowed values of pheromone. This is intended to prevent search stagnation, where all ants produce the same solution due to a high predominance of that path in the pheromone matrix [27]. A hybrid version of MMAS was employed in which, after every ant finishes the constructions phase, the best improvement local search is applied to the solution.

3) Robust Tabu Search: The Tabu Search algorithm is a simple meta-heuristic that accepts degrading movements when stuck in local optima. Then, in order to avoid returning to the same positions, it maintains a list of forbidden movements for a given number of iterations, or until the aspiration criteria is met, which is traditionally when the tabu movement results in a solution better than the best found so far [6]. The robust version introduces a randomly variable duration for considering each movement as tabu, and also an additional aspiration criteria which is met when the elements being swapped will be placed to positions that they have not being for the last defined number of iterations [34].

4) Breakout Memetic Algorithm: The BMA is a steady-state evolutionary algorithm that, in each generation, the BLS is applied on the new offspring solution. It uses the Uniform Crossover (UX) operator for reproduction. The UX creates an offspring by assigning at each position one element from either of the parents in that same position with equal probability, as long as the element has not been assigned yet. At the end, if there are unassigned positions, they are set randomly among the remainder values [35]. The BMA also applies an adaptive mutation procedure on the whole population when the best found solution has not changed for a certain amount of generations. The mutation operator exchanges a number of elements in order to create a solution with a determined Hamming distance, which is increased after each mutation appliance until being reset [35].

V. Experimental Results

For this dissertation, we conducted a series of incremental studies on automatic algorithm selection for the Quadratic Assignment Problem. At each time, we identified some weaknesses of the approach, and then proceeded to address them in the next study. With this, at the end we were able to reliably select algorithms with a multi-label approach using the described FLA based meta-features.

In all experiments, we used the Random Forest (RF) as classification model. Specifically, we used the implementation from the `scikit-learn` library [2]. The only parameters that we set were to use 100 tree estimators and a tree depth limit of 10, the other parameters were left as the default. The RF classifier is a model based on an ensemble of decision tree classifiers, where each tree is trained using a bootstrapped subset of the training samples and a random subset of the features [37]. The classification is then given by averaging all decision tree outputs.

For ranking the algorithms (which defines the meta-labels), we must compare their performance. Therefore, the executions of the meta-heuristics should have the same allowed effort, i.e., the same stopping criteria. Some works have used the maximum number of iterations for this end [17], [38], which can be hard to set if we are dealing with very different algorithms. For this same reason, using the number of solution evaluations as the stopping criteria [15], [22], [39] may disregard the cost of other operations present in the algorithms, which could even be more expensive or more frequently used [40].

Another possibility is to use the CPU time for this purpose [23], [35], [41], [42], which has the disadvantages of being sensible to implementation, compilation, and hardware settings

[2]http://scikit-learn.org/stable/index.html

[40]. Nevertheless, it is still a valid approach if some care is taken in the experiments.

Initially, for the single label experiment, we used the number of iterations as the stopping criteria and the CPU time for untying the performance (when the average objectives are the same). Thereafter, tn the multi-label experiments, we ran the meta-heuristics limited to CPU time (which varies according to the instance size).

For all datasets that we built, each meta-heuristic is executed 30 times for every QAP instance (always using the same set of 30 seeds).

In the first study [17], we investigated the meta-learning approach as a single label classification problem. Here, we built the dataset by extracting 12 meta-features and with the BLS, MMASBI and RO-TS algorithms as meta-labels.

We set as the stopping criteria for the meta-heuristics the number of iterations, which was chosen according to preliminary tests. The BLS and MMASBI were limited to $100 * n$ iterations, where each iteration contains a full local search appliance. For the RO-TS, the stopping criteria was $2000 * n$ iterations, because here each iteration is equivalent to one single swap movement.

Then, for labeling the instances, we ranked the three algorithms based on their average solution costs over the 30 executions and chose the one with the least average cost. In case of ties, the selection was based on the average execution time. Table I shows the resulting classes distribution.

TABLE I: Full dataset

Class	Instances
BLS	33
MMASBI	89
RO-TS	13

Besides the seven static features described in Section IV-A1, we extracted seven FLA meta-features, shown in Table II. The values of the acc and acl meta-features for all QAPLIB instances are given in [32]. The other five features are the same used in [13]. They are extracted using the sampling method proposed by [27], which consistis of collecting the solutions found by runs of the Iterated Local Search (ILS) and the Best Improvement algorithm (BI) (as described in Section IV-A2).

Initially, we trained the Random Forest for the complete dataset and with the 14 meta-features, in which, due to class imbalance, the minority classes had considerably worse performance. Therefore, some data cleansing was necessary to improve the results. For this, we inspected the optimization performance and noticed that for several instances, mostly with small sizes, the three meta-heuristic achieved the same average cost. So, a new and more balanced dataset was generated by removing those instances.

Following this, we managed to further improve the classification performance by reducing the dimensionality of the dataset. Because there were only 14 features, it was feasible to make an extensive search of the best subset of features. We acknowledge that doing it over the cross-validation folds rises the risk of overfitting the model, as is the case for any other supervised feature selection approach [43]. Even so, we wanted to discover if there was an optimal subset of features capable of improving both the accuracy and individual classes performances for the existing benchmark instances.

Finally, when observing that there was still a large difference in the individual classes performances, we concluded that it would be better to project our multi-class problem into sequential binary problems, resulting in the proposed cascading scheme.

In this approach, the multi-class problem is divided into two binary output problems, with each being tackled by a specifically trained Random Forest. At the first level, the model is trained to classify the class MMASBI (the majority class) against the joint of classes BLS and RO-TS. If the output of this classification is not MMASBI, then the input is forwarded to a second model that is trained to distinguish only the instances belonging to classes BLS and RO-TS. Moreover, in order to better discriminate the classes, the models were trained using their respective optimal subset of features, which were found after performing exhaustive searches. Figure 4 illustrates the proposed scheme, along with the best meta-features for each model.

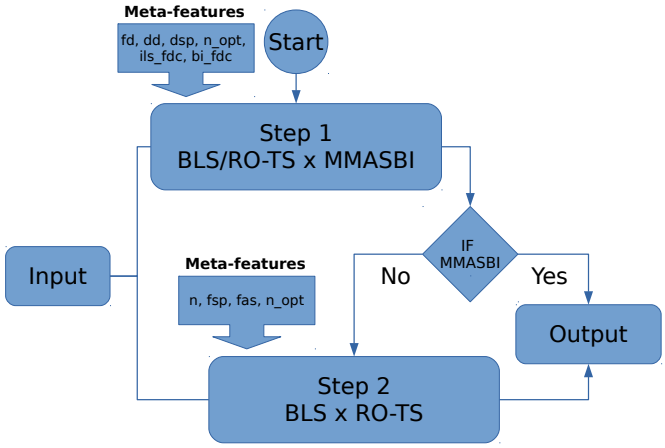

Fig. 4: Proposed cascade classification scheme

Table III shows the prediction performances (accuracy and F-score of the labels) of the cascade approach. The evaluation was made with the 10-Fold cross-validation strategy. With this, we were able to achieve a satisfactory accuracy and with similar performance on each class, which is important in a algorithm selection application.

However, the good classification results were achieved only after several manual steps, which is not desirable for an automated algorithm selection application. Therefore, in the next study [19] we investigated the algorithm selection task as a multi-label classification problem, using the same experimental setup as before: same QAP instances, meta-heuristics and meta-features (except the acc and acl meta-features).

In multi-label classification, the data entries can be assigned to more than one class simultaneously [44]. Since more than one meta-heuristic can have the same performance for a

TABLE II: FLA based meta-features in the single label experiment

Abbreviation	Name	Description
n_opt	Number of pseudo global optima	Best unique solutions found by ILS
ils_dst	Average distance to optima	Hamming distance to the closest pseudo optimum
bi_dst		
ils_fdc	Fitness Distance Correlation [29]	$FDC = \frac{\frac{1}{m}\sum_{i=1}^{m}(c_i-\mu_c)(d_i-\mu_d)}{\sigma_c.\sigma_d}$
bi_fdc		
acc	Autocorrelation coefficient [32]	Average fitness variation caused by a single step along the neighborhood
acl	Correlation length [32]	Number of required steps to make the correlation values statistically different

TABLE III: Evaluation metrics for the cascade scheme classification

Accuracy	F-score			
	BLS	MMASBI	RO-TS	Average
0.8765	0.875	0.8912	0.8333	0.8665

given instance, this approach is adequate to perform algorithm selection [23]. Moreover, we did not performed the extensive feature selection anymore.

Here, we used the CPU time as the stopping criteria. With the average times that the algorithms spent to reach their best found solutions in the previous study, we fit a polynomial function that gives the maximum CPU time for an instance according to its size. In this way, for an instance with size 150, for example, the meta-heuristics are executed for 17 minutes. An instance of size 256, in turn, requires 2 hours. Then, after comparing the average cost over the 30 runs, the class distribution ended up as displayed in Figure 5.

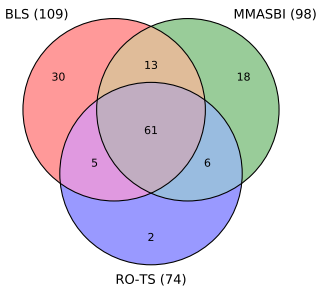

Fig. 5: Multi-label dataset (BLS/MMASBI/RO-TS)

We compared two approaches for dealing with multi-label classification. The first is based on transforming the dataset into a multi-class single label dataset by considering each subset of classes as a distinct label (powerset labels). Therefore, we turned the problem into a single label problem by treating each subset of classes as a unique label. In this way, the model is trained to predict among 7 different labels, shown in Table IV.

Alternatively, the learning algorithm can be trained directly on the multi-label configuration. In the literature, this approach is called algorithm adaptation [45]. In the *scikit-learn* RF implementation, the multi-label task is handled by, instead of assigning only the class with the highest mean probability, it assigns to all classes with a probability higher than 0.5.

TABLE IV: Powerset labels

Subset	Samples
only BLS	30
only MMASBI	18
only RO-TS	2
BLS/MMASBI	13
BLS/RO-TS	5
MMASBI/RO-TS	6
BLS/MMASBI/RO-TS	61

We reported the classification performance as the accuracy and the F-scores of each individual class. However, for multi-label classification, there are different types of accuracies that can be measured. In this work, we used the traditional instance accuracy, given by the Jaccard similarity coefficient between the true and predicted labels, and the subset accuracy, which requires that the predicted set of labels to be an exact match of the original label set [44].

Additionally, we used what we called the recommendation accuracy, that evaluates if the system is able to recommend only the best performing algorithms, i.e., it considers the classification as correct if there are no false positives for that instance. As far as we know, this type of accuracy is usually not discussed in the multi-label learning literature, and has not yet been applied in the context of algorithm selection for optimization problems.

Initially, we observed very high classification performance. However, almost half of our dataset was labeled to all three meta-heuristics (the BLS/MMASBI/RO-TS powerset), thus increasing the accuracies. Thus, we calculated the accuracy metrics (Table V) without those instances (using the Leave-one-out cross-validation strategy).

TABLE V: Accuracies of the multi-label classification on the reduced dataset

Approach	Instance	Subset	Recommendation
Powerset Labels	0.802	0.676	0.703
Multi-labels	0.818	0.703	0.720

Although the models did not achieve a very high performance, the recommendation accuracy was still considerably above random. Besides, this low performance was mainly due to the prediction of the minority class (RO-TS), as displayed in Table VI.

Therefore, the multi-label approach still demonstrated to be able to provide good prediction results without additional

TABLE VI: Labels F-scores on the reduced dataset

Approach	BLS	MMASBI	RO-TS	Average
Powerset Labels	0.835	0.909	0.538	0.760
Multi-labels	0.823	0.947	0.560	0.778

manual effort. Both strategies had similar performance, but letting the model learn directly on the multi-label dataset, instead of transforming it into powerset labels, had a small advantage in terms of classification.

For the next study [20], we targeted the weaknesses that we identified before. First, one of the meta-heuristics, the Robust Tabu Search (RO-TS) [34] had a considerably worse performance compared to the other algorithms, making the meta-learning dataset highly unbalanced.

Furthermore, Figure 6 shows the boxplot distributions of each meta-feature for each label from the single label dataset. We can notice that the instances labeled to RO-TS were very similar to those labeled as BLS, meaning that both meta-heuristics work better on instances of similar nature.

These similarities are actually not surprising, since both the RO-TS and BLS algorithms share same common properties. Both are single solution based with iterative improvement strategies. And, because the RO-TS outperformed the other algorithms on fewer instances, it ends up getting more disadvantaged from these similarities.

Because of that, we replaced the RO-TS by the more novel Breakout Memetic Algorithm [35]. By doing so, we also improved the diversity of the nature of our algorithm pool by having one perturbative (BLS), one constructive (MMASBI) and one populational evolutionary strategy (BMA).

Moreover, we increased the dataset by including additional QAP instances that were designed to be hard for heuristic search [24]. Specifically, they are 12 instances of the type `drexx` with sizes ranging from 15 to 132, and 5 instances of the type `taixxeyy`, all with the size of 343, which is very large in the context of the QAP.

Another key improvement of this study is that we compared the meta-heuristics using hypothesis test. Therefore, the meta-label for a particular instance is set as the algorithm with the lowest average cost (over the 30 runs), and all the algorithms with no statistical difference to it (with a confidence level of 95%). For this, we used the Kruskal-Wallis hypothesis test with the Nemenyi post-hoc test. Figure 7 shows the resulting dataset distribution.

Finally, and mainly, we used a new set of FLA based meta-features from the literature, such as the Accumulated Escape Probability and the Dispersion Metric (as discussed in Section IV-A2). Besides that, the main drawback of FLA features for algorithm selection is the high time consumption for their extraction [15]. Therefore, we investigated a cheaper sampling methodology, using the Metropolis-Hasting algorithm [28] (Section IV-A2).

We ran the MtH until a maximum of 5000 unique solutions were stored, which are then referenced as the set of base solutions. With this set, the two following meta-features are extracted:

- Accumulated Escape Probability (aep)
- Base Dispersion Metric (base_dm)

Then, for each base solution, we applied the Best Improvement local search and stored all unique local optima. This set of local optima is used to calculate the remaining 5 FLA meta-features:

- Optima Fitness Coefficient (opt_fit_coef)
- Average Descent (avg_descent)
- Fitness Distance Correlation (fdc)
- Average Distance to Optima (avg_dst)
- Optima Dispersion Metric (opt_dm)

Since one of our goals is to investigate the computational effort for using FLA features, we built five datasets considering different sizes of the MtH sampling: 100, 500, 1000, 2500, and 5000.

Table VII shows the performance over each of the five datasets with the Leave-one-out cross-validation strategy. Although all sampling sizes resulted in very high instance accuracies, this is due to the high imbalance present in the datasets (Figure 7). Nevertheless, we achieved satisfactory subset accuracies in all cases, with a small advantage for the meta-features extracted with the sample of size 1000.

TABLE VII: Classification accuracies

Sample size	Instance Accuracy	Subset Accuracy
100	0.92	0.84
500	0.93	0.85
1000	0.93	0.86
2500	0.93	0.85
5000	0.93	0.84

In fact, by looking at the subsets F-scores in Table VIII we can notice that the five datasets performed similarly well on the three majority subsets. Again, the sampling size of 1000 resulted in an overall slightly better performance, since it had at least the second-best F-score in all cases. However, the models failed to correctly classify the minority cases, which is reasonable considering the small number of instances on such subsets.

TABLE VIII: Subsets F-scores

Subset (size)	Sampling size				
	100	500	1000	2500	5000
only BLS (1)	0.00	0.00	0.00	0.00	0.00
only MMASBI (13)	0.72	0.66	0.76	0.76	0.76
only BMA (4)	0.00	0.00	0.00	0.00	0.00
BLS/MMASBI (2)	0.00	0.00	0.00	0.00	0.00
BLS/BMA (36)	0.88	0.90	0.89	0.87	0.86
MMASBI/BMA (7)	0.37	0.37	0.46	0.40	0.40
BLS/MMASBI/BMA (89)	0.92	0.93	0.92	0.92	0.91

Moreover, in order to investigate if there is any difference among the five datasets when learning a classification model, we compared their average Cross-Entropy Error (CCE). The CEE is computed by the sum of the logarithmic error of each true class. In our context, the error is given by $1 - p$, in which p is the class assignment probability given by the Random Forest.

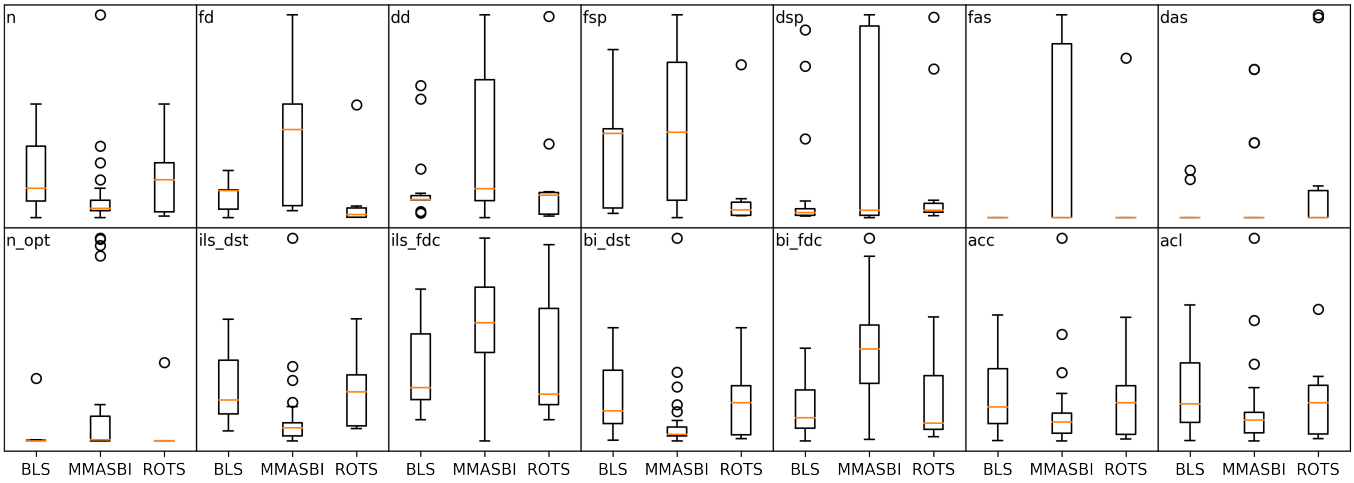

Fig. 6: Features distributions for each class

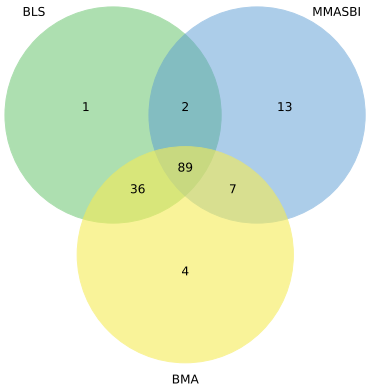

Fig. 7: Multi-label dataset (BLS/MMASBI/BMA)

Table IX displays the means and standard deviations of the CEEs. The datasets marked with gray background had no statistically difference ($p > 0.05$) from the one with the lowest mean (in boldface). We used a pairwise Wilcoxon hypothesis test with the Bonferroni correction for this comparison.

TABLE IX: Average Cross-Entropy Errors

Sample size	Cross-Entropy Error (std)
100	0.331 (1.318)
500	0.253 (0.957)
1000	0.191 (0.332)
2500	0.169 (0.269)
5000	**0.166 (0.306)**

Based on these results, we concluded that the sampling size of 1000 is an adequate choice for this algorithm selection task, since it had both high accuracies and good performance for training the model, as indicated by the CEE.

VI. Runtime Comparison

Additionally, in this study we compared the runtime consumption of the meta-learning approach using both sampling methodologies: the ILS/BI sampling that we used in the first two studies, and the Metropolis Hasting sampling with 1000 samples.

Figure 8 illustrates their CPU time performance by instance size. This includes the time required for running one selected meta-heuristic. The 1000 MtH required less time than the ISL/BI, whereas both were better than running all three meta-heuristics.

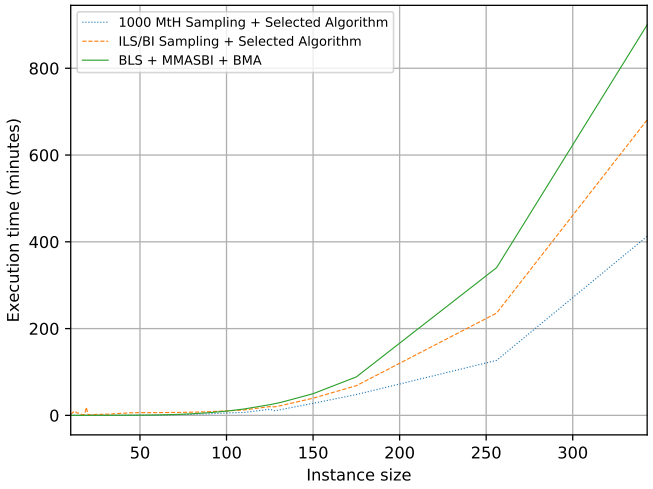

Fig. 8: CPU time of each approach by instance size

These differences on runtime between the sampling approaches are coherent if we consider the amount of total and partial solution evaluations. Both the MtH and BI are similar in the way that, for every iteration, an $O(n^2)$ evaluation is performed (Equation 1) on random solutions, followed by local searches with $\frac{n(n-1)}{2}$ $O(n)$ evaluations (Equation 4) and

several $O(1)$ evaluations (Equation 5). The main difference, however, is that the MtH is executed until 1000 random base solutions are accepted, whereas the BI requires 5000 unique local optima solutions. Furthermore, the ILS/BI sampling has still the cost of 1000 ILS appliances, meaning 1000 complete evaluations ($O(n^2)$) and the same linear and constant evaluations related to local search, with the additional $O(n)$ evaluations made due to the perturbation moves.

VII. Algorithm selection outcome

With the trained model from the last study (using the sample size of 1000), we investigated the efficiency of the selection approach in terms of the overall achieved QAP solutions. Therefore, we compared the objective values obtained by one selected algorithm against the objective values obtained with an Oracle approach, that always select the meta-heuristic with the best performance. However, because we are dealing with multi-label classification, there is more than one algorithm that could be chosen for execution, that, in turn, can yield different results. Hence, we consider the case in which the best correctly predicted algorithms for each instance are chosen (Selected (best)), i.e., only true positive labels, and the case that this choice is performed at random among the predicted classes (Selected (random)). The former is an idealistic scenario, whereas the latter is more realistic to an actual application.

Moreover, we also compared an approach that always chooses the algorithm related to the class with the highest probability estimate from de model (Selected (prob)), which would be the case in a single label classification task. Figure 9 shows, for each scenario, the number of instances where the selection approach was statistically equivalent to the Oracle approach. For comparing each two sets of 30 objective values (one approach versus the Oracle), we used the Kruskal-Wallis hypothesis test at a confidence level of 95%. Additionally, we also report the performance of a purely random selection strategy (Random) and the performance of each meta-heuristic individually (BLS, MMASBI, and BMA).

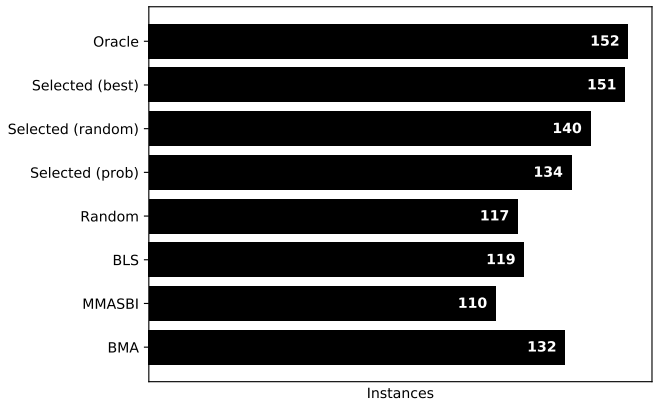

Fig. 9: Amount of instances that each selection approach achieved equivalent performance to the Oracle

In the Selected (best) case, our approach was very close to the Oracle, meaning that the classifier could correctly predict at least one true positive label for almost all instances. In turn, the more realistic Selected (random) scenario outperformed the Selected (prob), thus highlighting the advantage of using a multi-label approach for this task instead of treating it as a single label problem. Nevertheless, the meta-learning strategies were better than running only one meta-heuristic on all instances or making the selection at random.

VIII. Conclusion

This research presented a set of incremental studies about algorithm selection with meta-learning concepts and measures based on Fitness Landscape Analysis. Each progression of the work was responsible for increasing the robustness of the approach and also was fundamental to draw some different conclusions.

The first work highlights the potential of achieving high classification accuracy provided that the practitioner performs some steps to improve the results. It showed not only overall good accuracy, but the performance across the labels were also balanced.

Then, we changed our approach to a multi-label strategy, where more than one algorithm can be set as meta-label for a given instance. The results from this study show that, even without the manual effort made in the previous study, the selection approach yield reliable results. From this, we draw the conclusion that treating the algorithm selection problem as a multi-label task is adequate.

Finally, we investigated a meta-learning approach with a new set of meta-features based on Fitness Landscape Analysis. The results demonstrated that it is possible to obtain high classification performance with metrics that, in turn, can be extracted using low time consumption sampling, which is a recurring problem of FLA. Moreover, by using the meta-learning selection approach, we could improve the overall performance on the set of QAP instances, without adding too much effort in the process.

We used as a case study the Quadratic Assignment Problem, which is known for being a very hard problem and also being a generalization of other combinatorial problems. However, an interesting topic for future investigation is the performance of these FLA based meta-features on selecting algorithms for different optimization problems.

Moreover, one of the main drawbacks of the study was the relatively small number of instances. Therefore, in future work the dataset can be increased by introducing newly artificial instances created with a generator, which would improve the robustness of the results.

Additionally, the relatively low-cost extraction process indicates that this type of features could also be used in other automated techniques for solving optimization problems, such as hyper-heuristics and adaptive operator selection.

Acknowledgements

This work was financially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

REFERENCES

[1] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[2] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[3] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: Journal of the Econometric Society*, pp. 53–76, 1957.

[4] R. E. Burkard, E. Cela, P. M. Pardalos, and L. S. Pitsoulis, "The quadratic assignment problem," in *Handbook of Combinatorial Optimization*. Springer, 1998, pp. 1713–1809.

[5] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 555–565, 1976.

[6] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[7] J. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[8] M. Dorigo and G. D. Caro, "Ant colony optimization: a new metaheuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2, 1999, p. 1477.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] C. Schumacher, M. D. Vose, and L. D. Whitley, "The no free lunch and problem description length," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 565–570.

[11] K. Tang, F. Peng, G. Chen, and X. Yao, "Population-based algorithm portfolios with automated constituent algorithms selection," *Information Sciences*, vol. 279, pp. 94 – 104, 2014.

[12] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Applied Soft Computing*, vol. 11, no. 6, pp. 4135 – 4151, 2011.

[13] K. A. Smith-Miles, "Towards insightful algorithm selection for optimisation using meta-learning concepts," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 4118–4124.

[14] T. Cunha, C. Soares, and A. C. de Carvalho, "Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering," *Information Sciences*, vol. 423, pp. 128 – 144, 2018.

[15] E. Pitzer, A. Beham, and M. Affenzeller, "Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2013, pp. 109–120.

[16] E. Pitzer, M. Affenzeller, A. Beham, and S. Wagner, "Comprehensive and automatic fitness landscape analysis using heuristiclab," in *International Conference on Computer Aided Systems Theory*. Springer, 2011, pp. 424–431.

[17] A. L. Dantas and A. T. R. Pozo, "A meta-learning algorithm selection approach for the quadratic assignment problem," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8.

[18] R. E. Burkard, S. E. Karisch, and F. Rendl, "Qaplib – a quadratic assignment problem library," *Journal of Global Optimization*, vol. 10, no. 4, pp. 391–403, 1998.

[19] A. Dantas and A. Pozo, "Selecting algorithms for the quadratic assignment problem with a multi-label meta-learning approach," in *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, Oct 2018, pp. 175–180.

[20] ——, "On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem," *Theoretical Computer Science*, vol. 805, pp. 62–75, 2020.

[21] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[22] A. Beham, M. Affenzeller, and S. Wagner, "Instance-based algorithm selection on quadratic assignment problem landscapes," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 1471–1478.

[23] J. Kanda, A. Carvalho, E. Hruschka, and C. Soares, "Selection of algorithms to solve traveling salesman problems using meta-learning 1," *International Journal of Hybrid Intelligent Systems*, vol. 8, no. 3, pp. 117–128, 2011.

[24] Z. Drezner, P. M. Hahn, and É. D. Taillard, "Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods," *Annals of Operations Research*, vol. 139, no. 1, pp. 65–94, Oct 2005.

[25] P. Brazdil, C. G. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning - Applications to Data Mining*, ser. Cognitive Technologies. Springer, 2009.

[26] T. Stützle, "Iterated local search for the quadratic assignment problem," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1519–1539, 2006.

[27] T. Stützle and H. H. Hoos, "Max–min ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[28] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vérel, "Fitness clouds and problem hardness in genetic programming," in *Genetic and Evolutionary Computation – GECCO 2004*, K. Deb, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 690–701.

[29] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 184–192.

[30] G. Lu, J. Li, and X. Yao, "Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms," in *Evolutionary Computation in Combinatorial Optimization*, P. Merz and J.-K. Hao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 108–117.

[31] M. Lunacek and D. Whitley, "The dispersion metric and the cma evolution strategy," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 477–484.

[32] F. Chicano, G. Luque, and E. Alba, "Autocorrelation measures for the quadratic assignment problem," *Applied Mathematics Letters*, vol. 25, no. 4, pp. 698–705, Apr. 2012.

[33] U. Benlic and J.-K. Hao, "Breakout local search for the quadratic assignment problem," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4800–4815, Jan. 2013.

[34] É. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel computing*, vol. 17, no. 4-5, pp. 443–455, 1991.

[35] U. Benlic and J.-K. Hao, "Memetic search for the quadratic assignment problem," *Expert Systems with Applications*, vol. 42, no. 1, pp. 584 – 595, 2015.

[36] E. D. Taillard, "Comparison of iterative searches for the quadratic assignment problem," *Location Science*, vol. 3, no. 2, pp. 87–105, 1995.

[37] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[38] A. D. de León, E. Lalla-Ruiz, B. Melián-Batista, and J. M. Moreno-Vega, "Meta-learning-based system for solving logistic optimization problems," in *International Conference on Computer Aided Systems Theory*. Springer, 2017, pp. 339–346.

[39] X. Chu, F. Cai, C. Cui, M. Hu, L. Li, and Q. Qin, "Adaptive recommendation model using meta-learning for population-based algorithms," *Information Sciences*, vol. 476, pp. 192–210, 2019.

[40] J. Silberholz and B. Golden, "Comparison of metaheuristics," in *Handbook of metaheuristics*. Springer, 2010, pp. 625–640.

[41] E. S. Miranda, F. Fabris, C. G. Nascimento, A. A. Freitas, and A. C. Oliveira, "Meta-learning for recommending metaheuristics for the maxsat problem," in *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2018, pp. 169–174.

[42] H. Degroote, J. L. González-Velarde, and P. De Causmaecker, "Applying algorithm selection–a case study for the generalised assignment problem," *Electronic Notes in Discrete Mathematics*, vol. 69, pp. 205–212, 2018.

[43] P. Smialowski, D. Frishman, and S. Kramer, "Pitfalls of supervised feature selection," *Bioinformatics*, vol. 26, no. 3, pp. 440–443, 2009.

[44] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern recognition*, vol. 45, no. 9, pp. 3084–3104, 2012.

[45] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, 2014.